

Robust Convex Optimisation Techniques for Autonomous Vehicle Vision-based Navigation



Mohammed BOULEKCHOUR

Centre for Electronic Warfare
Cranfield University

This dissertation is submitted for the degree of
Doctor of Philosophy

Supervisor: Dr. Nabil AOUF

I would like to dedicate this thesis to my loving parents,
to my wife,
to Mahdi Amine,
to Didou (Imad Eddine),
to Merroula (Miral)

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements.

Mohammed BOULEKCHOUR

July 2015

Acknowledgements

I would like to thank my supervisor Dr Nabil Aouf for his support, guidance, optimism and enthusiasm throughout the past three years. I would also like to thank Professor Mark Richardson for his support and help during this research.

I must give special thanks to the Department of Military Manufacturing for giving me the opportunity to undertake this research.

I wish to express my gratitude to past and current members of Cranfield University and the Defence Academy of the UK, Shrivenham, for providing a good environment for research.

I treasure the time I spent in Shrivenham. My thanks go to all my friends and colleagues at the Autonomous Systems Laboratory who made my studies an enjoyable experience. Special thanks to Ivan Vitanov for his help in proof reading this thesis. Thanks to Abdenour A, Abdenour K, Ahmed, Özgün, Saif, Riad, Tarek, Lounis, Oualid, Xiaodong, Luke, Abdelmadjid, Amine, Abdelalim, Bilel and Rahim.

Thanks to my family for their love and concern. To my parents, to whom I owe almost everything: thank you ever so much for your prayers, love and support. To my brothers and sisters. Special thanks to the little Meriem.

Cheers to my wife Souheila for her love, support and invaluable advice. To Mahdi Amine and Imad Eddine for the enjoyable time we spent. To Merroula (Miral) for the brightness she brought with her. You are my everyday motivation, thank you for your encouragement and enduring patience.

Abstract

This thesis investigates new convex optimisation techniques for motion and pose estimation. Numerous computer vision problems can be formulated as optimisation problems. These optimisation problems are generally solved via linear techniques using the singular value decomposition or iterative methods under an L_2 norm minimisation. Linear techniques have the advantage of offering a closed-form solution that is simple to implement. The quantity being minimised is, however, not geometrically or statistically meaningful. Conversely, L_2 algorithms rely on iterative estimation, where a cost function is minimised using algorithms such as Levenberg-Marquardt, Gauss-Newton, gradient descent or conjugate gradient. The cost functions involved are geometrically interpretable and can statistically be optimal under an assumption of Gaussian noise. However, in addition to their sensitivity to initial conditions, these algorithms are often slow and bear a high probability of getting trapped in a local minimum or producing infeasible solutions, even for small noise levels.

In light of the above, in this thesis we focus on developing new techniques for finding solutions via a *convex optimisation* framework that are globally optimal. Presently convex optimisation techniques in motion estimation have revealed enormous advantages. Indeed, convex optimisation ensures getting a global minimum, and the cost function is geometrically meaningful.

Moreover, robust optimisation is a recent approach for optimisation under uncertain data. In recent years the need to cope with uncertain data has become especially acute, particularly where real-world applications are concerned. In such circumstances, robust optimisation aims to recover an optimal solution whose feasibility must be guaranteed for any realisation of the uncertain data. Although many researchers avoid uncertainty due to the added complexity in constructing a robust optimisation model and to lack of knowledge as to the nature of these uncertainties, and especially their propagation, in this thesis robust convex optimisation, while estimating the uncertainties at every step is investigated for the motion estimation problem.

First, a solution using convex optimisation coupled to the recursive least squares (RLS) algorithm and the robust H_∞ filter is developed for motion estimation. In

another solution, uncertainties and their propagation are incorporated in a robust L_∞ convex optimisation framework for monocular visual motion estimation. In this solution, robust least squares is combined with a second order cone program (SOCP). A technique to improve the accuracy and the robustness of the fundamental matrix is also investigated in this thesis. This technique uses the covariance intersection approach to fuse feature location uncertainties, which leads to more consistent motion estimates.

Loop-closure detection is crucial in improving the robustness of navigation algorithms. In practice, after long navigation in an unknown environment, detecting that a vehicle is in a location it has previously visited gives the opportunity to increase the accuracy and consistency of the estimate. In this context, we have developed an efficient appearance-based method for visual loop-closure detection based on the combination of a Gaussian mixture model with the KD-tree data structure.

Deploying this technique for loop-closure detection, a robust L_∞ convex pose-graph optimisation solution for unmanned aerial vehicle (UAVs) monocular motion estimation is introduced as well. In the literature, most proposed solutions formulate the pose-graph optimisation as a least-squares problem by minimising a cost function using iterative methods. In this work, robust convex optimisation under the L_∞ norm is adopted, which efficiently corrects the UAV's pose after loop-closure detection.

To round out the work in this thesis, a system for cooperative monocular visual motion estimation with multiple aerial vehicles is proposed. The cooperative motion estimation employs state-of-the-art approaches for optimisation, individual motion estimation and registration. Three-view geometry algorithms in a convex optimisation framework are deployed on board the monocular vision system for each vehicle. In addition, vehicle-to-vehicle relative pose estimation is performed with a novel robust registration solution in a global optimisation framework. In parallel, and as a complementary solution for the relative pose, a robust non-linear H_∞ solution is designed as well to fuse measurements from the UAVs' on-board inertial sensors with the visual estimates.

The suggested contributions have been exhaustively evaluated over a number of real-image data experiments in the laboratory using monocular vision systems and range imaging devices. In this thesis, we propose several solutions towards the goal of robust visual motion estimation using convex optimisation. We show that the convex optimisation framework may be extended to include uncertainty information, to achieve robust and optimal solutions. We observed that convex optimisation is a practical and very appealing alternative to linear techniques and iterative methods.

List of Publications

Journals

- Mohammed Boulekchour, Nabil Aouf and Mark Richardson. *Robust L_∞ convex optimisation for monocular visual odometry trajectory estimation*. Robotica Journal, Cambridge Journals, July 2014. (published);
- Mohammed Boulekchour, Nabil Aouf and Mark Richardson. *Robust L_∞ convex pose-graph optimisation for monocular localisation solution for UAVs*. Journal of Aerospace Engineering. December 2014 (published).
- Mohammed Boulekchour, Nabil Aouf and Mark Richardson. *Robust L_∞ Co-operative Motion Estimation for Multiple UAVs*. Journal of Field Robotics. (submitted).
- Saad Imran, Nabil Aouf and Mohammed Boulekchour. *L_∞ -Based Estimation Technique for Time-Varying Homographies with System Uncertainty*. Journal of Intelligent and Robotic System. (submitted).

Conferences

- Mohammed Boulekchour and Nabil Aouf. *Efficient Real-Time Loop Closure Detection Using GMM and Tree Structure*. IEEE/RSJ International Conference on Intelligent Robots and Systems , IROS 2014, Chicago Illinois Sept. 14-18, 2014.
- Mohammed Boulekchour and Nabil Aouf. *Robust Motion Estimation Using Covariance Intersection*. 22nd Mediterranean Conference on control and Automation, MED 2014, June 2014, Palermo, Italy.
- Mohammed Boulekchour and Nabil Aouf. *L_∞ Norm Based Solution for Visual Odometry*. Computer Analysis of Images and Patterns - 15th International Conference, CAIP 2013, York, UK;

- Mohammed Boulekchour and Nabil Aouf. *Improved Monocular Navigation - Closed Loop Concept*. 12th Electro-Optics and Infra-red Conference, Shrivenham, Defence Academy of the UK, June 2014.

Table of contents

List of figures	xxi
List of tables	xxv
Nomenclature	xxvii
1 Introduction	1
1.1 Motivation	1
1.2 Visual motion estimation	2
1.3 Optimisation in multiple view geometry	3
1.4 Visual SLAM and Visual Odometry	5
1.5 Filtering option within the SLAM (vSLAM) framework	7
1.6 Experimental set-up	8
1.6.1 Laboratory data	8
1.6.2 Urban environment data	9
1.6.3 Mars/Moon analogue site data	10
1.6.4 Platforms used for the experimental validations	11
1.6.5 Software tools	14
1.7 Assumptions for the proposed solutions	14
1.8 Commercially available platforms with vision and inertial systems	15
1.9 Summary and outline of the thesis	16
1.10 Published and submitted manuscripts	21
2 Image Geometry	23
2.1 Overview	23
2.2 Notation	24
2.3 Image formation	25
2.4 Projective geometry	25
2.4.1 Point representation	25
2.5 Geometry of cameras	27

2.5.1	The pinhole camera model	27
2.6	Camera model	30
2.7	Image interest points (image features)	35
2.8	RGB channels in colour images	36
2.9	Two-view geometry - epipolar geometry	37
2.9.1	The essential matrix	37
2.9.2	Pose extraction from the essential matrix	39
2.9.3	The fundamental matrix	41
2.10	The three-view geometry	42
2.11	Optical flow approach	44
2.11.1	The 2D and 3D motion constraint equations	45
2.11.2	Optical flow based motion estimation methods	47
2.12	Conclusion	48
3	Convex Optimisation	49
3.1	Notation	49
3.2	Optimisation problem classes	50
3.2.1	Least squares solution	50
3.2.2	Iterative methods	51
3.2.3	Convex optimisation	52
3.3	Basic notions of convex optimisation	53
3.3.1	Affine sets	53
3.3.2	Convex set	53
3.3.3	Operations that preserve convexity	55
3.3.4	Affine functions	56
3.3.5	Convex function	57
3.3.6	Quasi-convex functions	58
3.3.7	Operations that preserve quasi-convexity	61
3.4	Convex optimisation problem	61
3.4.1	Definitions	61
3.4.2	Convex optimisation form	62
3.4.3	Convex feasibility problems	63
3.5	Quasi-convex optimisation problem	64
3.6	Basic examples of convex optimisation	65
3.6.1	Linear programming	65
3.6.2	Quadratic programming	66
3.6.3	Quadratic constrained quadratic program	66
3.7	Second-order cone programming	67

3.7.1	SOCP feasibility problems	69
3.8	Robust convex optimisation	72
3.8.1	Robust solution using SOCP	73
3.8.2	Robust least-squares via SOCP	74
3.9	Branch and bound algorithms	75
3.10	Conclusion	77
4	Convex Optimisation and H_∞ Filtering for Motion Estimation	79
4.1	Overview	79
4.2	L_∞ Norm based solution for visual odometry	80
4.3	Scale ambiguity in monocular systems	81
4.4	Related work	81
4.4.1	Scale ambiguity issue	81
4.4.2	Visual odometry	83
4.5	The Proposed method	83
4.6	The L_∞ motion estimation	84
4.7	Robust solution for scale estimation	87
4.7.1	Recursive least squares method	89
4.7.2	H_∞ Filter method	90
4.8	Experimental evaluation	95
4.8.1	Motion estimation	97
4.8.2	Absolute scale estimation	98
4.9	Conclusions	99
5	Robust Motion Estimation Using Covariance Intersection	101
5.1	Overview	101
5.2	The proposed solution	104
5.3	Related work	105
5.4	Feature location uncertainty	105
5.4.1	Basic notions of uncertainty	106
5.4.2	Uncertainty estimation for the Harris corner detector	108
5.4.3	Uncertainty estimation for SIFT feature positions	109
5.5	Covariance intersection	112
5.6	Robust fundamental matrix estimation	113
5.6.1	Matching using features uncertainty	114
5.6.2	Robust fundamental matrix estimation	115
5.7	Experimental results	117
5.8	Conclusion	122

6	Robust L_∞ Convex Optimisation for Monocular Motion Estimation	125
6.1	Overview	125
6.2	The proposed solution	127
6.3	Robust convex optimisation	129
6.4	Uncertainty propagation	130
6.5	Feature location uncertainty	132
6.6	Uncertainty in the rotation matrix and the translation vector	137
6.6.1	Uncertainty estimation in the fundamental matrix	137
6.6.2	Uncertainty estimation in the essential matrix	141
6.6.3	Uncertainty estimation in the rotation and translation	141
6.7	Feature uncertainty propagation to the 3D reconstruction	142
6.8	Robust L_∞ motion estimation solution	144
6.8.1	Computational cost	145
6.8.2	Robust L_∞ triangulation with uncertain data	145
6.8.3	Robust L_∞ camera resectioning with uncertain data	147
6.9	Robust scale estimation	148
6.10	Experimental results	149
6.10.1	Motion estimation	150
6.10.2	Motion estimation robustness	154
6.11	Conclusions	156
7	Real-Time Loop-Closure Detection	159
7.1	Overview	159
7.2	Related Work	161
7.3	Tools for loop-closure detection	164
7.3.1	Bayes decision theory for loop-closure	165
7.3.2	Gaussian mixture modelling (GMM)	166
7.3.3	KD-tree Structure	167
7.3.4	Image features and their matching	168
7.3.5	Keyframe selection	169
7.4	Gaussian mixture Bayes solution	171
7.4.1	Bayesian loop-closure detection module	171
7.4.2	Gaussian mixture modelling (GMM) module	173
7.4.3	The multi-view geometry module	175
7.5	The GMM KD-tree solution for loop-closure detection	177
7.5.1	The first stage: GMM with KD-tree data structure	178
7.5.2	The second stage: Full representation stage	179
7.5.3	The third stage: geometric consistency check	181

7.6	Computational complexity of the solution	181
7.7	Experimental validation	182
7.7.1	The Gaussian mixture Bayes solution	183
7.7.2	The GMM KD-tree solution	185
7.7.3	Comparison between the first and the second solutions	188
7.8	Conclusions	190
8	Robust L_∞ Convex Pose-graph Optimisation for Monocular Localisation Solution for UAVs	193
8.1	Introduction	193
8.1.1	Pose-graph representation	195
8.1.2	Classical pose-graph optimisation	195
8.2	Convex optimisation formulation	197
8.3	Robust convex optimisation formulation	199
8.4	Overview of the proposed solution	199
8.4.1	Monocular motion estimation module	201
8.4.2	Loop-closure module	202
8.5	Pose-graph optimisation module	203
8.5.1	Robust convex pose-graph optimisation formulation	204
8.5.2	The optimisation problem formulation	205
8.5.3	Robust Convex pose-graph optimisation formulation	207
8.5.4	Computational complexity	208
8.6	Experimental validation	208
8.6.1	Pose-graph optimisation with BoW method	212
8.6.2	Pose-graph optimisation with GMM/KD-tree method	212
8.6.3	Effect of loop-closure detection method	216
8.6.4	Robust convex pose-graph optimisation	216
8.7	Conclusions	218
9	Robust L_∞ Cooperative Motion Estimation	219
9.1	Introduction	220
9.1.1	Motivation	220
9.1.2	Visual sensing	220
9.1.3	Related work	222
9.2	Global optimisation	224
9.3	Overview of the proposed solution	225
9.4	Three-view geometry motion estimation	228
9.4.1	Trifocal tensor estimation using convex optimisation	229

9.4.2	Camera motion estimation from the trifocal tensor	234
9.5	Registration problem	235
9.5.1	Registration using least squares estimation	236
9.5.2	Overview of the proposed solution for the registration	236
9.5.3	Implementation	241
9.6	Filtering based approach for relative pose estimation	241
9.6.1	Non-linear Filter design	244
9.6.2	State representation	245
9.6.3	Error State Representation	246
9.6.4	State Prediction	247
9.6.5	Measurements	248
9.6.6	Filter Update	249
9.7	Experimental validation	250
9.7.1	Motion estimation for each UAV	251
9.7.2	Relative pose estimation	257
9.8	Conclusions	265
10	Discussion and Conclusion	269
10.1	Overview	269
10.2	Summary and discussion of contributions	269
10.3	Future work	272
Appendix A	Convex Optimisation and Multiple View Geometry	275
A.1	Triangulation problem	275
A.1.1	Experiments	279
A.2	Camera resectioning	281
A.2.1	Experiments	282
A.3	Homography estimation	283
A.3.1	Experiments	285
Appendix B	The KD-tree Data Structure	287
B.1	Construction of a KD-tree:	287
B.2	Nearest-neighbour search in KD-trees	288
B.3	Computational complexity	290
Appendix C	The Fundamental Matrix and Image Interest Point Ex-	
	traction	293
C.1	The Fundamental matrix	293
C.2	Image interest point extraction	298

C.2.1	Harris feature detector	299
C.2.2	Scale invariant feature transform (SIFT)	300
C.2.3	Colour features	302
C.3	Feature position uncertainties	304
References		307

List of figures

1.1	Example of ground vehicle for visual odometry	3
1.2	The Triangulation Problem	4
1.3	Unmanned Autonomous Systems Laboratory (UASL)	9
1.4	Examples of images used in the experiments	10
1.5	Platform used to collect data in Mars/Moon analogue site	11
1.6	The overall thesis outline	18
2.1	Image representation	26
2.2	The pinhole camera model	29
2.3	Reference frame in the pinhole camera model	30
2.4	Transformation between camera and world reference frame	31
2.5	Example of a colour image and its RGB channels	36
2.6	Image matrix representation	36
2.7	Two-view geometry	38
2.8	Three-view geometry.	43
2.9	Optical flow as seen by ground vehicle	45
2.10	The 2D Motion Constraint	46
3.1	Local and global solutions	52
3.2	Affine sets	53
3.3	convex and non-convex sets	54
3.4	Examples of convex sets in \mathbb{R}^3	55
3.5	Example of affine function and affine set	57
3.6	Graph of a convex function	57
3.7	Quasi-convex functions	59
3.8	Quasi-convex function and their pointwise maximum	60
3.9	Example of the objective function	63
3.10	The geometric interpretation of the LP	66
3.11	The geometric interpretation of the QP	67
3.12	The branch and bound algorithm	76

4.1	Scale ambiguity problem in monocular vision systems	82
4.2	Illustration of the motion estimation pipeline.	84
4.3	The triangulation problem	85
4.4	Algorithm for recursive least squares estimation	90
4.5	Examples of images used in the experiments	95
4.6	Comparison between trajectory estimates	96
4.7	Illustration of feature matching	96
4.8	Camera motion estimation errors	97
4.9	Errors for the motion estimation using convex optimisation and LM algorithm	98
4.10	Motion estimation after convex optimisation	99
4.11	Motion estimation errors after convex optimisation	100
5.1	Pipeline of the proposed solution for motion estimation	103
5.2	Noise models	108
5.3	Harris features with position uncertainties	110
5.4	Harris features with position uncertainties for each RGB channel . . .	110
5.5	SIFT features with position uncertainties	111
5.6	SIFT features with position uncertainties for each RGB channel . . .	112
5.7	Example of covariance intersection	114
5.8	Examples of image pairs used for comparison	118
5.9	The fundamental matrix errors	119
5.10	Fundamental matrix estimation with the covariance intersection . . .	121
5.11	Comparison between using uncertainties from SIFT and from Harris .	122
5.12	A summary comparative graph	123
6.1	The main architecture of the proposed solution	128
6.2	Harris image features with location covariances	133
6.3	SIFT image features with location covariances	134
6.4	Average feature points localisation errors using Harris	135
6.5	Average feature points localisation errors using SIFT	135
6.6	3D points uncertainties against their depths	144
6.7	Set-up used in our indoor experimental	150
6.8	Camera motion estimation errors	152
6.9	Estimates using robust convex optimisation and the classical BA . . .	153
6.10	The H_{∞} filter and the robust least squares	154
6.11	Performance of the motion estimation algorithm	155
6.12	Re-projection errors for the four different scenarios	157

7.1	Image representation with the "bag-of-words" concept	162
7.2	A sample image from the outdoor environment	168
7.3	Main modules of the Gaussian mixture Bayes Solution	170
7.4	Combination of GMM with KD-trees for Loop-Closure Detection . . .	176
7.5	Example of key-frames forming a loop-closure	183
7.6	Loop closure detection using GMM with Bayesian decision theory . .	184
7.7	Key-frames forming a loop-closure in the outdoor dataset	186
7.8	Loop closure detection using GMM with Bayesian decision theory . .	186
7.9	Example of loop-closure	187
8.1	A typical pose-graph representation	196
8.2	Architecture of the proposed solution	200
8.3	Convex pose-graph optimisation problem	205
8.4	Set-up used in the indoor experiments	209
8.5	Convex pose-graph optimisation results on indoor experiment	210
8.6	Convex pose-graph optimisation results on outdoor experiment	211
8.7	Comparison of convex pose-graph optimisation to classical LM method	212
8.8	Comparison of convex pose-graph optimisation with classical	213
8.9	Effects of loop-closure detection method on motion estimates	215
8.10	Robust convex pose-graph optimisation	217
9.1	The set-up used in the cooperative motion estimation	225
9.2	Architecture of the proposed solution	227
9.3	Results on synthetic and real data	240
9.4	Set-up used for the experimental validation	251
9.5	Sample of images used to detect the loop-closure	252
9.6	Trifocal tensor computation results	254
9.7	Performance of motion estimation	255
9.8	The estimated trajectories	256
9.9	Relative pose estimation using the registration solution	257
9.10	Results for the relative rotation between the vehicles	259
9.11	Results for the relative pose between the vehicles	260
9.12	The relative rotation between the vehicles using the Non-linear H_∞ .	262
9.13	The relative translation using the registration and the NH_∞	264
9.14	Error evolution over time before and after loop-closure detection . . .	266
9.15	Error evolution over time before and after loop-closure detection . . .	267
A.1	The triangulation problem	276
A.2	Errors of the triangulation solution	280

A.3	The problem of camera resectioning	281
A.4	Camera resectioning problem	283
A.5	Homography Estimation	284
A.6	Errors for the the homographies estimation using convex optimisation	285
B.1	Nearest Neighbour Search illustration	288
B.2	Illustration of a KD-tree construction	289
B.3	Illustration of Nearest Neighbour Search in a KD-tree	291
C.1	The epipolar geometry	294
C.2	Image point correspondences using the epipolar constraints	295
C.3	Hue Saturation Value (HSV)	303
C.4	Example of a colour image and its HSV representation	303
C.5	SIFT and colour descriptors	304

List of tables

1.1	Technical Data - AscTec Firefly	12
1.2	Technical Data - AscTec Pelican	12
1.3	Technical Data - Mobile-Robots Pioneer P3DX	13
1.4	Technical Data - mvBlueFOX-MLC	13
4.1	Errors for the motion estimation using convex optimisation and LM algorithm	98
5.1	Average residual errors in pixels	120
6.1	Average feature points localisation errors using Harris and SIFT . . .	136
6.2	The four scenarios for robustness investigation	154
6.3	Average back projection errors	155
7.1	Performance in the indoor environment	184
7.2	Performance in the outdoor environment	185
7.3	Performance in the hallway environment	187
7.4	Computational time	188
7.5	Performance comparison: GMM-Bayesian and GMM-KD-tree solution	189
8.1	The two comparison scenarios	209
9.1	Errors for the relative pose estimation using NH_∞ and EKF	263

Nomenclature

Acronyms / Abbreviations

3D Three dimensional

BA Bundle Adjustment

CI Covariance Intersection

CVSLAM Cooperative Visual Simultaneous Localisation and Mapping

CVX Convex Optimisation

DLT Direct Linear Transform

EKF Extended Kalman Filter

GMM Gaussian Mixture Modelling

GPS Global Positioning System

ICP Iterative Closes Point

IMU Inertial Measurement Unit

INS Inertial Navigation System

KF Kalman Filter

LM Levenberg – Marquardt Algorithm

LP Linear Programming

LS Least-Squares

MAV Micro aerial vehicle

QCQP Quadratic Constrained Quadratic Program

QP Quadratic Programming

RanSaC RANdom SAmple and Consensus

RMS Root Mean Squares

SIFT Scale Invariant Feature Transform

SOCP Second Order Cone Programming

SVD Singular Value Decomposition

UAV Unmanned aerial vehicle

VO Visual Odometry

VSLAM Visual Simultaneous Localisation and Mapping

Chapter 1

Introduction

This work provides an introduction to the motion estimation problem in computer vision using novel convex optimisation techniques. The optimisation impact on visual motion estimation is revealed here and, in particular, convex optimisation method is discussed along with its competitors techniques, such as linear and iterative optimisation techniques. This would show the motivation and the advantage of using convex optimisation for motion estimation as an alternative to linear and iterative methods.

1.1 Motivation

The main objective of this thesis is to investigate the adaptation of ***convex optimisation*** in visual motion estimation problem. First, let us consider the scenario where a digital camera is moving through an unknown environment. Obviously, considerable visual information are being recorded and usually comprised of two-dimensional images. In this scenario, it is possible to match the images to each other, and then to recover camera displacement using multiple-view geometry algorithms [82, 200]. If one performs this displacement estimation and then optimises its parameters, we speak of ***motion estimation***.

In this thesis, we discuss how the optimisation in general can be efficiently used in the problem of motion estimation. In particular, we focus on convex optimisation and robust convex optimisation, where uncertainties are incorporated. As a global optimisation tool, branch and bound optimisation technique is considered as well in this thesis for non-convex problems. In contrast to linear and iterative methods, one might wish to formulate the motion estimation problem as a meaningful optimisation problem that converges to the global minimum. The main objective of this thesis is to develop techniques that are guaranteed to recover the robust and global solution.

Indeed, our goal is to extend the state-of-the-art for motion optimisation problems by formulating them as multiple-view geometry problems. In addition to motion estimation, our implementations include loop-closure detection, registration, feature matching, pose-graph optimisation and cooperative navigation. We focus on monocular systems, in which a single camera is deployed. Monocular systems have become an unavoidable solution for autonomous navigation systems, since they are practical and offer cheap and compact installations.

Unmanned and micro aerial vehicles will shortly be used in important operations, such as inspection, reconnaissance, surveillance and search and rescue. Moreover, they are expected to achieve similar performance as manned aircraft. The GPS and other satellite navigation systems may offer valuable assist for these aerial vehicles to accomplish their tasks. However, relying solely on satellite-based navigation systems is not fully reliable, especially in crucial and rapid operations. Urban- and indoor-environment operations present real handicaps to these navigation systems, where its availability is extremely limited or even does not exist at all. Inertial navigation system (INS) or GPS-aided INS systems might be used in these situations. However, the eventual growth in INS errors is prohibitive to these applications. Therefore, investigating other alternative solutions, such as visual systems, has become a priority for many research programmes. Indeed, vision systems are relatively more convenient for unmanned aerial vehicles (UAVs) due to their light-weight, low-cost and the great quality of information they provide.

Clearly, vision is the most significant sense for humans. It allows many tasks to be completed with simplicity, such as walking, directing themselves and recognising previously visited places. Recently, the decreasing cost of digital cameras has put their popularity in a steady increase. Thus, it is no longer difficult or expensive to set up an application that uses multiple cameras [82]. These systems have many benefits in real applications such as visual effects and scientific research.

1.2 Visual motion estimation

Several studies have been piloted during the last three decades on the theory and the geometry of single-camera systems, which are used to capture images from two, three and multiple viewpoints. However, complete and exhaustive solutions have not been given or applied yet.

This process of recovering the orientation and position information of a moving vehicle using only vision systems is known in the scientific community as the visual motion estimation or the 'visual odometry' problem (Figure 1.1). More precisely, visual odometry is the process of estimating the movement of a camera by matching

point features between pairs of consecutive image frames. No prior knowledge of the scene or the motion is necessary [188].



Fig. 1.1 Example of ground vehicle for visual odometry.

On the other hand, multiple view geometry is the subject where relations between the coordinates of the image points in different views are studied. The three main tasks in multiple view geometry are:

- **Points matching problem:** this problem deals with finding the corresponding image points between some extracted reference points in one image with the ones extracted in the other images;
- **Scene structure problem:** this problem tries to recover the 3D structure of the scene under process. This is known also as scene construction;
- **Motion estimation problem:** using visual input only, this problem estimates the rotations and translations of a moving camera along a path.

Therefore, the first step of any 'feature-based' multiple-view algorithm is to detect the image points (known also as *image features*) which are in correspondence. In general, image features in multiple views framework are said to be in **correspondence** if they represent the same feature in the 3D space. The problem of detecting corresponding image features is known as **the matching problem** [159]. Due to their extraction techniques, image features locations accuracy is heavily dependent on the variation in intensity within their neighbourhoods, from which their uncertainties are estimated. In addition to their extraction, in this thesis we investigate the incorporation of feature position uncertainties in motion estimation problems.

1.3 Optimisation in multiple view geometry

The main task of the optimisation in multiple views systems is to recover the optimal positions of both the image features and the cameras when a sufficient number of image features is measured, where the only source of information is images. It is important to note that the two tasks of reconstructing the structure and recovering

the camera positions are interlinked and should not be considered as two decoupled problems.

Let us consider for example the triangulation problem as illustrated in Figure 1.2, where a 3D scene point X is seen from two camera positions P_1 and P_2 . Let x_1 and x_2 be the projection of X into cameras P_1 and P_2 respectively. This theoretically means that the two lines of sight (P_1, x_1) and (P_2, x_2) intersect in the 3D point X . The question now is: do the two image points x_1 and x_2 represent a unique point in space? In fact there is noise in the measurements of x_1 , x_2 , P_1 and P_2 . This means that the two lines of sight $(P_1; x_1)$ and $(P_2; x_2)$ will not necessarily intersect, hence an optimisation procedure should be performed to select the 3D scene point X that projects to image points that are as close as possible to the measured image points x_1 and x_2 [123]. The best what we can do is to estimate a 3D point \hat{X} that projects to image points \hat{x}_1 and \hat{x}_2 via the cameras P_1 and P_2 . In this case, the projection error is the sum the squared Euclidean distance d_1 between x_1 and \hat{x}_1 and the squared Euclidean distance between x_2 and \hat{x}_2 . Therefore, the goal is to find \hat{X} that minimises this projection error:

$$\min_{\hat{X}} \left[d_1(x_1, \hat{x}_1)^2 + d_2(x_2, \hat{x}_2)^2 \right]. \quad (1.1)$$

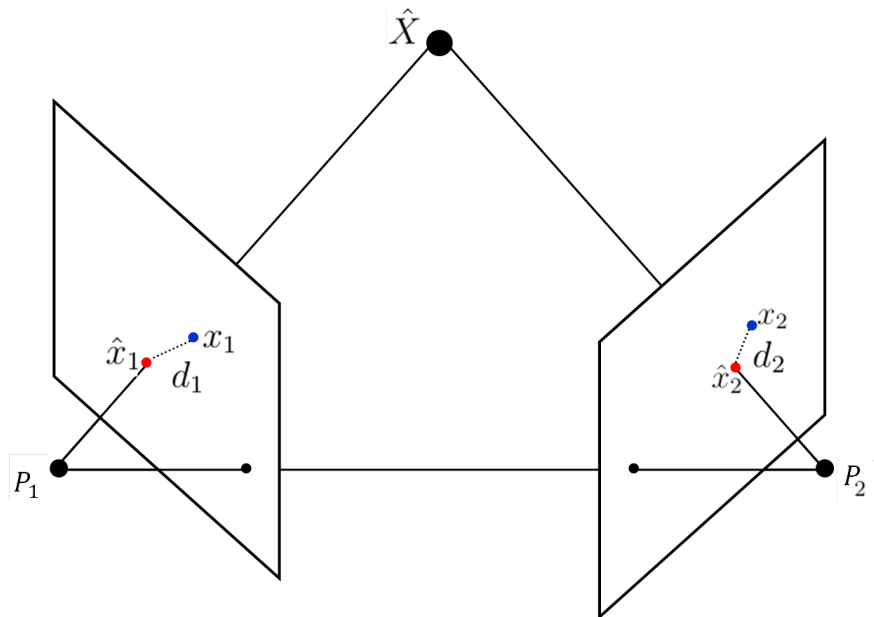


Fig. 1.2 The Triangulation Problem: recovering the 3D structure given two views of the same scene

Many algorithms have been proposed over the last few decades to solve this kind of optimisation problems. These algorithms can be classified into two main categories. The first category includes linear approaches, which is based on least squares optimisation by minimising the algebraic cost function, leading to using the singular value decomposition (SVD) [82, 84]. This approach is efficient and yields a closed-form solution. However, its main drawback is the fact that the quantity being minimised is not geometrically or statistically meaningful [25].

The second category includes iterative methods, where algorithms such as Levenberg-Marquardt, Gauss-Newton, Newton, gradient descent or conjugate gradient are used to minimise a geometric cost function in iterative way [199]. In this category the cost function has a geometrical meaning and, under an assumption of Gaussian noise, shown to be statistically optimal. However, the core problem with these methods, and notwithstanding of dependency on good initialisation, is related to the high probability of converging to a local minimum or even an infeasible solution.

As a powerful alternative, convex optimisation represents a third category offering the possibility of getting around problems that linear and iterative approaches have [25]. The cost function of this optimisation technique is geometrically meaningful, and has a single global minimum [94].

1.4 Visual SLAM and Visual Odometry

Imagine a digital camera is moving through its environment and acquiring sequence of images. Exploiting the rich amount of information in images, camera motion can be estimated by aligning the frames to each other and using the multiple view geometry. In fact, this motion consists on recovering the trajectory, which is built up from the estimated camera positions and orientations at different time steps. Specifically, the motion of image points (known as well as image features) can be used to determine the trajectory of the camera and the three dimensional structure of the scene. Two main motion estimation categories can be distinguished: "visual odometry - VO" (visual motion estimation) and "visual simultaneous localisation and mapping - visual SLAM (vSLAM)"

Though an exact discriminative line between the two categories is not fully defined, some properties of the employed algorithm can define its category. Specifically, if the algorithm relies on image feature matching between pairs of images, this algorithm belongs to the "visual odometry - VO" category (visual motion estimation). However, if the matching is performed between a map of the scene structure and the current image, it is identified as "visual simultaneous localisation and mapping - vSLAM" [210]. Our approach in this thesis belongs to the former category. Indeed, throughout

this thesis we discuss how image features between images can be used as a general tool for motion estimation via convex optimisation.

It is worth-noticing some basic notions about the SLAM category. In the SLAM framework, the problem is to estimate the motion of a moving vehicle as it continuously observes and maps its unknown environment using sensors which do not necessarily include cameras. When cameras are employed as the only exteroceptive sensor, it is called visual SLAM (vSLAM). In some applications this is referred as vision-based SLAM. Many studies have been presented in the last decade in which visual systems are used as the only external perception for SLAM systems [47, 103, 153, 162]. This is due to rich amount of information that cameras can provide.

The first notions of SLAM are started to appear during the period of 1985-1990, when Chatila and Laumond [34] and Smith et al. [181] proposed a mapping and localisation framework. After a while later, this problem took the name of SLAM (simultaneous localization and mapping). The key feature of SLAM is its capacity of building a global map of the environment and uses this map to deduce its own location at any time step [62]. In order to successfully do that, the system must possess exteroceptive (range lasers, sonar, cameras or GPS) and proprioceptive (encoders, accelerometers and gyroscopes) sensors. However, all these sensors are noisy and have limited range capabilities. Therefore, their ability to accurately estimate the vehicle position is compromised since errors are cumulative.

Simultaneous localisation and mapping (SLAM) algorithm is extensively formulated for the indoor and outdoor ground vehicle applications. This approach is stated as follows: starting from an initial position, the vehicle navigates through an unknown environment and obtains a set of sensor measurements at each position. The final aim is to process the sensor measurements to estimate the position while concurrently building a map of its environment. SLAM is known as an expensive algorithm, especially for the 6DOF implementation. Increasing the state size by including the sensor errors, and dealing with the sampling rates when using the inertial sensors, further intensifies the problem. In more details, when the number of features in the system state increases, then computational cost grows rapidly and consequently it becomes difficult to maintain the frame rate operation. To solve this problem, old features can be removed from the state to maintain a stable number of features. However, if old features are removed, then previous mapped areas cannot be recognized in the future. In the context of visual odometry, this fact is not considered as a problem.

In contract, visual odometry approach (visual motion estimation) is based on consecutive pairs of images to exclusively estimate the relative motion, neglecting scene

construction. This method can work in real time with significant less computational complexity.

As a summary, it is truth that the SLAM systems are able to achieve drift free position estimations of a camera relative to a jointly estimated map of landmarks (features). Due to the aforementioned computational complexity and in order to allow real-time operation, the map is kept quite sparse with usually only tens of landmarks visible in each frame. In contrast, visual odometry technique is allowed to use and track hundreds of features per frame. This certainly leads to a very accurate estimate of the relative camera motion, but without a persistent map, the estimate tends to drift over time.

1.5 Filtering option within the SLAM (vSLAM) framework

Two different approaches to deal with system uncertainties in real-time motion estimation applications, the optimisation and the filtering approaches. Both approaches have proven some successful, but they deal with the problem in completely different ways. Filtering option ignores past poses and relies on a probability distribution framework. The optimisation approach, on the other hand, uses the bundle adjustment (BA), but computationally must select only a small number of past frames to process.

In the SLAM technique, it is possible to create a consistent map of the environment. The main advantage of this technique is that repeated observation of the same landmark (or feature) ensures consistent trajectories where drifts over time are avoided. However, continuously and simultaneously building the map is computationally expensive. In fact, filtering option is adopted within the SLAM framework. Most proposed solutions for SLAM systems are based on the extended Kalman filter (EKF) and conditioned in the map size by the computational complexity, which certainly limits the number of feature matches [47, 210].

The filtering option (also referred as probabilistic filters) is the most commonly used approach due to its ability to provide the best results. This option is successful on small scale; however it is significantly limited when the vehicle is required to navigate in large environments, especially when including loop closure constraints [62].

The filtering-based approach to the SLAM is defined by a state vector composed mainly of the vehicle position and the map elements (landmarks positions). This state vector is recursively estimated from the non-linear models of transition and

observation. In this framework, the uncertainty is identified by probability density functions (pdf). Mainly, it is assumed that the propagation of the mean and covariance of these pdf are close to the optimal solution. However, the filtering option is known to be sensitive to outliers. A single inaccurate measurement can lead to the divergence. In addition, the SLAM quadratic complexity with respect to the number of the map features limits its deployment in large environments. Some techniques are introduced to cope with this problem such as atlas framework [22] and sparse extended information filter (SEIF) [194]. Other techniques based on particle generation process have managed to reduce the complexity to logarithmic, $\mathcal{O}(p \log n)$, where p is the number of particle and n is the number of features (landmarks) on the map. However an optimal number of particles in these methods is still not identified. Many other solutions have been proposed to increase the number maintained features; however they are required to perform within reduced scale environments. Davison presented in [46] a successful real-time monocular filtering system, called MonoSLAM. This solution employs a single digital camera to simultaneously estimate a 3D metric map and the vehicle position. However, this system is limited to work only in confined and indoor environments and suffers from the initialisation problem of the features (landmarks).

As a summary, in the filtering option the SLAM problem, all previously estimated poses are marginalised out after every frame and only features that can be observed again are maintained. However the main issue is the map size and the computational cost which grows extremely fast.

1.6 Experimental set-up

Experimentation validations of the proposed solutions have been conducted on real data from different environments. Indoor and outdoor experiments are conducted on ground and aerial platforms equipped with monocular vision systems. While indoor experiments are held on our laboratory, outdoor experiments include data collected from three different locations. The first one consists of data collected at the university campus, the second one are data collected from a vehicle travelling in an urban environment and the third one is a collection of data gathered at a Mars/Moon analogue site at Devon Island, Nunavut.

1.6.1 Laboratory data

Indoor experiments are conducted at the unmanned autonomous systems laboratory (UASL) at the university (Figure 1.3). In this laboratory, real experiments are

performed within a flying volume of about $15 \times 8 \times 10 \text{ m}^3$. During each experiment, the ground-truth is collected using the OptiTrack motion-capture system [1].

OptiTrack is a motion capture system designed by NaturalPoint Inc. Position and orientation data of the vehicles, streamed from the optitrack cameras, can be saved in CSV (comma separated values) format or sent over network in real-time. This system provides absolute position information with millimetre accuracy at 100 Hz [1]. This system employs a set of six cameras with 1.3 Megapixels resolution and 56 degrees field of view.



Fig. 1.3 Unmanned Autonomous Systems Laboratory (UASL)

1.6.2 Urban environment data

Data from the urban environment are collected via a vision system mounted on a vehicle, where a pointing-forward calibrated camera is mounted on the roof of this vehicle [67]. This sequence consists of high quality images, with a resolution of 1344×372 pixels. This dataset is known as the KITTI dataset and has been recorded from a moving vehicle while driving in and around Karlsruhe, Germany. It includes camera images, laser scans, high-precision GPS measurements and IMU accelerations from a combined GPS/IMU system.

Figure 1.4 shows a sample of images from this dataset. The sensor set-up of this dataset includes:

- $2 \times$ PointGray Flea2 grayscale cameras (FL2-14S3M-C), 1.4 Megapixels, 1/2" Sony ICX267 CCD, global shutter.
- $2 \times$ PointGray Flea2 color cameras (FL2-14S3C-C), 1.4 Megapixels, 1/2" Sony ICX267 CCD, global shutter.

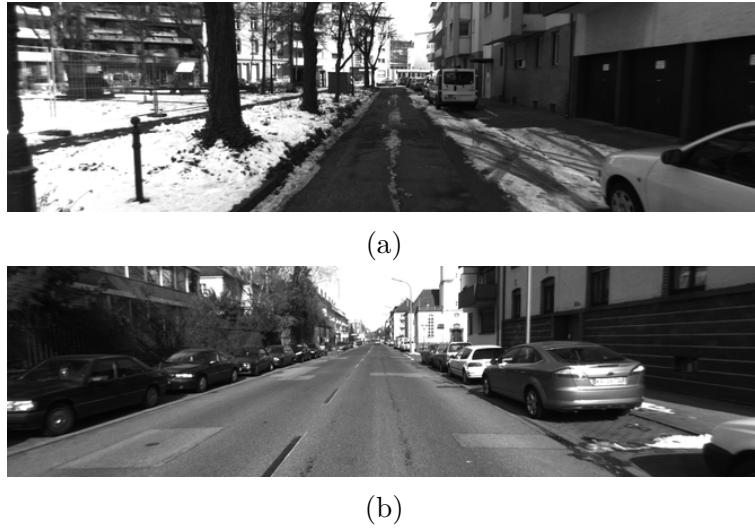


Fig. 1.4 Examples of images used in the experiments [68].

- 4 × Edmund Optics lenses, 4mm, opening angle $\sim 90^\circ$, vertical opening angle of region of interest (ROI) $\sim 35^\circ$.
- 1 × Velodyne HDL-64E rotating 3D laser scanner, 10 Hz, 64 beams, 0.09° angular resolution, 2 cm distance accuracy, collecting ~ 1.3 million points/second, field of view: 360° horizontal, 26.8° vertical, range: 120 m.
- 1 × OXTS RT3003 inertial and GPS navigation system, 6 axis, 100 Hz, L1/L2 RTK, resolution: 0.02m / 0.1° .

1.6.3 Mars/Moon analogue site data

The Devon Island rover navigation dataset is a collection of data gathered using a rover at a Mars/Moon analogue site on Devon Island, Nunavut in the Canadian High Arctic ($75^\circ 22'N$ and $89^\circ 41'W$). This dataset collects rover traverse data including stereo images, Sun vectors, inclinometer data, and differential global positioning system (DGPS) position [64]. The rover passes through areas with vegetation-free, planetary-analogue, rocky canyons, boulder fields, sandy flats and significant topographic relief terrain. For the ground-truth position pair of Magellan ProMark3 GPS units is used to produce post-processed DGPS data for the whole traverse. The images in the dataset are collected using an odometric trigger with approximately one image collected every 20 cm travelled. The sensor setup used to collect this dataset includes (Figure 1.5):

- Point Grey Research Bumblebee XB3 24 cm baseline stereo colour camera, with a resolution of 1280×960 pixels and a capture rate of approximately one image every 20 cm;

- A pair of Magellan ProMark3 GPS units (DGPS), with a capture rate of 1 Hz;
- Honeywell HMR-3000 inclinometer, estimating the pitch and the roll, with a capture rate of 1 Hz.

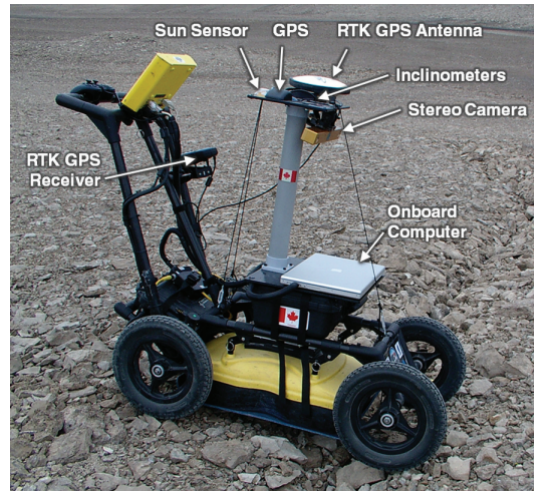


Fig. 1.5 Platform used to collect data in Mars/Moon analogue site [64].

1.6.4 Platforms used for the experimental validations

The main platforms used in our experimental validations in indoor and outdoor environments include:

- **AscTec MAV Firefly platform:** This platform is the most advanced UAV of the AscTec research line. The hexacopter offers the vibration damped slots for various payloads. Table 1.1 shows the technical data of this platform.
- **AscTec MAV Pelican platform:** This quadcopter offers plenty of space and various interfaces for individual components and payloads. Table 1.2 shows the technical data of this platform.
- **Mobile-Robots Pioneer P3DX:** The Pioneer 3DX (P3DX) from Mobile Robotstext™ is an advanced research robot that has an on-board PC, a range of sensors (including a laser range finder), and communicates via WiFi. Pioneer research robot is the world's most popular intelligent mobile robot. Table 1.3 shows the technical data of this platform.

In order to implement our visual motion estimation algorithms, these platforms are equipped with a mvBlueFOX-MLC camera. These kind of cameras are characterised by their on-board 8 M pixels memory. Calibration of these cameras is performed

Table 1.1 Technical Data - AscTec Firefly


	
UAV Type	Hexacopter
On-board computer	Intel® Core™ i7 processor
Size	605 × 665 × 165 mm
Max. take off, weight	1,6 kg
Max. payload	600 g
Flight time, incl. payload	12-14 min.
Range	4,500 m ASL, 1,000 m AGL
Max. airspeed	15 m/s
Wireless, communication	2,4 GHz XBee, link, 10-63 mW, WiFi
Inertial, guidance system	AscTec, AutoPilot with 1,000 Hz update rate
Flight modes	GPS Mode, Height Mode, Manual Mode
Emergency, modes	Direct, landing, Come-home straight, Come-home high

Table 1.2 Technical Data - AscTec Pelican


	
UAV Type	Quadcopter
On-board computer	Intel® Core™ i7 processor
Size	651 × 651 × 188 mm
Max. take off, weight	1,65 kg
Max. payload	650 g
Flight time, incl. payload	16 min.
Range	4,500 m ASL, 1,000 m AGL
Max. airspeed	15 m/s
Wireless, communication	2,4 GHz XBee, link, 10-63 mW, WiFi
Inertial, guidance system	AscTec, AutoPilot with 1,000 Hz update rate
Flight modes	GPS Mode, Height Mode, Manual Mode
Emergency, modes	Direct, landing, Come-home straight, Come-home high

Table 1.3 Technical Data - Mobile-Robots Pioneer P3DX

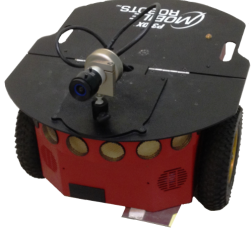
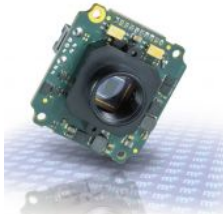
	
Robot Weight	9 kg
Operating Payload	17 kg
Max. Forward/Backward Speed	1.2 m/s
Rotation Speed	300 °/s
Max. Traversable Step	2.5 cm
Traversable Terrain	Indoor, wheelchair accessible
Run Time	8-10 hours w/3 batteries
Charge Time	12 hours
Batteries	Supports up to 3 at a time, Voltage: 12 V
User Control Panel	MIDI programmable piezo buzzer

Table 1.4 Technical Data - mvBlueFOX-MLC

	
Resolution	1280 × 960
Mpixels	1.2
Shutter type	Rolling
Rotation Speed	300 °/s
Sensor size	1/3"
Pixel size [μm]	3.75 × 3.75
Exposure time	100 μs - 10 s
SNR	40 dB
Focal length	[1638.63 1634.15] ^T
Focal length uncertainty	[5.30 5.26] ^T
Principal point	[627.46 479.09] ^T
Principal point uncertainty	[7.78 6.35] ^T
Skew coefficient uncertainty	0.00
Distortion coefficients	[-0.47 0.19 - 0.0028 0.00075 0.00] ^T

t our laboratory using the camera calibration toolbox for Matlab™ given in [92]. Table 1.4 shows the technical data of this camera.

1.6.5 Software tools

OpenCV computer vision library is employed the in our solutions within the C/C++ programming tool. In addition, Matlab™ software is used with two the main toolboxes: SeDuMi [187] and Yalmip [116].

1.7 Assumptions for the proposed solutions

As we aim in our monocular motion estimation algorithms for generic trajectories, we explicitly avoid constraints on the vehicles' motion and derive our solutions for the full 6 DoF case. However, few assumptions for simplicity are still required, which however do not affect the proposed solutions.

Mainly, in matching task, the brightness constancy assumption is made. This assumption simply says that if \mathbf{x} and \mathbf{x}' are two matching points, then the intensity of \mathbf{x} in the first image should be reasonably similar to the intensity of \mathbf{x}' . Hence, given \mathbf{x} , we may find its match by finding the point \mathbf{x}' in the second image that has the closest intensity to \mathbf{x} .

In addition, within reasonable assumptions on lighting condition, surface reflective property, and geometry of objects, the proposed solutions are able to fully capture information of a three-dimensional scene by a collection of two-dimensional images taken from different camera positions. These assumptions usually include:

- Lighting condition remains relatively stable while all the images are taken for a scene;
- Colour and brightness intensity of a point on a surface does not heavily change with the camera position;
- Objects in the scene consist of geometric primitives: vertex, edge, plane, and some simple curves or surfaces.

Our solutions also include another important assumption related to the nature of the scene being captured. It makes sense to require that there is always sufficient variation of colour and reflectance among these primitives such that it is possible to distinct them from one or another - a scene of purely white objects against a white background does not reveal itself well from any view.

An important parameter of the imaging system is the field of view (FOV). The field of view is twice the angle between the optical axis and the end of the retinal plane (CCD array), covering the scene under consideration. Therefore, in our monocular

vision systems it is acceptable to assume that consecutive cameras have a sufficient overlapping field of view, as is the case in standard stereo scenarios. This scene is assumed to be relatively static and includes sufficient texture and visual features, allowing consistent image feature extraction and matching. However, the developed solutions have the ability to cope with dynamic information that might occur during navigation. The proposed solutions are autonomous and not depending on any external source of information or human interaction.

In our multi-sensor systems, the inter-sensor calibration is crucial to the robustness of the motion estimation algorithms. While we assume the intrinsic camera parameters to be accurately estimated and remain unchanged, the inter-sensor calibration parameters describing the 6 DoF pose between the inertial measurement unit (IMU) and the camera are unknown. There exist various methods in the literature to calibrate these unknowns. In the real-world case, a good extrinsic calibration can be obtained using off-the-shelf toolboxes as presented in [115].

We also assume the flying UAV and the ground vehicles to have sufficient motion with a reasonable speed at any point in time. The proposed algorithms ensure real time performance as well. This includes all the operations, from image and inertial data acquisition to all the processes in motion estimation. A frame rate of 5 frames per second (fps) is considered as sufficient to achieve accurate navigation.

Under these assumptions, we may expect that full motion estimation can be obtained from multiple images. The following chapters of this thesis are going to show how.

1.8 Commercially available platforms with vision and inertial systems

Nowadays, digital video cameras, low-cost MEMS-based IMUs and GNSS devices are commercially available and some producers have already integrated such devices into their systems. Our research aims to develop robust algorithms for visual navigation systems in challenging environments where GPS signal is denied. A portion of our study includes as well the IMU sensors, where a solution fusing vision and inertial measurements is presented. Our research final goal is to present algorithms that can be implemented on the commercially available platforms equipped with vision and inertial systems.

Many platforms have become commercially available where camera is the dominant sensor for perceptions. A non-exhaustive list of these systems can include:

- Platforms used in our experiments, AscTec MAV Firefly and AscTec MAV Pelican for example, can be used in domestic operations.
- In addition to these two platforms, AscTec Falcon 8 platform is equipped with high resolution camera. This platform can be used in operations such as inspection and surveillance of big installations such as dams and electrical stations.
- AscTec Hummingbird Drone is another platform that can integrate our algorithms, exploiting its embedded inertial and vision systems.
- QuestUAV is another range of small unmanned airborne systems (sUAS) in the 3kg to 4kg weight range. These systems (Q-200 and Q-300) can operate payloads up to 1500g using high resolution vision systems.
- Zephyr2UAV is a platform characterised for its high speed of 30-40 MPH and high payload capabilities.
- For the ground vehicles, our platform used in the experimental validation in this thesis, Mobile Robots™ Pioneer 3-DX, is considered as the world's most popular intelligent mobile robot. This platform is featured for its mobility and high operational payload.
- The Arlo Robotic System is another ground mobile system. In addition to the embedded inertial and vision systems, this platform can also include a laser range-finder and IR distance sensors.
- Scitos G5 is a mobile robot platform, which is able to move the 60 kg platform at a speed of up to 1.4 m/s and handles payloads of up to 50 kg.

1.9 Summary and outline of the thesis

The thesis outline is illustrated in Figure 1.6. This thesis can be divided into two main parts: multiple view geometry problems and convex optimisation as a solution tool. The connection between these parts is the main objective of this thesis which is to robustly and optimally estimate the motion. In addition, the structure of the thesis is designed in a progressive-research presentation, where a developed solution in a later chapter is based on the previous solution presented in a previous chapter. This is shown using red arrows in Figure 1.6.

Our scenario consist of using monocular vision systems, where a vehicle equipped with single camera takes a sequence of images as it moves. We wish to estimate the

camera rotations and translations relying exclusively on visual inputs. In addition to image analysis, another challenge is added to our scenario consisting on the scale ambiguity estimation due to the projection effects.

In the next two chapters, we will present an overview of the theoretical basics which are relevant for the thesis. Particularly, **Chapter 2** introduces the basic ideas of multiple view geometry, focussing on camera geometry and projective camera model. We show the mathematical background of the multiple-view geometry problems. Attention is given to the fundamental and essential matrices estimation problems, especially their mathematical formulation problem. Three-view geometry is considered as well in this chapter, in which a scenery is sequentially captured by a moving camera from three different positions.

In **Chapter 3**, we provide details on convex optimisation, robust convex optimisation and quasi-convex optimisation, along with some notions about least-squares optimisation, linear optimisation, iterative optimisation, robust and recursive filtering. Furthermore, we introduce convex optimisation tools, such as the second-order cone programming (SOCP). Details on global optimisation methods (Branch and bound in particular) are given as well. This chapter can be considered as a preamble for the deployment of optimisation in motion estimation.

Using tools introduced in Chapter 2 and Chapter 3, in **Chapter 4** we present first contribution in this thesis. Using monocular systems makes the motion estimation challenging due to the absolute scale ambiguity caused by the projective effects. For this, we use robust tools to estimate both the trajectory of a moving object and the unknown absolute scale ratio between consecutive image pairs. Thus, the novel approach presented in this chapter consists of a two-stage solution used to efficiently solve the monocular visual odometry problem. The first stage of which pertains using convex optimisation with the L_∞ norm in motion estimation. For the second solution, we propose to use two new methods such as the recursive least squares (RLS) algorithm and more robust one such as the H_∞ filter to solve the scaling estimation problem. Both techniques follow on nicely from the first one and capable of dealing with system ambiguities in frame to frame absolute scale estimation. The proposed solution uses as input only images provided by a single camera mounted on the roof of a ground vehicle.

Typically, the camera pose is recovered from the available corresponding points between two or more views and the camera calibration parameters. These correspondences lead to estimate the fundamental matrix, which is the key for any motion estimation. Generally, the fundamental matrix is the critical link that represents the vision geometry between two views in the pinhole camera model. Thus, in **Chapter 5**, we present a new technique to robustly and accurately estimate this fundamental

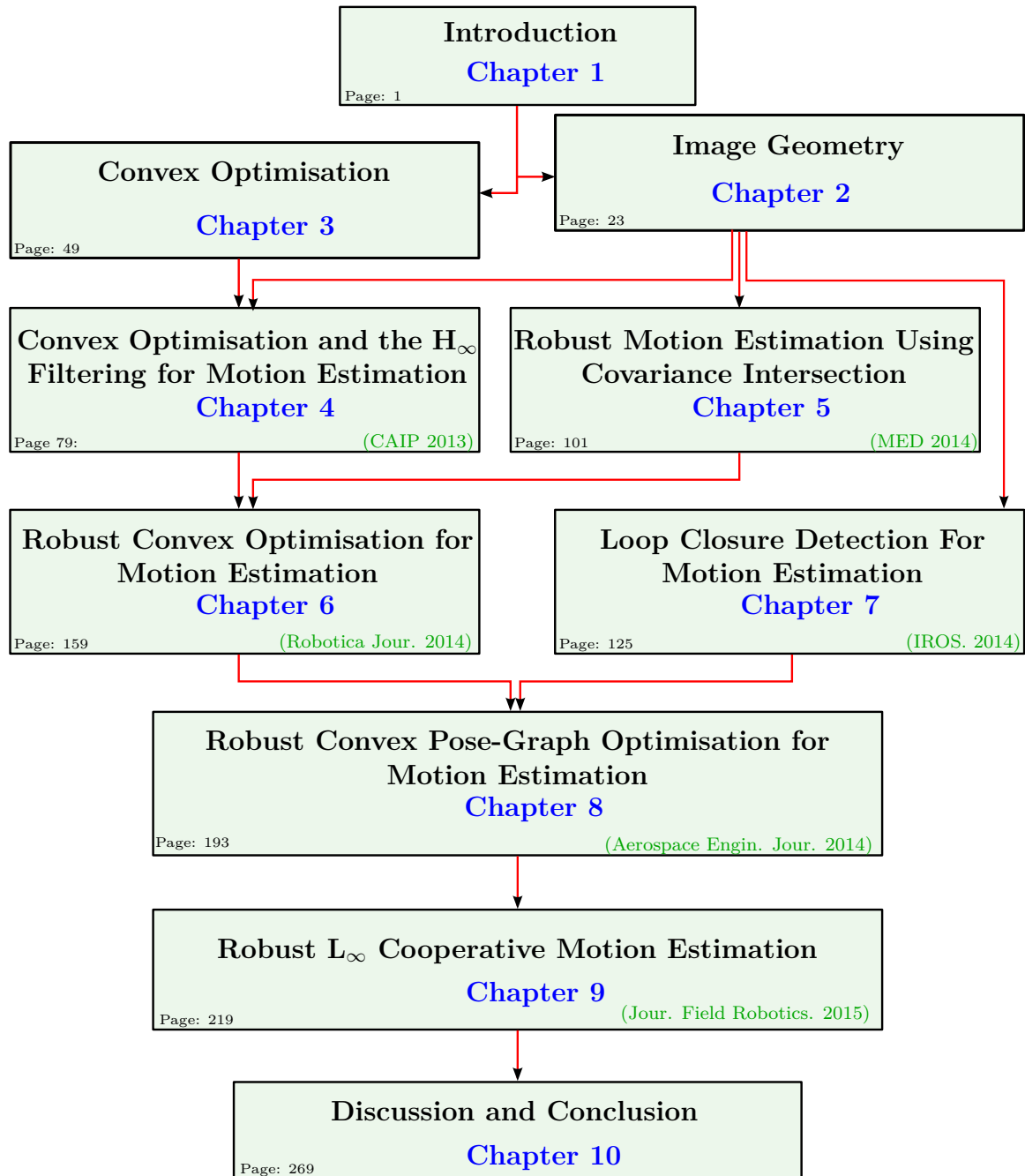


Fig. 1.6 The overall thesis outline.

matrix. In most vision applications, colour images are converted first to grey-level images, leading to a serious loss of information. In our solution, however, each RGB channel of colour images is processed separately. Then a fusion mechanism is employed to combine the information. After having estimated the uncertainties in features locations in each channel, covariance intersection is used as a fusion solution, resulting on considerable decrease in the measurement errors, which leads in turn to more accurate estimation of the fundamental matrix.

Exploiting the uncertainty estimation techniques from Chapter 5, and convex optimisation tools in Chapter 4, in **Chapter 6** we present a robust convex optimisation solution for monocular motion estimation systems. Critical implementations in the computer vision systems are based on robust features extraction, matching and tracking. Due to their extraction techniques, image feature locations accuracy is heavily dependent on the variation in intensity within their neighbourhoods, from which their uncertainties are estimated. In this chapter, we incorporate the uncertainty estimation in feature positions via the SIFT and the Harris derivative approaches along with their propagation.

In practice, for any navigation system, errors in position estimates are continuously growing due to the integration of noisy measurements over time and imperfect computational techniques. This unavoidable drift in motion estimation, due to inherent inaccuracy of the devices as well, needs to be corrected. Thus, providing additional correction tools would have a crucial impact on the final estimates of the navigation solution. Indeed, after long navigation into an unknown environment, detecting that the vehicle has returned to a previously visited location offers the opportunity to correct and to increase the accuracy and the consistency of the vehicle motion estimates. In computer vision, this is known as detecting loop-closures. In **Chapter 7**, we present a novel appearance-based technique for visual loop-closure detection. The widely used techniques based on the Bag-of-Words image representation have shown some limitations, especially with the perceptual aliasing problem. Our solution, however, uses both local invariant and colour features. Moreover, the proposed solution combines Gaussian mixture modelling (GMM) with the KD-tree data structure. In doing so, this solution takes advantage of the robustness of the KD-tree data structure and the efficiency of the Gaussian mixture modelling representation. Experimental validation using datasets from different environments has been conducted. We show that due to their efficiency and complementarity, a combination of KD-trees with GMM could be an alternative for real-time loop-closure detection for mobile robots navigation.

In **Chapter 8**, we present a new robust convex pose-graph optimisation solution for UAVs monocular motion estimation systems. Pose-graph formulation is an

intuitive way to address the pose estimation problem. The nodes of the graph represent the vehicle's poses and the edges encodes measurements that constrain the connected poses. Solving the pose-graph problem involves finding the optimal configuration of the nodes that best satisfies the constraints. Most methods in the literature utilise standard approaches, like the Gauss-Newton or Levenberg-Marquardt algorithms. However, with these methods there is no guarantee of convergence to the global minimum. Furthermore, they could lead to an infeasible solution. As such, these methods are also very dependent on good initialisations. Alternatively, the proposed solution recovers the optimal position configuration by using convex optimisation through the adoption of more robust norms such as the L_∞ norm. Furthermore, uncertainty estimations, based on the SIFT derivative approach and their propagation through multi-view geometry algorithms are included in this solution. Once a loop-closure is detected using technique presented in Chapter 7, convex pose-graph optimisation solution performs the correction of any drift occurred during the monocular motion estimation.

After developing robust solutions for visual navigation systems, in which an autonomous vehicle can estimate its own localisation independently, a need for a cooperative solutions has been risen. Thus, **Chapter 9** deals with cooperative navigation using convex optimisation. In this chapter, a system for cooperative monocular visual motion estimation with multiple aerial vehicles is proposed. The distributed system between vehicles allows efficient processing in both computational time and estimates accuracy. The global cooperative motion estimation employs state-of-the-art approaches for optimisation, individual motion estimation and registration. Three-view geometry algorithms in a convex optimisation framework are deployed on board the monocular vision system for each vehicle. In addition, vehicle-to-vehicle relative pose estimation is performed with a novel robust registration solution in a global optimisation framework. In parallel, and as a complementary solution for the relative pose, a robust non-linear H_∞ filter is designed as well to fuse measurements from the UAVs' on-board inertial sensors with the visual estimates.

This thesis is concluded in **Chapter 10** where we summarise the obtained results and achievements, and finally point out future directions.

In addition, this thesis includes three appendices. Appendix A presents a review on some multiple-view geometry problems that can be solved using convex optimisation. Particular attention is given to problems of triangulation estimation, camera resectioning and homography. These tasks were implemented in this thesis using the second-order cone programming (SOCP) on benchmark datasets, familiar to the computer vision community.

As a complementary extension to Chapter 7, Appendix B details the KD-tree data structure, presenting an illustrative example on the tree construction and nearest neighbour search through it.

Appendix C focuses on the fundamental matrix estimation, giving the main algorithms used throughout this thesis in the multi-view geometry framework. Image interest point extraction techniques are also presented in this appendix, such as the Harris corner detector, the scale invariant feature transform (SIFT) and colour features.

1.10 Published and submitted manuscripts

Journals

- Mohammed Boulekchour, Nabil Aouf and Mark Richardson. *Robust L_∞ convex optimisation for monocular visual odometry trajectory estimation*. Robotica Journal, Cambridge Journals, July 2014. (published);
- Mohammed Boulekchour, Nabil Aouf and Mark Richardson. *Robust L_∞ convex pose-graph optimisation for monocular localisation solution for UAVs*. Journal of Aerospace Engineering. December 2014 (published).
- Mohammed Boulekchour, Nabil Aouf and Mark Richardson. *Robust L_∞ Co-operative Motion Estimation for Multiple UAVs*. Journal of Field Robotics. (submitted).
- Saad Imran, Nabil Aouf and Mohammed Boulekchour. *L_∞ -Based Estimation Technique for Time-Varying Homographies with System Uncertainty*. Journal of Intelligent and Robotic System. (submitted).

Conferences

- Mohammed Boulekchour and Nabil Aouf. *Efficient Real-Time Loop Closure Detection Using GMM and Tree Structure*. IEEE/RSJ International Conference on Intelligent Robots and Systems , IROS 2014, Chicago Illinois Sept. 14-18, 2014.
- Mohammed Boulekchour and Nabil Aouf. *Robust Motion Estimation Using Covariance Intersection*. 22nd Mediterranean Conference on control and Automation, MED 2014, June 2014, Palermo, Italy.

- Mohammed Boulekchour and Nabil Aouf. *L_∞ Norm Based Solution for Visual Odometry*. Computer Analysis of Images and Patterns - 15th International Conference, CAIP 2013, York, UK;
- Mohammed Boulekchour and Nabil Aouf. *Improved Monocular Navigation - Closed Loop Concept*. 12th Electro-Optics and Infra-red Conference, Shrivenham, Defence Academy of the UK, June 2014.

Chapter 2

Image Geometry

In this chapter, we introduce the basic notions of image formation and multi-view geometry. The aim of this chapter is to familiarise the reader with concepts of computer vision that are the basis for the contributions made in this thesis. First we give a brief review of image representation as digital data, where intensity values are parametrised in a 2D array. Then, we present the basis of the projective camera model, giving some details about the pinhole camera projective mapping. Details concerning estimation of the relative rotations and translations between two cameras in different positions are given. Three-view geometry is considered as well in this chapter, in which a scenery is sequentially captured by a moving cameras from three different positions.

In this thesis, we concentrate on motion estimation using multiple view geometry. In this scenario a ground or an aerial mobile platform is moving through an unknown environment and recording digital images. Using image geometry principles, it is possible to align images to each other and then estimate the camera motion. If one performs these steps, then one is doing visual motion estimation - known in computer vision as visual odometry.

2.1 Overview

Digital image processing started to emerge during the mid 1960s, when the cost of the corresponding devices was rather high. The first digital consumer camera was sold in 1991, which was a 1/3-inch, 376×240 pixel model CCD (charged coupled device) with 256 grey levels. This camera was designed by Dycam, and could be directly connected to a personal computer to transfer images. Recently, digital cameras have seen substantial development and have become widely available. In addition, current generation of digital cameras are inexpensive and easy to use. Another important

feature of digital cameras is their compact design, which ensures low weight and low power consumption. Due to these new technologies, digital cameras can also be deployed in different harsh environments.

Throughout this thesis, we will validate our results using monocular vision systems, which consist of a single camera with low power supply. These cameras are easily embedded on ground and small aerial vehicles. As a summary, digital cameras are small, light, low-power and inexpensive. Therefore, they are ideal for our work on visual navigation systems.

On the other hand, using monocular vision systems poses many challenging tasks. Some drawbacks of such vision systems comprise the need of depth information, poor resolution, image occlusion and the necessity for wide data interpretation. In contrast to range/bearing sensors that are able to estimate the travelled distance via estimating the time-of-flight, visual monocular systems are not able to estimate this distance directly. Instead, they extract and track some scene features from different camera positions and use the triangulation technique to estimate the non-scaled distance to the scene and then recover the absolute depth scales using optimisation tools. One more challenging task is how to deal with the large amount of information that a camera is acquiring as it moves. Extracting relevant data under real-time constraints would be a difficult task as well.

2.2 Notation

Throughout this thesis, the transpose of a vector x is represented by x^\top . A vector $(x_1, x_2, x_3)^\top$ denotes a column vector and (x_1, x_2, x_3) denotes a row vector.

Suppose we have a rigid body rotating about a stationary point O in a three-dimensional Euclidean space \mathbb{E}^3 . The space of orthogonal matrices in $\mathbb{R}^{3 \times 3}$ is denoted throughout this thesis by:

$$SO(3) = \{R \in \mathbb{R}^{3 \times 3} \mid R^\top R = I \text{ and } \det(R) = +1\}$$

where I is the identity matrix, $R \in \mathbb{R}^{3 \times 3}$ is the rotation matrix and $\det(R)$ is the determinant of R .

The set of rigid body motion, is a (Lie) group and denoted as $SE(3)$. This refers also to the special Euclidean transformation defined by:

$$SE(3) = \{g = (R, T) \mid R \in SO(3) \text{ and } T \in \mathbb{R}^3\} = SO(3) \times \mathbb{R}^3$$

where T is the translation vector.

Suppose, \mathbf{x} is a 3-element vector, where $\mathbf{x} = (x_1, x_2, x_3)^\top$. Then, we denote by $[\mathbf{x}]_\times$ a 3×3 skew-symmetric matrix of the form:

$$[\mathbf{x}]_\times = \begin{bmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{bmatrix}$$

2.3 Image formation

Digital images are constructed after the incoming light hits the image plane passing the camera lens. In most cases, this plane is called the CCD (Charged Coupled Device). The CCD consists of an array of $n \times m$ rectangular grid of photo-sensors, each one is sensitive to light intensity. Then, the light energy is converted into voltage by any element of the CCD. After that, an electronic device called a frame grabber receives an electric signal from the CCD and digitises it into 2D rectangular array of $N \times M$ integer values and stores it in a memory buffer [200]. Thus, a digital image is characterised, at this point, by a matrix Ω of size $N \times M$. An entry $\Omega(i, j)$ of this matrix is an integer quantity ranging from 0 to 255, representing the image brightness at the position (i, j) and called pixels (an acronym for picture elements) [82, 200]. The quantities N and M represent then the image size in pixels along each direction.

In a mathematical way, an image is a two-dimensional brightness array. More precisely, an image is a map I defined on a surface Ω of a two-dimensional surface, whose entries are positive real [82, 122, 200]:

$$I : \Omega \subset \mathbb{R}^2 \rightarrow \mathbb{R}_+ \quad (x, y) \mapsto I(x, y) \quad (2.1)$$

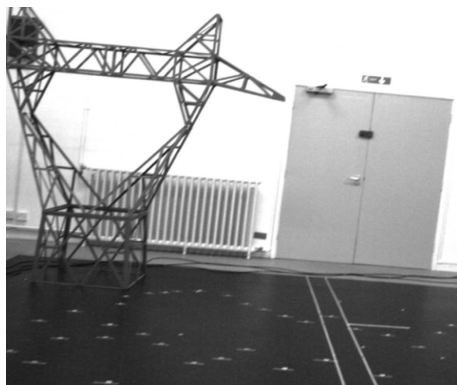
where (x, y) are image coordinates in 2D space.

Figure 2.1 shows the image representation for a grey scale image. Plot (b) in this figure shows the intensity at the z -axis, while x and y -axis represent the N rows and M columns respectively. Colour images include three monochromatic component images RGB (red, green and blue), therefore giving three overlapping matrices.

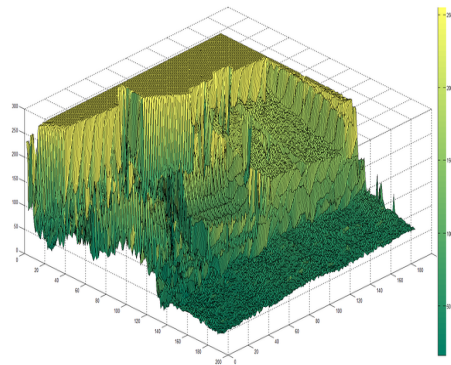
2.4 Projective geometry

2.4.1 Point representation

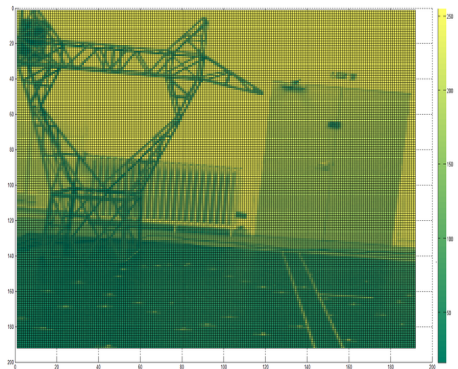
A point in 2D Euclidean space \mathbb{R}^2 is represented with an ordered pair of real numbers (x, y) . This representation is called an inhomogeneous vector $\mathbf{x} = (x, y)^\top$. However,



(a)



(b)



(c)

232	209	87	220	241	238	142	96	88	151	112	238	252	253	254	252	255	255	254	254	254
234	200	93	219	242	245	134	97	109	184	123	163	252	253	254	252	255	255	254	254	254
236	191	97	238	243	231	124	157	110	184	175	115	219	253	253	254	255	255	255	254	255
227	178	95	234	247	167	200	154	95	157	197	152	152	244	252	254	255	255	255	254	255
229	159	112	185	194	121	218	148	97	107	189	202	140	198	255	254	253	255	255	255	254
229	142	76	79	84	89	94	97	107	83	124	170	140	126	233	254	251	253	254	254	255
224	129	59	46	74	63	62	86	86	74	90	78	69	87	163	246	247	254	254	254	255
209	113	56	53	54	39	40	89	68	54	82	86	69	72	91	185	239	252	253	253	254
212	104	52	53	41	41	64	83	72	80	63	78	86	76	80	103	215	249	251	255	255
196	98	61	59	37	34	53	75	72	55	64	62	86	91	64	74	152	243	251	253	253
181	89	67	42	39	53	78	75	52	37	37	36	58	90	128	80	83	199	253	255	251
175	86	61	40	60	78	60	83	46	38	35	40	43	76	92	102	142	123	235	247	251
168	80	58	68	69	43	51	75	42	38	44	35	59	72	98	180	206	117	181	254	250
165	81	77	62	38	38	53	70	40	39	42	60	71	75	104	121	211	186	114	225	252
119	80	58	47	42	42	53	67	43	45	69	64	49	70	174	115	157	234	145	148	244
68	78	75	72	75	74	76	78	66	79	68	48	54	74	198	166	109	205	214	111	202
95	77	61	41	36	45	64	73	74	69	73	76	75	82	102	107	74	115	181	129	126
103	75	72	63	38	24	27	30	37	42	47	42	51	77	160	156	89	82	122	93	93
86	64	49	72	57	34	21	17	30	33	32	33	39	49	83	92	78	96	175	139	77
83	60	26	46	74	61	30	31	36	39	36	36	52	70	79	106	99	84	62	58	44
66	66	40	39	51	69	61	80	59	50	74	59	81	82	181	216	109	118	110	213	172

(d)

Fig. 2.1 (a) Original image taken in our laboratory. (b) The same image represented as a two dimensional surface. The intensity is represented at the z-axis ranging from 0-255. (c) Shows the top view of the surface in (b). (d) Part of the same image represented as a two-dimensional array of integers.

it is possible to add a third coordinate to represent the same point as $(x, y, 1)$. For a non-zero value k , this point can be represented with homogeneous coordinates (kx, ky, k) . Therefore, it is possible to extend the 2D Euclidean space \mathbb{R}^2 to a projective space \mathbb{P}^2 by representing points as homogeneous vectors. In this projective space, points at infinity can be represented with homogeneous coordinates in which the third coordinate is zero [82].

Throughout this thesis we will be using the homogeneous vector notation $\hat{x} = (x_1, x_2, x_3)^\top$ to represent a point in projective space \mathbb{P}^2 . The relationship between the homogeneous and inhomogeneous coordinates of the same point in 2D is:

$$x = \frac{x_1}{x_3}, \quad y = \frac{x_2}{x_3} \quad (2.2)$$

Note that any homogeneous vector is defined to a scale; this means that $(x_1, x_2, x_3)^\top$ and $k(x_1, x_2, x_3)^\top$, where $k \neq 0$, refer to the same point.

Similarly, in 3D Euclidean space \mathbb{R}^3 , a point is represented with an inhomogeneous vector $X = (X, Y, Z)^\top$ and in 3D projective space \mathbb{P}^3 with a homogeneous vector $\hat{X} = (X_1, X_2, X_3, X_4)^\top$. Similarly, the relationship between the homogeneous and inhomogeneous coordinates of the same point in 3D is:

$$X = \frac{X_1}{X_4}, \quad Y = \frac{X_2}{X_4}, \quad Z = \frac{X_3}{X_4} \quad (2.3)$$

2.5 Geometry of cameras

As discussed in the preceding section, the world can be characterised using a projective space \mathbb{P}^3 , where the structures and the shapes of objects are represented using points in the form of a homogeneous 4-element vectors, such as \hat{X} . Relying on the projective geometry, the main goal of this section is to define the link between the positions of 3D scene points and their corresponding image points. The motion of these points is represented by a 3×3 rotation matrix $R \in SO(3)$ and a 3-element vector translation $T \in \mathbb{R}^3$ [82, 122, 200].

2.5.1 The pinhole camera model

A pinhole camera is a mapping π from Euclidean 3D space \mathbb{E}^3 to Euclidean 2D space \mathbb{E}^2 :

$$\pi : \mathbb{R}^3 \rightarrow \Omega, \quad X \mapsto \mathbf{x} = \pi(X), \quad (2.4)$$

where $\Omega \subset \mathbb{R}^2$ and $\mathbf{X} = (X, Y, Z)^\top \in \mathbb{R}^3$ are the coordinates of a point in space expressed in the camera coordinate frame, and $\mathbf{x} = (x, y)^\top$ is the corresponding 2D coordinates.

The optical system of this model gathers of a lens set used to guide the light. This means, lenses are used to control the direction of the light propagation by means of refraction, reflection and diffraction. A simpler model of this optical system includes thin lenses. This model is defined by an axis, called *the optical axis*, and the focal plane, which is perpendicular to this axis (Figure (2.2a)). Thin lenses are characterised by their *focal length*, which is the distance from *the optical centre* (O in Figure (2.2a)) and to the intersection point, called *the focus*, of rays entering the aperture in parallel to the optical axis (Figure 2.2a). One other important property for thin lenses is that oncoming rays through the optical centre do not deflect. This is illustrated in Figure 2.2b, where there is a point $p \in \mathbb{R}^3$ at a distance Z from the optical centre on the optical axis. The first ray from the point p through the optical centre remains undeflected, while the second ray from p and parallel to the optical axis will be deflected and cross the optical axis at the focus. The first and the second ray intersect at a point \mathbf{x} . This point is the image of p . If z is its distance along the optical axis from the optical centre, then:

$$\frac{1}{Z} + \frac{1}{z} = \frac{1}{f}. \quad (2.5)$$

Equation 2.5 means that, a thin lens with a focal length f is focusing the light from a plane at a distance Z from the lens at a distance z behind the lens [190]. This equation is called the fundamental equation of thin lenses. If all the rays go through a single point, the optical centre, and the lens aperture is compacted to zero, then all rays remain undeflected. In this scenario we speak about pinhole model which is adopted throughout this thesis. This model is depicted in more details in Figure (2.3).

Briefly, the camera is a mapping between the 3D real world and a 2D image (the image plane) [82]. Let us consider a 3D point $\mathbf{X} = (X, Y, Z)^\top$ expressed relative to a reference frame centred at the optical centre O , where the Z -axis is the optical axis. This reference is called the camera reference frame. The image of this point is \mathbf{x} , which is where the line from \mathbf{X} to the optical centre intersects the image plane. Four principal and critical reference frames ought to be detailed (Figure 2.3):

- **The pixel frame:** a pixel represents information from the real world on the image. The two coordinates on this frame are measured in pixels [73], representing number of pixels from upper-left corner of the image plane in

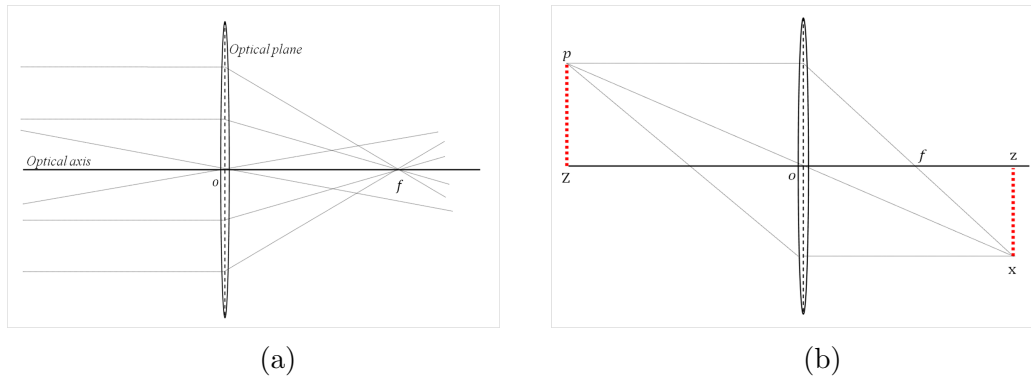


Fig. 2.2 (a) The rays parallel to the optical axis meet at the focus. (b) x is the image of the point p , which is the intersection of the rays parallel to the optical axis and the ray through the optical centre [122].

the horizontal and vertical directions. Usually, u is used as the horizontal coordinate and v as the vertical coordinate.

- **Image plane frame or focal plane:** This frame is identified by its origin which is what called the principal point. This origin, located at the centre of the image plane, corresponds to the projection of the optical centre O on the image plane.
- **Camera frame:** in this frame, the origin is the optical centre, and the distance between this origin and the image plane is the focal length f . The 3D point $X = (X, Y, Z)^\top$ is expressed in this frame.
- **The world reference frame:** defined by the user as an external global positioning reference.

As shown in Figure 2.3b, x is the image of the 3D scene point X . Consider the camera reference frame, if $X = [X, Y, Z]^\top$ and $x' = [x, y, z]^\top$ are expressed in this reference frame, then we can see from Figure 2.3b that the coordinates of x and X are related by:

$$x = -f \frac{X}{Z}, \quad y = -f \frac{Y}{Z}, \quad z = -f \quad (2.6)$$

Observe that any 3D scene point that lies on the line through X will be projected at the same point $x' = [x, y, z]^\top$. As previously observed, this model is called the pinhole camera. In (2.6), coordinates are expressed with a negative sign. Similarly to the retina of the eye, this sign makes objects appear upside down on the image plane (dotted green arrow in Figure 2.3b). The way to remove this effect is to place the image plane in front of the optical centre at $Z = f$ instead of $Z = -f$ as shown in Figure (2.3b). Then, it is sufficient to apply a transformation: $(x, y, z) \mapsto (-x, -y, -z)$.

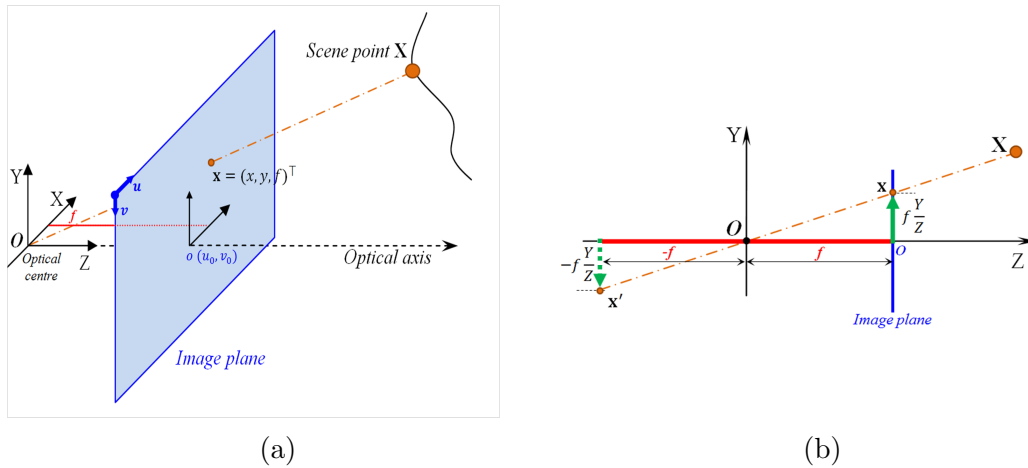


Fig. 2.3 (a) In the image plane, o is the principal point and x is the image of the point X , which is the intersection of the ray going through the optical centre O . (b) The image plane is placed at a distance f in front of the camera centre [82, 122, 200].

This transforms the point x' to x :

$$x = f \frac{X}{Z}, \quad y = f \frac{Y}{Z}, \quad (2.7)$$

$$\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix} = \frac{f}{Z} \begin{bmatrix} X \\ Y \end{bmatrix}. \quad (2.8)$$

In other words, we are saying that point $X = [X, Y, Z]^T$ is mapped to the point $\mathbf{x} = [x, y]^T = [f \frac{X}{Z}, f \frac{Y}{Z}]^T$ on the image plane. Note that x and y are expressed in 2D coordinates on the image plane centred at the principal point.

2.6 Camera model

Now, let us consider the two reference frames $\{O, X^w, Y^w, Z^w\}$ and $\{o, x^c, y^c, z^c\}$ in Figure 2.4. These frames are the world reference frame and the camera reference frame respectively. In order to remove any ambiguity, in this section we will be using the superscripts and the subscripts w and c to refer to the world and the camera reference frames respectively. A 3D scene point X is expressed in the camera reference frame with the coordinates: $X_c = [X_c, Y_c, Z_c]^T$. Now, in order to express this point in the world reference frame X_w , we need to define a transformation g_{cw} from the camera to the world reference frame, where:

$$X_c = g_{cw}(X_w) \quad (2.9)$$

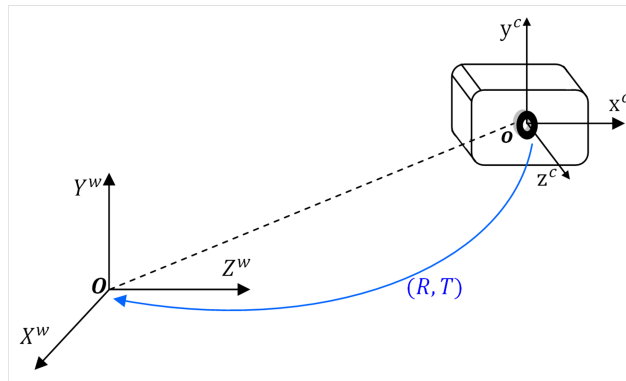


Fig. 2.4 Transformation between camera and world reference frame

This transformation is defined by a rigid motion, $g_{cw} = (R_{cw}, T_{cw})$, where $R_{cw} \in SO(3)$ is the rotation matrix and $T_{cw} \in \mathbb{R}^3$ is the translation vector. Thus, this gives:

$$X_c = g_{cw}(X_w) = R_{cw}X_w + T_{cw} \quad (2.10)$$

On the other hand, $X_w = g_{wc}(X_c)$, where $g_{wc} = g_{cw}^{-1}$ is the inverse mapping from the world reference frame to the camera reference frame. Equation (2.10) can be reformulated in a compact form as:

$$\begin{bmatrix} X_c \\ 1 \end{bmatrix} = \begin{bmatrix} R_{cw} & T_{cw} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_w \\ 1 \end{bmatrix}. \quad (2.11)$$

Now, using homogeneous coordinates, equation (2.8) can be rewritten as:

$$\begin{aligned} Z \begin{bmatrix} x \\ 1 \end{bmatrix} &= \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_c \\ 1 \end{bmatrix}, \\ Z \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} &= \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix}. \end{aligned} \quad (2.12)$$

Since the Z coordinate represents the depth, which is not known, we can denote it with a positive scalar $\lambda \in \mathbb{R}_+$. Thus:

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} \quad (2.13)$$

Notice as well that:

$$\begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = K_f P \quad (2.14)$$

where:

$$K_f = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{and} \quad P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Then, using (2.11), (2.13) and (2.14), we can write the geometric model:

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R_{cw} & T_{cw} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \quad (2.15)$$

Or in matrices:

$$\lambda \hat{x} = K_f P \hat{g} \hat{X}_w \quad (2.16)$$

Unfortunately, the only known coordinates in practice are the pixels location (u, v) , in which the image upper-left corner is the origin (Figure 2.3a). The challenge now is to identify the transformation between the pixel coordinates and the image plane coordinates. Now, suppose that (x, y) are expressed in metric units (in millimetres for instance), and (u', v') are in pixels, then this transformation is given as follow [82, 122]:

$$\begin{bmatrix} u' \\ v' \end{bmatrix} = \begin{bmatrix} s_u & 0 \\ 0 & s_v \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \quad (2.17)$$

where s_u and s_v represent the number of pixels per unit distance (e.g. *pixel/mm*) along the image coordinates in the u and v directions. In other words, s_u and s_v represent the relation between the pixels and the metric units. However, the origin has to be translated to the upper-left corner, hence (see Figure 2.3a):

$$u = u' + u_0 \quad ; \quad v = v' + v_0 \quad (2.18)$$

where (u_0, v_0) are the the image centre coordinates (expressed in pixels). Hence (2.17) becomes:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} s_u & 0 & u_0 \\ 0 & s_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (2.19)$$

A more general form has to be considered in the case where the pixels are not rectangular:

$$\begin{aligned} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} &= \begin{bmatrix} s_u & s_\theta & u_0 \\ 0 & s_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \\ \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} &= K_s \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \text{ where } K_s = \begin{bmatrix} S_u & S_\theta & u_0 \\ 0 & S_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \end{aligned} \quad (2.20)$$

The quantity s_θ is the skew factor.

Combining (2.13) with (2.20) gives the transformation model between a 3D point relative to the camera frame and its image expressed in pixels using homogeneous coordinates [82, 122]:

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} s_u & s_\theta & u_0 \\ 0 & s_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} \quad (2.21)$$

Now consider a new matrix K , where:

$$\begin{aligned}
 K &= \begin{bmatrix} s_u & s_\theta & u_0 \\ 0 & s_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \in \mathbb{R}^{3 \times 3} \\
 K &= \begin{bmatrix} fs_u & fs_\theta & u_0 \\ 0 & fs_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \\
 &= \begin{bmatrix} \alpha_u & \gamma & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \in \mathbb{R}^{3 \times 3} \tag{2.22}
 \end{aligned}$$

This new triangular 3×3 matrix K gathers the parameters of a camera, and called the calibration matrix. Specifically:

- u_0 is the u -position of the point where the optical axis intersects the plane;
- v_0 is the v -position of the point where the optical axis intersects the plane;
- $\alpha_u = fs_u$ is the size in pixels of the focal length in the horizontal direction;
- $\alpha_v = fs_v$ is the size in pixels of the focal length in the vertical direction;
- $\gamma = fs_\theta$ is the skew of the pixels, often close to zero.

In the case where the effect of the skew factor is considerable, this parameter must be integrated in the camera parameter matrix.

Now, back to equation (2.15), points in space may be expressed in the two coordinate frames: the world and the camera coordinate frames (Figure 2.4). We want to define the geometric relationship between image point coordinates $[u, v]^T$ expressed in pixels and their corresponding 3D scene point coordinates $[X_w, Y_w, Z_w]^T$ expressed in the world reference frame $\{O, X^w, Y^w, Z^w\}$. Using (2.15), (2.21) and (2.22), we can write the general geometry as:

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha_u & \gamma & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R_{cw} & T_{cw} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{2.23}$$

Or in matrix form:

$$\lambda \hat{x}_{im} = KP\hat{X}_c = KP\hat{g}\hat{X}_w \tag{2.24}$$

A more compact form can represent this projection using homogeneous coordinate as [82, 209]:

$$\hat{x} = KR_{cw}[I | -\tilde{C}] \hat{X}_w \quad (2.25)$$

$$\hat{x} = P \hat{X}_w \quad (2.26)$$

where:

$$P = KR_{cw}[I | -\tilde{C}]$$

Specifically:

- \tilde{C} : collects the coordinates of the camera centre in the world reference frame;
- \hat{x} : image point represented by a homogeneous 3-vector;
- \hat{X}_w : 3D scene point in space expressed in the world reference frame, on a homogeneous 4-element vector in 3D projective space \mathbb{P}^3 ;
- K : camera calibration matrix (intrinsic parameters);
- R_{cw} : 3×3 rotation matrix in $SO(3)$;
- P : 3×4 homogeneous camera projection matrix.

The camera projection matrix P has 11 degrees of freedom: 5 for K , 3 for R_{cw} and 3 for \tilde{C} . Equation (2.26) is known as the projective equation.

2.7 Image interest points (image features)

In visual autonomous navigation systems, image points (or usually called *features*) are of great interest in estimating the camera translations and the rotations. In fact, for any computer vision problem, recorded images are the main source of information. Indeed, extracting these image points is the first step for any algorithm. Matching and tracking these image points is not less important than the extraction task, especially for problems of registration and recognition [106]. Having images in hand as inputs, the first thing to look for is points that can be recognised in other images. Therefore, interest points like corners and edges seem to be valid options. These image points are commonly called image features in computer vision. Other names such as keypoints, interest points, corners, control points and edges are used as well. Other tasks in computer vision apart from motion estimation such as object classification and recognition, use image features in their algorithms [27, 119, 218] as well.

Good quality features must be easy to track, be extracted with precision and invariant to illumination and geometrical changes such as rotation, translation and

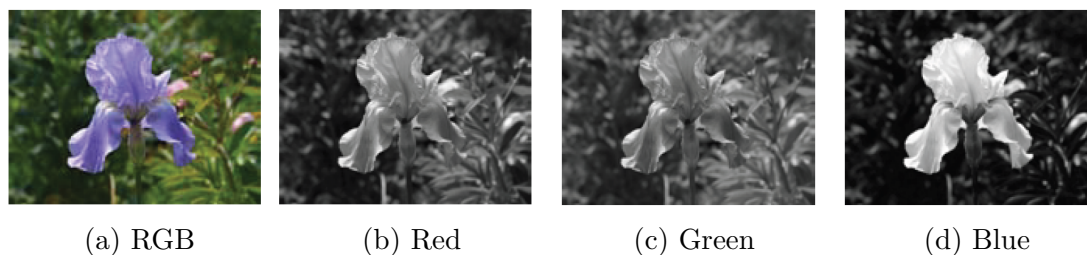


Fig. 2.5 Example of a colour image and its RGB channels [190].

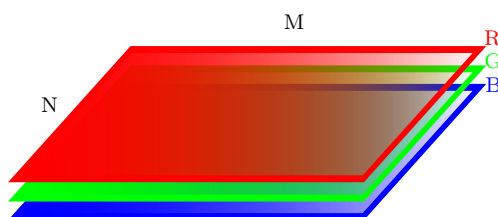


Fig. 2.6 Image matrix representation.

scale. Obviously, best scene features must have similar appearance from different camera positions. Two main tasks can be distinguished when dealing with image features: detection and description. Detection involves processing the image information to obtain regions, corners or even edge segments. The description consists of creating a descriptive vector based on the visual appearance in the image. Invariant features to changes in scale and rotation are of great interest in visual autonomous navigation systems since they ensure accurate matching and data association.

2.8 RGB channels in colour images

In computer vision, it is known that the incoming light rays in different wavelengths are integrated into the discrete red, green, and blue (RGB) colour values to form a digital colour image [190]. This digital image is represented by a numerical matrix, E , with N rows and M columns. The matrix element $E(i, j)$ represents the value at the pixel (i, j) , which consist of three RGB numeric components. Colour images need three components (red, green and blue) to represent a pixel value, therefore the numerical matrix E is of size $M \times N \times 3$ (Figure 2.6). The pixel colour is given by three values, where each one is ranging from 0 to 255. This means that a pixel takes one of 16 million colours (256 shades of red, 256 of green, and 256 of blue, so $256 \times 256 \times 256 = 16.8$ million possible combinations). Black is an RGB value of $(0, 0, 0)$ and white is of $(255, 255, 255)$.

Red, green and blue colours are considered as additive colours. That is the colours are added together to form the final colour image. This can be explained by the fact that these colours are the dominant colours that our eyes can see. Combining them together we can effectively generate almost every colour our eyes can see.

2.9 Two-view geometry - epipolar geometry

Views are defined as images taken by a one or more cameras at different locations [82, 122, 200]. If the same camera is used, each view has the same image size and the same calibration parameters. In two-view geometry, the two images are captured by a single camera that is shifted from one place to another, or they can be captured by two cameras at two different positions in space [89]. Multiple views imply that a single camera is used to capture multiple images, which form a single image sequence, from n different positions.

It is possible to estimate correspondences between image points of the same 3D scene points at each camera position assuming that the scene is relatively static.

Now, let us consider an arbitrary 3D scene point X in the scene (Figure 2.7), where $\hat{X} = [X \ 1]^\top$. From (2.26), this scene point X will be projected to image point $x_i = P_i \hat{X}$ in the left image l_i and to image point $x_j = P_j \hat{X}$ in the right image l_j . Hence, x_i and x_j are corresponding (matching) image points of the same 3D point X . Also, it can be seen from Figure 2.7 that scene point X , the matching image points x_i and x_j , and the camera centres C_i and C_j are in the same plane, denoted as π . In the case where x_j is known, then our task is to recover its corresponding x_i . The distance between the two camera centres is called the baseline. Knowing that x_i is on the plane π , therefore the search area must be restricted to the line, where the left image plane π_i intersects with the plane π . This important line is called the Epipolar line.

Now the question is: what is the geometrical relationship between the two points x_i and x_j ? This is discussed in the following section.

2.9.1 The essential matrix

Let us consider the scenario where the point X is seen by two cameras P_i and P_j from different locations. Therefore, points x_i and $x_j \in \mathbb{R}^3$ represent the coordinates of the projection of the same 3D scene point X . As stated before, each camera has its own reference frame (Section 2.5.1). Let $X_i \in \mathbb{R}^3$ and $X_j \in \mathbb{R}^3$ be the 3D coordinates of the same point X in each camera frame respectively. Then X_i and X_j are related

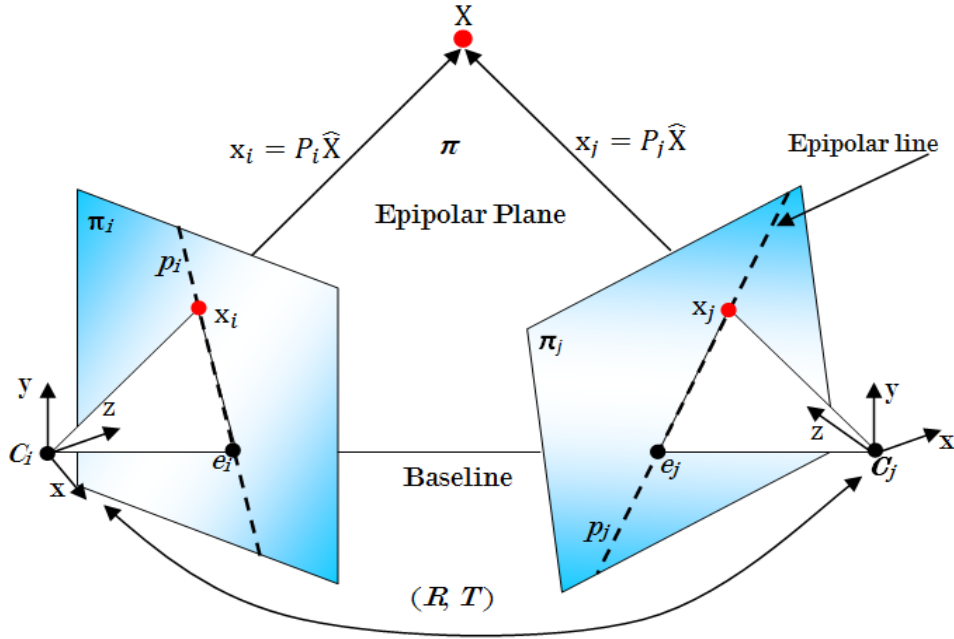


Fig. 2.7 Two-view geometry

by the rigid body motion:

$$X_j = RX_i + T \quad (2.27)$$

where R is the rotation matrix $\in SO(3)$ and $T \in \mathbb{R}^3$ is the translation. Here we assume that the second camera P_j is the reference frame. The transformation $g = (R, T) \in SE(3)$ gathers the location and the orientation of the camera P_i . This equation may be written using the image points x_i and x_j as:

$$\lambda_j x_j = R\lambda_i x_i + T \quad (2.28)$$

where λ_i and λ_j are again the depths. In order to eliminate these depths, one may multiply both sides of (2.28) by $[T]_{\times}$, and since $[T]_{\times}T = 0$, then:

$$\lambda_j [T]_{\times} x_j = [T]_{\times} R\lambda_i x_i \quad (2.29)$$

Since $[T]_{\times} x_j = T \times x_j$ is perpendicular to the vector x_j , the inner product is $x_j^{\top} [T]_{\times} x_j = 0$, then $x_j^{\top} [T]_{\times} R\lambda_i x_i = 0$. We know that the depth λ_i is positive ($\lambda_i > 0$), then:

$$x_j^{\top} [T]_{\times} R x_i = 0 \quad (2.30)$$

This equation is of great importance in computer vision and is called *the essential constraint* or *epipolar constraint*.

The geometric interpretation of this equation is shown in Figure 2.7. The camera centres C_i and C_j , the 3D scene point X and its image points x_i and x_j are coplanar (i.e. lie in the same plane π). Therefore (2.30) is simply the co-planarity constraint expressed in the reference frame of camera P_j . Let the matrix $E = [T]_{\times} R \in \mathbb{R}^{3 \times 3}$, then (2.30) could be rewritten as:

$$x_j^{\top} E x_i = 0 \quad (2.31)$$

Matrix E is called the **Essential Matrix**. This matrix constraints the relative translation T and rotation R between the two cameras P_i and P_j [2, 72, 82, 89, 122, 200]. Note that the E matrix deals with points expressed in the camera coordinate frame. In this thesis, we define a space for this kind of matrices in $\mathbb{R}^{3 \times 3}$ called the Essential space denoted by \mathcal{E} :

$$\mathcal{E} = \{ [T]_{\times} R \mid R \in SO(3), T \in \mathbb{R}^3 \} \subset \mathbb{R}^{3 \times 3} \quad (2.32)$$

$[T]_{\times}$ is the 3×3 skew-symmetric matrix of T .¹ Now let us define this theorem [82, 89, 200]:

Theorem 1. A non-zero matrix $E \in \mathbb{R}^{3 \times 3}$ is an essential matrix if and only if E has a singular value decomposition (SVD): $E = U \Sigma V^{\top}$, where $\Sigma = \text{diag}\{\sigma, \sigma, 0\}$ for some $\sigma \in \mathbb{R}_+$ and $U, V \in SO(3)$.

Proof to this theorem is given in [123].

2.9.2 Pose extraction from the essential matrix

Obviously, when we have the relative rotation R and the relative translation T between the two cameras, we can immediately estimate the essential matrix by just putting $E = [T]_{\times} R$. Now the question is how can we estimate the pose T and R by knowing the essential matrix E ? This problem is known as pose recovery from the essential matrix. Prior to that, we have to estimate first this matrix. This is performed using the corresponding image points. Then, recovering the translation and rotation from the estimated E . However, before performing the second step, we have to make sure that the recovered E is really an essential matrix, i.e. $E \in \mathcal{E}$.

First let us show how to estimate the E matrix itself. Equations (2.30) and (2.31) imply that corresponding image points are connected by the E matrix. Thus, it is

¹ $[T]_{\times}$ is the 3×3 skew-symmetric matrix of T as defined in the Notation Section in this chapter.

possible to estimate the E matrix given these correspondences. This can be done using what is called *the eight-point algorithm*.

The eight-point linear algorithm

Longuet-Higgins [117] was the first to develop the 8-point algorithm in computer vision. This algorithm estimates the essential matrix using 8 pairs of matching points across two views [2]. The 3×3 essential matrix E can be derived from the essential constraint (2.31):

$$\mathbf{x}_j^\top E \mathbf{x}_i = (x_j, y_j, z_j)^\top \begin{bmatrix} e_1 & e_2 & e_3 \\ e_4 & e_5 & e_6 \\ e_7 & e_8 & e_9 \end{bmatrix} (x_i, y_i, z_i) = 0 \quad (2.33)$$

Let the vector $\mathbf{e} \in \mathbb{R}^9$ contain the elements of the essential matrix:

$$\mathbf{e} = (e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9)^\top \in \mathbb{R}^9 \quad (2.34)$$

If we have n correspondences, then the $n \times 9$ matrix $A \in \mathbb{R}^{n \times 9}$ is given by:

$$A = \begin{pmatrix} x_{j_1} x_{i_1} & x_{j_1} y_{i_1} & x_{j_1} z_{i_1} & y_{j_1} x_{i_1} & y_{j_1} y_{i_1} & y_{j_1} z_{i_1} & z_{j_1} x_{i_1} & z_{j_1} y_{i_1} & z_{j_1} z_{i_1} \\ x_{j_2} x_{i_2} & x_{j_2} y_{i_2} & x_{j_2} z_{i_2} & y_{j_2} x_{i_2} & y_{j_2} y_{i_2} & y_{j_2} z_{i_2} & z_{j_2} x_{i_2} & z_{j_2} y_{i_2} & z_{j_2} z_{i_2} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{j_n} x_{i_n} & x_{j_n} y_{i_n} & x_{j_n} z_{i_n} & y_{j_n} x_{i_n} & y_{j_n} y_{i_n} & y_{j_n} z_{i_n} & z_{j_n} x_{i_n} & z_{j_n} y_{i_n} & z_{j_n} z_{i_n} \end{pmatrix} \quad (2.35)$$

The epipolar geometry constraint (2.31) can be simply formulated as the inner product of A and \mathbf{e} . This leads to the linear equation in the entries of \mathbf{e} :

$$A\mathbf{e} = 0 \quad (2.36)$$

This linear equation may be solved for the vector \mathbf{e} . The rank of the matrix A needs to be exactly eight in order for the solution to be unique. This requires at least 8 corresponding points, i.e. $n \geq 8$. However, even with a sufficient number of corresponding points, the linear equation (2.36) may have no solution due to the noise. In this case, the one available option is to recover the entries of \mathbf{e} that minimise $\|A\mathbf{e}\|^2$.

Before extracting the relative pose from the recovered E matrix, this matrix must satisfy the essential constraint, i.e. $E \in \mathcal{E}$ (2.32). Enforcing this involves

orthogonally projecting it onto the essential space. Let us first consider the following theorem [82, 122]:

Theorem 2. Let $H \in \mathbb{R}^3$ be a real matrix and its $\text{SVD}(H) = U \text{diag}(\lambda_1, \lambda_2, \lambda_3) V^\top$, where $U, V \in SO(3)$ and $\lambda_1 \geq \lambda_2 \geq \lambda_3$. The optimal essential matrix $E \in \mathcal{E}$ is the one that minimises the cost function: $\|E - H\|_f^2$, given by $E = U \text{diag}(\sigma, \sigma, 0) V^\top$, where $\sigma = (\lambda_1 + \lambda_2)/2$. The subscript f designates the Frobenius norm.

The recovered E matrix is to an unknown scale, since the corresponding image points are expressed in homogeneous coordinates. A typical solution to deal with this ambiguity is to select an E whose non-zeros singular values are 1, i.e. $E = U \text{diag}\{1, 1, 0\} V^\top$.

Now after estimating E , let us consider the following theorem [82, 122]:

Theorem 3. For a non-zero Essential matrix $E = U \text{diag}(1, 1, 0) V^\top \in \mathcal{E}$, there exist four possible choices for the relative poses (R, T) , where $R \in SO(3)$ and $T \in \mathbb{R}^3$:

- $[R|T] = [UWV^\top \mid +u_3]$ or $[R|T] = [UWV^\top \mid -u_3]$ or
- $[R|T] = [UW^\top V^\top \mid +u_3]$ or $[R|T] = [UW^\top V^\top \mid -u_3]$.

where $W = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ is the rotation by an angle of $\frac{\pi}{2}$ around the Z-axis, and u_3 is the last column of U . Proof of this theorem can be found in [82].

So we end up with a four-fold ambiguity. This ambiguity may be solved by the reconstruction of a scene point X , which must be in front of both cameras in only one of these four solutions. This is known as the cheirality constraint. This constraint means the condition that points in an image must obviously lie in front of the camera and not at the back. The E matrix here gives four possible solutions for R and T . However, there is just one combination of R and T that guarantees the depth of the 3D reconstructed points is positive. Therefore, three out of the four solutions will be infeasible and hence will be discarded [72, 82, 84, 122, 200].

A structured form of the eight-point algorithm is given in Appendix C (Algorithm 4, page 298).

2.9.3 The fundamental matrix

The primary aim of two-view geometry is to recover the relative pose (R, T) between two views, even in non-calibrated camera scenarios circumstances. Let us again consider the rigid body motion between two views [2, 72, 82, 122, 200]:

$$\lambda_j x_j = R \lambda_i x_i + T, \quad (2.37)$$

where λ_i and λ_j are again the unknown depths. Let K be the calibration matrix, then, by multiplying both sides of (2.37) by K we get:

$$\lambda_j K \mathbf{x}_j = KR \lambda_i \mathbf{x}_i + KT, \quad (2.38)$$

$$\lambda_j \mathbf{x}'_j = KRK^{-1} \lambda_i \mathbf{x}'_i + KT, \quad (2.39)$$

$$\lambda_j \mathbf{x}'_j = KR \lambda_i K^{-1} \mathbf{x}'_i + T'. \quad (2.40)$$

where $\mathbf{x}'_j = K \mathbf{x}_j$ and $T' = KT$. Notice that the coordinates in \mathbf{x}'_i and \mathbf{x}'_j are now in pixels. Similarly to the case in (2.29) and (2.30), the depths λ_i and λ_j may be eliminated from (2.40) by multiplying both sides by the cross product ($\mathbf{x}'_j \times T'$):

$$\mathbf{x}'_j{}^\top K [T']_\times R K^{-1} \mathbf{x}'_i = 0 \quad (2.41)$$

Then:

$$\begin{aligned} \mathbf{x}'_j{}^\top F \mathbf{x}'_i &= 0 \\ F &= K [T']_\times R K^{-1} \in \mathbb{R}^{3 \times 3} \end{aligned} \quad (2.42)$$

Matrix F is called *the fundamental matrix* and $\mathbf{x}'_j{}^\top F \mathbf{x}'_i = 0$ is called *the epipolar constraint*. Note that when $K = I$, the fundamental matrix is equal to the essential matrix $[T']_\times R$. The relationship between the fundamental and essential matrices is then given by:

$$E = K^\top F K \quad (2.43)$$

Therefore, the fundamental matrix, F , is another algebraic representation of epipolar geometry (Figure 2.7) [82]. This matrix, however, deals with image points measured in pixels. Therefore, it is of great importance for motion estimation, since the image point locations in two frames are measured in pixels. More detail about estimating this matrix is given in Appendix C (Section C.1, page 293).

2.10 The three-view geometry

The two-view geometry is the dominant concept in computer vision even for monocular systems. Suppose two images of a reasonably stationary scene are taken from a moving vehicle with an on-board camera. As we have seen in the previous section, two-view geometry is mainly characterised by an algebraic representation that abstracts the geometry between these two images [82]. In fact, this algebraic entity represents the relationship between the corresponding image points in the two images through

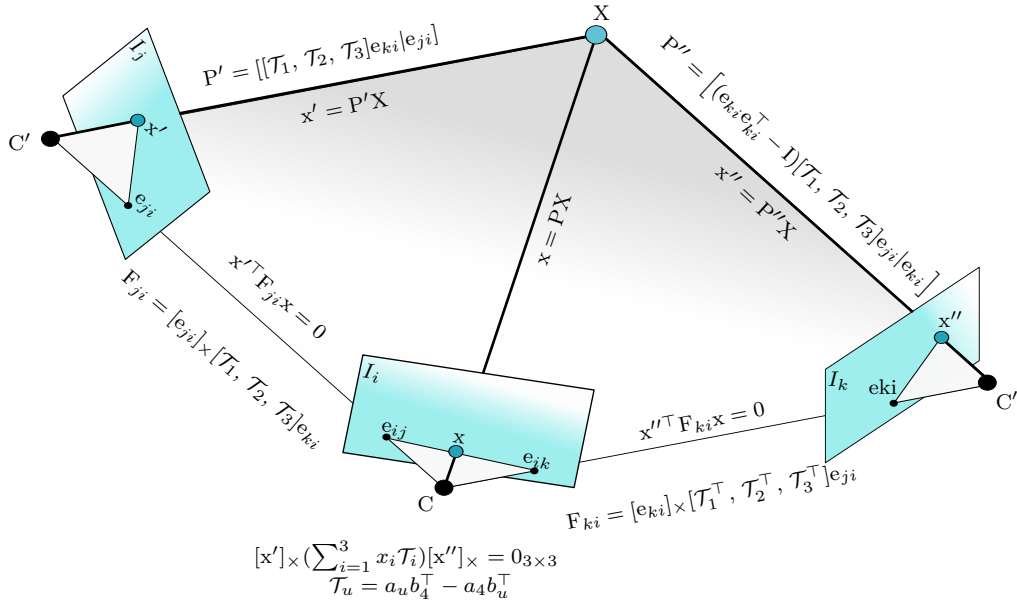


Fig. 2.8 Three-view geometry.

the Essential or the Fundamental matrix, from which the vehicle motion between the two positions is estimated.

The three-view geometry, on the other hand, is an extension of these geometrical concepts to three views, where three images of a scene are taken from three different camera locations. This geometry is hardly deployed in computer vision on applications involving localisation and reconstruction. The essential matrix role in two-view geometry is played by its analogous entity in three-view geometry called *the trifocal tensor* [82]. This latter entity encapsulates the geometry between three views via the three image point correspondences.

The three-view geometry concept was first introduced in [184, 208], where it was deployed for scene reconstruction from corresponding lines; however, without considering it as a tensor [83]. There is common agreement that the first work to refer to this entity as a tensor was introduced in [206]. In an independent study, author in [173] introduced the trifocal tensor's parameters for a system of four independent trilinearity constraints, which relate the coordinates of matching points in three frames. Later on, a linear approach for estimating trifocal tensor's parameters from no more than seven points is presented in [174].

Figure 2.8 summarises the three-view geometry, where point-point-point relationship, $x \leftrightarrow x' \leftrightarrow x''$, is given in function of the trifocal tensor's parameters \mathcal{T} . Let the 3×4 camera matrices for three views be $P = [I|0]$, $P' = [a_1 \ a_2 \ a_3 \ a_4] = [A|a_4]$ and $P'' = [b_1 \ b_2 \ b_3 \ b_4] = [B|b_4]$, where I is the identity matrix, A and B are 3×3 matrices and the vectors a_i and b_i are the i^{th} columns of the respective camera

matrices for $i = 1, \dots, 4$. Let $\mathbf{x} = (x_1, x_2, x_3)$, $\mathbf{x}' = (x'_1, x'_2, x'_3)$ and $\mathbf{x}'' = (x''_1, x''_2, x''_3)$ be corresponding image points in the three views respectively. Then, the relationship between these three image point correspondences $\mathbf{x} \leftrightarrow \mathbf{x}' \leftrightarrow \mathbf{x}''$ is given by the point-point-point relation [82, 174]:

$$\begin{aligned} \mathcal{T}_u &= a_u b_4^\top - a_4 b_u^\top, u = 1, \dots, 3 \\ [\mathbf{x}']_{\times} \left(\sum_{i=1}^3 x_i \mathcal{T}_i \right) [\mathbf{x}'']_{\times} &= 0_{3 \times 3} \end{aligned} \quad (2.44)$$

In these equations, the set of three matrices $\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3$ constitute the $3 \times 3 \times 3$ trifocal tensor, where \mathcal{T}_i is a 3×3 with rank 2. Note that the relationship between the three cameras is completely described by $\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3$.

2.11 Optical flow approach

In addition to multiple view geometry approach for vision-based motion estimation, optical flow is another option that can be used for this purpose. In real-life the apparent visual motion that one can experience as he moves is referred as optical flow. Certainly, when one is sitting in a moving train for example, all external objects (trees, building, ground, etc.) appear to move. This motion is referred as optical flow. One can realise that distant objects such as mountains or clouds move relatively slower, whereas, closer objects such as trees appear to move faster (Figure 2.9).

A mathematical model can represent this optical flow. In this model, the magnitude of the optical flow must define the position of the objects, whether they are close or distant. This model should also take in consideration the angle to the objects. Obviously, objects in the direction of the movement move much slower than objects to side of the direction of the movement.

Indeed, given camera optics (camera calibration parameters) and scene depth, there is a one-to-one correspondence between camera motion and optical flow. That means that the optical flow can be used to determine the motion of the vehicle. Formally, given a sequence of images, optical flow is an approximation of local image motion based upon local derivatives. The 2D image sequences are formed under perspective projection via the relative motion of a camera and scene objects as described in the previous sections. The crucial property in optical flow is that moving patterns cause varieties of the image brightness. These temporal intensity changes are assumed to occur due to motion only.

Many algorithms have been presented in the literature to estimate the optical flow. The well-known technique is the Luca-Kanade method. This algorithm employs the

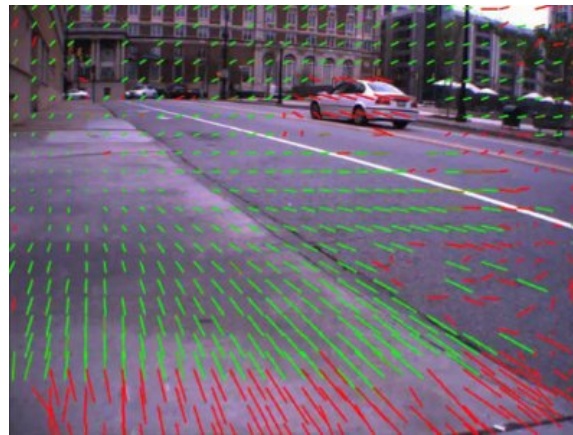


Fig. 2.9 Optical flow as seen by ground vehicle

gradients over multiple frames and analyses them spatially and temporally. Other techniques use feature tracking to estimate motion parameters. Elementary motion detector (EMD) method can be used as well to obtain motion measurements. The diversity of methods introduced in the literature illustrates how difficult the optical flow problem is. Having said that, it seems that estimating the motion from the optical flow has been addressed in different ways and there is no global solution applicable in all cases. However, some aspects must be considered when using the optical flow based solution:

- **Translation and rotation:** Translational optical flow indicates the motion of objects in the environment. In the case of rapid rotations, the measurements of translational optical flow may be illuminated by the rotational optical flow. External sensor such as the gyro can be used in these situations.
- **Low texture contrast:** Texture is crucial in the optical flow framework. Obviously in perfectly clean and smooth surfaces it is not possible to track any texture. Therefore, good images and good feature extractor algorithms are critical.
- **Mechanical jitter:** In any vision system, mechanical vibrations can significantly affect the algorithm. Tools to overcome this problem should be incorporated during the whole navigation.

2.11.1 The 2D and 3D motion constraint equations

In 2D (images) optical flow defines how much each image pixel moves between adjacent images while in 3D it specifies how much each volume in 3D moves between

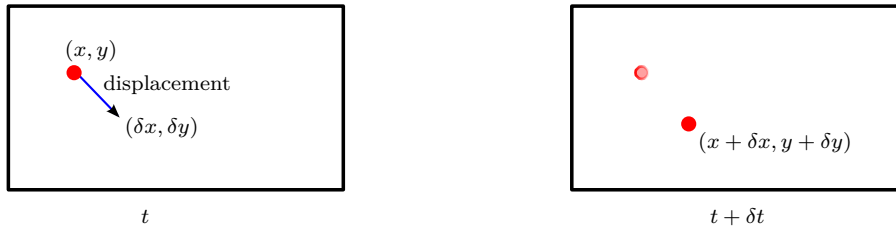


Fig. 2.10 The image at position (x, y, t) and time step t is the same as at $(x + \delta x, y + \delta y, t + \delta t)$ and time step $t + \delta t$.

adjacent volumes. The motion constraint equation is the basis of differential optical flow.

The 2D motion constraint equation

Let $I(x, y, t)$ be the centre pixel in an $n \times n$ neighbourhood. Assume this pixel moves by δx and δy to $I(x + \delta x, y + \delta y, t + \delta t)$ (Figure 2.10). One can see that $I(x, y, t)$ and $I(x + \delta x, y + \delta y, t + \delta t)$ are images of the same 3D scene point X . Then, one may write:

$$I(x, y, t) = I(x + \delta x, y + \delta y, t + \delta t) \quad (2.45)$$

Equation (2.45) represents the basis of the 2D motion constraint equation, which is illustrated at Figure 2.10. The first order Taylor series expansion about $I(x, y, t)$ in equation (2.45) is given as:

$$I(x + \delta x, y + \delta y, t + \delta t) = I(x, y, t) + \frac{\partial I}{\partial x} \delta x + \frac{\partial I}{\partial y} \delta y + \frac{\partial I}{\partial t} \delta t + \Psi \quad (2.46)$$

where Ψ represents the higher order terms. In fact, Ψ is assumed to be small and can safely be ignored. Using (2.45) and (2.46), one can get:

$$\frac{\partial I}{\partial x} v_x + \frac{\partial I}{\partial y} v_y + \frac{\partial I}{\partial t} = 0 \quad (2.47)$$

where $v_x = \frac{\partial x}{\partial t}$ and $v_y = \frac{\partial y}{\partial t}$ are the x and y components of **image velocity** or **optical flow**. By putting:

$$I_x = \frac{\partial I}{\partial x}, I_y = \frac{\partial I}{\partial y}, I_t = \frac{\partial I}{\partial t} \quad (2.48)$$

equation (2.47) may be written as:

$$(I_x, I_y) \cdot (v_x, v_y) = -I_t \quad (2.49)$$

or as:

$$\nabla I \cdot \vec{v} = -I_t \quad (2.50)$$

where $\nabla I = (I_x, I_y)$ is the spatial intensity gradient and $\vec{v} = (v_x, v_y)$ is the image velocity or optical flow at pixel (x, y) at time t .

Equation (2.50) is of great interest in optical flow framework and called 2D motion constraint equation. This constraint is the basis of optical flow based motion estimation.

The 3D motion constraint equation

The 3D motion constraint equation is an extension of the 2D motion constraint equation. This 3D constraints depicts an $n \times n \times n$ block at (x, y, z) at time t moving by $(\delta x, \delta y, \delta z)$ to $(x + \delta x, y + \delta y, z + \delta z)$ over time δt . Similarly to the 2D case, the 3D motion constraint equation is given as:

$$\nabla I \cdot \vec{v} = -I_t \quad (2.51)$$

where $\nabla I = (I_x, I_y, I_z)$ is the 3D spatial intensity gradient and $\vec{v} = (v_x, v_y, v_z)$ is the 3D velocity.

2.11.2 Optical flow based motion estimation methods

Many techniques have been presented in the literature for motion estimation through the optical flow. Among all the presented approaches we can list:

- **Gradient based approach:** This approach was first presented by Horn and Schunck [88]. This technique combines the gradient constraint (2.50) with a global smoothness term to constrain the estimated velocity field $\vec{v} = (v_x, v_y)$. Then, Lucas and Kanade presented in [120] a second technique through the gradient-based approach. This technique relies on an iterative least-squares minimisation.
- **Correlation-based approach:** This method is very close to block matching. In this approach, the image is divided into fixed-size blocks. The matching is performed using the correlation technique.

- **Spatio-temporal energy based approach:** In this approach, frequencies define the velocity of translational moving objects. Motion can be evaluated by using spatio-temporally oriented filters.
- **Phase-based approach:** this approach is based on spatially filtering the images using a bank of quadrature pair filters.

2.12 Conclusion

We have introduced in this chapter the basics of image formation and multi-view geometry. More attention is given to the pinhole camera model, describing some optical principles behind image formation. We discussed in detail the mathematical representation of the camera geometry and the link between the positions of 3D scene points with their corresponding image points, which is defined as a mapping from the Euclidean 3-space to Euclidean 2-space. We then discussed the two-view geometry, the epipolar geometry and the relationship between two image points of the same 3D scene point. We have detailed in depth the essential matrix, which constrains the relative translation and rotation between two cameras. We presented also some techniques that might be used to estimate this matrix. The fundamental matrix and its estimation methods were investigated as well. Three-view geometry is also presented in this chapter, where we showed that the trifocal tensor is the algebraic representation of this geometry. This chapter includes as well the basic notions of the optical flow approach.

Chapter 3

Convex Optimisation

In this chapter, we provide a brief introduction to some basic notions that are used in this thesis. First, we start by giving a short overview of the subclasses of optimisation problems and their implementations. We then give a brief introduction to the basic concepts in convex optimisation that are used extensively throughout the thesis. Specifically, we detail the concepts of convex sets, convex functions, convex and quasi-convex optimisation problems. As a special case, we will focus on the second order cone program (SOCP), which is used as a solution tool for our optimisation models.

Robust convex optimisation is also considered in this chapter along with global optimisation methods, such as the branch and bound algorithm.

3.1 Notation

The set of real numbers is denoted with the symbol \mathbb{R} , and the set of non-negative real numbers is denoted with $\mathbb{R}_+ = \{x \in \mathbb{R} | x \geq 0\}$; while the positive numbers set is denoted with $\mathbb{R}_{++} = \{x \in \mathbb{R} | x > 0\}$. The n -dimensional Euclidean space is denoted with \mathbb{R}^n and the set of real $m \times n$ matrices is denoted with $\mathbb{R}^{m \times n}$. The non-negative numbers is denoted with $\mathbb{R}_+^n = \{x_i \geq 0, \text{ for } i = 1, \dots, n\}$, while $\mathbb{R}_{++}^n = \{x_i > 0, \text{ for } i = 1, \dots, n\}$ is for the positive orthant.

The symbol $\mathbf{1}$ denotes a vector with a given dimension all of whose components are one. A vector is denoted with x and its i^{th} component is denoted with x_i . The symmetric $k \times k$ matrices set is denoted with \mathbb{S}^k , while \mathbb{S}_+^k and \mathbb{S}_{++}^k are used to represent the symmetric positive semi-definite and definite $k \times k$ matrices respectively.

3.2 Optimisation problem classes

Firstly, let us frame the optimisation problem and why we should optimise. Assuming we have a set of measurements x_i , our goal is to fit a parametrised model to these measurements. Now consider some model values \hat{x}_i , then the residuals are given by: $\delta_i = \|x_i - \hat{x}_i\|$, where $\|\cdot\|$ is any norm in the measurement space. Optimisation involves finding the model parameters that minimise the residuals δ_i .

Many algorithms are presented in the literature to solve this optimisation problem. All of these algorithms can be subdivided into three main classes. The first covers linear approaches such as least squares optimisation. The second approach concerns the iterative methods, where a geometric cost function is minimised using iterative algorithms such as Levenberg-Marquardt or Gauss-Newton; and the third method, which is important for us in this work, is convex optimisation.

3.2.1 Least squares solution

Generally, the least squares solution is said to be the optimal solution [84], and it is one of the most commonly used methods in numerical computation. The least squares solution minimises the cost function:

$$\min_x \|\Delta\| = \sum_i \|x_i - \hat{x}_i\|^2. \quad (3.1)$$

The objective of this optimisation problem is to find a solution that minimises the sum of squared errors between observed and estimated values. This optimisation assumes that the measurements are estimated from values that are spoiled with Gaussian noise with variance σ^2 . Hence, the probability of the set of measurements x_i given true values \hat{x}_i is obtained as:

$$P(\{x_i|\hat{x}_i\}) = K \prod_i \exp(-\|x_i - \hat{x}_i\|^2 / 2\sigma^2) \quad (3.2)$$

where K is a normalising constant. Minimisation after taking logarithms shows that the values of \hat{x}_i that maximise this probability also minimise the cost function in (3.1). Therefore, under an assumption of Gaussian noise in the measurements, we can say that the least squares solution is the maximum likelihood (ML) estimate. However, in multi-view geometry, the justification of this assumption is not straight forward. In addition, the optimality of the Maximum Likelihood is not always guaranteed [84].

The more general form of Least Squares (LS) minimisation problem is given as:

$$\min_x f_o(x) = \|Ax - b\|_2^2 = \sum_i (a_i^\top x - b_i)^2 \quad (3.3)$$

The objective function $f_o(x)$ is a summation of squares: $(a_i^\top x - b_i)$. Note here that there are no constraints. A set of linear equations can be used to solve the least-squares problem (3.3):

$$(A^\top A)x = A^\top b \quad (3.4)$$

This form has the analytical solution $x_{opt} = (A^\top A)^{-1}A^\top b = A^\ddagger b$, where $A^\ddagger = (A^\top A)^{-1}A^\top$ is the pseudo-inverse of A . This can also be called the Batch least-squares solution.

3.2.2 Iterative methods

In general, iterative estimation methods minimise a geometric cost function using iterative algorithms such as Levenberg-Marquardt, Gauss-Newton, gradient descent or conjugate gradient. Let us consider the functional relation: $X = f(P)$, where X is the measurement vector and P is the parameter vector. The aim here is to find some parameters \hat{P} that satisfy as far as possible the functional relation. In other words, we try to minimise the difference between X and $f(\hat{P})$. Thus, this involves minimising $\|\varepsilon\|$, where $\|\varepsilon\| = f(P) - X$.

When f is not-linear, the approach is to start with an initial value of parameters \hat{P} , let us denote it as the set of parameters P_0 , and continuously refine it until a satisfactory ending threshold is met. Let $\varepsilon_0 = f(P_0) - X$, then we assume that the function f is approximated at P_0 by:

$$f(P_0 + \Delta) = f(P_0) + J\Delta \quad (3.5)$$

where J is the linear mapping represented by the Jacobian matrix $J = \partial f / \partial P$ [82]. We now look for some updated parameters P_1 , where $P_1 = P_0 + \Delta$, that minimises $f(P_1) - X = f(P_0) + J\Delta - X = \varepsilon_0 + J\Delta$. Thus, we have to minimise $\varepsilon_0 + J\Delta$ in Δ , which can be done by solving the linear system:

$$(J^\top J)\Delta = -J^\top \varepsilon_0 \quad (3.6)$$

The solution is then updated by adding the solution Δ^* of (3.6) to the initial guess:

$$P_{i+1} = P_i + \Delta^* \quad (3.7)$$

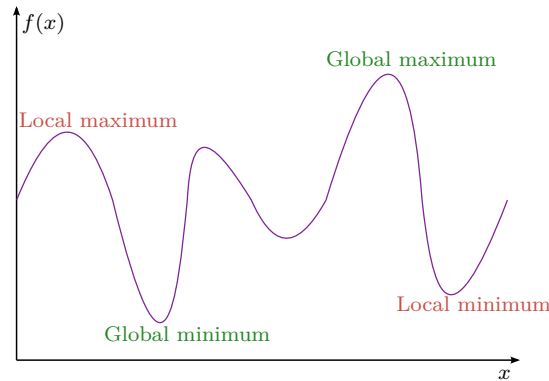


Fig. 3.1 Local and global solutions.

The Gauss-Newton algorithm iterates the equations (3.5) and (3.6) and updates (3.7). The previous solution Δ^* is used as the linearisation point and the initial guess. This procedure is iterated until a satisfactory convergence standard is achieved; usually until a predefined termination criterion is met.

The Levenberg–Marquardt (LM) algorithm is slightly different from Gauss-Newton. The LM algorithm adds a new variable λ to equation (3.6):

$$(\mathbf{J}^\top \mathbf{J} + \lambda \mathbf{I})\Delta = -\mathbf{J}^\top \varepsilon_0 \quad (3.8)$$

where \mathbf{I} is the identity matrix. If the recovered Δ by solving (3.8) leads to a decrease in the error, the update is accepted and λ is updated (usually divided by a factor of 10). But, if the error increases, then λ is multiplied by the same factor and (3.8) is solved again. This process is repeated until a value of Δ is recovered to decrease the error. However, and notwithstanding their dependency on a good initialisation guess, these algorithms have high probabilities of convergence to a local minimum value, or does not converge at all (Figure 3.1).

3.2.3 Convex optimisation

Our aim in this thesis is to extend the state-of-the-art techniques for motion estimation in computer vision. Thinking of more accurate and robust techniques would be a valid option. Thus, global optimisation methods such as convex and quasi-convex optimisation offer the possibility of getting around the drawbacks of linear and iterative techniques and recovering robust and global solutions. Convex functions necessary have a single minimum solution to the problem due to their shape.

3.3 Basic notions of convex optimisation

Convex optimisation is well detailed in [25, 131], where basic notions are given, specifically: convex sets, convex functions including affine functions, convex and quasi-convex functions. Second-order cone programming (SOCP) is also detailed in this textbook, referring to problems that can be solved using this technique.

First, let us consider two different points x_0 and x_1 in \mathbb{R}^n (Figure 3.2). The line passing through x_0 and x_1 has the form:

$$y = (1 - \alpha)x_0 + \alpha x_1, \text{ where } \alpha \in \mathbb{R}$$

Note that when $\alpha = 0$, then $y = x_0$, and when $\alpha = 1$, then $y = x_1$. When α is between 0 and 1, $0 \leq \alpha \leq 1$, this corresponds to the line segment between x_0 and x_1 (Figure 3.2).

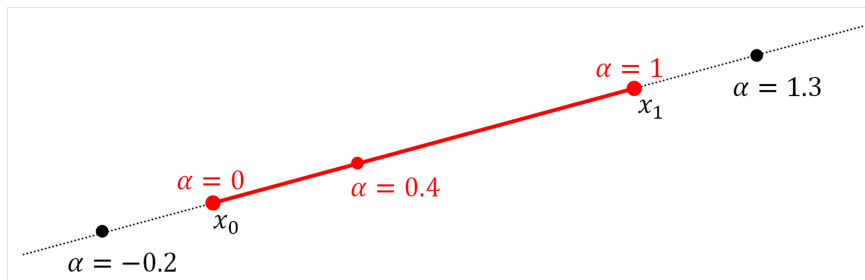


Fig. 3.2 The line passing through x_0 and x_1 is identified by $(1 - \alpha)x_0 + \alpha x_1$. The (red) solid line segment between x_0 and x_1 is parametrised when α is between 0 and 1.

3.3.1 Affine sets

If the segment passing any two different points in set $C \subseteq \mathbb{R}^n$ stays in C , then this set is affine. In other words:

Definition 1. A set $C \subseteq \mathbb{R}^n$ is an affine set, if for any x_0 and $x_1 \in C$ and $\alpha \in \mathbb{R}$, we have $(1 - \alpha)x_0 + \alpha x_1$ in C .

3.3.2 Convex set

A set C in \mathbb{R}^n is supposed to be convex, if the line segment joining any two points of the set also belongs to the set [25, 131]. In more details, if x_0 and x_1 are in C , then $(1 - \alpha)x_0 + \alpha x_1$ must belong to C for each $\alpha \in [0, 1]$.

Definition 2. A set $C \subseteq \mathbb{R}^n$ is a convex set, if for any x_0 and $x_1 \in C$ and $0 \leq \alpha \leq 1$, then, $(1 - \alpha)x_0 + \alpha x_1$ is in C .

Obviously, all affine sets are convex, but not all convex sets are affine sets. Figure 3.3 illustrates the notion of a convex set. Note that the line segment joining x_0 and x_1 in the middle plot in this figure does not entirely lie inside the set.

In general, a point x of the form:

$$x = \sum_{i=1}^n \alpha_i x_i, \text{ where } \sum_{i=1}^n \alpha_i = 1, \quad 0 \leq \alpha_i \leq 1 \text{ and } x_i \in C \quad (3.9)$$

is said to be a convex combination of x_1, \dots, x_n . In these conditions, if the non-negativity condition of α_i is dropped, the combination is known as an affine combination.

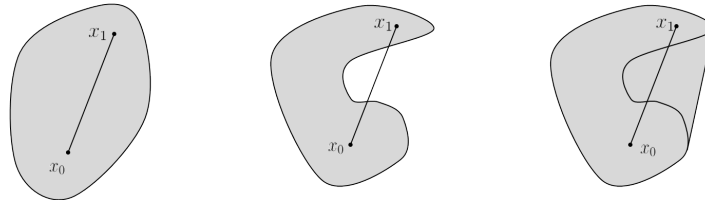


Fig. 3.3 Left: A convex set. Middle: A non-convex set. Right: The convex hull of the middle set.

Examples of convex sets

Norm Balls. A norm ball in \mathbb{R}^n has the form:

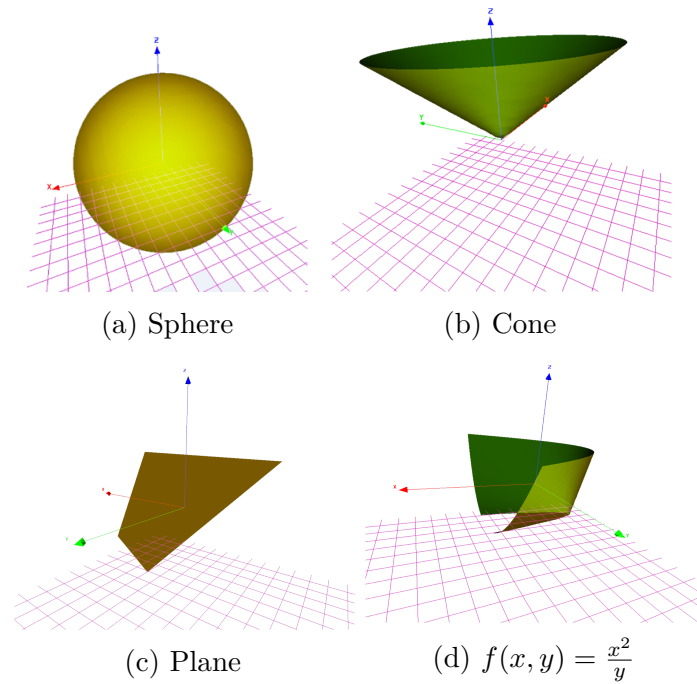
$$B(x_{ctr}, r) = \{x \mid \|x - x_{ctr}\| \leq r\}$$

where $x_{ctr} \in \mathbb{R}^n$ is the centre of the ball, the scalar $r > 0$ is the radius of the ball and $\|\cdot\|$ is a norm on \mathbb{R}^n . An Euclidean ball is a norm ball using the norm $\|x\| = (x^\top x)^{\frac{1}{2}}$. That is [25]:

$$B_{Euclid.}(x_{ctr}, r) = \{x \mid \|x - x_{ctr}\|_2 \leq r\}$$

Norm Cones. A norm cone in \mathbb{R}^n has the form [25]:

$$C = \{(x, t) \in \mathbb{R}^n \times \mathbb{R} \mid x \in \mathbb{R}^n, t \in \mathbb{R}; \|x\| \leq t\}$$

Fig. 3.4 Examples of convex sets in \mathbb{R}^3 .

A second-order cone of dimension n is a norm cone, where the Euclidean norm is used. That is:

$$C_n = \{(x, t) \in \mathbb{R}^{n-1} \times \mathbb{R} \mid \|x\| \leq t\} \quad (3.10)$$

A second-order cone is also called a quadratic cone, a Lorentz cone or an ice-cream cone (Figure 3.4b). In \mathbb{R}^3 this cone has the equation:

$$C_3 = \{(x, t) \in \mathbb{R}^2 \times \mathbb{R} \mid \sqrt{x_1^2 + x_2^2} \leq t\}$$

Planes or hyperplane. A plane in \mathbb{R}^3 has the following equation:

$$C_4 = \{(x_1, x_2, x_3) \in \mathbb{R}^3 \mid ax_1 + bx_2 + cx_3 = d\} \in \mathbb{R}^3$$

In general, $C_4 = \{X \in \mathbb{R}^n \mid \mathbf{P}^t X = \delta\}$ forms a hyperplane in \mathbb{R}^n , where \mathbf{P} is a nonzero vector in \mathbb{R}^n and δ is a scalar.

3.3.3 Operations that preserve convexity

The two main categories of operations that preserve convexity of a set are the intersection and the mapping via an affine function.

Intersection

Let C_1 and C_2 be convex sets in \mathbb{R}^n , then $C_1 \cap C_2$ is a convex set. Convexity is preserved under intersection.

Mapping via an affine function

Let $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$ be an affine function with the form: $f(x) = Ax + b$ where $A \in \mathbb{R}^{n \times m}$ and $b \in \mathbb{R}^m$. If $C \subseteq \mathbb{R}^n$ is a convex set, then the image of C under f given by:

$$f(C) = \{f(x) | x \in C\}$$

is a convex set.

3.3.4 Affine functions

In the case where a function $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$ is defined by a sum of a constant and a linear function, then this function is said affine. Thus, this function is defined as: $f(x) = Ax + b$, where $A \in \mathbb{R}^{n \times m}$ and $b \in \mathbb{R}^m$. Let us consider the linear function $f(x) = ax + b$, where $x \in \mathbb{R}$. This is simply the equation of a line in which a is the slope and b is the y -intercept of this line. In \mathbb{R}^2 , this function, $f(x) = a_1x_1 + a_2x_2 + b$, is the equation of a plane. Using higher dimensions, a function $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$ is affine in the case where $f(x) = a^\top x + b$, here $a, x \in \mathbb{R}^n$ and $b \in \mathbb{R}^m$ which is the equation of a hyperplane. Now the first function $f(x) = Ax + b$, has several affine functions $f_i(x) = a_i^\top x + b_i$, for $i = 1, \dots, m$, where a_i are the row of the matrix A .

Now, according to Definition 1, an affine set C is any set where the line segment passing any two different points in C lies in C . Similarly, if for any x_0 and $x_1 \in C$ and $\alpha \in \mathbb{R}$, then $\alpha x_0 + (1 - \alpha)x_1 \in C$.

It is easy to confuse the concepts of affine sets and affine functions. To clarify these two concepts, consider this example in \mathbb{R}^2 : the affine function $f(x) = x_0 + x_1 + 1$ is the equation of a plane (red-sloped plane in Figure 3.5) and the affine function $f(x) = 6$ is the equation of a plane (blue-horizontal plane in Figure 3.5). The affine set, on the other hand, $\{x \in \mathbb{R}^2 | f(x) = 6\}$ is all points on the line $\{x \in \mathbb{R}^2 | x_0 = 5 - x_1\}$ where the red and the blue planes intersect.

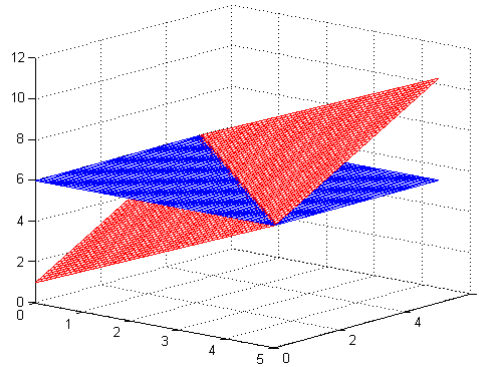


Fig. 3.5 Example of affine function and affine set.

3.3.5 Convex function

A convex function is a function f that has the domain, C , of a convex set such that for all $x, y \in C$ and $0 \leq \alpha \leq 1$:

$$f((1 - \alpha)x + \alpha y) \leq (1 - \alpha)f(x) + \alpha f(y) \quad (3.11)$$

This inequality means that the line segment between $(x, f(x))$ and $(y, f(y))$ lies above the graph (Figure 3.6).

Definition 3. A function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is convex if its domain C is a convex C and for all $x, y \in C$: $f((1 - \alpha)x + \alpha y) \leq (1 - \alpha)f(x) + \alpha f(y)$.

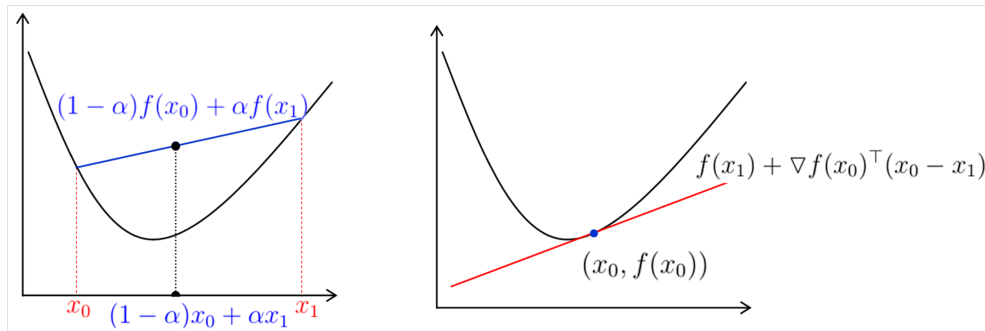


Fig. 3.6 Graph of a convex function.

If strict inequality holds in (3.11) whenever $x \neq y$ and $0 < \alpha < 1$, then, function f is strictly convex. We say f is concave if $-f$ is convex.¹

¹Most theorems given in this chapter are simply stated without proof since they are well known and whose proofs may be found in books such as [25]. Proofs are only provided for results that are crucial to the thesis.

Theorem 4. A function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is convex if its domain C is a convex C and for all $x, y \in C : f((1 - \alpha)x + \alpha y) \leq (1 - \alpha)f(x) + \alpha f(y)$.

Suppose we have a differentiable function, then, the first-order convexity condition is that its first-order approximation is a global under-estimator, that is:

$$\forall x_0, x_1 \in C, f(x_0) \geq f(x_1) + \nabla f(x_0)^\top (x_0 - x_1) \quad (3.12)$$

If f is differentiable then f is convex if and only if (3.12) holds for all $x_0, x_1 \in C$. The geometric interpretation of this means that f is convex if and only if f lies above its tangent plane at all points. Figure 3.6 illustrates the geometry of the definition. This illustrates why convexity is of great importance in optimisation. It means that the tangent is global under the estimator of the function. ²

Theorem 5. A function f is convex if and only if its domain C is convex and $f(x_0) \geq f(x_1) + \nabla f(x_0)^\top (x_0 - x_1)$ holds for all $x_0, x_1 \in C$.

If $f(x)$ is twice differentiable, which means its Hessian $\nabla^2 f$ exist for all points x in its domain C , then f is convex if and only if C is convex and its Hessian is positive semi-defined for all points $x \in C$, that is,

$$\forall x \in C, \nabla^2 f(x) \succeq 0 \quad (3.13)$$

The geometrical interpretation of (3.13) is that the graph of $f(x)$ has an upward curvature at x .

Theorem 6. A function f is convex if and only if its domain C is convex and its Hessian $\nabla^2 f(x)$ is a positive, semidefinite matrix.

Some examples of functions used in this thesis that are convex functions:

- **Norms:** Every norm on \mathbb{R}^n is convex;
- **Affine functions:** $f(x) = a^\top x + c$, where $a \in \mathbb{R}^n, c \in \mathbb{R}$;
- **Quadratic functions:** $f(x) = x^\top A x + 2b^\top x + c$ if and only if $A \succeq 0$ since A is its Hessian;
- **Exponential functions:** $f(x) = e^{\alpha x}$, where $\alpha \in \mathbb{R}$.

3.3.6 Quasi-convex functions

Commonly, convex problems do not come up in multi-view geometry [84]. Most problems are formulated as quasi-convex optimisations instead. Quasi-convex functions are defined as:

²[T]_× is the 3×3 skew-symmetric matrix of T as defined in the Notation Section in this chapter.

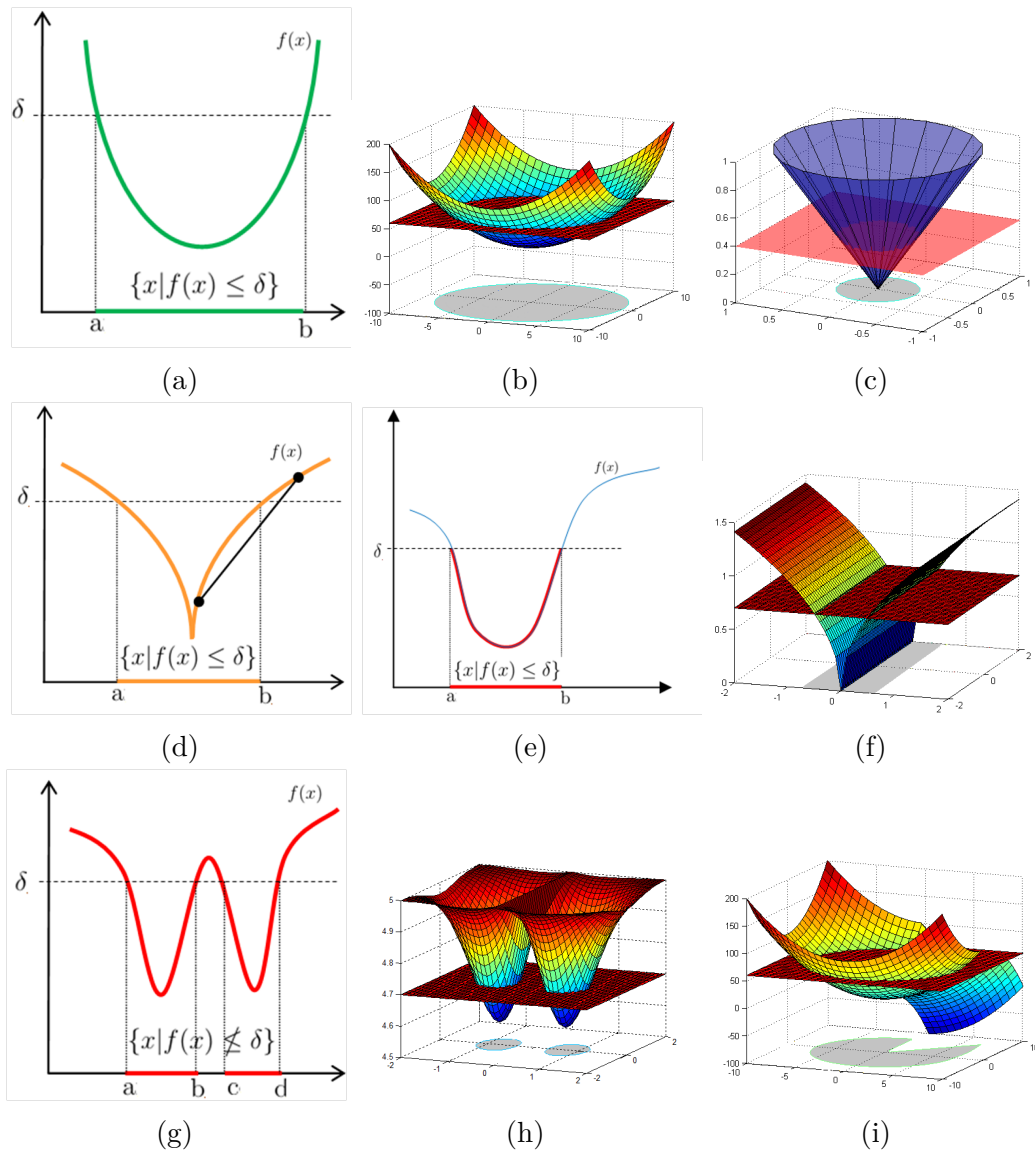


Fig. 3.7 Quasi-convex functions.

Definition 4. A function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is quasi-convex if its domain C and its sub-level sets $S_\delta = \{x \in C \mid f(x) \leq \delta\}$ are convex for $\delta \in \mathbb{R}$.

The geometrical interpretation of quasi-convexity is shown at Figure 3.7. In this figure, functions in the first and the second rows (plots from (a) to (f)) are quasi-convex. For a given value of δ , the δ -sub-level sets, $S_\delta = \{x \in C \mid f(x) \leq \delta\}$, of these functions are convex. These sets are the intervals $S_\delta = [a, b]$ in \mathbb{R} for the plots (a), (d) and (e); and the shaded (gray areas) in \mathbb{R}^2 for the plots (b), (c) and (f). It is easy to see that these δ -sub-level sets are convex. First row functions (plots (a), (b) and (c)) are convex and quasi-convex at the same time. However, functions in the second row (plots (d), (e) and (f)) are not convex since not all line segments between two points lie above their corresponding graph. Therefore, clearly, any convex function is also quasi-convex. However, not all quasi-convex functions are convex. Functions on the third row in this figure (plots (g), (h) and (i)) are not convex for the same reason. Furthermore, they are not quasi-convex, either since their δ -sub-level sets, intervals $S_\delta = [a, b] \cup [c, d]$ in \mathbb{R} for the plot (g) and the shaded areas (gray areas) in \mathbb{R}^2 for the plots (h) and (i), are not convex.

Two important properties of quasi-convex functions are:

- Quasi-convex functions have no local minima apart from the global minimum;
- If functions f_1, f_2, \dots, f_m are quasi-convex, the function $f = \max_{i=1, \dots, m} f_i$, the point-wise maximum, is also quasi-convex (proof is given in the next section). This is illustrated in Figure 3.8 for the one dimensional case.

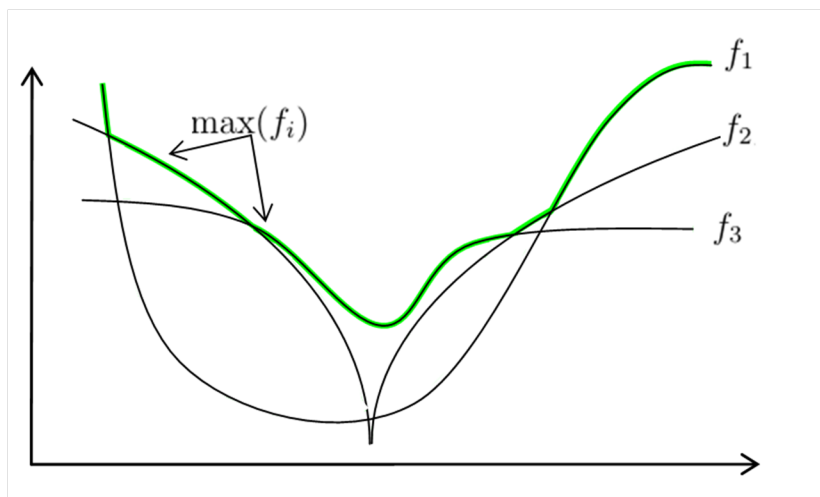


Fig. 3.8 Quasi-convex function and their pointwise maximum.

3.3.7 Operations that preserve quasi-convexity

Theorem 7. If functions f_1, \dots, f_m are quasi-convex, then their pointwise maximum f , defined by: $f(x) = \max\{f_1, \dots, f_m\}$ with $\text{dom } f = \bigcap_{i=1}^m \text{dom}(f_i)$ is also quasi-convex.

Proof. First we need to prove that the domain of f is convex then we shall prove that its sub-level sets are convex.

- $\text{dom } f = \bigcap_{i=1}^m \text{dom}(f_i)$ is the domain of f . We know that the domain of each f_i is convex since they are quasi-convex. We know that convexity is preserved under intersection; thus, $\text{dom}(f)$ must be convex.
- The δ -sub-level set of f is:

$$\begin{aligned} S_\delta(f) &= \{x \in \text{dom } f \mid f(x) \leq \delta\} \\ &= \{x \mid \max f_i \leq \delta, i = 1, \dots, m\} \\ &= \bigcap_{i=1}^m \{x \mid f_i \leq \delta, i = 1, \dots, m\} \\ &= \bigcap_{i=1}^m S_\delta(f_i) \end{aligned}$$

Again, since the intersection preserves the convexity, then the δ -sublevel set of f is convex [94]. \square

3.4 Convex optimisation problem

3.4.1 Definitions

First, we will give some definitions concerning optimisation problems. Let us consider this notation:

$$\begin{aligned} \min_x \quad & f_0(x) \\ \text{subject to} \quad & f_i(x) \leq 0 \text{ for } i = 1, \dots, m \\ & c_i(x) = 0 \text{ for } i = 1, \dots, p \end{aligned} \tag{3.14}$$

This means that we wish to find all components of x that provide the minimum value of the function $f_0(x)$, but these components must satisfy all the m conditions $f_i(x) \leq 0$ and all the p conditions $c_i(x) = 0$.

Here, $x \in \mathbb{R}^n$ is called the optimisation variable and $f_0(x) : \mathbb{R}^n \rightarrow \mathbb{R}$ is the objective function (or cost function). The equalities $c_i(x) = 0$ are called the equality constraints and the corresponding functions $c_i(x)$ are called the equality constraint

functions. The inequalities $f_i(x) \leq 0$ are called the inequality constraints, and the corresponding functions $f_i(x)$ are called the inequality constraint functions.

The set of points for which the objective and all constraint functions are defined,

$$\mathcal{D} = \text{dom } f_0 \cap \bigcap_{i=1}^m \text{dom } f_i \cap \bigcap_{i=1}^p \text{dom } c_i \quad (3.15)$$

is called the optimisation problem domain. A point $x \in \mathcal{D}$ is said to be feasible if it satisfies the inequality and the equality constraints $f_i(x)$ and $c_i(x)$. If there is no x that satisfies this condition, then the problem is said infeasible. The set of all feasible points is called the feasible set.

Definition 5. p^* is said an optimal value to the optimisation problem (3.14), if

$$p^* = \inf \{f_0(x) \mid f_i(x) \leq 0, i = 1, \dots, m, \quad c_i(x) = 0, \text{ for } i = 1, \dots, p\} \quad (3.16)$$

Definition 6. x^* is said an optimal solution to the optimisation problem (3.14), if x^* is feasible and $f_0(x^*) = p^*$. A feasible point x with $f_0(x) \leq p^* + \varepsilon$, where $\varepsilon > 0$, is called ε -suboptimal.

3.4.2 Convex optimisation form

Similarly to problem (3.14), a convex optimisation problem has the form:

$$\begin{aligned} & \text{find} && f_0(x) \\ & \text{subject to} && f_i(x) \leq 0 \text{ for } i = 1, \dots, m \\ & && a_i^\top x = b_i \text{ for } i = 1, \dots, p \end{aligned} \quad (3.17)$$

where:

- The objective function f_0 must be convex;
- The inequality constraint functions $f_i(x) \leq 0$ for $i = 1, \dots, m$ must be convex;
- The equality constraint functions $a_i^\top x = b_i$ for $i = 1, \dots, p$ must be affine.

The feasible set of a convex optimisation problem is convex. Therefore, in a convex optimisation problem, we are minimising a convex objective function (which has a single minimum) over a convex set.

A point $x \in \mathcal{D}$ is a feasible point if it satisfies the constraints $f_i(x) \leq 0$ for $i = 1, \dots, m$ and $c_i(x) = 0$ for $i = 1, \dots, p$ ($c_i(x) = a_i^\top x - b_i$). The feasible set is the set of all feasible points. An optimisation problem is said to be feasible if there exists at least one feasible point and infeasible otherwise.

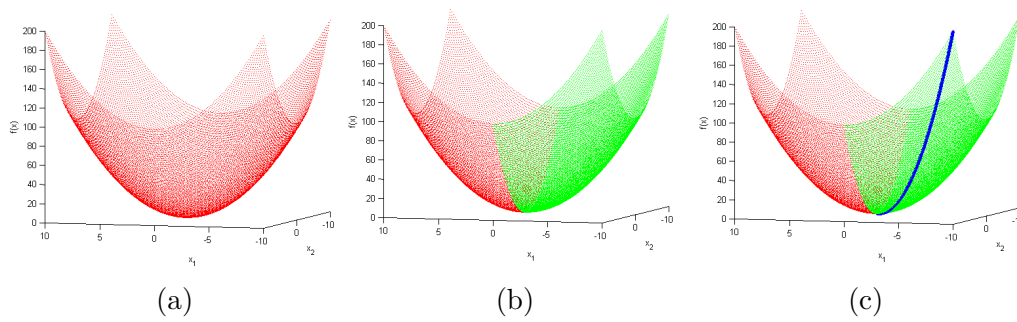


Fig. 3.9 a. Graph of the objective function $f_0(x) = x_1^2 + x_2^2$, note $f_0(x)$ is clearly a convex function. b. After adding the inequality constraint function $f_1(x) = x_1 \leq 0$, the feasible set has been reduced to the green set. c. The quality constraint function $c_1(x) = x_1 + x_2 = 0$ has even reduced the feasible set to the blue curve.

Example

In order to get a clear idea, let us consider this problem in \mathbb{R}^2 :

$$\begin{aligned}
 \min_x \quad & f_0(x) = x_1^2 + x_2^2 \\
 \text{subject to} \quad & f_1(x): x_1 \leq 0 \\
 & c_1(x): x_1 + x_2 = 0
 \end{aligned} \tag{3.18}$$

This is a convex optimisation problem since the objective function $f_0(x)$ is convex, the inequality constraint function is convex and the equality constraint function is affine. The feasible set is convex (Details is given in the caption of Figure 3.9).

3.4.3 Convex feasibility problems

One form of convex optimisation problems that is used extensively in this thesis is the quasi-convex optimisation via convex feasibility problems. The most attractive property of a convex problem is that it has one single minimum, which is at the same time the global minimum [25, 84].

A feasibility problem has the following form:

$$\begin{aligned}
 \text{find} \quad & x \\
 \text{subject to} \quad & f_i(x) \leq 0 \text{ for } i = 1, \dots, m \\
 & c_i(x) = 0 \text{ for } i = 1, \dots, p
 \end{aligned} \tag{3.19}$$

Note that there is no objective function here. Therefore, the feasibility problem seeks to find out if the constraints are consistent and, if so, recover the points that satisfy them.

3.5 Quasi-convex optimisation problem

The general form of a quasi-convex optimisation problem is:

$$\begin{aligned} \min_x \quad & f_0(x) \\ \text{subject to} \quad & f_i(x) \leq 0 \text{ for } i = 1, \dots, m \\ & a_i^\top x = b_i \text{ for } i = 1, \dots, p \end{aligned} \tag{3.20}$$

where the objective function $f_0(x)$ must be quasi-convex, the inequality constraint functions $f_i(x) \leq 0$ for $i = 1, \dots, m$ must be convex and the equality functions $c_i = a_i^\top x - b_i$ for $i = 1, \dots, p$ must be affine. The most important feature here is the fact that the feasible set of the quasi-convex optimisation problem is convex.

One may notice that a quasi-convex optimisation problem is similar to a convex optimisation problem, except that the objective function $f_0(x)$ is quasi-convex. Though, quasi-convex optimisation problems share most agreeable properties of convex optimisation [84]. The most important one is that a quasi-convex function has no local minima apart from the global minimum.

To solve this quasi-convex optimisation problem, its sub-level sets are represented using inequalities of convex functions $\varphi_\delta : \mathbb{R}^n \rightarrow \mathbb{R}$ with:

$$f_0(x) \leq \delta \Leftrightarrow \varphi_\delta(x) \leq 0 \tag{3.21}$$

Let us consider the following problem:

$$\begin{aligned} \text{find} \quad & f_0(x) \\ \text{subject to} \quad & \varphi_\delta(x) \leq 0 \text{ for } i = 1, \dots, m \\ & f_i(x) \leq 0 \text{ for } i = 1, \dots, m \\ & a_i^\top x = b_i \text{ for } i = 1, \dots, p \end{aligned} \tag{3.22}$$

One may notice that this problem is similar to the convex feasibility problem (3.19), where the aim is to determine whether the constraints are consistent or not. Thus, let p^* be the optimal solution of the problem (3.20). In the case where the problem (3.22) is feasible, then $p^* \leq \delta$. On the contrary, if it is infeasible, then $p^* \geq \delta$.

This leads to solve the quasi-convex problem (3.20) using what is called a bisection algorithm (Algorithm 1). In each step of this algorithm, we check the feasibility of the problem. We start with the lower bound δ_l and the upper bound δ_u , assuming that the problem is feasible within the interval $[\delta_l, \delta_u]$ with an optimal solution p^* . Then we solve the convex feasibility problem at the interval midpoint $\delta = \frac{(\delta_l + \delta_u)}{2}$, to

Algorithm 1 Bisection algorithm for solving the optimisation problem.

Given: optimal value $\delta_{opt} \in [\delta_l, \delta_u]$ and tolerance $\varepsilon > 0$

while ($\delta_l - \delta_u < \varepsilon$) **do**

$$\delta = \frac{1}{2}(\delta_l + \delta_u)$$

Solve the convex feasibility problem (3.22)

if feasible **then**

$$\delta_u = \delta$$

else

$$\delta_l = \delta$$

end if

end while

check whether the optimal solution p^* is within the lower or the upper bound. This allows us to dismiss the second half in which the optimal solution does not exist. This produces a new narrower interval (half the width of the initial interval) that certainly contains the optimal solution.

3.6 Basic examples of convex optimisation

3.6.1 Linear programming

A linear program (LP) is appropriate for optimisation when the objective function and the constraint functions are all affine. Therefore the general form of a linear program is:

$$\begin{aligned} \min_x \quad & f_0 : c^\top x + d \\ \text{subject to} \quad & Gx \leq h \\ & Ax = b \end{aligned} \tag{3.23}$$

where $G \in \mathbb{R}^{m \times n}$ and $A \in \mathbb{R}^{p \times n}$.

The geometric interpretation of a LP is shown in Figure 3.10. We seek a point in the polyhedron \mathcal{P} where function $c^\top x + d$ has the smallest value, if such a point exists. The number of constraints defines the dimension of this polyhedron. The objective function $c^\top x + d$ is affine so its level curves $\{x | f_0(x) = t\}$ for some value $t \in \mathbb{R}$ are hyperplanes $\{x | c^\top x + d = t\}$ that are orthogonal to vector c . The optimal point x^* is the point in \mathcal{P} which is as far as possible in the direction $-c$. Decreasing values of the cost function are in the negative direction of the gradient direction $-\Delta f_0 = -c$.

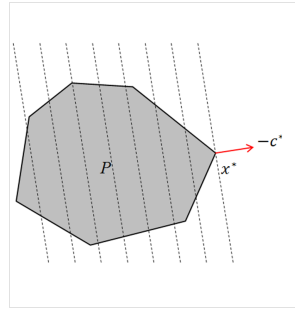


Fig. 3.10 The geometric interpretation of the LP. The shaded area is the feasible set. The dotted lines are the level sets of the cost function, and the red arrow indicates the direction in which we are optimising, defined by $-\nabla f_0 = -c$.

3.6.2 Quadratic programming

Let us consider the optimisation problem (3.17). If the objective function of this problem is quadratic, and the constraint functions are affine, this problem is called a quadratic program (QP)(Figure 3.11):

$$\begin{aligned} \min_x \quad & f_0 : x^\top P x + 2q^\top x + r \\ \text{subject to} \quad & Gx \leq d \\ & Ax = b \end{aligned} \tag{3.24}$$

where $S_+^n \in \mathbb{R}^{m \times n}$.

3.6.3 Quadratic constrained quadratic program

A quadratic constrained quadratic program (QCQP) is a problem of the form:

$$\begin{aligned} \min_x \quad & x^\top P_0 x + 2q_0^\top x + r_0 \\ \text{subject to} \quad & x^\top P_i x + 2q_i^\top x + r_i, \quad \text{for } i = 1, \dots, m \\ & a_i^\top x = b_i \quad \quad \quad \text{for } i = 1, \dots, p \end{aligned} \tag{3.25}$$

where $x, q_i, a_i \in \mathbb{R}^n, P_i \in S_+^n$ for $i = 0, \dots, m, b_i, r_i \in \mathbb{R}$. In a QCQP, a convex quadratic function is minimised over a region which is the intersection of ellipsoids (when $P_i > 0$).

Observe that quadratic programs (QP) include linear programs (LP) as a special case, by taking $P = 0$ in (3.24).

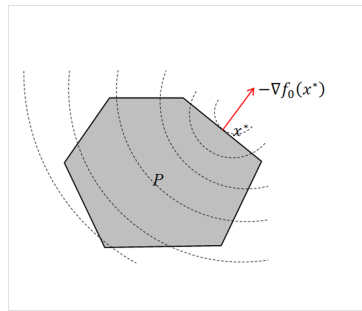


Fig. 3.11 The geometric interpretation of the QP. The shaded area is the feasible set. The dotted curves are the level sets of the cost function, and the red arrow indicates the direction in which we are optimising, defined by $-\nabla f_0(x^*)$.

3.7 Second-order cone programming

The Second-Order Cone Programming (SOCP) is a convex optimisation technique used to solve convex and quasi-convex problems. In a second order cone program (SOCP), a linear function is minimised over the intersection of an affine set and the product of second order cones. SOCP are close to quadratic programs (QP). Let us consider the second order cone program (SOCP):

$$\begin{aligned} \min_x \quad & f^\top x \\ \text{subject to} \quad & \|A_i x + b_i\| \leq c_i^\top x + d_i, \text{ for } i = 1, \dots, m \\ & g_i^\top x = h_i \text{ for } i = 1, \dots, p \end{aligned} \quad (3.26)$$

where $x \in \mathbb{R}^n$ is the optimisation variable, and $f \in \mathbb{R}^n$, $A_i \in \mathbb{R}^{(n_i-1) \times n}$, $b_i \in \mathbb{R}^{n_i-1}$, $c_i, g_i \in \mathbb{R}^n$ and $d_i, h_i \in \mathbb{R}$ are the problem parameters. The norm $\|\cdot\|_2$ is the standard Euclidean norm $\|u\|_2 = \sqrt{(u^\top u)}$.

The constraint:

$$\|A_1 x + b_1\| \leq c_1^\top x + d_1 \quad (3.27)$$

is called the second order cone constraint of dimension n_i [84, 135]. This is called so because the set of points that satisfy this second order cone constraint is the inverse image of the unit second order cone of dimension k , defined as:

$$\mathcal{L}_k = \left\{ \begin{bmatrix} u \\ t \end{bmatrix} \mid u \in \mathbb{R}^{k-1}, t \in \mathbb{R}, \|u\| \leq t \right\} \quad (3.28)$$

under an affine mapping:

$$\|A_i x + b_i\| \leq c_i^\top + d_i \Leftrightarrow \begin{bmatrix} A_i \\ c_i^\top \end{bmatrix} x + \begin{bmatrix} b_i \\ d_i \end{bmatrix} \in \mathcal{L}_{n_i} \quad (3.29)$$

In other words, the set of points satisfying a second-order cone constraint of dimension n_i is the inverse image of the second-order cone of dimension n_i under an affine mapping. Therefore, the feasible set of the problem (3.26) is the intersection of m second-order cones $\{x \mid \|A_i x + b_i\| \leq c_i^\top + d_i\}$ with p hyperplanes $g_i^\top x = h_i$. Both second-order cones and hyperplanes are convex. Therefore, their intersection produces a convex set. Hence, the SOCP in (3.26) is a convex optimisation problem as the objective function is a convex function and the constraints define a convex set.

Note that when $n_i = 1$ for $i = 1, \dots, m$, the SOCP reduces to the linear program (LP):

$$\begin{aligned} \min_x \quad & f^\top x \\ \text{subject to} \quad & c_i^\top x + d_i \geq 0, \text{ for } i = 1, \dots, m \\ & g_i^\top x = h_i \text{ for } i = 1, \dots, p \end{aligned} \quad (3.30)$$

Furthermore, SOCP integrates numerous significant classes of convex optimisation problems, such as Quadratic Program QP and Quadratic Constrained Quadratic Program QCQP. To see that, let us consider the QCQP problem given in (3.25). This problem can be rewritten as [135]:

$$\begin{aligned} \min_x \quad & \|P_0^{\frac{1}{2}} x + P_0^{-\frac{1}{2}} q_0\|^2 + r_0 - q_0^\top P_0^{-1} q_0 \\ \text{subject to} \quad & \|P_i^{\frac{1}{2}} x + P_i^{-\frac{1}{2}} q_i\|^2 + r_i - q_i^\top P_i^{-1} q_i, \text{ for } i = 1, \dots, m \\ & a_i^\top x = b_i \text{ for } i = 1, \dots, p \end{aligned} \quad (3.31)$$

with the assumption that matrices P_i are positive definite. Introducing a new optimisation variable t , problem (3.31) can be formulated and solved as SOCP:

$$\begin{aligned} \min_x \quad & t \\ \text{subject to} \quad & \|P_0^{\frac{1}{2}} x + P_0^{-\frac{1}{2}} q_0\| \leq t \\ & \|P_i^{\frac{1}{2}} x + P_i^{-\frac{1}{2}} q_i\| \leq (q_i^\top P_i^{-1} q_i r_i)^{\frac{1}{2}}, \text{ for } i = 1, \dots, m \\ & a_i^\top x = b_i \text{ for } i = 1, \dots, p \end{aligned} \quad (3.32)$$

The optimal solution of problems (3.25) and (3.32) are equal to constant and a square root. If $c_i = 0$, for $i = 1, \dots, m$ in problem (3.26), the SOCP reduces to a QCQP. Therefore, SOCPs are more general than QCQPs, QPs and LPs.

3.7.1 SOCP feasibility problems

SOCP feasibility problems are of great importance in this thesis. In addition to the detail given in the previous sections, this section gives more analysis to quasi-convex optimisation problem. Most problems throughout this thesis are formulated as SOCP feasibility problems. First, let us consider the following feasibility problem:

$$\begin{aligned} \text{Find} \quad & x \\ \text{subject to} \quad & \|A_i x + b_i\| \leq \delta (c_i^\top x + d_i), \quad \text{for } i = 1, \dots, m \\ & c_i^\top x + d_i \geq 0, \quad \text{for } i = 1, \dots, m \end{aligned} \quad (3.33)$$

This problem is a SOCP feasibility problem, which involves finding a solution x that satisfies the second-order cone constraint.

Example of a quasi-convex function

A particular form of a quasi-convex function that is used frequently in this thesis is given as:

$$f(x) = \frac{f_1(x)^2 + f_2(x)^2}{f_3(x)^2} \quad (3.34)$$

where $f_i(x) = a_i^\top x + b_i$.

As given in Section 3.3.4 (page 56), $f_i(x)$, $i = 1, \dots, 3$ are affine, since they have the form: $f_i(x) = a_i^\top x + b_i$. To prove that this function is a quasi-convex function, let us show first that its domain is convex.

The domain of $f(x)$ is the intersection of convex sets:

$$\mathcal{D}_{f(x)} = \text{dom } f_1 \cap \text{dom } f_2 \cap \text{dom } f_3$$

Since convexity of a set is preserved under intersection, $\mathcal{D}_{f(x)}$ is also a convex set.

Now, we have to prove that the sub-level sets of $f(x)$ are convex. First, let us define $S_\delta(f)$ as a δ -sublevel of f , then:

$$\begin{aligned} S_\delta(f) &= S_\delta(f) = \{x \in \mathcal{D}_{f(x)} \mid f(x) \leq \delta\} \\ &= \left\{x \mid \frac{f_1(x)^2 + f_2(x)^2}{f_3(x)^2} \leq \delta, f_3(x) > 0\right\} \\ &= \left\{x \mid f_{i1}(x)^2 + f_{i2}(x)^2 \leq \delta f_{i3}(x)^2, f_{i3} > 0\right\} \\ &= \left\{x \mid \sqrt{f_{i1}(x)^2 + f_{i2}(x)^2} \leq \sqrt{\delta} f_{i3}(x), f_{i3} > 0\right\} \\ &= \left\{x \mid \|f_{i1}(x), f_{i2}(x)\|_2 \leq \sqrt{\delta} f_{i3}(x), f_{i3} > 0\right\} \end{aligned} \quad (3.35)$$

where $\delta \geq 0$. One can see that (3.35) is the equation of a second-order cone of dimension 3 (see equation (3.10) page 55), which is clearly a convex set [84, 94]. The function $f(x) = \frac{f_1(x)^2 + f_2(x)^2}{f_3(x)^2}$ is quasi-convex. Furthermore, for any function where the numerator is the sum of a number of squared affine functions, then this function: $f(x) = \frac{f_1(x)^2 + f_2(x)^2 + \dots + f_{n-1}(x)^2}{f_n(x)^2}$ is still quasi-convex

Theorem 8. Let $f_i(x)$, $i = 1, \dots, 3$ are affine functions, having the form: $f_i(x) = a_i^\top x + b_i$. Then function $f(x) = \frac{f_1(x)^2 + f_2(x)^2}{f_3(x)^2}$, in the domain $\{x | f_3(x) > 0\}$ is quasi-convex.

Let us consider now the optimisation problem that minimises the maximum of a number optimisation functions given in (3.34):

$$\begin{aligned} \min_x \quad & f_0(x) = \max_{i=1, \dots, m} f_i(x) = \max_{i=1, \dots, m} \left(\frac{f_{i1}(x)^2 + f_{i2}(x)^2}{f_{i3}(x)^2} \right) \\ \text{subject to} \quad & f_{i3}(x) > 0, \quad \text{for } i = 1, \dots, m. \end{aligned} \quad (3.36)$$

Note that f_{i1}, f_{i2}, f_{i3} are all affine functions of the optimisation variable x . According to Theorem 8, $f_i(x) = \frac{f_{i1}(x)^2 + f_{i2}(x)^2}{f_{i3}(x)^2}$ is quasi-convex function. In addition, according to Theorem 7 (page 61), the function $f_0 = \max_{i=1, \dots, m} \left(\frac{f_{i1}(x)^2 + f_{i2}(x)^2}{f_{i3}(x)^2} \right)$ is also quasi-convex since the point-wise maximum is also convex. The constraint function has a convex domain $\{x | f_{i3}(x) > 0, \text{ for } i = 1, \dots, m\}$, therefore problem (3.36) is a quasi-convex optimisation problem [94, 101].

To solve this optimisation problem, suppose δ is an upper bound for the objective function $f_0(x) = \max_{i=1, \dots, m} f_i(x)$. Obviously, δ is also an upper bound for each of the functions $f_i(x)$. Then,

$$\begin{aligned} f_i(x) &\leq \delta \\ \frac{f_{i1}(x)^2 + f_{i2}(x)^2}{f_{i3}(x)^2} &\leq \delta \\ \frac{\sqrt{f_{i1}(x)^2 + f_{i2}(x)^2}}{f_{i3}(x)} &\leq \delta \end{aligned} \quad (3.37)$$

Since $f_{i3}(x) > 0$ for $i = 1, \dots, m$, and by using the norm $\|u\|_2 = (u^\top u)^{\frac{1}{2}}$, then the function in 3.37 may be reformulated as:

$$\|f_{i1}(x), f_{i2}(x)\|_2 \leq \delta f_{i3}(x), \quad i = 1, \dots, m. \quad (3.38)$$

Thus, the quasi-convex optimisation problem in (3.36) can be rewritten as:

$$\begin{aligned} \min_{\delta, x} \quad & \delta \\ \text{subject to} \quad & \|f_{i1}(x), f_{i2}(x)\|_2 \leq \delta f_{i3}(x), \quad i = 1, \dots, m. \end{aligned} \quad (3.39)$$

By introducing the new variable δ , the new composed optimisation variable Φ is defined as: $\Phi = (\delta, x)$. This makes the optimisation problem (3.39) a non-SOCP problem because the right hand side of the constraint function: $\delta (a_i^\top x + b_i)$ is quadratic in the optimisation variable $\Phi = (\delta, x)$. The solution for this issue is to fix the value of δ , which makes the expression $\delta (c_i^\top x + d_i)$ linear in the optimisation variable x .

The optimisation problem (3.39) may then be formulated for a given value of δ as:

$$\begin{aligned} \text{Find} \quad & \delta \\ \text{subject to} \quad & \|f_{i1}(x), f_{i2}(x)\|_2 \leq \delta f_{i3}(x), \quad i = 1, \dots, m. \end{aligned} \quad (3.40)$$

In this formulation, we are checking the feasibility of the problem for a given value of δ [25]. Similarly to problem (3.20) in Section 3.5 (page 64), this problem can be solved by a sequence of SOCP feasibility problems. This is done through the Bisection algorithm (Algorithm 1, page 65) for finding the global optimum. Note that in every iteration of the bisection algorithm, the search space is halved.

The L_∞ -norm

Given an n -dimensional vector $x = (x_1, \dots, x_n)^\top$, the L_p norm of vector x for $p = 1, 2, \dots$ is defined as:

$$\|x\|_p = \left(\sum_{i=1}^n |x_i|^p \right)^{\frac{1}{p}} \quad (3.41)$$

With the special case where $p = \infty$, the L_∞ -norm is defined as:

$$\|x\|_\infty = \max_i |x_i| \quad (3.42)$$

The most commonly encountered vector norm in computer vision is the L_2 norm (also called the Euclidean norm) given by:

$$\|x\|_2 = \left(\sum_{i=1}^n |x_i|^2 \right)^{\frac{1}{2}} = \sqrt{x_1^2 + x_2^2 + \dots + x_n^2} \quad (3.43)$$

3.8 Robust convex optimisation

In the recent years, the need to deal with uncertain data has become crucial, especially when real life applications are involved. In these circumstances, robust optimisation aims to recover an optimal solution whose feasibility must be guaranteed for any realisation of the uncertain data [20]. Robust optimisation, for which the data are not specified exactly, explicitly incorporates uncertainty to protect the decision-maker against parameter ambiguity and stochastic uncertainties [65].

We have seen that convex optimisation offers the possibility of getting around the problems that linear and non-linear approaches have in terms of convergence. We have also seen that the cost function in convex optimisation is geometrically meaningful, and has a single global minimum [25, 94]. In this thesis, most problems are formulated in terms of parameter estimation from image-based measurements. Since these measurements are subject to deterministic perturbations, more studies have started to focus on how this parameter estimation might be improved if uncertainties in these data are integrated [26].

Therefore, in this thesis more interest is given to robust optimisation and robust convex optimisation in particular. Throughout this thesis we will be dealing with uncertain data resulting from inexact measurements. Inaccuracy of the devices could be considered as an important source of uncertainties as well. Furthermore, data uncertainty results in uncertain constraints and an uncertain objective function as well.

Robust optimisation is a recent approach to optimisation under uncertain data, where the uncertainty model is not stochastic, but rather deterministic. Even though it is still considered as a new approach to optimisation problems under uncertainty, an increasing number of real applications have already proved its efficiency. Robust optimisation is mainly designed to allow uncertainty-affected optimisation problems to provide guarantees about the performance of the solution [20, 65]. In other words, in this optimisation, instead of recovering the solution in some probabilistic sense under stochastic uncertainty, the optimiser builds a solution that is optimal for any realisation of the uncertainty in a given set [18]. In cases where the optimality of a solution is affected by the uncertainty, the robust optimisation main goal will be then to seek a solution that performs well enough for any value taken by the unknown coefficients. Many studies are conducted toward different robustness methods, however, the most known one is to optimise the worst-case objective function [65].

Robust optimisation, in general form, deals with two sets of entities, decision and uncertain variables. The first aim of robust optimisation is to recover the optimal

solution on the decision variables such that the worst-case is minimised and the constraints are robustly feasible, while the uncertainty is allowed to take arbitrary values in a defined uncertainty set [116]. The optimal solution is evaluated using the realisation of the uncertainty that is most unfavourable [65].

The general form of this robust optimisation is given by:

$$\begin{aligned} \min_x \quad & \max_{\omega} f_0(x, \omega) \\ \text{subject to} \quad & f_i(x, \omega_i) \leq 0; \forall \omega_i \in \mathcal{W}, i = 1, \dots, m \end{aligned} \quad (3.44)$$

where $x \in \mathbb{R}^n$ are the decision variable, f_0 and f_i are objective function and the inequality constraint functions, $\omega_i \in \mathbb{R}^k$, are the uncertain variables and $\mathcal{W} \subseteq \mathbb{R}^k$ are the uncertainty sets. If the objective function and the inequality constraints are convex, the problem (3.44) is considered as a robust convex optimisation problem. The aim of problem (3.44) is to recover the optimal solution, x^* , of the cost function among all feasible solutions, allowing the uncertain variables ω_i to take any realisation within \mathcal{W}_i .

3.8.1 Robust solution using SOCP

In this thesis we focus on the SOCP, as most problems are solved using this approach. In our work, uncertainties, and their propagation through all multiple-view geometry algorithms, are estimated. Therefore, uncertainties are included in our optimisation problems. Let us consider the following SOCP problem:

$$\begin{aligned} \min_x \quad & f^\top x \\ \text{subject to} \quad & \|A_i x + b_i\| \leq c_i^\top x + d_i, \quad \text{for } i = 1, \dots, m \\ & g_i^\top x = h_i, \quad \text{for } i = 1, \dots, p \end{aligned} \quad (3.45)$$

Due to these uncertainties, the problem parameters A_i, b_i, c_i and d_i are allowed to accept any values within the set. In these situations, we refer to checking the feasibility of the SOCP for all realisations of the uncertain parameters as a robust feasibility problem. The problem of finding the set of robust feasible solutions is called the robust counterpart. Thus, problem (3.45) will be written as:

$$\begin{aligned} \min_x \quad & f^\top x \\ \text{subject to} \quad & \|A_i x + b_i\| \leq c_i^\top x + d_i; \forall (A_i, b_i, c_i, d_i) \in \mathcal{W}_i \\ & g_i^\top x = h_i, \text{ for } i = 1, \dots, p \end{aligned} \quad (3.46)$$

where $\mathcal{W}_i \subseteq \mathbb{R}^k$ are again the uncertainty sets.

3.8.2 Robust least-squares via SOCP

First let us remind the reader on least squares (LS) minimisation. Consider the LS problem given in (3.3):

$$\min_x f_o(x) = \|Ax - b\|_2^2 = \sum_i (a_i^\top x - b_i)^2 \quad (3.47)$$

The objective function here is the sum of squares $(a_i^\top x - b_i)$. Note that the LS problem has no constraints. The objective function, in fact, is a (quadratic) convex function since $f_o(x)$ may be written as $x^\top A^\top Ax - 2b^\top Ax + b^\top b$. Thus, a set of linear equations can be used to solve the least-squares problem (3.47):

$$(A^\top A)x = A^\top b \quad (3.48)$$

Thus,

$$\mathbf{A}x = \mathbf{b} \quad (3.49)$$

where $\mathbf{A} = (A^\top A)$ and $\mathbf{b} = A^\top b$. This is similar to finding a solution to an overdetermined set of equations $Ax \cong b$. Figure 3.9a (page 63) shows a plot of the objective function. Obviously, the problem has a single minimum. Similarly to other problems, this optimal solution x_{opt} is obtained by putting its gradient $\nabla f_o(x) = 2x^\top A^\top A - 2b^\top A = 0$. Therefore, the analytical solution $x_{opt} = (A^\top A)^{-1}A^\top b = A^\ddagger b$, where $A^\ddagger = (A^\top A)^{-1}A^\top$ is the pseudo-inverse of A .

Now, in the scenario where the parameters of the equation (3.49) are subject to unknown but bounded uncertainties $\Delta\mathbf{A}$ and $\Delta\mathbf{b}$, where $\|\Delta\mathbf{A}\| \leq \rho$ and $\|\Delta\mathbf{b}\| \leq \xi$, then our new Least-Squares problem has the form:

$$\min_x \max_{\|\Delta\mathbf{A}\| \leq \rho, \|\Delta\mathbf{b}\| \leq \xi} \|(\mathbf{A} + \Delta\mathbf{A})x - (\mathbf{b} + \Delta\mathbf{b})\| \quad (3.50)$$

This problem is referred as the Robust Least-Squares and introduced by El Ghaoui and Lebret in [52], Miguel Soma Lobo in [135], Chandrasekaran in [32, 33] and Sayad et al. in [5]. The problem (3.50) may be formulated in closed form as:

$$\begin{aligned} & \max_{\|\Delta\mathbf{A}\| \leq \rho, \|\Delta\mathbf{b}\| \leq \xi} \|(\mathbf{A} + \Delta\mathbf{A})x - (\mathbf{b} + \Delta\mathbf{b})\| \\ &= \max_{\|\Delta\mathbf{A}\| \leq \rho, \|\Delta\mathbf{b}\| \leq \xi} \max_{\|y\| \leq 1} y^\top (\mathbf{A}x - \mathbf{b}) + y^\top \Delta\mathbf{A}x - y^\top \Delta\mathbf{b} \\ &= \max_{\|z\| \leq 1} \max_{\|y\| \leq 1} y^\top (\mathbf{A}x - \mathbf{b}) + z^\top x + \xi \\ &= \|\mathbf{A}x - \mathbf{b}\| + \rho\|x\| + \xi \quad (3.51) \end{aligned}$$

The Robust Least-Squares solution to this problem computes the exact value of the optimal worst-case residuals using convex Second-Order Cone Programming (SOCP). This problem can be formulated as a SOCP [52]:

$$\begin{aligned} \min_{\lambda} \quad & \lambda \\ \text{subject to} \quad & \|\mathbf{A}x - \mathbf{b}\| \leq \lambda - \tau \\ & \left\| \begin{bmatrix} x \\ 1 \end{bmatrix} \right\|_2 \leq \tau \end{aligned} \tag{3.52}$$

The unique solution to this problem is then given by:

$$S = \begin{cases} (\mu I + \mathbf{A}^\top \mathbf{A})^{-1} \mathbf{A}^\top \mathbf{b} & \text{if } \mu = (\lambda - \tau)/\tau > 0 \\ \mathbf{A}^\dagger \mathbf{b} & \text{else,} \end{cases} \tag{3.53}$$

3.9 Branch and bound algorithms

In this thesis we will be dealing with some non-convex optimisation problems. Clearly, in these situations we cannot use convex optimisation to recover the global solution. Fortunately, there are some tools to overcome these kinds of problems. One way to solve these problems is to use branch and bound algorithms. These are iterative approaches for recovering the global solution. The outcome of these algorithms is an ε -suboptimal solution. Changing the value of ε to its smallest value allows us to get to the possible global optimum.

The basic idea of branching and bounding is simple but at the same time very reliable. Figure 3.12 illustrates an example of the branch and bound algorithm. Suppose we are looking for the global minimum of an objective function $\Phi(x)$ over a domain Q_0 . If $\Phi(x)$ is non-convex, finding such a minimum directly is very difficult. One way to do that is to bound $\Phi(x)$ over Q_0 with a simpler function, $\Phi_{lower}(x)$, and then minimise this new function. Minimising this simpler function $\Phi_{lower}(x)$ should be much easier than minimising $\Phi(x)$. Let us consider $\Phi_{lower}(x)$ as the lower bound of $\Phi(x)$ over Q_0 , then $\Phi_{lower}(x) \leq \Phi(x)$ for all $x \in Q_0$. After finding the minimum of the bounding function $\Phi_{lower}(x)$, let us denote it as x_{lower}^* . We evaluate the cost functions of both functions: the bounding function, $y_{lower}^* = \Phi_{lower}(x_{lower}^*)$, and the original function $y^* = \Phi(x_{lower}^*)$. If $y_{lower}^* - y^* \leq \varepsilon$, then the ε -suboptimal solution is recovered and the algorithm stops. Otherwise, we subdivide (**branch**) the domain Q_0 into subsets (which we refer as rectangles) and perform (**bounding**) on each

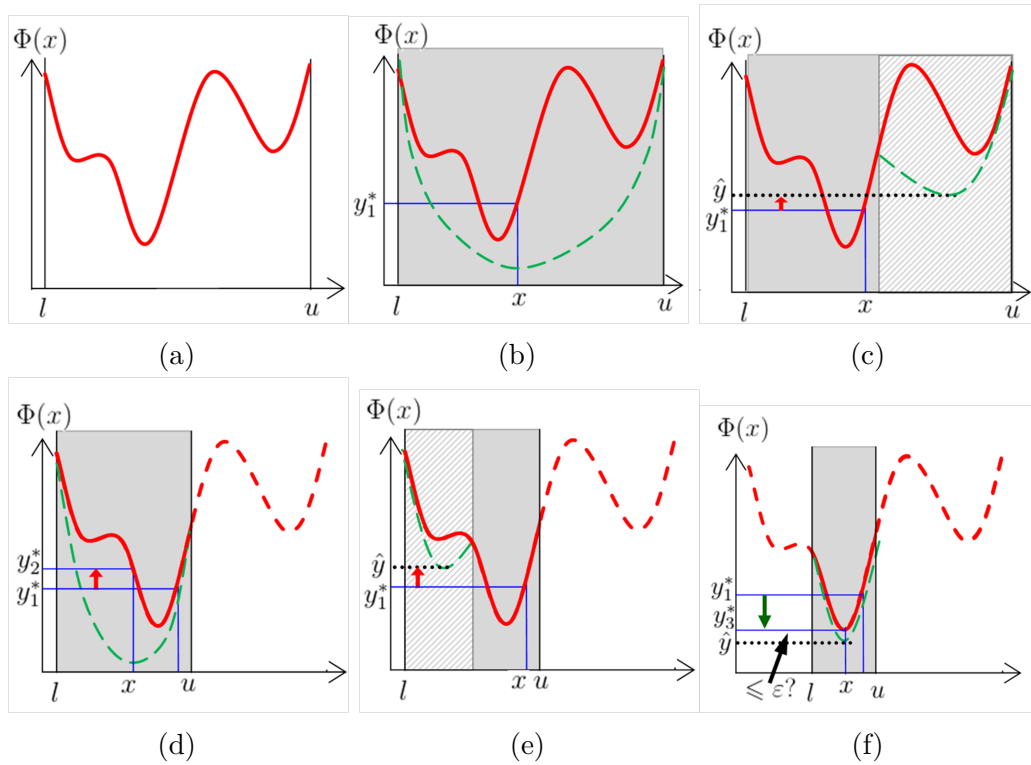


Fig. 3.12 Plots depicting the branch and bound algorithm for the problem of non-convex minimisation. (a) shows the non-convex function $\Phi(x)$ and the interval $l \leq x \leq u$. (b) shows the first bounding via the convex relaxation of the $\Phi(x)$, indicated by a (green) dashed curve and its minimum value at x . y_1^* is the corresponding cost of the original function $\Phi(x)$. (c) Shows results after the first branching. The right hand side subset (hatched in light gray) can be discarded since the minimum \hat{y} of its bounding function is higher than the value of the best solution found so far y_1^* . (d) The function $\Phi(x)$ in the discarded subsets is shown using (red) bold dashed curves. The function is bounded in the remaining subset and the new minimum y_2^* is estimated. This new estimate y_2^* is dismissed as it is greater than the best solution found so far y_1^* . (e) Shows further branching in which the left side subset is discarded as the minimum \hat{y} of its bounding function is higher than y_1^* . (f) shows the bounding in the remaining subset. A new estimate $\Phi(x^*) = y_3^*$ of the minimum value of $\Phi(x)$ is found. The algorithm terminates if $y_3^* - \hat{y} \leq \varepsilon$, and in this case x^* is said to be the ε -suboptimal solution.

subset. The subset (rectangle) in which the cost function is higher than the best optimum so far is eliminated. We continue bounding and branching as long as the difference between the lower bounding function and the cost function is larger than ε .

3.10 Conclusion

We have introduced in this chapter the basic notions of convex optimisation that are used extensively throughout the thesis. An overview on other optimisation subclasses and their implementations are given as well. Details on the convex sets; convex functions; convex and quasi-convex optimisation problems are also given. Special attention is made to the Second Order Cone Program (SOCP), which is used as a solution tool for our optimisation model along with the quasi-convex optimisation formulation. Robust convex optimisation is considered as well in this chapter along with global optimisation methods (branch and bound in particular). For more details about using the the SOCP on some multiple-view geometry problems, the reader may refer to Appendix A (page 275).

Chapter 4

Convex Optimisation and H_∞ Filtering for Motion Estimation

In this chapter a new approach for the monocular visual odometry problem is presented. The L_∞ norm and H_∞ filter are employed alongside convex optimisation. In addition to the H_∞ filter, the use of a recursive least squares (RLS) technique is investigated. Such techniques have the ability to cope with noisy data and to provide optimal solutions [114]. Using monocular systems makes the motion estimation challenging due to the absolute scale ambiguity caused by projective effects. For this, we propose robust tools to estimate both the trajectory of a moving vehicle and the unknown absolute scale ratio between consecutive image pairs. The proposed solution uses as input only images provided by a single camera mounted on the roof of a ground vehicle. Experimental evaluations show that convex optimisation with the L_∞ norm and the robust H_∞ filter clearly outperforms classical methods based on least squares and solved using iterative algorithms.

4.1 Overview

In this chapter we present our first contribution in this research. The proposed motion estimation solution, which is based on the L_∞ norm with convex optimisation for the triangulation problem is detailed. Techniques that are introduced in Chapter 2 and Chapter 3 are used in this solution throughout this chapter. In this solution, we describe the mechanism to estimate the unknown absolute scale between successive pairs of frames using the H_∞ filter and recursive least squares (RLS). In the last section of this chapter, we provide and analyse the obtained results, where comparisons with standard and classical structure-from-motion algorithms for motion estimation are given.

Convex optimisation has attracted many researchers in computer vision in the last decade. In [80], Hartley and Schaffalitzky introduced the L_∞ norm in the cost function for the multi-view triangulation problem and the motion reconstruction problem for omni-directional images. Then, in [94] Kahl presented a convex optimisation technique for multi-view problems such as triangulation, homography estimation, the reconstruction problem and the camera resectioning problem using SOCP. In [101] similar problems were formulated as quasi-convex optimisation problems with known rotations using both SOCP and linear programming (LP). Appendix A (page 275) gives a detailed discussion about these multi-view geometry problems, where experiments on real-image data using convex optimisation are performed.

4.2 L_∞ Norm based solution for visual odometry

In autonomous navigation systems, visual odometry estimates the pose of moving vehicles using visual inputs only from either a single camera (monocular system), stereo cameras or multi camera systems [109, 204]. Visual odometry is a particular case of Structure from Motion (SfM), where the latter tackles the estimation of both the relative camera poses and three-dimensional structure [60, 166]. Visual odometry, on the other hand, emphasises on estimating the motion of the camera sequentially [60, 82, 166]. Optimisation approaches such as Bundle Adjustment (BA) are used to refine the estimates of the trajectory [60]. More robust optimisation techniques such as convex optimisation for motion estimation have began to attract more interest among computer vision researchers as well.

As we have seen in Chapter 2, the camera pose can be recovered from the available corresponding points between two views and the camera calibration parameters. Commonly, the essential matrix estimation is performed through the eight-point algorithm (Algorithm 4, page 298), which is then used for recovering the camera relative pose by solving least squares problems via singular value decomposition (SVD) [82, 109, 204].

In [36], an approach on stereo visual odometry which consists of determining the essential matrix by minimising the algebraic error through convex optimisation is presented. As a consequence of the convexity of its optimisation function, the main advantage of convex optimisation consists in avoiding local minima and approximating non-linear terms when compared to other competitor methods for minimising algebraic cost functions. A successful method is presented in [145] for real time motion estimation of a single camera and a stereo rig. In [130] an algorithm based on minimising the L_∞ norm of the re-projection error is proposed, where the problem is

divided into two successive tasks by fixing the parameters of one sub-problem while optimising the remaining sub-problem.

4.3 Scale ambiguity in monocular systems

A fundamental issue with the monocular visual odometry solution is related to the scale ambiguity due to the projective effects. A measured point $\mathbf{x} = (u, v)^\top$ in a particular image can represent the projection of an infinite number of 3D scene points in the world. As shown in Figure 4.1, the projective nature of monocular systems generates a scale ambiguity in both the 3D scene structure and in motion estimation. One way to estimate the global scale is to use known information about the real world. An INS/IMU and GPS could be used to gather information about the real world, and the camera travelling distance to remove the scale ambiguity. In addition to the global scale ambiguity, the estimated translation vectors from frame-to-frame image points (features) correspondences suffers as well from scale ambiguity. This ambiguity is known in the computer science community as a local scale problem.

Stereo visual odometry takes advantage of the known baseline to directly remove any scale ambiguity. However, it suffers from a substantial lack of accuracy when the distance to the 3D scene points is much larger than the baseline. Therefore, monocular visual odometry becomes an unavoidable solution and ought to be used since the stereo system reduces to the monocular case [166].

4.4 Related work

4.4.1 Scale ambiguity issue

Many algorithms have been successfully implemented to implicitly solve the problem of scale ambiguity. In [46], initially known landmarks are used to build a well scaled map. In [39], initially known landmarks are not required as an un-delayed landmark initialisation technique is used. Nutzi et al. presented in [147] an approach to estimate the unknown absolute scale in a monocular SLAM frames by fusing data from an inertial measurement unit (IMU) and vision system with an extended Kalman filter (EKF). However, this method is tested only on simulated two-meter-cube data which limits its applicability.

Previous methods used information gathered from IMU and GPS or from a known fixed landmark, to recover the scale ambiguity. Others in the literature solved the scale ambiguity implicitly. Real world information could also be used to estimate the global scale factor, as showed in [167], where the camera's height to

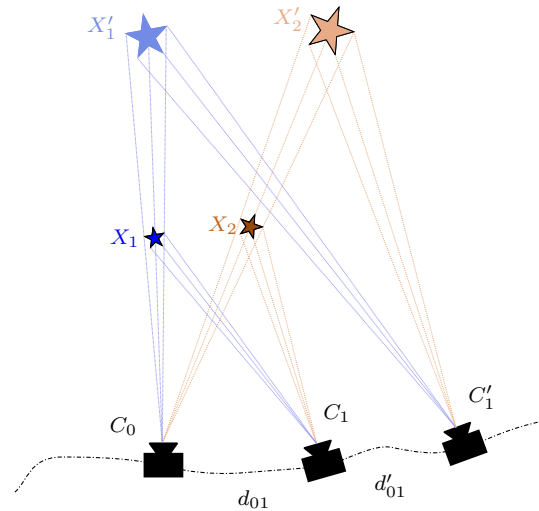


Fig. 4.1 Scale ambiguity problem in monocular vision systems in motion estimation. Pose C_1 could only be estimated up to an unknown scale. The camera at C_0 projects X_1 and X'_1 as a single image point and projects X_2 and X'_2 at another single image point as well. Suppose that the camera has moved to C_1 , then it projects X_1 and X_2 and match them to image points when the camera was in C_0 . However, in pose C'_1 , the camera matches the image points of X'_1 and X'_2 to the same image points in C_0 as it was in C_1 . Therefore, it is impossible to distinguish whether the cameras has moved from C_0 to C_1 or from C_0 to C'_1 .

the ground plane and position with respect to the vehicle axis are used. However, assuming that the camera is moving at a known and fixed height over ground reduces its applicability. Another technique is presented in [168] where the non-holonomic constraints of wheeled vehicles are exploited to estimate the absolute scale. Other techniques estimate the distance travelled by the camera using GPS or other devices such as an odometer to recover the global scale [168]. Although these approaches are used extensively in the literature, they suffer from a number of weaknesses.

One major drawback is the reliance on external devices such as IMU/INS or GPS that are not necessarily accurate. This will allow errors to be propagated to the estimated motion. Correspondences between image points and 3D scene points are used to solve the camera pose, in which the scale is implicitly recovered [29, 48, 133, 199]. However, 2D-3D-correspondence methods suffer from a serious problem of error propagation between triangulation and back projection tasks. For the local scale estimation, explicit methods are deployed in the literature. A practical technique in [145] for real time motion estimation of a single camera or stereo rig is presented. In [54], I. Esteban et al. presented a successful monocular visual odometry algorithm relying on a linear computation of the scale ratio between frame pairs.

4.4.2 Visual odometry

For the visual odometry task, the main goal of the optimisation is to find both the 3D point positions and camera parameters that minimise the re-projection error between the measured image points (features) and the projection of the estimated corresponding 3D scene points [188]. Bundle adjustment is frequently formulated as a non-linear least squares problem in which the 3D point positions are recovered using triangulation, where the cost function is minimised using a numerical optimisation method such as Levenberg-Marquardt(LM) using the L_2 norm [82, 166] . The differentiability of the cost function for this norm, which allows the use of gradient- and Hessian-based optimisation methods, makes this choice reasonable [130].

Gauss-Newton and Levenberg-Marquardt [146] are the most popular algorithms for solving non-linear least squares problems, and considered as the algorithms of choice for bundle adjustment. These algorithms have some nice convergence properties near a local minimum. However, apart from their computational complexity and memory requirement, the main issue with these methods is the cost functions dealt with, which are highly non-linear and non-convex. Therefore they have many local minima and guaranteeing to end up in the global minimum is very difficult.

On the other hand, the L_∞ norm optimisation, contrary to iterative L_2 based optimisation methods using Levenberg-Marquardt, ensures the global optimum of the error function [25]. Indeed this function, which is geometrically meaningful, is of quasi-convex type that can be efficiently minimised by the bisection method as a sequence of second order cone programs (SOCP) feasibility problems (Section 3.5, page 64) [80, 172].

4.5 The Proposed method

In this chapter we propose to use more robust norms such as the L_∞ norm. This norm deals with minimising the worst-case error or min-max estimation. Therefore, from the perspective of getting more accurate 3D point positions, our method uses convex optimisation for the triangulation task based on the L_∞ norm formulation. Image re-projection errors can be efficiently minimised, which leads to a global minimum. The only drawback of using the L_∞ norm is its sensitivity to outliers which are critical as well to the ordinary L_2 -norm based algorithms.

Our scenario deals with a calibrated camera taking a sequence of images as it moves along an unknown trajectory. The proposed motion estimation algorithm accepts as inputs the intrinsic calibration parameters and a sequence of grey-scale undistorted images, including the previous and the current views. Harris feature

extraction is performed on each new frame (Section C.2.1, page 299). For each pair of consecutive images, correspondences between image points are recovered using a maximum correlation technique followed by an outlier-rejection scheme using the RANSAC algorithm (Algorithm 2, page 296). Then the relative frame-to-frame motion is estimated using convex optimisation. These relative motion estimates are accumulated, resulting in a 6 DOF trajectory.

The navigation solution proposed in this work can be summarised in the following steps (Figure 4.2):

- Computing image point tracks (using Harris detector [79] and RANSAC [58]).
- Estimating the essential matrix using the 8-point algorithm (Algorithm 4, page 298) [82].
- Estimating relative rotations R_i and the translation T_i .
- Optimising the estimated parameters using L_∞ norm-based bundle adjustment with convex optimisation for the triangulation problem.
- Computing the unknown absolute scale ratio using robust H_∞ filter and recursive least squares algorithm (RLS) for motion estimation [54].

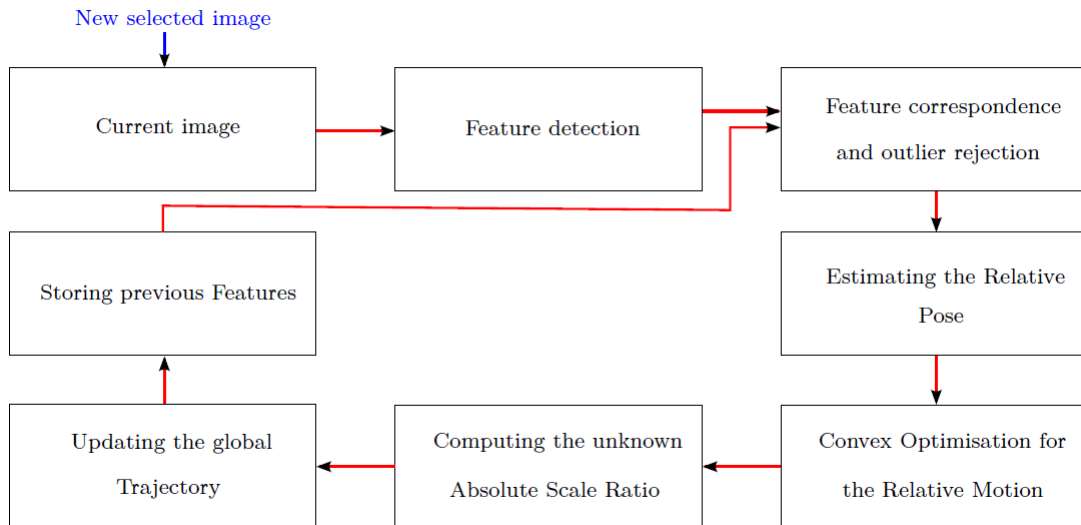


Fig. 4.2 Illustration of the motion estimation pipeline.

4.6 The L_∞ motion estimation

Triangulation is a critical part for the subsequent motion estimation. Therefore, an efficient and optimal algorithm for estimating its parameters is essential. Thus, our approach is based on convex optimisation with the L_∞ Norm.

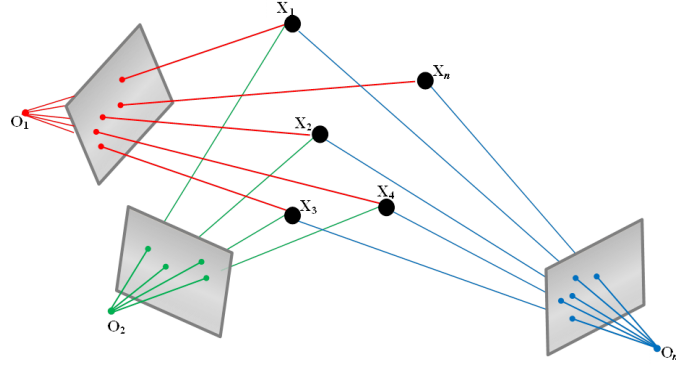


Fig. 4.3 The triangulation problem. Each point is seen from two or more cameras. The goal is to estimate the position of the 3D scene points X_j .

The geometry of a multiple views system is depicted in Figure 4.3, where the cameras are modelled using the pinhole model (Section 2.5.1, page 27) [82]. Assume we have m views of a scene point $\hat{X}_i = [X^\top, 1]^\top$, where \hat{X} is a vector represented by homogeneous coordinates and $X = [X, Y, Z]^\top \in \mathbb{R}^3$. This scene point is mapped to image points $\hat{x}_i = [u_i, v_i, 1]^\top$ via camera matrices P_i . The triangulation problem is detailed in Section A.1 (Appendix A, page 275), presenting its solution within the convex optimisation framework. In general, this problem is recovering the 3D space position of points \hat{X} such that $\hat{x}_i = P_i \hat{X}$, for $i = 1, \dots, m$. The optimisation variable is then $x = (X, Y, Z)^\top \in \mathbb{R}^3$. Since the motion parameters are estimated from image-to-image correspondences, these quantities are related by the projection equations:

$$u_i = \frac{p_i^1 \hat{X}}{p_i^3 \hat{X}} \quad ; \quad v_i = \frac{p_i^2 \hat{X}}{p_i^3 \hat{X}} \quad (4.1)$$

where p_i^j denotes the j^{th} row vector of the 3×4 camera matrix P_i . Then, for a given image point i , the error residual may be rewritten as:

$$\varepsilon_i = d(\hat{x}_i, P_i \hat{X}) \quad (4.2)$$

where $d(\cdot)$ denotes image-space Euclidean distance between two points in the image plane, the measured and the projected points. Our aim is then to recover the value of \hat{X} that minimises the maximum of this re-projection error across the two images.

Given:

- the camera matrices $\tilde{P}_i = K_i^{-1} P_i = [R_i | t_i]$, where $R_i \in RO(3)$ is the rotation matrix, $t_i \in \mathbb{R}^3$ is the translation vector, and K_i is the camera calibration matrix. Let the rotation matrix be $R_i = [r_{i1}, r_{i2}, r_{i3}]$, and the translation vector t_i be $t_i = [t_{i1}, t_{i2}, t_{i3}]^\top$,

- $\hat{X} = [X, 1]$ and their corresponding normalised images $\tilde{x}_i = K_i^{-1}x = [\tilde{u}_i, \tilde{v}_i]$. Therefore, the residual error ε_i on the i^{th} image can be written as:

$$\begin{aligned}
\varepsilon_i &= d(\hat{x}_i, P_i \hat{X}) \\
&= \sqrt{\left(\tilde{u}_i - \frac{r_{i1}^\top X + t_{i1}}{r_{i3}^\top X + t_{i3}}\right)^2 + \left(\tilde{v}_i - \frac{r_{i2}^\top X + t_{i2}}{r_{i3}^\top X + t_{i3}}\right)^2} \\
&= \sqrt{\frac{(\tilde{u}_i(r_{i3}^\top X + t_{i3}) - r_{i1}^\top X + t_{i1})^2 + (\tilde{v}_i(r_{i3}^\top X + t_{i3}) - r_{i2}^\top X + t_{i2})^2}{(r_{i3}^\top X + t_{i3})^2}} \quad (4.3) \\
&= \sqrt{\frac{f_{i1}(x)^2 + f_{i2}(x)^2}{f_{i3}(x)^2}}
\end{aligned}$$

The point \hat{X} is seen from m cameras, this gives m error residuals: $\varepsilon = (\varepsilon_1, \varepsilon_2, \dots, \varepsilon_m)^\top$. Therefore, the optimal 3D scene point \hat{X} is the one that minimises the norm of the error residuals vector: $\|\varepsilon\|$.

Most classical methods use the L_2 norm for the cost function $\|\varepsilon\|_2$ or, equivalently, $\|\varepsilon\|_2^2 = \sum_{i=1}^m \varepsilon_i^2 = \sum_{i=1}^m d(\hat{x}_i, P_i \hat{X})^2$. This optimisation problem is shown in the literature to have multiple local minima [80, 82, 94, 101]. Therefore, iterative methods can easily get trapped in one of these local minima instead of ending up in the global minimum.

To get around this problem, the L_∞ norm is used instead. Optimising the L_∞ norm of ε leads to the cost function: $\|\varepsilon\|_\infty = \max_i |\varepsilon_i| = \max_i |d(\hat{x}_i, P_i \hat{X})|$. The goal of the optimisation problem now is to minimise the maximum error between the projected points and the measured image points, hence:

$$\begin{aligned}
\min_x \quad & f_0(x) = \max_{i=1, \dots, m} f_i(x) = \max_{i=1, \dots, m} \left(\frac{f_{i1}(x)^2 + f_{i2}(x)^2}{f_{i3}(x)^2} \right) \quad (4.4) \\
\text{subject to} \quad & f_{i3}(x) > 0, \quad \text{for } i = 1, \dots, m.
\end{aligned}$$

Note that f_{i1}, f_{i2}, f_{i3} are all affine functions of the optimisation variable x . According to Theorem 8 (Chapter 3, page 70), $f_i(x) = \frac{f_{i1}(x)^2 + f_{i2}(x)^2}{f_{i3}(x)^2}$ is quasi-conex function. In addition, according to Theorem 7 (Chapter 3, page 61), the function $f_0 = \max_{i=1, \dots, m} \left(\frac{f_{i1}(x)^2 + f_{i2}(x)^2}{f_{i3}(x)^2} \right)$ is also quasi-convex since the pointwise maximum is also convex. The constraint function has a convex domain $\{x | f_{i3}(x) > 0, \text{ for } i = 1, \dots, m\}$, therefore problem (4.4) is a quasi-convex optimisation problem [94, 101].

Let δ be the upper bound of the optimisation function in (4.4), then each projection defines a conical surface where the bound δ is the radius of this cone and the camera centre is its apex. It is noteworthy here that each image measurement adds one conical constraint to (4.4).

This problem is solved using the technique introduced in Section A.1 (Appendix A, page 275) via the bisection algorithm with a sequence of second order cone programs (SOCP) feasibility problems. The 3D points recovered using convex optimisation for triangulation are then used in the optimisation algorithm for recovering the camera motion. Thus, given the efficiently estimated 3D point positions and their corresponding measured image locations, the goal of bundle adjustment is then to exclusively find the camera parameters that minimise the re-projection error.

4.7 Robust solution for scale estimation

After having robustly estimated the camera motion using L_∞ norm-based bundle adjustment, ambiguities in the translation scale can nevertheless still occur. Unlike in the stereo scheme, the monocular visual odometry estimates both the relative motion and the 3D structure up to an unknown scale. This absolute scale cannot be estimated unless information about the real world is provided.

Inspired of the work in [54], as we have seen in Section 2.6 (Chapter 2, page 30), we assume that m 3D scene points $\hat{X}_i = [X^\top, 1]$ are mapped into image points $\hat{x}_i = [u_i, v_i, 1]^\top$ via the normalised camera matrix $P = [R|t]$ then [46, 82]:

$$\lambda \hat{x}_i = P \hat{X}_i \quad (4.5)$$

where λ is the unknown depth factor that takes into account the projection plane ambiguity. This parameter should be chosen to obtain image points and not image plane points [46, 82]. Hence:

$$\lambda \hat{x}_i = [R|t] \hat{X}_i \quad (4.6)$$

$$\lambda \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \hat{X}_i \quad (4.7)$$

By introducing the unknown scale S we get:

$$\lambda \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & S t_x \\ r_{21} & r_{22} & r_{23} & S t_y \\ r_{31} & r_{32} & r_{33} & S t_z \end{bmatrix} \hat{X}_i \quad (4.8)$$

Hence:

$$\lambda u_i = r^1 X_i + S t_x \quad (4.9)$$

$$\lambda v_i = r^2 X_i + S t_y \quad (4.10)$$

$$\lambda = r^3 X_i + S t_z \quad (4.11)$$

where r^i denotes the i^{th} row vector of R_i and $\hat{X}_i = [X_i, 1]$. By substituting (4.11) in (4.9) to remove the image scale factor λ , we get:

$$(r^3 X_i + S t_z) u_i = r^1 X_i + S t_x \quad (4.12)$$

$$(t_z u_i - t_x) S = (r^1 - r^3 u_i) X_i \quad (4.13)$$

$$\mathbf{A} S = \mathbf{b} \quad (4.14)$$

where $\mathbf{A} = (t_z u_i - t_x)$ and $\mathbf{b} = (r^1 - r^3 u_i) X_i$. In [54], this problem is solved in the least squares sense by minimising $\| \mathbf{A} S - \mathbf{b} \|^2$. Here we propose two new methods to solve this scaling estimation problem. These methods are the recursive least squares (RLS) and the more robust H_∞ filter.

To formulate the unknown scale estimation problem in a mathematical way, suppose x is the unknown constant and y is a vector containing noisy measurements. Our aim is to find the best estimation \hat{x} of x . Measurements y must be related to the unknown vector x coupled the accumulation of some measurement noise; hence, we can write [178]:

$$y = Hx + v \quad (4.15)$$

where H is a known observation matrix and v is some additional noise. This matrix relates the unknown vector x to the measurements y , and the difference is compensated with the error vector v . Thus, for a given estimate \hat{x} , the corresponding error is given by:

$$\varepsilon = y - H\hat{x} \quad (4.16)$$

This error is the difference between the vector $H\hat{x}$ and the noisy measurements y . Solving this problem in a least squares form leads us to minimise the sum of the squared errors between the elements of the measurement vector and those of the vector given by $H\hat{x}$. In the least squares algorithm, the measurement noise is assumed to be zero-mean and white with known variance.

Suppose we have k measurements, then, the cost function to be minimised is given by:

$$\begin{aligned}
 J &= \varepsilon_1^2 + \dots + \varepsilon_k^2 \\
 J &= \varepsilon^\top \varepsilon \\
 J &= (y - H\hat{x})^\top (y - H\hat{x}) \\
 J &= y^\top y - \hat{x}^\top H^\top y - y^\top H\hat{x} + \hat{x}^\top H^\top H\hat{x}
 \end{aligned} \tag{4.17}$$

Minimising J with respect to \hat{x} means computing its partial derivative and setting it equal to zero:

$$\frac{\partial J}{\partial \hat{x}} = -y^\top H - y^\top H + 2\hat{x}^\top H^\top H = 0 \tag{4.18}$$

$$\hat{x} = (H^\top H)^{-1} H^\top y \tag{4.19}$$

$$\hat{x} = H^\dagger y \tag{4.20}$$

Therefore, the best estimate of \hat{x} in least squares form is given by (4.19). Note that H^\dagger (4.20) is the pseudo inverse of the matrix H . Thus, H must be of a full rank, i.e. simply the number of measurements k has to be greater than the number of variables n . Note that when H is equal to 1, this solution simply computes the average of all measurements.

4.7.1 Recursive least squares method

We propose to find the optimal value of \hat{x} in a recursive manner. In this section, we show how to recursively compute the least squares estimate of the absolute scale. Using the recursive least squares (RLS) algorithm, which is based on continuously estimating \hat{x} while relying on its previous estimate. Figure 4.4 depicts the main steps of this algorithm. After estimating \hat{x} from $(k-1)$ measurements, the challenge is how to update our estimate \hat{x} when a new measurement y_k is obtained without rebuilding and solving the whole system given in (4.19).

First, let us consider this linear estimator:

$$y_k = H_k x + v_k \tag{4.21}$$

$$\hat{x}_k = \hat{x}_{k-1} + K_k (y_k - H_k \hat{x}_{k-1}) \tag{4.22}$$

We can see from (4.22) that the new estimate \hat{x}_k is computed from the previous estimate \hat{x}_{k-1} and from the new measurement y_k . The term $(y_k - H_k \hat{x}_{k-1})$ is called the corrector term. Note that K_k is the estimator gain matrix, which is obtained via

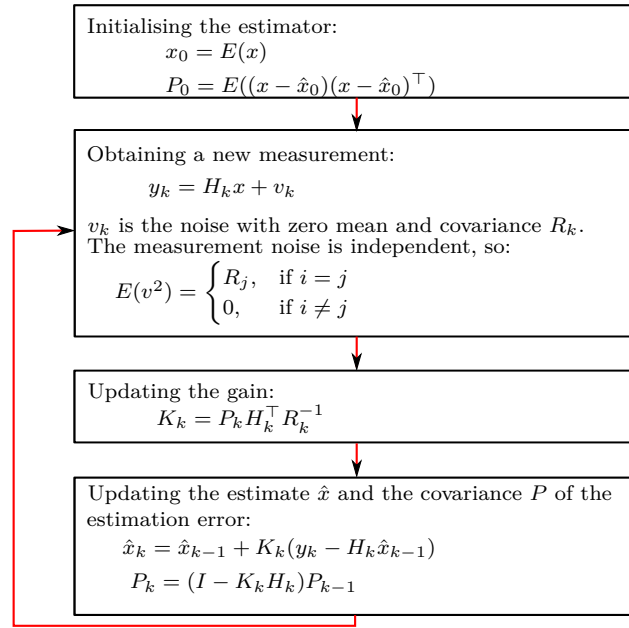


Fig. 4.4 Algorithm for recursive least squares estimation.

[178]:

$$K_k = P_k H_k^\top R_k^{-1} \quad (4.23)$$

Here R_k is the covariance of the noise vector v_k and P_k is the estimation error covariance given by:

$$P_k = (I - K_k H_k) P_{k-1} \quad (4.24)$$

Equations (4.21) to (4.24) form our recursive least squares algorithm (Figure 4.4). The initial value of P_0 depends on our knowledge of \hat{x} . The better our knowledge of \hat{x} is, the smaller the value for P_0 will be. In other words, in case of no prior knowledge of \hat{x} is available, we can assume $P_0 = \infty I$. In the case of perfect prior knowledge, we put $P_0 = 0$.

4.7.2 H_∞ Filter method

The least squares algorithm assumes a measurement with white noise, zero-mean and known variance. Unfortunately, these assumptions are not always valid. Therefore, using more robust estimators would be a reasonable option to investigate. The first estimator that one might think of is the Kalman filter due to its efficiency in estimating a system state.

The Kalman filter took its name from its inventor Rudolf E. Kalman [183]. This filter is mostly suitable for linear systems with Gaussian process and white observation noise. This provides analytical solutions to the Bayesian prediction and update step. The Kalman filter is famous for its success in aerospace applications in the 1960s. This success urged industrial applications to deploy it as well. However, a quick mismatch between the filter assumptions and their problems is observed [178]. This mismatch is induced by the lack of knowledge of the nature of the process noise in industrial applications in comparison to aerospace applications. These problems have stimulated new research to develop filters that are able to cope with modelling errors and system uncertainties. These new filters are called robust since they can handle such uncertainties. The H_∞ filter is one of the newer filters that were designed for robustness. We use this filter to estimate the optimal absolute scale in the presence of noisy measurements of an unknown nature.

Before developing our H_∞ filter, an overview on the Kalman filter seems to be necessary as the structure of both filters is similar. A linear dynamic system is given by:

$$x_{k+1} = F_k x_k + w_k \quad (4.25)$$

$$y_k = H_k x_k + v_k \quad (4.26)$$

here w_k and v_k are stochastic process noise terms with covariances Q_k and R_k respectively. The Kalman filter's main goal is to estimate the state of this linear system. In this model, x_k is the state vector and F_k is the transition matrix. In fact, the process noise w_k is injected into the linear system because of the uncertainties in the transition matrix F_k . This process noise is supposed to be uncorrelated with zero-mean, with covariance:

$$E(w_k) = 0 \quad \forall k$$

$$E(w_k w_j^\top) = \begin{cases} Q_k, & \text{if } k = j \\ 0, & \text{if } k \neq j \end{cases} \quad (4.27)$$

In the observation model given in (4.26), H_k relates the state vector at time k to the observation vector. The observation noise v_k is assumed to be zero-mean, uncorrelated random sequence, where:

$$E(v_k) = 0 \quad \forall k$$

$$E(v_k v_j^\top) = \begin{cases} R_k, & \text{if } k = j \\ 0, & \text{if } k \neq j \end{cases} \quad (4.28)$$

More importantly, the process and the observation noise are assumed to be uncorrelated:

$$E(w_k v_j^\top) = 0 \quad \forall k, j \quad (4.29)$$

Kalman filter estimates the state x_k based on the known system dynamics and the availability of the noisy measurements y_k . The Kalman filter's equations are given as follow:

$$\begin{aligned} \hat{x}_{k+1}^- &= F_k \hat{x}_k^- + F_k K_k (y_k - H_k \hat{x}_k^-) \\ K_k &= P_k^- F_k (I + H_k^\top R_k^{-1} H_k P_k^-)^{-1} H_k^\top R_k^{-1} \\ P_{k+1}^- &= F_k P_k^- (I + H_k^\top R_k^{-1} H_k P_k^-)^{-1} F_k^\top + Q_k \end{aligned} \quad (4.30)$$

The available measurements y_k up to time k are used in the estimation of x_k to form what is called the a-posteriori estimate, denoted by \hat{x}_k^+ . If we have available measurements before time k , then we can form an a-priori estimate denoted by \hat{x}_k^- . Therefore, \hat{x}_k^- is the estimate of \hat{x}_k before the measurement y_k is taken into account, and \hat{x}_k^+ is our estimate of \hat{x}_k , after the measurement y_k is taken into account [178].

These procedures perform well, but only under certain conditions [178]:

- Firstly, the mean and the correlation of w_k and v_k need to be accurately known at each time step.
- Secondly, the covariances Q_k and R_k need to be known as well.
- Thirdly, the system matrices F_k and H_k need to be perfectly known.
- Finally, and most importantly, the cost function needs to be minimised over the least potential standard deviation. Hence, the smallest variance estimator is given if the noise is Gaussian. If the noise is not Gaussian, a linear minimum variance estimator is provided. This fact signifies that minimising different cost functions does not guarantee optimal estimates.

The problem is: what happens if one of these conditions is not satisfied? And what should we do in cases where no information is available about the noise? Moreover, what should we do if we want to minimise different cost functions such as the worst-case estimation error?

Applying the Kalman filter without satisfying of all these conditions will certainly result in a less accurate estimation or sub-optimal solutions. Satisfying all these conditions in our case is difficult, since the image-based measurements are subject to unknown perturbations with an unknown distribution law. Moreover, in most vision applications the noise statistics are not identified.

To get around the problems that the Kalman filter and other equivalent filters have, one might consider using robust filters. The H_∞ filter is based on a min-max

estimation method, leading to an important difference with the Kalman filter: the former is optimal in terms of minimising the L_∞ -norm between ranges of disturbances [178]. Therefore, the H_∞ filter is a more robust alternative to the Kalman filter. The key feature of the H_∞ filter is the fact that no assumptions about the noise are required. Moreover, it minimises the worst-case estimation error.

Let us consider the H_∞ based system as filter model as in (4.25) and (4.26):

$$x_{k+1} = F_k x_k + w_k \quad (4.31)$$

$$y_k = H_k x_k + v_k \quad (4.32)$$

where w_k and v_k are the noise terms with an unknown distribution law. They may have a non-zero mean as well. Under these circumstances, we aim to estimate a linear combination of the system states given by:

$$z_k = L_k x_k \quad (4.33)$$

where L_k is a user defined matrix. Note that we can set $L = I$ in the case where x_k is estimated directly as in the Kalman filter. Now, at time step $(N - 1)$, our aim is to estimate z_k based on all previous measurements.

The estimate \hat{z}_k is found after minimising the cost function J_1 as $J_1 < \frac{1}{\theta}$, where θ is the performance bound specified by the designer and J_1 is defined as [178]:

$$J_1 = \frac{\sum_{k=0}^{N-1} \|z_k - \hat{z}_k\|_{S_k}^2}{\|x_0 - \hat{x}_0\|_{P_0}^2 + \sum_{k=0}^{N-1} (\|w_k\|_{Q_k}^2 + \|v_k\|_{R_k}^2)} \quad (4.34)$$

where P_0 , Q_k , R_k and S_k are chosen matrices with the condition of being symmetric positive-definite. Re-organising this equation gives:

$$J = -\frac{1}{\theta} \|x_0 - \hat{x}_0\|_{P_0}^2 + \sum_{k=0}^{N-1} \left[\|z_k - \hat{z}_k\|_{S_k}^2 - \frac{1}{\theta} (\|w_k\|_{Q_k}^2 + \|v_k\|_{R_k}^2) \right] < 1 \quad (4.35)$$

Hence, the min-max problem is defined as:

$$J^* = \min_{\hat{z}_k} \max_{w_k, v_k, x_0} J \quad (4.36)$$

The worst case is obtained when w_k , v_k and x_0 are chosen to maximise J . The solution then is to find an estimate \hat{z}_k , which minimises this maximum. Since $y_k = H_k x_k + v_k$, hence $v_k = y_k - H_k x_k$, then the norm of the noise $\|v_k\|_{R_k}^2$ will be:

$$\|v_k\|_{R_k}^2 = \|y_k - H_k x_k\|_{R_k}^2 \quad (4.37)$$

Since $z_k = L_k x_k$ and $\hat{z}_k = L_k \hat{x}_k$, then:

$$\begin{aligned} \|z_k - \hat{z}_k\|_{S_k}^2 &= (z_k - \hat{z}_k)^\top S_k (z_k - \hat{z}_k) \\ &= (x_k - \hat{x}_k)^\top L_k^\top S_k L_k (x_k - \hat{x}_k) \\ &= \|x_k - \hat{x}_k\|_{\bar{S}_k}^2 \end{aligned} \quad (4.38)$$

where:

$$\bar{S}_k = L_k^\top S_k L_k \quad (4.39)$$

We can then substitute the results in (4.35) to get:

$$\begin{aligned} J &= -\frac{1}{\theta} \|x_0 - \hat{x}_0\|_{P_0^{-1}}^2 + \sum_{k=0}^{N-1} \left[\|x_k - \hat{x}_k\|_{\bar{S}_k}^2 - \frac{1}{\theta} \left(\|w_k\|_{Q_k^{-1}}^2 + \|y_k - H_k x_k\|_{R_k^{-1}}^2 \right) \right] \\ J = \psi(x_0) &+ \sum_{k=0}^{N-1} \mathcal{L}_k \end{aligned} \quad (4.40)$$

This leads to the filter description below [178]:

$$\begin{aligned} \bar{S}_k &= L_k^\top S_k L_k \\ K_k &= P_k [I - \theta S_k P_k + H_k^\top R_k^{-1} H_k P_k]^{-1} H_k^\top R_k^{-1} \\ \hat{x}_{k+1} &= F_k \hat{x}_k + F_k K_k (y_k - H_k \hat{x}_{k-1}) \\ P_{k+1} &= F_k P_k [I - \theta S_k P_k + H_k^\top R_k^{-1} H_k P_k]^{-1} F_k^\top + Q_k \end{aligned} \quad (4.41)$$

In order for this estimator to be a solution to the problem 4.40, this condition must hold at each time step k :

$$P_k^{-1} - \theta S_k + H_k^\top R_k^{-1} H_k > 0 \quad (4.42)$$

Note that the output and the input of our system are gathered from the vectors \mathbf{A} and \mathbf{b} in (4.14), which are constructed using 3D scene points and their measured corresponding image point projections.

Using the observed data in \mathbf{A} and \mathbf{b} , errors are calculated as follows:

$$e_k = y_k - H x_k \quad (4.43)$$

where H_k and y_k are always collected from the vectors \mathbf{A} and \mathbf{b} respectively. Matrix \mathbf{A} and vector \mathbf{b} in (4.14) can be constructed in four different ways to solve the system for S [54]. Choosing the best method depends on the reference system used and

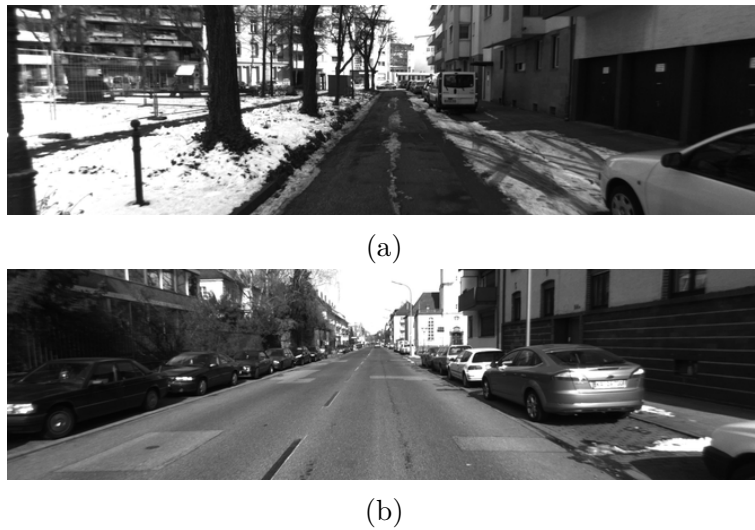


Fig. 4.5 Examples of images used in the experiments [68].

where most of the motion occurs: whether in the x -axis direction, the y -axis direction or a combination of both.

4.8 Experimental evaluation

This section presents an experimental evaluation of the proposed method for motion estimation using convex optimisation coupled to the recursive least squares and the H_∞ filter. A comparison with the classical bundle adjustment based on the LM algorithm is given. A further comparison, estimating the absolute scale, is made between the proposed methods based on the H_∞ filter and the recursive least squares (RLS) and the previous method using the batch least squares.

Real data from urban environments are used to validate the proposed solution. The data are collected via a vision system mounted on a vehicle travelling in the city of Karlsruhe [68], where a forward-pointing calibrated camera is mounted on the roof of the vehicle. These sequences consist of high quality images, with a resolution of 1344×372 pixels, for travelled distances up to 460 metres. Ground truth is provided using an OXTS RT3003 inertial and GPS navigation system, 6 axis, 100 Hz, L1/L2 RTK, resolution: 0.02m / 0.1 degrees. Figure 4.5 shows a sample of images from this dataset (Section 1.6, Chapter 1, page 8). In our experiment we used SeDuMi toolbox for optimisation in SOCP problems [187].

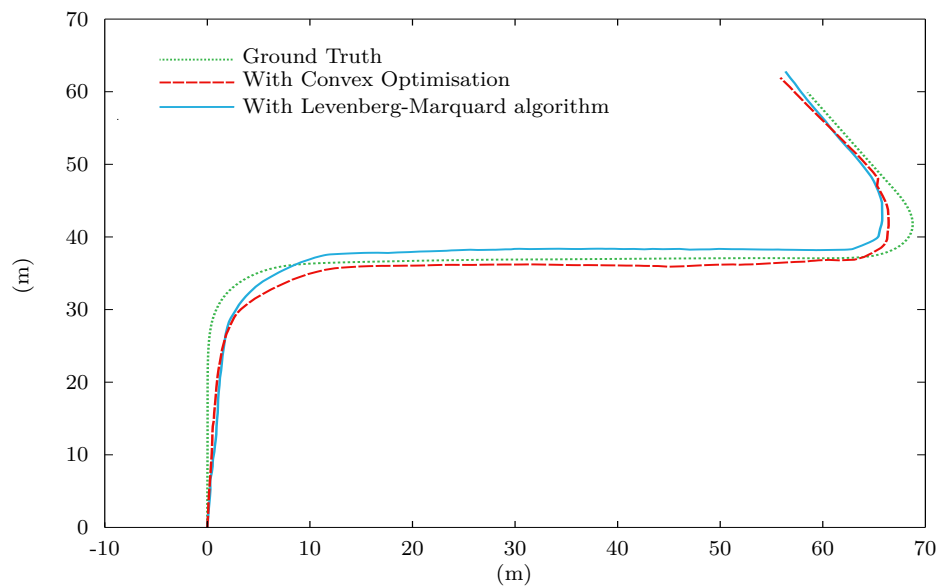


Fig. 4.6 Comparison between trajectory estimates using convex optimisation and the Levenberg-Marquardt algorithm for triangulation.



Fig. 4.7 Illustration of feature matching on sample of images used in motion estimation.

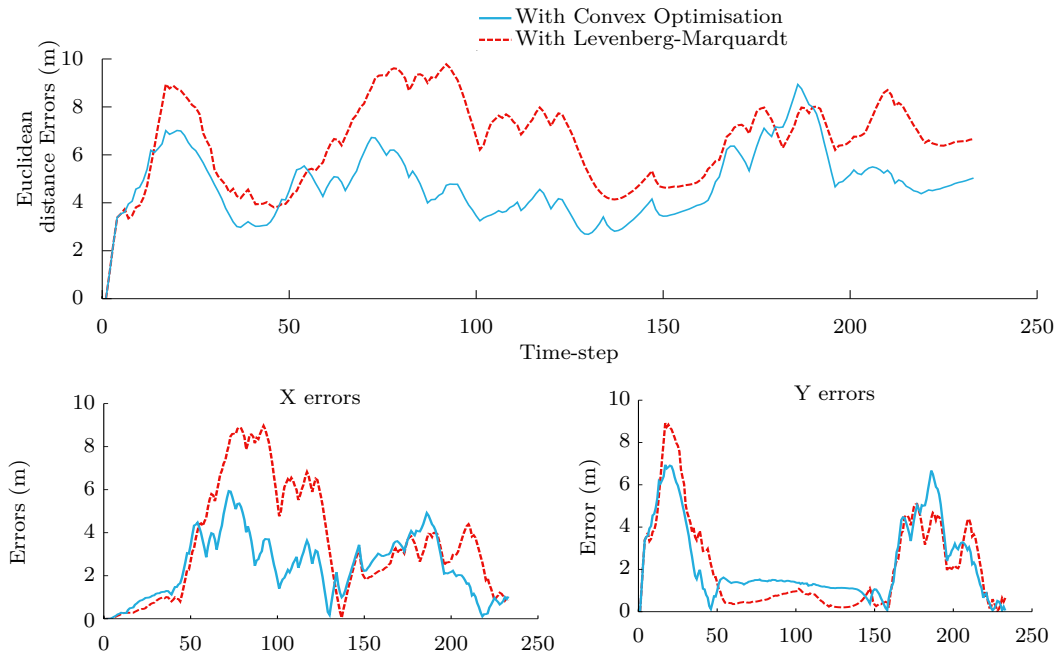


Fig. 4.8 Camera motion estimation errors.

4.8.1 Motion estimation

Our first aim is to investigate the robustness of the proposed technique on real data and in the presence of high levels of noise. Figure 4.6 shows the trajectory estimates obtained with the convex optimisation approach and those obtained using the LM approach superimposed on the ground truth data. Figure 4.7 illustrates feature matching on a sample of images used in motion estimation. Euclidean distance errors between these series and reference data are shown in Figure 4.8.

Achieved results are motivating, reaching errors smaller than 2% and normally bounded by 4 – 9% in terms of travelled error, defined as:

$$\text{Travelled error} = 100 \frac{\text{abs(error)}}{\text{Travelled distance}} \quad (4.44)$$

Several experiments were performed on different datasets in [68]. Table 4.1 summarises the RMS, the minimum and the maximum errors from different experiments. The corresponding plot of these figures is illustrated on Figure 4.9

It is clearly seen that convex optimisation is able to more accurately estimate the motion than traditional methods based on the LM algorithm. This is in conformity with theory in terms of global optimality. By analysing these results, we can establish that convex optimisation with the min-max error scheme performs better over the the experimental set. Equally, this shows that our algorithm is suited for estimating

Table 4.1 RMS, maximum and minimum errors for the motion estimation using convex optimisation (CVX) and Levenberg-Marquardt (LM) algorithm

Trajectory		1	2	3	4	5	6	7	8	9	10	11
Travelled distance [m]		431.06	271.20	462.28	213.65	225.22	309.74	92.97	412.3	184.94	62.69	234.46
RMS [m]	CVX	5.32	2.52	4.95	1.75	4.56	5.36	1.21	3.48	2.03	1.86	1.48
	LM	6.06	3.31	4.85	3.25	4.29	6.02	1.89	4.23	2.64	2.23	2.59
Min [m]	CVX	1.23	1.02	1.59	0.86	1.13	1.28	0.78	2.03	1.03	0.56	1.85
	LM	2.03	1.03	1.56	1.26	1.09	1.19	1.23	3.04	1.65	0.86	2.32
Max [m]	CVX	8.86	5.26	8.45	4.85	3.89	5.86	4.33	5.36	3.25	3.25	6.25
	LM	7.96	7.25	9.25	5.06	4.82	5.92	5.05	6.35	4.95	2.89	6.85

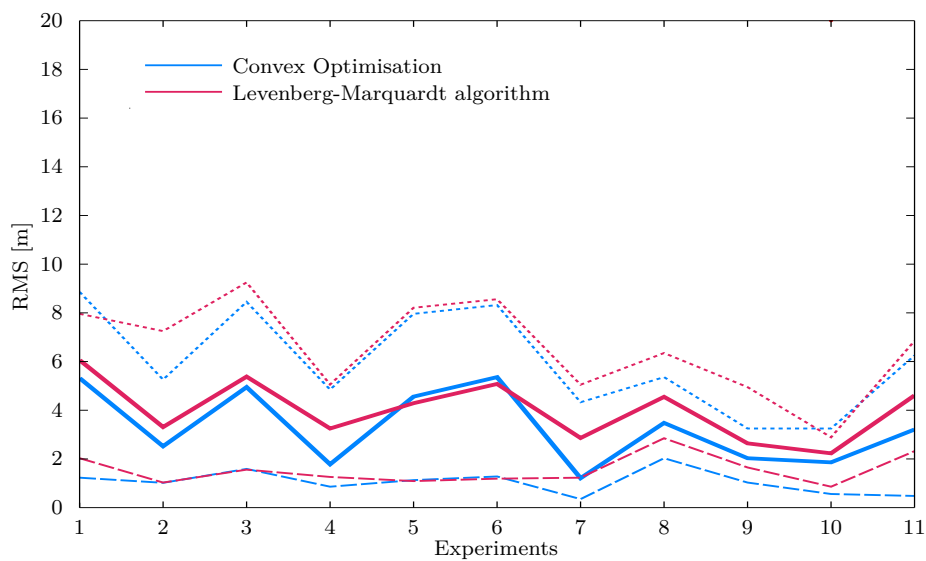


Fig. 4.9 RMS, maximum and minimum errors for the motion estimation using convex optimisation (CVX) and Levenberg-Marquardt algorithm

the motion of a vehicle travelling in an urban environment, where high levels of noise of unknown type are likely to persist. A clearer comparison is shown in Figure 4.8. Indeed, the Euclidean distance errors of camera motion estimates demonstrate better accuracy of results with the convex optimisation approach.

4.8.2 Absolute scale estimation

We now show the capacity of the H_∞ filter and the recursive least squares algorithm to estimate the frame-to-frame absolute scale in the presence of high level of noise. After having estimated the scale free motion, we apply our two techniques on the motion estimates. Figure 4.10 compares the trajectory estimates obtained using the three methods. The first method is linear, using a batch least squares algorithm, as

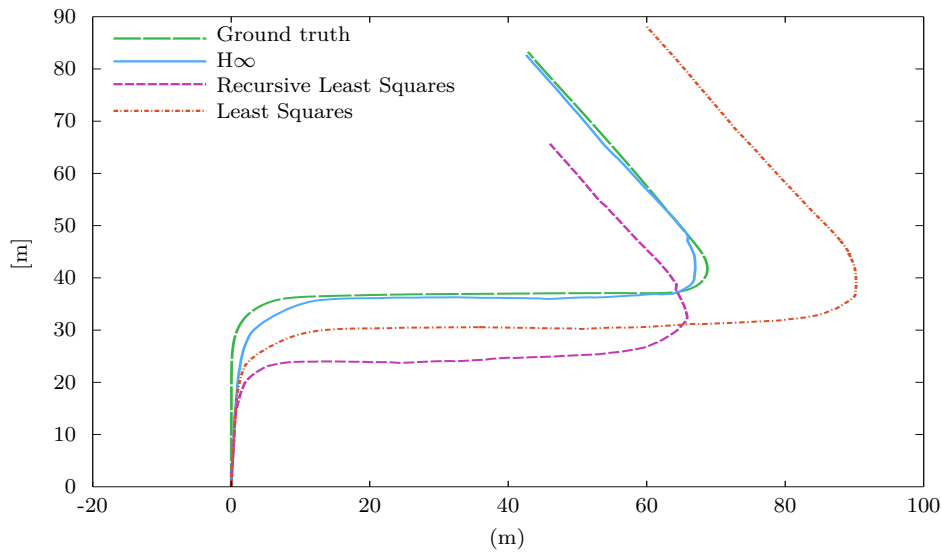


Fig. 4.10 Motion estimation after convex optimisation and absolute scale computation using three different methods: H_∞ , recursive least squares and Least squares. Trajectory estimates are superimposed with the ground truth data.

described in [54, 82] and the two others are using recursive schemes: the H_∞ filter and the recursive least square (RLS).

The obtained results show that recursive methods perform better in comparison to the batch least squares method. Additionally, Figure 4.11 compares camera motion estimation errors at each successive frame pair. These represent the Euclidean distance errors, as well as errors over the x -axis and y -axis. The results show that the H_∞ filter remarkably outperforms the recursive least squares algorithm as well. This is likely due to its tendency towards minimising the worst case. These results justify our final choice for the global solution, which consists of using convex optimisation for motion estimation followed with an H_∞ filter for absolute scale estimation.

4.9 Conclusions

We have presented in this chapter two novel contributions to efficiently solve the monocular visual odometry problem. The first of these pertains to using convex optimisation with the L_∞ norm in motion estimation for the triangulation problem. Our second contribution, which follows on nicely from the first one, is to use an H_∞ filter capable of dealing with system noise for frame-to-frame absolute scale estimation. Although solutions to the motion estimation problem based on least squares are liable to offer good results, they also impose limitations that a solution based on the L_∞ norm is capable to overcome. Through several experiments, the

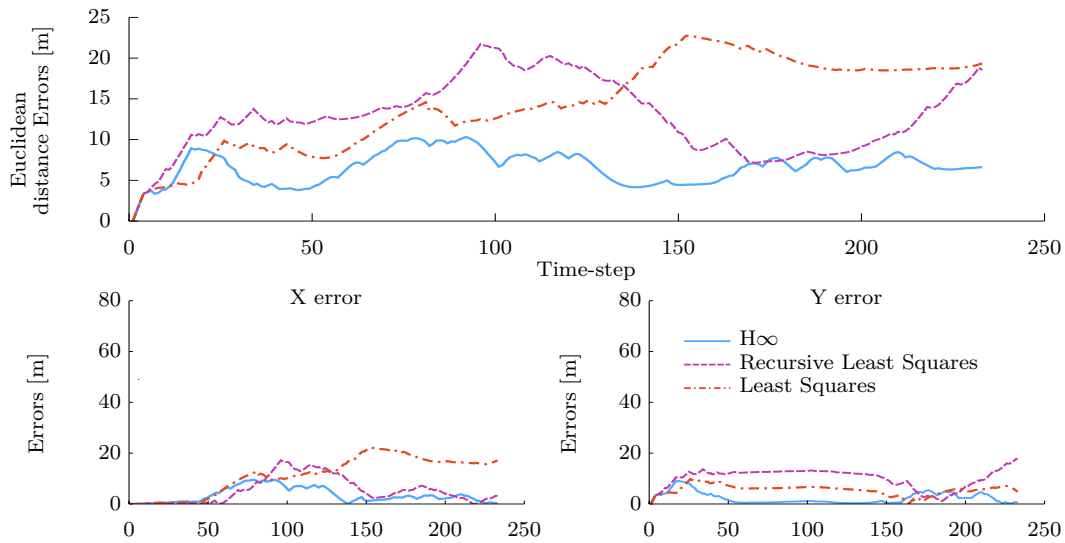


Fig. 4.11 Motion estimation errors after convex optimisation and absolute scale computation using three different methods: H_∞ , recursive least squares and Least squares.

results confirm that the proposed technique clearly outperforms classical techniques, which use the bundle adjustment based on the Levenberg-Marquardt algorithm for motion and the least squares algorithm for absolute scale estimation.

In the next chapter, we will introduce a solution that improves the accuracy and the robustness of the fundamental matrix, and consequently the motion estimate of a ground vehicle equipped with a monocular visual system. This solution takes into consideration the feature position uncertainties in each RGB channel of colour images for the optimisation problem.

Chapter 5

Robust Motion Estimation Using Covariance Intersection

In the present chapter, a novel technique for robust motion estimation is presented. In this technique, we use information from all three RGB channels of colour images. The novelty of our approach consist of fusing feature localisation errors to robustly estimate the motion of a camera via the fundamental matrix. Robust features extraction, matching and tracking are crucial for many applications in computer vision systems. The feature location accuracy is dependent to the variation in intensity within their neighbourhoods, from which their uncertainties are estimated. Our approach tries to improve the accuracy and the robustness of the fundamental matrix, and consequently the motion estimate by considering these uncertainties.

In our solution, rather than converting colour images to gray-level images, which results in a serious loss of information as most vision applications do, each RGB channel of colour images is processed separately. Then, the covariance intersection (CI) technique is used to fuse all the uncertainties in each channel. Through several experimental results in different environments, we show that including the fused feature uncertainties from all three channels, better estimates of the fundamental matrix are obtained.

5.1 Overview

Colour information is of great importance in computer vision due to its ability to describe the world around us. Interestingly, each RGB channel of colour images provides its own description of the scene into consideration. This description is completely different from other channels' response. Figure 2.5 shows a colour image with the associated RGB channels, where each channel is encoded using a gray scale.

We can see from these images that each channel generates a different response to the incoming light rays.

Most solutions in the literature ignore the valuable information that each channel provides. Usually, these solutions convert colour images to gray-level images, resulting in a substantial loss of information. The proposed solution however, processes each channel independently, then the covariance intersection technique is used to fuse all information in each channel.

Motion estimation via the fundamental matrix

We have seen in Chapter 2 (and in Appendix C) that the fundamental matrix is the algebraic representation that characterises the geometry between two views in the pinhole camera model [82]. Two main approaches can be distinguished for its estimation: iterative and linear algorithms. In general, the fundamental matrix is recovered from image point (feature) correspondences between images [42, 81, 121, 129, 217]. Only eight correspondences are sufficient to estimate the fundamental matrix since it is defined to a scale factor [189]. Knowing that the motion estimation steps rely heavily on the estimation of this matrix, a rational interest on recovering its parameters should be allocated.

Indeed, optimally estimating the fundamental matrix is a hard task since image point (feature) locations are always affected by noise and the correspondences are spoiled by outliers [42]. Random sample consensus (RANSAC) (Algorithm 2, page 296) [58] is a well-known robust statistical solution for this type of problems. Unfortunately, RANSAC and similar solutions are able to detect outliers, but the inaccuracy in the image point locations is still not estimated. Therefore, modelling the uncertainty in estimating the fundamental matrix is of great interest for the motion estimation robustness.

Uncertainty in image-based measurements and Covariance Intersection (CI)

Motion estimation is one of the main problems of computer vision, which can be formulated as parameter estimation from image-based measurements. Knowing that these image-based measurements are corrupted with noise, more interest has recently been given to investigate how parameter estimation might be improved if the uncertainty is incorporated [37, 96, 108]. Commonly, the uncertainty in these applications is represented as covariance matrices.

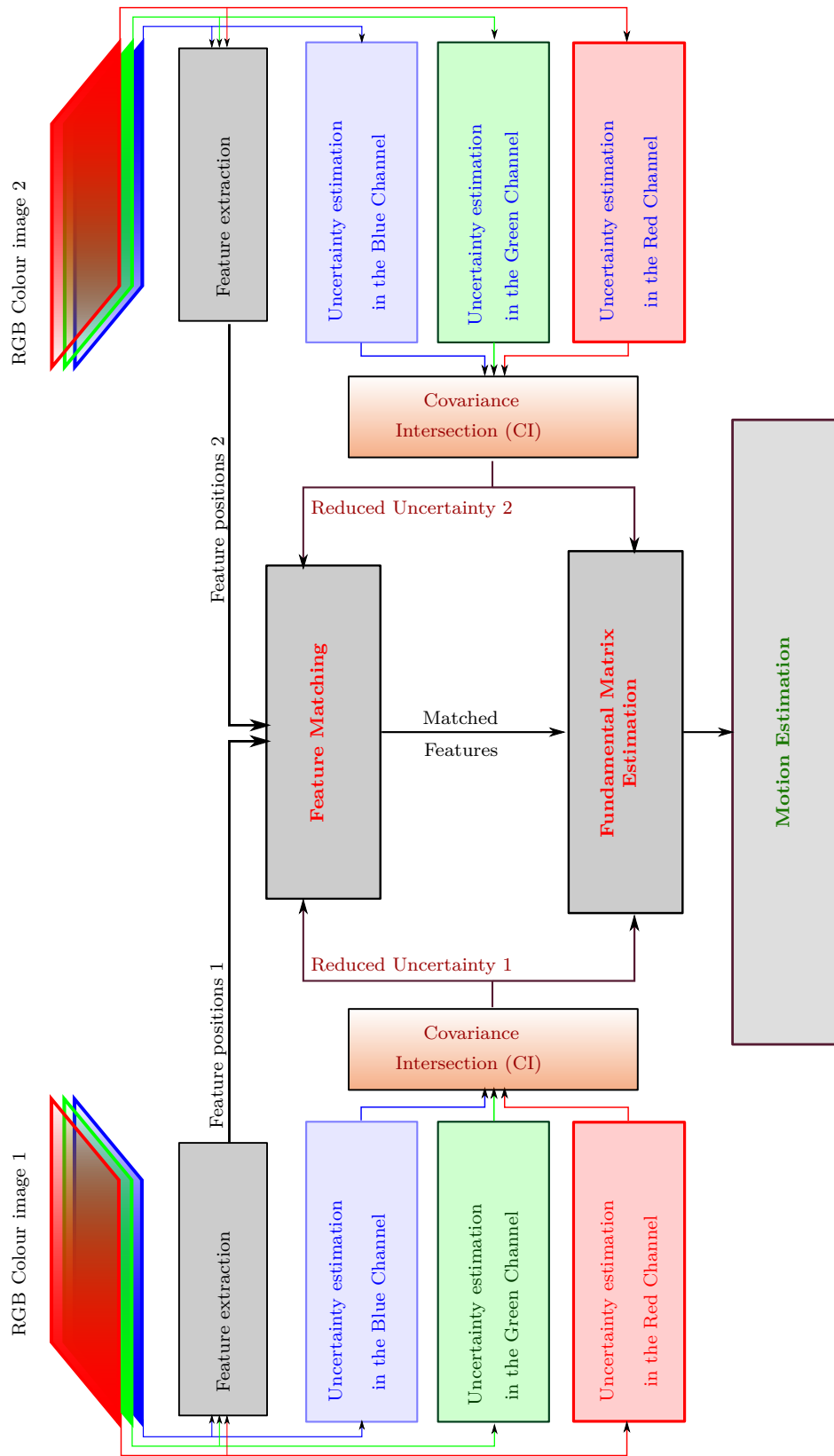


Fig. 5.1 Pipeline of the proposed solution for motion estimation.

In this chapter, we investigate how the integration of feature position uncertainties for each RGB channel of colour images might help to improve the accuracy of the fundamental matrix estimates. As we have seen in Chapters 2 and 4, image description i.e. feature extraction, is a decisive step for motion estimation. Regardless the algorithm used to extract them, feature positions have always some uncertainties [99]. The problem is how one can evaluate this uncertainty and does the estimated uncertainty really characterise the accuracy of feature position? Moreover, does the integration of this uncertainty in the optimisation problem really improve the final estimate of the fundamental matrix?

To investigate that, two feature extraction techniques are used in this work, mainly the SIFT extractor and the Harris feature detector. Feature extraction for each technique is performed along with the associated uncertainties and for each RGB channel. Then, we use the covariance intersection technique (CI) to fuse the uncertainty information from the three RGB channels for each feature. This fusion technique is of great importance in our solution since it allows the reduction of feature position errors. Then, the reduced position errors are used to estimate a more accurate fundamental matrix, F , by formulating it as an optimisation problem.

5.2 The proposed solution

An overview of the proposed solution is depicted in Figure 5.1. As shown in this diagram, the input to the solution is a sequence of colour images. Relative motion is estimated between consecutive pairs (depicted as RGB Colour image 1 and RGB Colour image 2). For each colour image, feature extraction is performed along with their uncertainties in each channel. Thus, each feature in each image is associated with three uncertainty estimates. Covariance intersection algorithm (CI) is then used to reduce feature position uncertainty by fusing the three uncertainties of the three RGB channels. Since the fundamental matrix estimation algorithm accepts only features with low uncertainty, our fusion technique via CI allows more features to contribute in the estimation of this matrix. Hence, more accurate estimations would be obtained. Notice from Figure 5.1 that in our solution, feature uncertainties after CI are used as well in matching features between the two images. Details of each block in this diagram is given in the following sections.

5.3 Related work

We have shown in Chapter 2 that the fundamental matrix can be estimated using linear algorithms from corresponding image points in two images under Gaussian noise assumption, that is not necessary homogeneous or isotropic [97]. Many studies tried to improve the accuracy of this matrix by formulating it as an optimisation problem with the incorporation of feature position uncertainties [26, 99, 215].

In [99], authors presented a framework for introducing feature uncertainties to estimate the homography and the fundamental matrix using renormalisation technique. In that work, authors compared the results between using default values for the covariance matrices (the identity matrix $\text{diag}(1, 1, 0)$), and using the estimated covariance matrices from gray level images. They raised the issue of usefulness of feature uncertainties in this optimisation problem. In addition, they stated that the improvement of the accuracy cannot be ensured by incorporating such uncertainties due to its nature, which is isotropic and homogeneous. On the other hand, Brooks et al. showed in [26] that the quality of the fundamental might be considerably improved by incorporating these uncertainties. They demonstrated a significant reduction in the residual errors for the fundamental matrix when feature uncertainties are incorporated.

In [161], authors presented a procedure to decrease the feature position uncertainty by employing the covariance intersection in all channels of a colour image in order to decrease the homography estimation error by using the Harris features detector. In addition, Zeist et al. illustrated in [215] the enhancement of the bundle adjustment estimations by incorporating the uncertainty for scale invariant feature points.

In the same optic, our solution here uses the iterative renormalisation technique [98] to compute the fundamental matrix. In addition to the matched features, this technique accepts as inputs their reduced uncertainties, which are estimated by fusing uncertainties in the three RGB channels using covariance intersection (Figure 5.1). Our solution exploits as well these feature uncertainties to improve the feature matching operation before estimating the fundamental matrix.

5.4 Feature location uncertainty

The extraction of image features is associated with some localisation errors. Many studies have been dedicated to investigate these localisation errors in the computer vision community [26, 42, 99, 185, 215]. Most of these studies dealt with feature uncertainties based on the pixel noise. Commonly, it is supposed that the error model is Gaussian, leading to characterising the uncertainty by 2D covariance matrices. The

ultimate objective of all these methods is to estimate a feature position uncertainty that really reflects their localisation errors. In this section, we investigate a method to recover feature uncertainty in an image.

Two kinds of feature spaces are considered in this chapter. The first one is the Harris corner detector [171] (Section C.2.1, Appendix C, page 299) and the second is the scale invariant feature transform (SIFT) [119] (Section C.2.2, Appendix C, page 300). Local invariant features such as the SIFT have been extensively used in these imaging based applications. These features have impressive robustness qualities to changes in the orientation and the scale. However, they impose some limited robustness to illumination and affine changes. Over the last decades, the Harris corner detector is considered as the most employed algorithm in the computer vision community. To detect corner pixels, this algorithm relies on the second moment matrix, which is the second order derivative matrix. This matrix in reality describes the curvature distribution around a point.

A comparative study between the two algorithms, regarding their uncertainties for motion estimation, is given. For both algorithms, feature extraction is performed with associated localisation errors.

5.4.1 Basic notions of uncertainty

Before introducing the method for estimating the uncertainties in feature positions, let us introduce some basic notions about the covariance matrix. Suppose we have a measurement of a scalar p , in which its true value is \bar{p} . If we assume that the measurement process of this scalar has a zero-mean Gaussian distribution, then p can be regarded as a particular sample figure of a random variable, where:

$$\begin{aligned} p &= \bar{p} + \Delta p \\ \Delta p &\sim \mathcal{N}(0, \sigma^2) \end{aligned} \tag{5.1}$$

Here, Δp is a random variable describing the errors and σ is a positive value representing the standard deviation (σ^2 is the variance). The expression $\sim \mathcal{N}(0, \sigma^2)$ represents the Gaussian distribution with zero mean and a variance σ^2 of Δp . The expectations of Δp and Δp^2 are given as [26]:

$$E[\Delta p] = 0, \quad E[(\Delta p)^2] = \sigma^2 \tag{5.2}$$

Now, Let us upgrade the scalar p to the two-dimensional case. In this case, the quantity p has a two-entry measurement, $p = (x, y)^\top$. Similarly, let $\bar{p} = (\bar{x}, \bar{y})^\top$ be its true value and the measurement errors of each entry is of zero-mean with Gaussian

distribution. Then, we may write:

$$\begin{aligned} p &= \bar{p} + \Delta p \\ \Delta p &\sim \mathcal{N}(0, \Lambda) \end{aligned} \quad (5.3)$$

where $\mathcal{N}(0, \Lambda)$ is the Gaussian distribution with zero mean and a covariance matrix Λ . Obviously, $\Delta p = (\Delta x, \Delta y)^\top$ and the parameters of this distribution are given by:

$$\begin{aligned} (E[\Delta x], E[\Delta y])^\top &= 0 \\ E[(\Delta p)(\Delta p)^\top] &= \Lambda = \begin{bmatrix} E[\Delta x^2] & E[\Delta x \Delta y] \\ E[\Delta x \Delta y] & E[\Delta y^2] \end{bmatrix} \end{aligned} \quad (5.4)$$

This is equivalent to:

$$\Lambda = \begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix} \quad (5.5)$$

Note that if $\sigma_x = \sigma_y$ then the probability distribution is said to be *isotropic* and the level-set is represented with a circle (Figure 5.2a and 5.2b). On the other hand, if $\sigma_x \neq \sigma_y$ then the probability distribution is said to be *anisotropic* and the level-set is represented with an ellipse (Figure 5.2c and 5.2d).

Now, let us move to the case where we have a number of k two-dimensional quantities $p_i = (x_i, y_i)^\top$, where $i = 1, \dots, k$. Then, the covariance matrix of a particular p_i is given by:

$$\Lambda_i = \begin{bmatrix} \sigma_{i_x}^2 & 0 \\ 0 & \sigma_{i_y}^2 \end{bmatrix} \quad (5.6)$$

If all quantities $p_i = (x_i, y_i)^\top$, where $i = 1, \dots, k$ have the same distribution then, the measurement error is said to be *homogeneous*. However, if the standard deviation may change from one quantity to other, then the measurement error is said to be *inhomogeneous*. Therefore, four cases might be distinguished here:

- If $\sigma_x = \sigma_y$ with the same distribution, then the probability distribution is said to be *isotropic homogeneous* (Figure 5.2a).
- If $\sigma_x = \sigma_y$ with the different distributions, then the probability distribution is said to be *isotropic inhomogeneous* (Figure 5.2b).
- If $\sigma_x \neq \sigma_y$ with the same distribution, then the probability distribution is said to be *anisotropic homogeneous* (Figure 5.2c).
- If $\sigma_x \neq \sigma_y$ with the different distributions, then the probability distribution is said to be *anisotropic inhomogeneous* (Figure 5.2d).

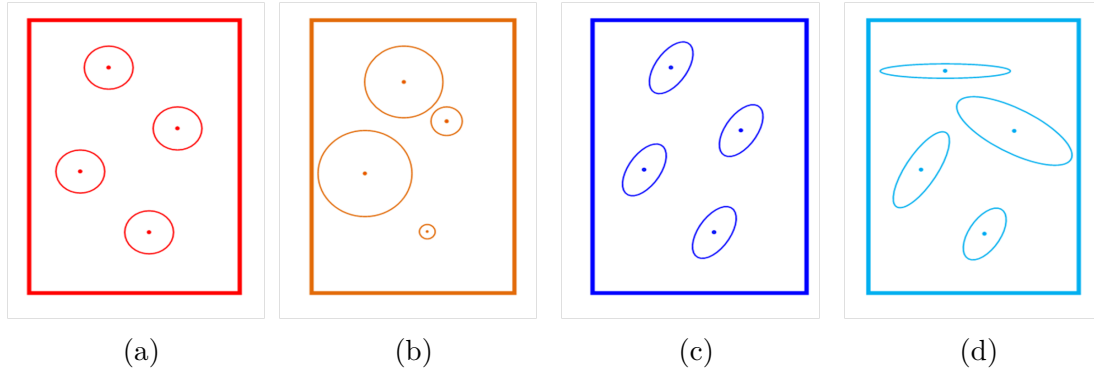


Fig. 5.2 Noise models. (a) Isotropic homogeneous. (b) Isotropic inhomogeneous. (c) Anisotropic homogeneous. (d) Anisotropic inhomogeneous.[26][99]

5.4.2 Uncertainty estimation for the Harris corner detector

The Harris corner detector has been extensively used over the last decades in computer vision community. As mentioned in Section C.2.1 (Appendix C, page 299), this detector uses the second order derivative matrix, evaluated from the intensity around corners. This matrix in reality is the second moment matrix, which describes the curvature distribution. Regardless the techniques used for extraction, Harris feature positions have a certain level of uncertainty.

Let $\mathbf{x} = (x, y)^\top$ be the pixel coordinates of an image feature extracted using the Harris corner detector (which is a gradient based feature extractor) and $\bar{\mathbf{x}} = (\bar{x}, \bar{y})^\top$ be the correct location of this feature [171]. Errors in its location are then given by $\Delta \mathbf{x} = (\Delta x, \Delta y)^\top = (x - \bar{x}, y - \bar{y})^\top$. Considering Δx and Δy as random variables and referring to (5.4), the covariance matrix of this feature measurement is given by [99]:

$$\Lambda = \begin{bmatrix} E[\Delta x^2] & E[\Delta x \Delta y] \\ E[\Delta x \Delta y] & E[\Delta y^2] \end{bmatrix} \quad (5.7)$$

where $E[\cdot]$ represents the expectation. This matrix in reality provides the extent or the spread of the uncertainty of a feature $\mathbf{x} = (x, y)^\top$ in each axis (x -axis and y -axis), which is characterised as a Gaussian distribution.

Two main techniques are used in the literature for determining Λ . The first one is a residual technique, while the second is a derivative approach. These techniques are detailed in [99]. We employ the latter technique for computational reasons. To extract a feature using this approach, a Hessian matrix is defined for the coordinates x and y as follows:

$$\mathcal{H} = \begin{bmatrix} \sum_{(x,y) \in \mathcal{N}_x} \omega_{xy} I_x^2 & \sum_{(x,y) \in \mathcal{N}_x} \omega_{xy} I_x I_y \\ \sum_{(x,y) \in \mathcal{N}_x} \omega_{xy} I_y I_x & \sum_{(x,y) \in \mathcal{N}_x} \omega_{xy} I_y^2 \end{bmatrix} \quad (5.8)$$

where I_x and I_y are the partial derivatives and ω_{xy} is a weight function, commonly has a Gaussian nature and \mathcal{N}_x is a rectangular grid whose centre is the feature point $\mathbf{x} = (x, y)^\top$.

Known as the second moment matrix, this expression is the basis of all gradient-based feature extractors, which depicts the curvature distribution around the feature point $\mathbf{x} = (x, y)^\top$. Therefore, this matrix decides, in the extraction step, whether a feature in this position is good or not. The greater the change in curvature, the more accurately the corner can be located and vice versa. Therefore, a quick change in the gray level around a point corresponds to a large Hessian. This means that this corner will be accurately located. Therefore, the inverse of this expression will certainly define the covariance of this feature:

$$\Lambda = \text{inv}(\mathcal{H}) \quad (5.9)$$

The estimated covariance of a feature can be visualised through an error ellipse as illustrated in Figure 5.3. We can see clearly from these images that the noise is anisotropic and inhomogeneous. Figure 5.4 shows some Harris features with the associated position uncertainties for each RGB channel of colour images. We can see from this figure that each RGB channel behaves in different way and using Covariance Intersection will certainly reduce these uncertainties.

5.4.3 Uncertainty estimation for SIFT feature positions

Similarly, the SIFT [119] and the SURF [15] features inherit also some feature localisation errors. Zeist et al. presented in [215] a similar method to the Harris detector, for estimating the uncertainty of the SIFT and the SURF features. In addition to their robust detection of keypoints in an image, these algorithms are also able to detect interest regions. Most applications in the literature have been focusing on matching features to assess their localisation errors but giving less interest to the detection accuracy [128, 171].

The SIFT region detector has become one of the most popular algorithms for feature extraction problem, and the algorithm of choice for many vision applications. This algorithm uses Laplacian for feature detection and scale selection. Using the derivative based approach; the location uncertainty of the SIFT features is also calculated as the inverse of the Hessian matrix as well [215]. Neighbouring sample



Fig. 5.3 Harris features with position uncertainties visualised through error ellipses.



Fig. 5.4 Harris features with position uncertainties for each RGB channel visualised through error ellipses. Covariances from each channel is coloured according to the channel.



Fig. 5.5 SIFT features with position uncertainties visualised through error ellipses.

points are used to calculate the derivatives:

$$\mathcal{H} = \sum_{(x,y) \in \mathcal{N}_x} \omega(i,j) \begin{bmatrix} D_{xx}(i,j, \delta_x) & D_{xy}(i,j, \delta_x) \\ D_{yx}(i,j, \delta_x) & D_{yy}(i,j, \delta_x) \end{bmatrix} \quad (5.10)$$

$$\Lambda = \text{inv}(\mathcal{H})$$

where D_{xx} , D_{xy} and D_{yy} are the second order derivatives around the point $\mathbf{x} = (x, y)^\top$, σ_x is the scale, and \mathcal{N}_x is the image neighbourhood. The Hessian is calculated as a Gaussian weighted sum $\omega(i, j)$. Usually, the neighbourhood is taken as regions of 3×3 to 5×5 pixels. Again, the estimated feature covariances are visualised using error ellipses as shown in Figure 5.5.

Figure 5.6 shows some SIFT features with the associated position uncertainties for each channel. We can see from this figure that each channel RGB of a colour image behaves in different way. Using Covariance Intersection will certainly reduce these uncertainties.



Fig. 5.6 SIFT features with position uncertainties for each RGB channel visualised through error ellipses. Covariances from each channel is coloured according to the channel.

5.5 Covariance intersection

After having estimated the covariance of each feature in each RGB channel, covariance intersection is used to fuse information from all the three channels. Covariance intersection is a filtering approach based on the combination of information matrices in order to get better estimate. This technique uses a data fusion architecture, where information about a signal is incomplete [142]. In distributed fusion systems, different nodes or sensors provide different estimates with unknown degree of cross-correlation. Therefore, covariance intersection could be a good option, which gives much better and reliable estimates for local cross-correlated estimates [59, 142].

Given $\hat{x}_1, \hat{x}_2, \dots, \hat{x}_n$, the unbiased estimates of a quantity x_0 , where:

$$E[\hat{x}_i] = x_0, \quad \text{for } i = 1, 2, \dots, n. \quad (5.11)$$

The associated covariance matrices for each estimate are given by P_1, P_2, \dots, P_n . These covariance matrices are assumed to be consistent, which means:

$$P_i - \tilde{P}_i \geq 0, \quad \text{for } i = 1, 2, \dots, n. \quad (5.12)$$

where:

$$\tilde{P}_i = E[(\hat{x}_i - x_0)(\hat{x}_i - x_0)^\top] = E[\tilde{x}_i \tilde{x}_i^\top] \quad (5.13)$$

The matrix \tilde{P}_i represents the covariance matrix of the i^{th} estimate \hat{x}_i , i.e., the correlation matrix of the i^{th} estimation error $\tilde{x}_i = \hat{x}_i - x_0$. The covariance intersection is then specified by:

$$P_0^{-1} = \sum_{i=1}^n \omega_i P_i^{-1} \quad (5.14)$$

$$P_0^{-1} x_0 = \sum_{i=1}^n \omega_i P_i^{-1} \hat{x}_i \quad (5.15)$$

where $0 \leq \omega_i \leq 1$. Detail on estimating these weights is presented in [142]. For $n \geq 2$, these weights must satisfy the linear constraint:

$$\omega_1 + \omega_2 + \dots + \omega_n = 1 \quad (5.16)$$

and:

$$\text{tr}(P_i)\omega_i - \text{tr}(P_{i+1})\omega_{i+1} = 0, \quad \text{for } i = 1, 2, \dots, n-1 \quad (5.17)$$

Combining the two constraints (5.16) and (5.17) yields to the linear system:

$$\begin{bmatrix} \varepsilon_1 & -\varepsilon_2 & 0 & \cdots & 0 \\ 0 & \varepsilon_2 & -\varepsilon_3 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & \cdots & 0 & \varepsilon_{n-1} & -\varepsilon_n \\ 1 & \cdots & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \cdots \\ \omega_{n-1} \\ \omega_n \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \cdots \\ 0 \\ 1 \end{bmatrix} \quad (5.18)$$

where $\varepsilon_i = \text{tr}(P_i)$.

Figure 5.7 shows an example of error ellipses of three original estimates and the error after using the covariance intersection algorithm. Results from this simple example confirm our analysis made above.

In our case, n equals to three, representing the three channels of the colour image. In each colour image, each feature is represented with three covariance matrices. Figure 5.4 and Figure 5.6 illustrate some features with their uncertainties on images taken from the dataset presented in [69].

5.6 Robust fundamental matrix estimation

Our framework relies on the estimated uncertainties in the feature position after fusing all covariances of each RGB channel of colour images. These uncertainties

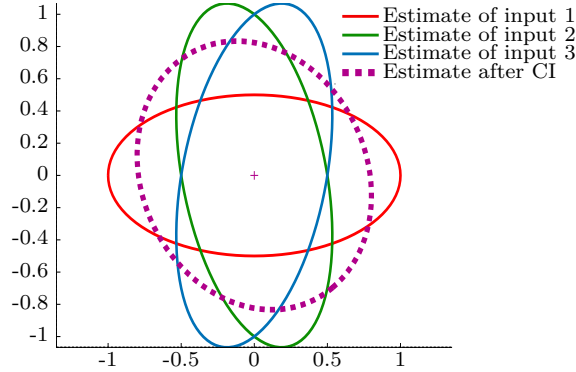


Fig. 5.7 Example of covariance intersection. Red, green, and blue lines represent three covariances of three inputs illustrated via error ellipses. The magenta error ellipse shows the estimate from CI.

are then incorporated in the motion estimation algorithm. Prior to that, these uncertainties are used in improving the feature matching task.

5.6.1 Matching using features uncertainty

The availability of more accurate feature-uncertainty information will be adopted in our work to improve the matching task before estimating the motion parameters (Figure 5.1). Instead of using the standard RANSAC algorithm, we exploit the feature uncertainty and their propagation to reject outliers. This technique is introduced in [10] and [155]. Given a set of corresponding image points $\mathbf{x}_i \leftrightarrow \mathbf{x}_j$, where $\mathbf{x}_i = (x_i, y_i, 1)^\top$ and $\mathbf{x}_j = (x_j, y_j, 1)^\top$, then the 2D homography matrix \mathcal{H} is the 3×3 matrix that links between \mathbf{x}_i and \mathbf{x}_j . Detail about estimating the 2D homography is given in Appendix A (Section A.3, page 283).

In the same uncertainty framework, and after having estimated:

- the 9×9 covariance matrix $\Lambda_{\mathcal{H}}$ that encodes the uncertainty of the homography matrix \mathcal{H} using technique introduced in [10], and
- the 3×3 covariance matrices $\Lambda_{\mathbf{x}_i}$ and $\Lambda_{\mathbf{x}_j}$ that encodes the uncertainty of the points \mathbf{x}_i and \mathbf{x}_j respectively using techniques introduced in the previous Sections 5.4 and 5.5.

the covariance of the projection point of \mathbf{x}_i in the second image, denoted $\hat{\mathbf{x}}_i$, is then given by [155]:

$$\Lambda_{\hat{\mathbf{x}}_j} = A\Lambda_{\mathcal{H}}A + \mathcal{H}\Lambda_{\mathbf{x}_i}\mathcal{H} \quad (5.19)$$

where A is 3×9 matrix given by:

$$\begin{bmatrix} \mathbf{x}_i^\top & 0_{1 \times 3} & 0_{1 \times 3} \\ 0_{1 \times 3} & \mathbf{x}_i^\top & 0_{1 \times 3} \\ 0_{1 \times 3} & 0_{1 \times 3} & \mathbf{x}_i^\top \end{bmatrix}$$

The covariance $\Lambda_{\hat{\mathbf{x}}_j}$ in (5.19) models the uncertainty in $\hat{\mathbf{x}}_j$ due to both the measurement uncertainty in \mathbf{x}_i and in the estimated homography \mathcal{H} . Note that the uncertainty $\Lambda_{\mathbf{x}_j}$, of the point \mathbf{x}_j in the second image, has already been estimated using techniques presented in Sections 5.4 and 5.5 as well. Then, rejecting matching outliers is performed by checking the intersection between the uncertainty in the projected point $\hat{\mathbf{x}}_j$ and the uncertainty in the measured point \mathbf{x}_j . Clearly, these uncertainties (in the measured point in the right image \mathbf{x}_j and the projected point $\hat{\mathbf{x}}_j$) are represented using error ellipses. Therefore, if the two ellipses intersect in the image plane, this means that this correspondence can be considered as a true matching, otherwise this matching is said to be an outlier.

5.6.2 Robust fundamental matrix estimation

After having rejected matching outliers using uncertainty information, the fundamental matrix is then estimated using again the reduced feature position uncertainties (Figure 5.1). We have seen in Section 2.9.3 (and in Appendix C), that the fundamental matrix, F , is the algebraic representation of epipolar geometry. It is well known that for each pair of images I_i and I_j , point correspondences $\mathbf{x}_i \leftrightarrow \mathbf{x}_j$, where $\mathbf{x}_i = (x_i, y_i, 1)^\top$ and $\mathbf{x}_j = (x_j, y_j, 1)^\top$, then the 3×3 fundamental matrix, F , can be derived from the following equation system:

$$\mathbf{x}_j F \mathbf{x}_i^\top = (x_j, y_j, 1)^\top \begin{bmatrix} f_1 & f_2 & f_3 \\ f_4 & f_5 & f_6 \\ f_7 & f_8 & f_9 \end{bmatrix} (x_i, y_i, 1) = 0 \quad (5.20)$$

Let the column vector \mathbf{f} contain the nine elements of F , where:

$$\mathbf{f} = (f_1, f_2, f_3, f_4, f_5, f_6, f_7, f_8, f_9)^\top$$

The epipolar geometry constraint, ($\forall k \in [1, n], \mathbf{x}_{j_k} F \mathbf{x}_{i_k}^\top = 0$) simply leads to the matrix equation $M\mathbf{f} = 0$ [82]. If we have n correspondences, then this $n \times 9$ matrix

M is given by:

$$M = \begin{pmatrix} x_{j_1}x_{i_1} & x_{j_1}y_{i_1} & x_{j_1} & y_{j_1}x_{i_1} & y_{j_1}y_{i_1} & y_{j_1} & x_{i_1} & y_{i_1} & 1 \\ x_{j_2}x_{i_2} & x_{j_2}y_{i_2} & x_{j_2} & y_{j_2}x_{i_2} & y_{j_2}y_{i_2} & y_{j_2} & x_{i_2} & y_{i_2} & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{j_n}x_{i_n} & x_{j_n}y_{i_n} & x_{j_n} & y_{j_n}x_{i_n} & y_{j_n}y_{i_n} & y_{j_n} & x_{i_n} & y_{i_n} & 1 \end{pmatrix} \quad (5.21)$$

The well-known method for solving this problem is the singular value decomposition (SVD) of M and putting the smallest singular value of F to zero. Obviously, data are still corrupted with noise. Therefore, recovering an exact solution is not possible. All standard solutions for solving the matrix equation $M\mathbf{f} = 0$ ignore the uncertainties in the feature locations even the well-known 8-point algorithm.

In an optimisation framework, to estimate the fundamental matrix from point correspondences between two images by taking into account the feature uncertainties, [98], the cost function to be minimised is defined as:

$$J_F = \frac{1}{n} \sum_{k=1}^n \mathcal{W}_k (x_{i_k}, Fx_{j_k})^2 \quad (5.22)$$

$$\mathcal{W}_k = \frac{1}{(x_{j_k}, F^\top V_0[x_{i_k}]Fx_{j_k}) + (x_{i_k}, F^\top V_0[x_{j_k}]Fx_{i_k})} \quad (5.23)$$

Specifically, (a, b) denotes the inner product of vectors a and b , and $V_0[x_{i_k}]$ for a particular image feature x_{j_k} is given by:

$$V_0[x_{i_k}] = \begin{bmatrix} P^{x_{i_k}} & 0 \\ 0 & 0 \end{bmatrix} \quad (5.24)$$

where P is the covariance of that feature location.

The optimisation is performed in two main stages. Firstly, the fundamental matrix F is computed without taking in account the constraint $\det(F) = 0$. Then, this constraint is enforced in the second stage. During the first stage, the weighted least-squares solution is estimated by using the renormalisation technique.

In the cost function (5.22), features with low uncertainty are included more, and thus, they contribute more to the final results. This is because the weighting term \mathcal{W}_k has an inverse proportion with the uncertainty, as shown in (5.23). In [26, 97, 98] features covariance matrices are evaluated using two methods. In the first one, they used the derivative approach from the grey-level images and in the second approach

they just used default values ($V_0[\mathbf{x}_{i_k}] = \text{diag}(1, 1, 0)$). In this work however, features covariances are estimated for each RGB channel, and then the covariance intersection filter is applied to fuse the three estimates. Therefore, feature uncertainties are reduced and consequently more features, which are optimally matched as explained in Section 5.6.1, will be able to contribute to the final estimate of the fundamental matrix. This method will lead to more robust and accurate estimates.

After robustly estimating the fundamental matrix, motion parameters (translation vectors and rotation matrices) are then extracted using our techniques introduced in Section 4.5 (Chapter 4, page 83).

5.7 Experimental results

This section presents the experimental evaluation of the proposed method using covariance intersection in motion estimation. Uncertainties in feature positions have been taken into consideration in this implementation. First, we illustrate the need for separately dealing with each RGB channel of colour images, and then using the covariance intersection filter. To do that, the error function used to evaluate the performance of the proposed solution is defined by:

$$f_F = \frac{1}{n} \sum_{k=1}^n d(\mathbf{x}_{j_k}, F\mathbf{x}_{i_k})^2 + d(\mathbf{x}_{i_k}, F^\top \mathbf{x}_{j_k})^2 \quad (5.25)$$

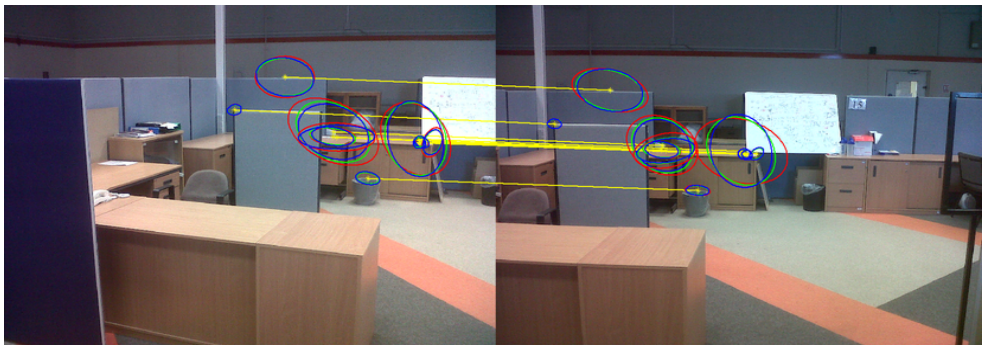
where $d(\mathbf{x}_{j_k}, F\mathbf{x}_{i_k})$ is the Euclidean distance error between the point \mathbf{x}_{j_k} and $F\mathbf{x}_{i_k}$. Note that the quantity $F\mathbf{x}_{i_k}$ is in fact the epipolar line corresponding to \mathbf{x}_{i_k} (Section C.1, page 293). Since the fundamental matrix defines a point-line mapping, the error is the average over all n matches of the squared distance between the epipolar line of a point and its matching point in the other image.

The implementation of the proposed technique is conducted on real data from different environments (Section 1.6, Chapter 1, page 8), showing the diversity of the image qualities. The first dataset is gathered from a vehicle travelling in an urban environment in the city of Karlsruhe [68] (Figure 5.8a). The second is a sequence of images taken from our laboratory (Figure 5.8b) and the third one is a collection of data gathered at a Mars/Moon analogue site on Devon Island, Nunavut [64] (Figure 5.8c). Few matching features along with their uncertainties are shown in this figure (Figure 5.8).

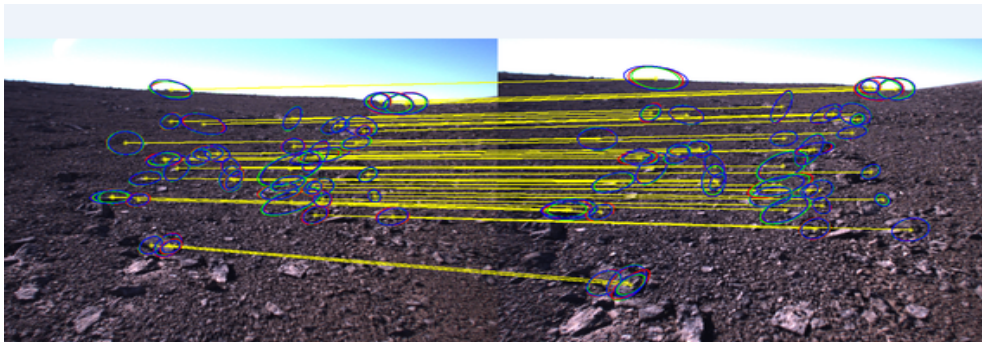
The corresponding results of these experiments are given in Figure 5.9. For each environment, residual errors are shown against the number of matched points used to compute F . Note that these errors are calculated over all matched points



(a) Urban environment

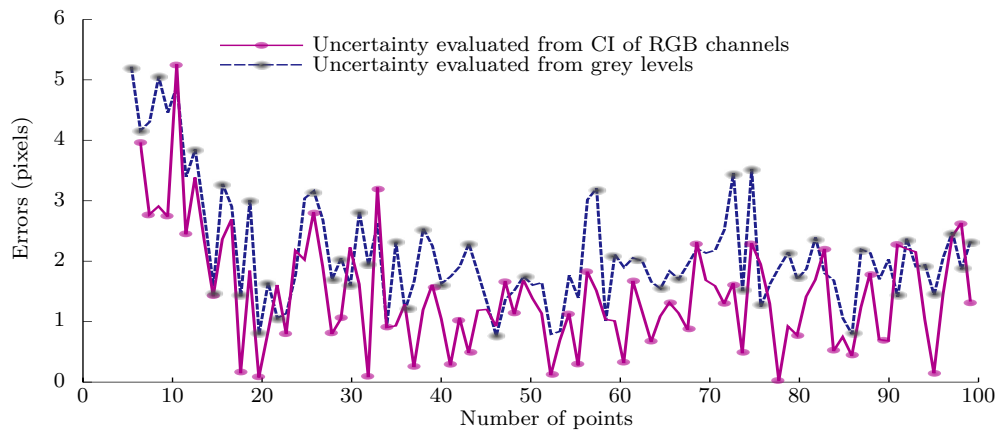


(b) Indoor environment

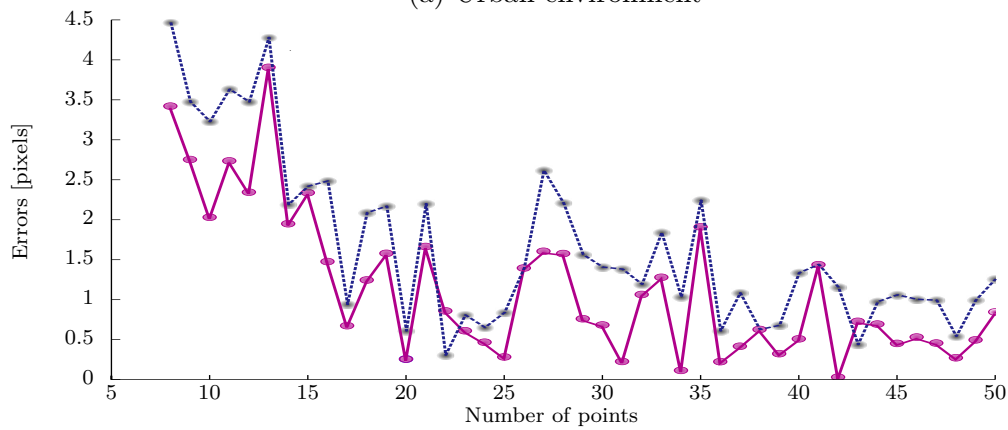


(c) Moon/Mars analogue environment

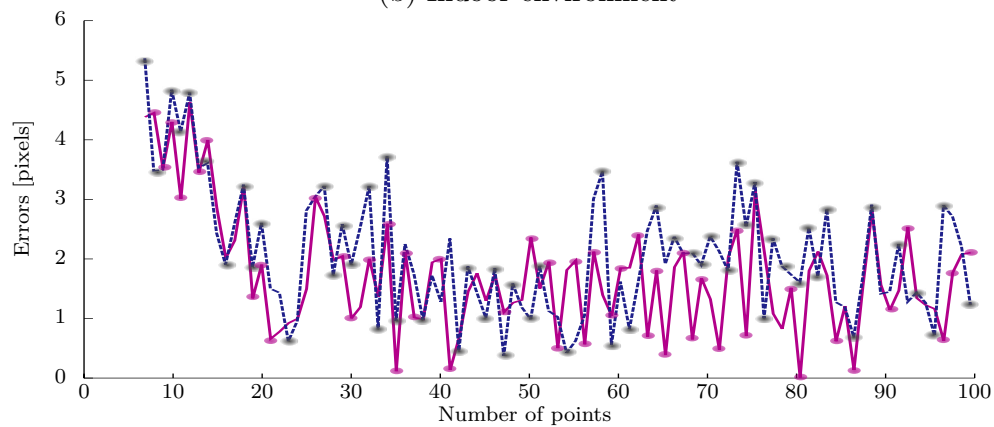
Fig. 5.8 Examples of image pairs used for comparison. Features and their uncertainties in these environments are extracted using the SIFT algorithm. Few matched points are illustrated. Error bounds for each channel are given as ellipses coloured according to the channel from which they were estimated.



(a) Urban environment



(b) Indoor environment



(c) Mars/Moon analogue site

Fig. 5.9 Errors plotted against the number of points used to compute the fundamental matrix. Dashed lines (blue) show the results using covariance matrices evaluated from the grey levels. Solid lines (red) show results using covariance intersection to fuse uncertainties from all the three RGB channels of colour images. In all environments, using covariance intersection is noticeably better than using grey-levels, though the latter also shows good results.

and not just the ones used to compute F . To illustrate the advantage of using colour information, comparison between using the covariance intersection over the three RGB channels for estimating the feature uncertainties and using just the grey levels is shown. Figure 5.9 shows the significant impact on the final estimate of the fundamental matrix when using the decreased feature location uncertainty via the covariance intersection. One can see that the improvement is not considerable in the Moon/Mars analogue environment. This is due to the image qualities and also to the nature of the landscape. However, this is not the case for the remaining environments, where considerable improvement can be noticed. Therefore, including features position uncertainties yields to remarkably better results. Obviously, for all environments, the accuracy is improved as the number of matched points used to estimate F is increased.

These results are summarised as well in Table 5.1.

Table 5.1 Average residual errors in pixels for each environment using covariance intersection for feature uncertainties of all RGB channels (E_{RGB}) and using covariances from grey levels (E_{Gray})

	E_{gray}	E_{RGB}
Urban environment	1.88	1.16
Indoor environment	1.36	0.74
Moon/Mars analogue environment	2.17	2.02

Comparison with similar algorithms for estimating the fundamental matrix, such as the 8-point algorithm, is conducted. Figure 5.10, shows the residual errors from the urban environment experiment as an illustrative example. The proposed solution uses the renormalisation technique with reduced feature uncertainties. This uncertainty reduction is obtained by employing the covariance intersection to the feature location errors. This technique iteratively removes bias of weighted least squares, instead of minimising a cost function.

Both methods, the 8-point algorithm and the proposed technique, use the same well matched points. It is clearly shown in Figure 5.10 that when using the covariance intersection, more features will be able to contribute to the final solution. Thus, this leads to better estimates of the fundamental matrix.

Comparative experiments when using two feature extractors, such as the Harris corner detector and the SIFT algorithm, are conducted as well in this experiment. The two main investigated tasks in this part are the matching robustness and the uncertainty estimation of each detector. We have seen that the Harris corner detector relies on the image intensity changes to detect corners using the second moment matrix. Matching in the Harris corner detector algorithm is performed using the

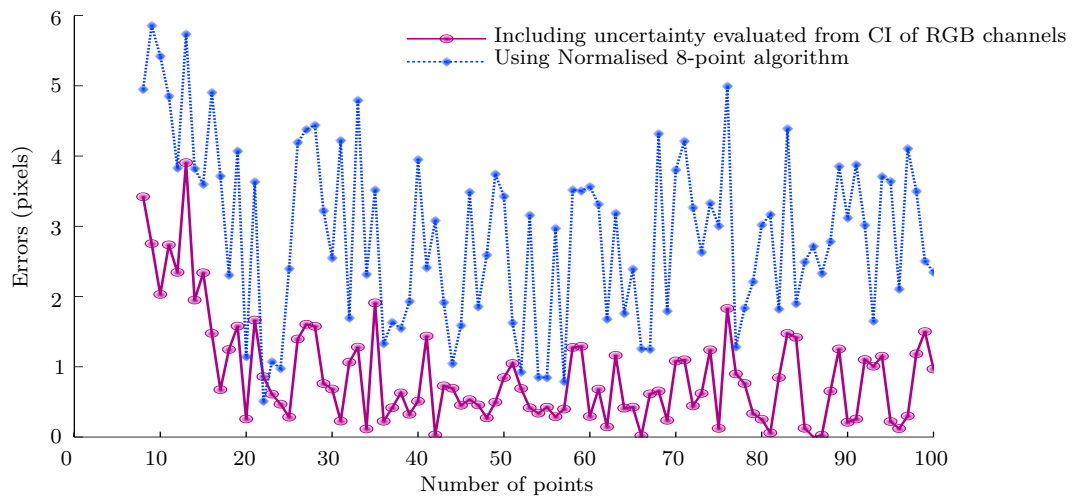


Fig. 5.10 The fundamental matrix estimation using the covariance intersection algorithm. Residual errors are plotted as a function the number of deployed points. Dotted blue line shows the results using the normalised eight-point algorithm. Solid red line shows results using the normalisation technique with reduced feature uncertainties by employing the covariance intersection. The residual errors from the covariance intersection algorithm are substantially smaller than those from the normalised eight-point algorithm, regardless the number points.

cross correlation technique between local image patches (Section C.2.1, Appendix C, page 299). This means that only features that correlate most strongly with each other in both directions are accepted. The SIFT features on the other hand, use the Euclidean distance between the feature descriptors as a similarity criteria, and use the nearest neighbour algorithm to match features (Section C.2.2, Appendix C, page 300). This technique has proved its efficiency among the computer vision researchers [128]. Note that while the Harris corner detector is able to match larger number of features in relatively shorter time, this significantly compromises its robustness. The SIFT robustly performs that, but relatively in more time.

Our experiments reveal that, even though, features uncertainties using the Harris corner are relatively smaller than those estimated using SIFT, the latter detector is more representative to the real uncertainties in the features positions. In our experiments, the average of the residual errors using the Harris corner detector is 1.48 pixels, while it is 1.16 pixel when SIFT is used. The obtained results are plotted in Figure 5.11. These results confirm that uncertainties from SIFT features are less conservative than Harris. Hence, more robust estimates are obtained, which justify our preference for the SIFT algorithm. A summary graph is illustrated in Figure 5.12 comparing the proposed solution using CI of uncertainty in the three RGB

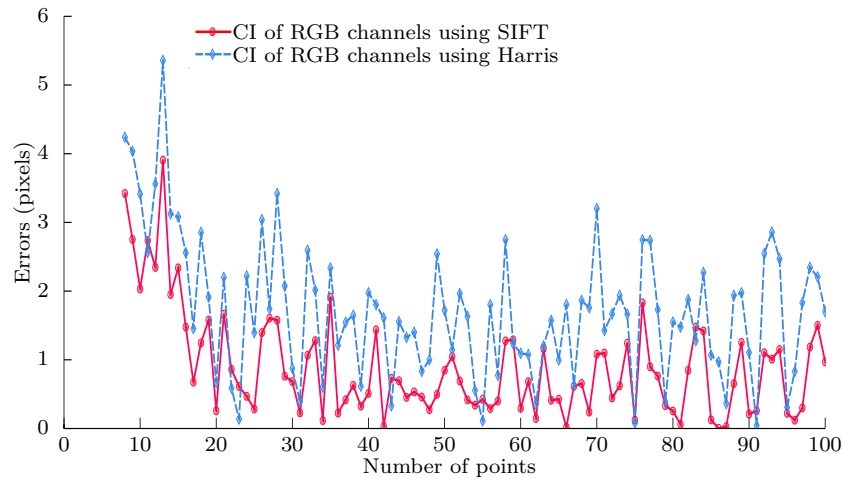


Fig. 5.11 Comparison between using uncertainties from the SIFT extractor and from the Harris corner detector features with the covariance intersection technique to estimate the fundamental matrix. Performance with the SIFT features is better than with Harris corner detector, though the latter algorithm also gives encouraging results.

channel of colour images with the solution using uncertainty from grey-level images, and with the traditional eight-point Algorithm.

5.8 Conclusion

In this chapter, a technique for robust and accurate estimation of the fundamental matrix is presented. In most vision applications, colour images are converted first to gray-level images leading to a serious loss of information. In our solution, however, each RGB channel of colour images is processed separately. Then, a fusion mechanism is employed to combine these information. After having estimated the uncertainties in feature locations in each channel, covariance intersection filter is used. This results on a reduction of the measurement error, leading to more accurate estimates of the fundamental matrix. The iterative technique for this matrix estimation is adopted, which takes as inputs the uncertainties in feature locations. Before estimating the fundamental matrix, the available feature uncertainty information is used as well to improve the matching task rather than using only the standard RANSAC algorithm.

Through several experimental results in different environments, we showed that including feature uncertainties from all three RGB channels of images leads to more accurate estimates of the fundamental matrix and consequently to more accurate estimates of the motion parameters. A comparative study between employing feature

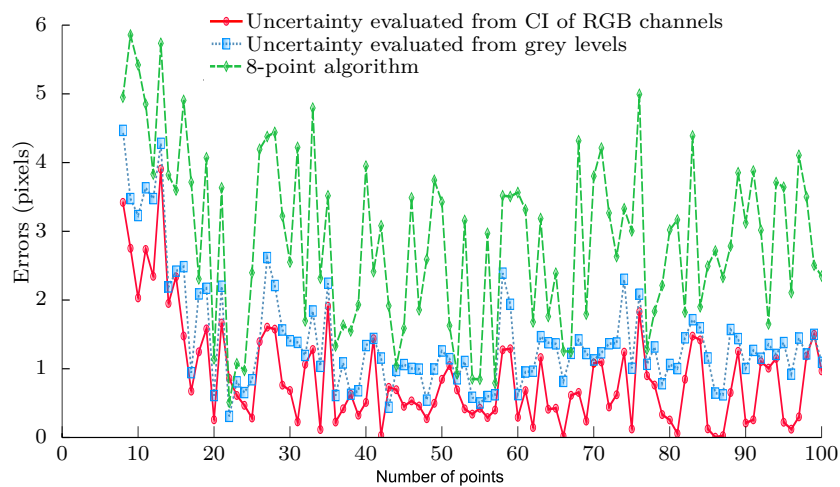


Fig. 5.12 A summary graph comparing the proposed solution using the covariance intersection algorithm on the uncertainty in the three RGB channel of colour images, the solution using uncertainty from grey-level images, and the traditional eight-point Algorithm.

uncertainties extracted from the Harris corner detector and the SIFT algorithm is given as well, where the latter exhibits better performance.

Chapter 6

Robust L_∞ Convex Optimisation for Monocular Motion Estimation

In chapter 4, we presented a solution for motion estimation using convex optimisation in the triangulation task and the H_∞ filter in the scale ambiguity problem. In Chapter 5, we introduced a solution that incorporates feature position uncertainties in estimating the fundamental matrix before recovering the camera motion parameters. In the present chapter, a more global and robust L_∞ norm-based optimisation solution for monocular motion estimation systems is presented. In addition to exploiting the uncertainty estimation techniques from the previous chapter, this solution propagates this uncertainty through the multiple-view geometry algorithms and incorporates it at each stage of the solution.

More precisely, we introduce in this chapter the robust convex optimisation notion in our solution, where the propagated uncertainties to the relative rotations and translations, and to the 3D scene points contribute in improving the global motion estimation. Rather than using the H_∞ filter to solve the scale ambiguity problem in our monocular system, we set up a robust least squares algorithm using the SOCP approach, capable of handling the system uncertainties. Experimental evaluations showed that robust convex optimisation with the L_∞ norm under uncertain data and the robust least squares via the SOCP clearly outperform classical methods based on least squares and Levenberg-Marquardt algorithms.

6.1 Overview

Visual Odometry (VO) types of approaches have been widely studied in the last decade as a possible solution for autonomous navigation systems. Optimisation techniques, such as bundle adjustment (BA) are used to deliver trajectory estimates.

Throughout this thesis, monocular system is used for motion estimation. As we have seen in Chapter 4, this system is known for its frame-to-frame scale ambiguity, in which some successful solutions dealing with this problem have been listed. For the optimisation task, the bundle adjustment algorithm has been widely used in motion estimation for both stereo and monocular systems. This optimisation technique is used to refine the visual reconstruction jointly with the optimal camera pose estimates by minimising the re-projection errors. This optimisation problem is usually formulated as a non-linear least squares problem, where the error is the squared differences between the observed feature locations and the projection of the corresponding 3D points on the image plane. The Levenberg-Marquardt (LM) algorithm is the most popular algorithm in the computer vision community for solving non-linear least squares problems. It is considered as the algorithm of choice for bundle adjustment. However, the main issue with these methods, and notwithstanding of dependency on good initialisation, is related to the high probability of converging to a local minimum or even to infeasible solutions.

As a powerful alternative, convex optimisation (denoted in this chapter as CVX) offers the possibility of getting around these issues when dealing with these types of non-linear minimisation problems [25, 94]. A projective bundle adjustment algorithm using L_∞ norm is proposed in [130], based on minimising the L_∞ norm of re-projection error, where the problem is divided into two successive tasks, by fixing the parameters of one sub-problem while optimising the remaining sub-problem using convex optimisation. The first sub-problem consists of recovering the camera parameters while keeping the structure parameters fixed. The second one looks for the camera and the structure parameters. In [36], a visual odometry approach to estimate the essential matrix by minimising the algebraic error through a convex optimisation is presented.

Dealing with uncertain data and more specifically in our visual uncertain measurements has become crucial in the computer vision community. Robust optimisation in general is an optimisation procedure, which is able to recover an optimal solution, that guarantees its feasibility for any realisation of the uncertain data [20]. Indeed, robust optimisation, for which data are not precisely specified, can explicitly incorporate uncertainty [65]. Knowing that image-based measurements are subject to deterministic perturbations, more studies have started to focus on how parameters estimation using these measurements might be improved if additional information characterising the uncertainty of the data are available [26].

Robust convex optimisation, on the other hand, would be a valid option to develop for visual odometry, for which image-based measurements are associated

with uncertainties. It is worth noticing here, that the common way of expressing this uncertainty information is in terms of covariance matrices.

We have seen in the previous chapters, that extracting feature points is the first step in many vision applications, including the motion estimation problem. Position uncertainty estimation for these feature points is presented as well in Chapter 5, where details on the uncertainty on the SIFT extractor and the Harris corner detector are given. The same techniques are used in this solution, where comparisons between the solution performance using the Harris corner detector and the SIFT extractor are given.

In the present solution, more interest is given to the uncertainty propagation. More studies have started to consider the impact of the propagated uncertainties on the final estimate accuracy. Haralick described in [78] how to propagate additive random perturbations through the various stages of a vision-based algorithm. A similar idea is adopted by Leo et al., [49], who presented a methodology for propagating the measurement uncertainty, from the initial stages through the stereo calibration, to the uncertainty in the triangulation. However, a complete method adopting robust convex optimisation scheme using covariance information to get better estimates of the camera motion for monocular systems has not been given yet.

6.2 The proposed solution

We aim, in this chapter, to robustly improve the camera motion estimations of an innovative monocular visual odometry solution by incorporating feature localisation uncertainties and their propagation through the multiple view geometry. The proposed approach provides a framework to estimate robust and global solution under the L_∞ norm. This approach, which relies on modern optimisation methods, is more efficient in dealing with uncertain data than their traditional gradient-based counterparts. Therefore, robust and global solutions are guaranteed.

Although most researchers avoid uncertainty due to the added complexity in constructing the robust optimisation model, and to the lack of knowledge of the nature of the uncertainty, especially its propagation, our work focuses on developing a robust convex optimisation solution along with estimating the uncertainties in every step of the algorithm, starting from uncertainties in features positions. First, we propose a technique that minimises the errors by incorporating these uncertainties from all sources and their propagation to the rotations, to the translations and to their corresponding 3D scene points, using robust L_∞ convex optimisation via the second-order cone programming (SOCP). Secondly, we propose to use the robust least squares solution via the SOCP as well, in dealing with system uncertainties for frame-

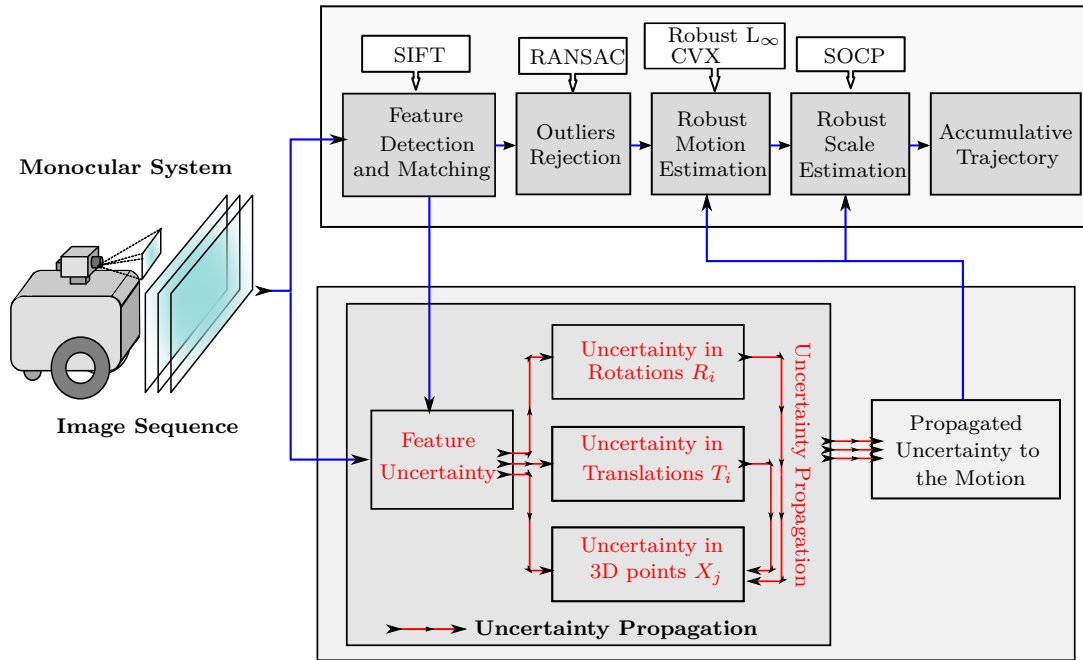


Fig. 6.1 A block diagram showing the main architecture of the proposed solution.

to-frame absolute scale estimation. The proposed solution, which is geometrically meaningful, does not possess any element of randomisation, in which the global optimality is ensured since its local minimum is guaranteed, by definition, to be the global minimum.

Thus, our first set of contributions in this chapter consists of estimating robust and globally optimal solutions to problems that arise in monocular motion estimation using robust convex optimisation under the L_∞ norm. Our goal is to provide a solution with a priori guaranteed feasibility when the uncertain problem parameters vary within the approved uncertainty set defined after propagation. Implementation of these techniques is conducted on real challenging data collected in our autonomous system laboratory, in an urban environment and data gathered from a Mars/Moon analogue site.

The proposed solution, described in Figure 6.1, assumes a fully calibrated system with known intrinsic parameters K . Using a vehicle equipped with a single camera, capturing sequence of images, the final goal is to estimate the camera pose at each time step, relying only on these images and incorporating the uncertainties. Thus, the main steps of the proposed algorithm are:

- Extraction of image feature points using the SIFT/Harris detector and estimating their uncertainties.

- Estimating the initial relative rotations R_i and the translation T_i via the essential matrix.
- Estimation of the propagated feature uncertainties to R_i and T_i through the normalised eight-point algorithm and the SVD.
- Estimation of the 3D scene points using convex optimisation along with their uncertainties.
- Optimising the motion using robust L_∞ convex optimisation, taking in consideration all sources of uncertainty with a sequence of camera resectioning/triangulation.
- Computing the unknown absolute scale ratio using the robust least squares algorithm via the SOCP solution.

Details about each block of the diagram illustrated in Figure 6.1, are given in the coming sections. The basic concepts of the robust convex optimisation are discussed in the next section. Then, we describe the propagation of uncertainties in feature positions to the rotation matrices and to the translation vectors. After that, we present the techniques used in estimating the uncertainties in the reconstructed 3D points. Robust L_∞ motion estimation and robust scale estimation algorithms are detailed in last two sections of this chapter before giving the main conclusions.

6.3 Robust convex optimisation

In general, optimisation is important in engineering and control design, where most applications assume a complete knowledge of the problem data. However, most optimisation problems deal in fact with data that are not completely known, and without taking their uncertainties into consideration. Two main sources of uncertainty exist: data which are not exactly known or cannot be exactly measured, and inherent inaccuracy of the devices used in the applications [20]. This uncertainty results in uncertain constraints and objective functions.

As we have seen in Section 3.8 (Chapter 3, page 72), robust optimisation is a recent alternative to the optimisation under uncertain data, where the uncertainty model is not stochastic, but rather deterministic. In this optimisation, instead of recovering the solution in some probabilistic sense under stochastic uncertainty, the optimiser builds a solution that is optimal for any realisation of the uncertainty in a given set [18]. In cases where the optimality of a solution is affected by the uncertainty, the robust optimisation main goal will be then to seek a solution that performs relatively well for any value taken by the unknown coefficients. While a common approach is to optimise the worst-case objective, more studies are conducted toward other robustness methods [65].

Robust optimisation, in general form, deals with two sets of entities, decision variables and uncertain variables. In this context, the first aim of worst-case robust optimisation is to recover the optimal solution on the decision variables such that the worst-case is minimised and the constraints are robustly feasible, while the uncertainty is allowed to take arbitrary values in a defined uncertainty set [116]. The optimal solution is evaluated using the realisation of the uncertainty that is most unfavourable [65]. The general form of this robust optimisation is given by:

$$\begin{aligned} \min_x \quad & \max_{\omega} f(x, \omega) \\ \text{subject to} \quad & g(x, \omega) \leq 0; \forall \omega \in \mathcal{W} \end{aligned} \quad (6.1)$$

where ω is the uncertain variables, \mathcal{W} is the uncertainty set and x is the decision variables.

In the main part of this solution, we will be dealing with convex optimisation problems for monocular motion estimation, for which the data are uncertain and known to belong to a given uncertainty set \mathcal{W} . These problems can be efficiently recast and solved using second-order cone programming (SOCP). A SOCP constraint, which is a conic quadratic constraint, is of the form:

$$\|A_i x + b_i\|_2 \leq c_i^\top x + d_i, \quad A_i \in \mathbb{R}^{(m) \times n} \quad b_i \in \mathbb{R}^m \quad c_i \in \mathbb{R}^n \quad d_i \in \mathbb{R} \quad (6.2)$$

where x is the variable vector and A_i, b_i, c_i and d_i are the constraints parameters [94, 101]. The robust counterpart is the problem of finding x such that:

$$\begin{aligned} \min_x \quad & f^\top x \\ \text{subject to} \quad & \|A_i x + b_i\|_2 \leq c_i^\top x + d_i, \quad \forall (A_i, b_i, c_i, d_i) \in \mathcal{W}_i \\ & g_i^\top x = h_i \text{ for } i = 1, \dots, p \end{aligned} \quad (6.3)$$

Boni et al. showed in [20] that a convex quadratic constraint with ellipsoidal uncertainty errors can be implemented as a system of conic quadratic constraint. In addition, they showed that a conic quadratic constraint with ellipsoidal uncertainty error can be reformulated as a set of nearly conic quadratic constraints.

6.4 Uncertainty propagation

The expression of *uncertainty* refers to doubt, and in the measurement framework, it stands for doubt about the validity of the measurement results. This measurement uncertainty characterises the dispersion of the values, that could be associated to the

quantity being measured (the measurand) [91]. This uncertainty in reality reflects the lack of knowledge of the value of the measurand. More precisely, the uncertainty indicates the upper and the lower values that an uncertain variable may assume; after all systematic biases have been corrected.

Estimating the uncertainty could be not sufficient in some vision applications. One of the most important tasks is to evaluate its propagation through a particular model. To illustrate that, let us consider a system with inputs $X = (X_1, \dots, X_m)^\top$, and outputs $Y = (Y_1, \dots, Y_n)^\top$, where:

$$Y = f(X), \quad f = (f_1, f_2, \dots, f_m)^\top \quad (6.4)$$

where f is the measurement model. Given an estimate x of X , then an estimate of Y is:

$$y = f(x) \quad (6.5)$$

The covariance matrix of the dimension $m \times m$ of the output y is:

$$\underline{\Lambda}_y = \begin{bmatrix} u(y_1, y_1) & \cdots & u(y_1, y_m) \\ \vdots & \ddots & \vdots \\ u(y_m, y_1) & \cdots & u(y_m, y_m) \end{bmatrix} \quad (6.6)$$

and given by:

$$\underline{\Lambda}_y = \underline{J}_x \underline{\Lambda}_x \underline{J}_x^\top \quad (6.7)$$

where \underline{J}_x is the input Jacobian matrix, called also the sensitivity matrix of dimension $m \times n$, and given by:

$$\underline{J}_y = \begin{bmatrix} \frac{\partial f_1}{\partial X_1} & \cdots & \frac{\partial f_1}{\partial X_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial X_1} & \cdots & \frac{\partial f_m}{\partial X_n} \end{bmatrix} \quad (6.8)$$

The uncertainty of the output given in (6.7) is estimated in fact through a first order Taylor series approximation [50]:

$$u_y^2 = \sum_{i=1}^m \sum_{j=1}^{m-1} \left(\frac{\partial f}{\partial X_i} \right) \left(\frac{\partial f}{\partial X_j} \right) u(x_i, x_j) \quad (6.9)$$

where $u(x_i, x_j)$ is the covariance of x_i and x_j , and when $i = j$, the $\sqrt{(u(i, x_i))} = u_{x_i}$ is the uncertainty of x_i . Equation (6.9) can be written in a more general form:

$$u_y^2 = \underline{\Lambda}_y = \underline{J}_x \underline{\Lambda}_x \underline{J}_x^\top \quad (6.10)$$

where $\underline{\Lambda}_x$ is the input covariance matrix, and \underline{J}_y is the input Jacobian matrix, namely the matrix of partial derivatives. Equation (6.10) is known as the propagation property of the uncertainty through non-linear systems [50].

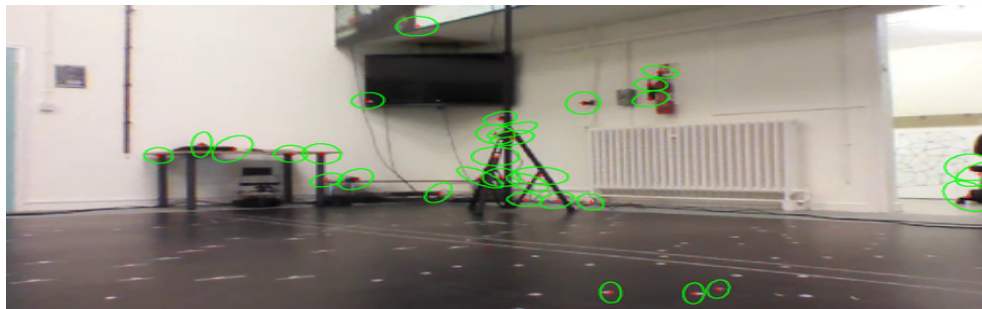
6.5 Feature location uncertainty

Selecting the most appropriate feature extraction technique for any vision application is not obvious. In the literature, a relatively large variety of feature detectors are usually used in such applications. Interestingly, the final performance of these solutions vary from one feature extraction technique to the other. Similarly to the previous solution, in this chapter we investigate the performance of our solution using the Harris corner detector and the SIFT extractor. Since the detected feature points, regardless the nature of the detector, have some uncertainty, the proposed solution in this chapter to estimate the motion uses the robust convex optimisation scheme based on those uncertainties and their propagations.

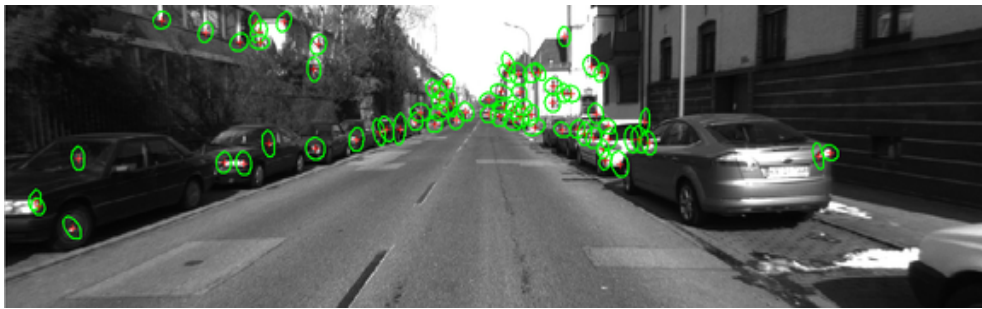
We have introduced in Chapter 5 (Section 5.4, page 105) techniques that can be used to estimate the uncertainty in feature positions from the Harris corner detector and from the SIFT extractor. The derivative approach is used as well in this solution, where the covariance matrix is recovered as the inverse of the Hessian matrix.

Implementation of these techniques is conducted on challenging datasets and shown in Figure 6.2 and Figure 6.3. The first one is collected in our laboratory using a Pioneer P3-DX platform with a fully-calibrated forward-looking camera (Section 1.6, Chapter 1, page 8). The second is gathered from a vehicle travelling in an urban city environment, where a forward-pointing calibrated camera is mounted on this vehicle [68]. The third one is a collection of data from a Mars/Moon analogue site at Devon Island, Nunavut [64].

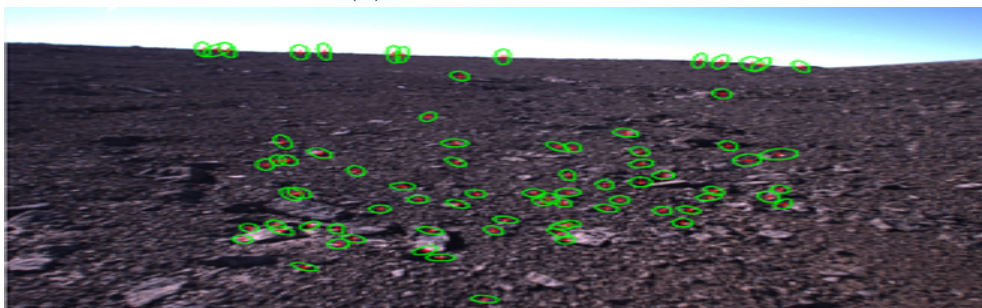
Figure 6.4 and Figure 6.5 show clearly that feature points localisation uncertainties using the Harris corner detector are relatively smaller than those estimated using SIFT for all environments. Results are summarised in Table 6.1 as well. The average error for Harris corner detector in urban environment, for example, is in the order of 0.04 pixels, whereas it reaches 0.15 pixels using the SIFT extractor. In the Moon/Mars analogue environment, and due to its nature, these uncertainties have remarkably increased (0.05 pixels for Harris corner detector and 0.25 pixels for SIFT extractor), which directly affects the subsequent motion estimations. For the indoor environment, the same pattern is recorded. The average errors is 0.03 pixels for the Harris detector and about 0.12 pixels using the SIFT extractor. However, overall, lower errors are noticed here in comparison to the two other environments. This is due to nature of the environment, where features are relatively closer.



(a) Indoor environment

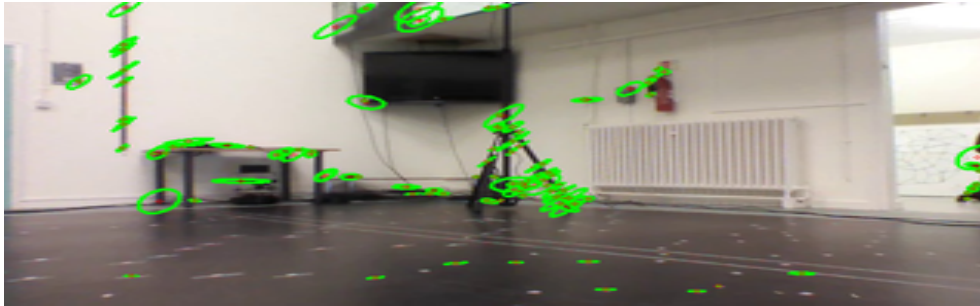


(b) Urban environment



(c) Moon/Mars analogue environment

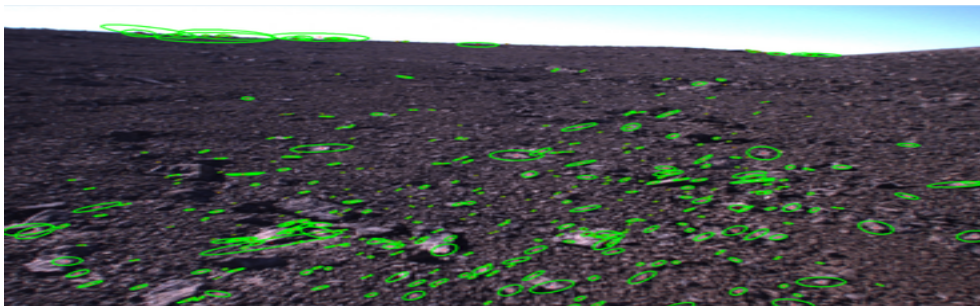
Fig. 6.2 Harris image features with location covariances visualised via error ellipses.



(a) Indoor environment



(b) Urban environment



(c) Moon/Mars analogue environment

Fig. 6.3 SIFT image features with location covariances visualised via error ellipses.

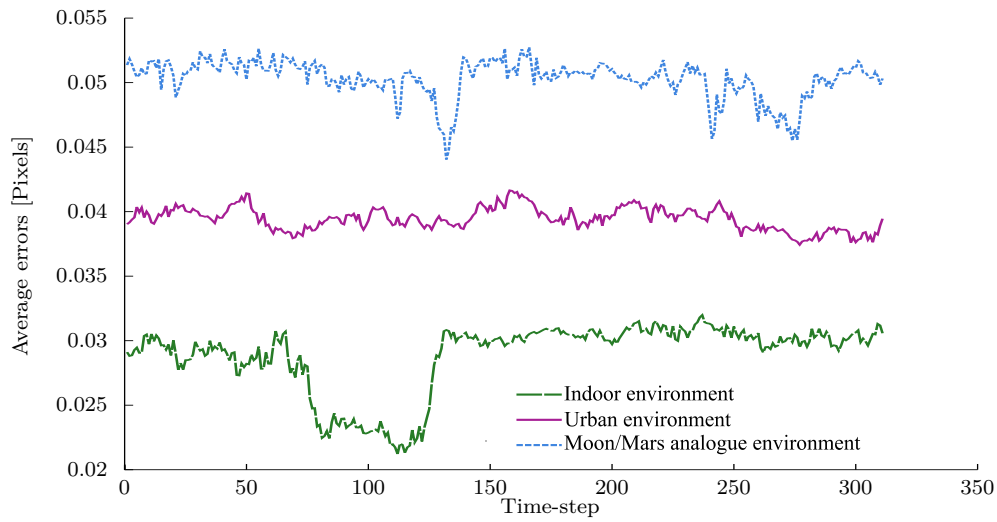


Fig. 6.4 Average feature points localisation errors using Harris.

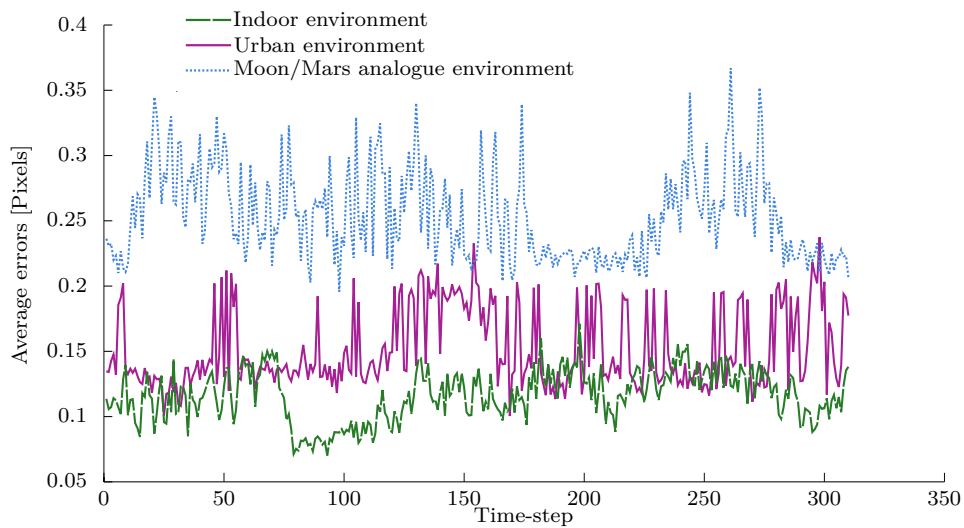


Fig. 6.5 Average feature points localisation errors using SIFT.

Table 6.1 Average localisation errors of the extracted feature points using Harris and SIFT.

	Average localisation errors [pixels]	
	Harris	SIFT
Indoor environment	0.03	0.12
Urban environment	0.04	0.15
Moon/Mars analogue environment	0.05	0.25

Matching in the Harris corner detector algorithm is performed using the cross correlation between local image patches. This means that only features that correlate most strongly with each other in both directions are accepted. Therefore, as a serious drawback, the matching accuracy and robustness, in this algorithm, is completely depending on the actual transformation between views. On the other hand, the SIFT extractor uses the Euclidean distance between two feature point vectors as a similarity criteria of the two keypoints and uses the nearest neighbour algorithm to match each other, which increases significantly its accuracy. Even the matching using the Harris corner detector can be performed with low time consumption, its accuracy is compromised comparing to the high accuracy and robustness matching that is provided by the SIFT algorithm.

By analysing the obtained results shown in Figure 6.4 and Figure 6.5, for the SIFT extractor and the Harris corner detector in all environments, the latter confirms its ability to provide relatively more stable and conservative uncertainty estimations. The SIFT quality is necessary to discard underestimated uncertainty as in the Harris detector, which could influence on the performance of motion estimations. This, in addition to matching accuracy of the SIFT extractor, justify our deployment of the SIFT extractor along with the covariance intersection of their uncertainties in each RGB channel for our monocular motion estimation algorithm.

In addition, one of the important novelty parts of our solution in this chapter is the estimation of the propagated uncertainties from the feature positions to the rotation matrices and the translation vectors and to the 3D scene points. In the following two sections, we introduce the techniques used to estimate these propagated uncertainties.

The uncertainties in the rotation matrices and in the translation vectors are estimated by propagating the feature position uncertainties through the eight-point algorithm and the singular value decomposition (SVD) algorithm. These new uncertainties in the rotations and translations, in addition to the original uncertainties in feature positions, are propagated even more to the 3D scene points through the triangulation algorithm.

Our robust optimisation motion estimation algorithm takes these propagated uncertainties at each stage, in order to robustly estimate the camera trajectory (as shown in Figure 6.1).

6.6 Uncertainty in the rotation matrix and the translation vector

We have seen in Chapter 2, that the entity encoding the translation and rotation comprising the 3D motion is the essential matrix, E . This matrix is defined by $E = [T]_{\times} R^1$, where T and R represent respectively the translation vector and the rotation matrix. In calibrated vision systems, this matrix is deduced from the fundamental matrix F , which is estimated via the eight-point algorithm, followed by an SVD estimation process. Starting from the estimated uncertainties in feature points, and before estimating the propagated uncertainties to T and R , it is necessary to estimate first, the uncertainties in the fundamental matrix and then the uncertainties in the essential matrix. For that, we present in this section techniques adopted in estimating these propagated uncertainties.

6.6.1 Uncertainty estimation in the fundamental matrix

Given two views with camera matrices P_i and P_j , a pair of matching image points $x_i \leftrightarrow x_j$ must satisfy: $x_j^T F x_i = 0$, where F is the fundamental matrix (Appendix C, Section C.1, page 293). Estimating the fundamental matrix is the key stage for any motion estimation algorithm, where information has to be retrieved from several images as a unique source. Since F is defined to a scale factor, it can be estimated with only eight correspondences [189]. Since the subsequent motion estimation steps rely heavily on the estimation of this matrix, a rational attention on recovering its parameters should be paid. Indeed, estimating an optimal F is a hard task, since point locations are noisy and the correspondences are spoilt by outliers. RANSAC (Algorithm 2, page 296) is a well-known robust statistics solution for this type of problems [58].

Unfortunately, RANSAC and similar solutions are able to detect outliers, but the inaccuracy in the image point locations is still not estimated. In the literature, two main methods are used for estimating the covariance of F : Monte-Carlo simulations, and the derivation of a closed-form formula. The uncertainty estimation method adopted in this work was originally introduced in [189].

¹ $[T]_{\times}$ is the 3×3 skew-symmetric matrix of T as defined in Section 2.2 in Chapter 2, page 24.

It is well known that for each pair of images I_i and I_j , point correspondences $\mathbf{x}_i \leftrightarrow \mathbf{x}_j$, where $\mathbf{x}_i = (x_i, y_i, 1)^\top$ and $\mathbf{x}_j = (x_j, y_j, 1)^\top$, the 3×3 fundamental matrix F can be derived from the following system equation:

$$\mathbf{x}_j F \mathbf{x}_i^\top = (x_j, y_j, 1)^\top \begin{bmatrix} f_1 & f_2 & f_3 \\ f_4 & f_5 & f_6 \\ f_7 & f_8 & f_9 \end{bmatrix} (x_i, y_i, 1) = 0 \quad (6.11)$$

The epipolar geometry constraint, that is,

$$\forall k \in [1, n], \mathbf{x}_j^\top F \mathbf{x}_i = 0$$

simply leads to the matrix equation $\mathbf{M}\mathbf{f} = 0$, where \mathbf{M} is given in (5.21), and the column vector \mathbf{f} contains the nine elements of F , where:

$$\mathbf{f} = (f_1, f_2, f_3, f_4, f_5, f_6, f_7, f_8, f_9)^\top.$$

The well-known method for solving the problem: $\mathbf{M}\mathbf{f} = 0$ is by using singular value decomposition (SVD) of \mathbf{M} and putting the smallest singular value of F to zero (rank 2 constraint enforcement) (Theorem 2 - page 41). This makes it necessary to choose F in a (generally) 1-dimensional singular. To do that, we choose to set $f_9 = 1$. Obviously, data are still corrupted with noise; therefore recovering an exact solution is not possible. All standard solutions for solving the matrix equation $\mathbf{M}\mathbf{f} = 0$ ignore the uncertainties in the feature locations, even the well-known eight-point algorithm.

The matrix F is then given as the solution of the linear system $\tilde{\mathbf{M}}\tilde{\mathbf{f}} = \mathbf{c}$, where $\tilde{\mathbf{M}}$ is the sub-matrix of \mathbf{M} containing the first eight columns, $\tilde{\mathbf{f}}$ is equal to \mathbf{f} without the last element f_9 and the 8×1 vector $\mathbf{c} = -[\mathbf{1}_8]^\top$. In this case, we recover F by solving:

$$\tilde{\mathbf{f}} = \tilde{\mathbf{M}}^{-1} \mathbf{c} \quad (6.12)$$

Using the propagation property of the uncertainty through non-linear systems given in (6.10), a first order approximate gives an approximation of the covariance matrix $\Lambda_{\tilde{\mathbf{f}}}$ of the system in (6.12). This approximation is given by:

$$\Lambda_{\tilde{\mathbf{f}}} = \mathbf{J}_{\mathbf{X}} \Lambda_{\mathbf{X}} \mathbf{J}_{\mathbf{X}}^\top, \quad (6.13)$$

Note that (6.13) involves the estimation of the Jacobian $\mathbf{J}_{\mathbf{X}}$ of this transformation from \mathbf{X} to F , where \mathbf{X} is a vector containing the entries of \mathbf{M} . i.e. $\mathbf{X} = (x_{i_1}, y_{i_1}, x_{j_1}, y_{j_1}, x_{i_2}, y_{i_2}, x_{j_2}, y_{j_2}, \dots, x_{i_n}, y_{i_n}, x_{j_n}, y_{j_n})^\top$. The method used to compute this Jacobian is given in the following section. The quantity $\Lambda_{\mathbf{X}}$ is the covariance

matrix, representing the feature position uncertainties, before propagation, and estimated using technique presented in Chapter 5 (Section 5.4, page 105).

Computation of $\mathbf{J}_\mathbf{X}$, the Jacobian of $\tilde{\mathbf{f}}$:

Equation (6.12) gathers three transformations from the vector \mathbf{X} , which contains the coordinates of the matched points to the vector $\tilde{\mathbf{M}}^{-1}\mathbf{c}$. ($\mathbf{X} \mapsto \tilde{\mathbf{M}}^{-1}\mathbf{c}$). These three transforms are : $\Theta : A \mapsto A\mathbf{c}$, $\Psi : A \mapsto A^{-1}$ and $\Phi : \mathbf{X} \mapsto \tilde{\mathbf{M}}$, hence, $\tilde{\mathbf{f}} = \Theta \circ \Psi \circ \Phi(\mathbf{X})$ and the 8×32 Jacobian matrix of the transform is given by:

$$\mathbf{J}_\mathbf{X} = \mathbf{J}_\Theta \cdot \mathbf{J}_\Psi \cdot \mathbf{J}_\Phi(\mathbf{X}) \quad (6.14)$$

where:

- $\mathbf{J}_\Phi(\mathbf{X})$ is a 64×32 matrix containing the derivative of \mathbf{M} with respect to all element of X ;
- $\mathbf{J}_\Psi(\mathbf{X})$ is a 64×64 matrix where its entries are calculated as follow: for any element $\mathbf{J}_\Psi(i, j) = -A^{-1}\Gamma_{ij}A^{-1}$ where Γ_{ij} is the matrix with entries equal to 0 except in position (i, j) which is 1;
- $\mathbf{J}_\Theta(\mathbf{X})$ is a 8×64 constant matrix.

After estimating the Jacobian $\mathbf{J}_\mathbf{X}$, then the covariance matrix $\tilde{\mathbf{f}}$ is estimated using (6.13).

Computation of the Jacobian of the SVD:

Since the fundamental matrix must be enforced to have a rank equals to two, which is done using the SVD, then the computation of the Jacobian of this SVD is necessary. If $F = UDV^\top$, is the SVD decomposition of F ; then imposing $\text{rank}(F) = 2$ is simply performed by putting the element the last element of D to 0 or putting [152]:

$$F = UD \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} V^\top \quad (6.15)$$

Let f_{ij} be the $(i, j)^{th}$ element of F , then the entries of the 8×8 Jacobian matrix of the SVD, $\mathbf{J}_{\text{SVD}}(F)$, are then given by:

$$\frac{\partial F}{\partial f_{ij}} = \frac{\partial U}{\partial f_{ij}} D \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} V^\top + U \frac{\partial D}{\partial f_{ij}} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} V^\top + UD \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \frac{\partial V^\top}{\partial f_{ij}} \quad (6.16)$$

To estimate the derivatives $\frac{\partial U}{\partial f_{ij}}$, $\frac{\partial D}{\partial f_{ij}}$ and $\frac{\partial V}{\partial f_{ij}}$ in (6.16), we use the method introduced in [152]. Note that, $\forall (k, l) \neq (i, j)$, $\frac{\partial f_{kl}}{\partial f_{ij}} = 0$ and $\frac{\partial f_{ij}}{\partial f_{ij}} = 1$.

Since U in (6.15) and (6.16) is an orthogonal matrix, then:

$$U^\top U = I \Rightarrow \frac{\partial U^\top}{\partial f_{ij}} U + U^\top \frac{\partial U}{\partial f_{ij}} = \Omega_U^{ij\top} + \Omega_U^{ij} = 0 \quad (6.17)$$

where:

$$\Omega_U^{ij} = U^\top \frac{\partial U}{\partial f_{ij}} \quad (6.18)$$

Similarly to (6.15) and (6.16) we can get from (6.14):

$$\Omega_V^{ij} = \frac{\partial V^\top}{\partial f_{ij}} V \quad (6.19)$$

Notice that from (6.18) and (6.19) Ω_U^{ij} and Ω_V^{ij} are antisymmetric matrices. By multiplying (6.16) by U^\top and V from the left and right respectively, and (6.18) and (6.19), the following relation is obtained:

$$U^\top \frac{\partial F}{\partial f_{ij}} V = \Omega_U^{ij} D \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \frac{\partial D}{\partial f_{ij}} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} + D \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \Omega_V^{ij} \quad (6.20)$$

The derivatives of the singular values in (6.20) are given by

$$\frac{\partial D_{kk}}{\partial f_{ij}} = U_{ik} V_{jk} \quad (6.21)$$

where D_{kk} are the diagonal elements of D . Now, for computing $\frac{\partial U}{\partial f_{ij}}$ and $\frac{\partial D}{\partial f_{ij}}$, we first compute Ω_U^{ij} and Ω_V^{ij} . A set of 2×2 linear systems are used to compute these matrices using the off-diagonal entries of the matrices in (6.20):

$$\begin{aligned} D_{ll} \Omega_U^{ij} + D_{kk} \Omega_V^{ij} &= U_{ik} V_{jl} \\ D_{kk} \Omega_U^{ij} + D_{ll} \Omega_V^{ij} &= -U_{ik} V_{jl} \end{aligned} \quad (6.22)$$

where the index ranges are $k = 1, \dots, 3$ and $l = k + 1, \dots, 3$. Then :

$$\frac{\partial U}{\partial f_{ij}} = U \Omega_U^{ij} \quad \text{and} \quad \frac{\partial V}{\partial f_{ij}} = -V \Omega_V^{ij} \quad (6.23)$$

After having estimated $\frac{\partial U}{\partial f_{ij}}$, $\frac{\partial D}{\partial f_{ij}}$ and $\frac{\partial V}{\partial f_{ij}}$ we can compute J_{SVD} from (6.16).

The covariance of the fundamental matrix:

After estimating all these quantities, and,

- since the coefficient f_9 is fixed;

- and after estimating $\Lambda_{\tilde{\mathbf{f}}}$, the covariance of the sub-matrix $\tilde{\mathbf{f}}$ using (6.13);
- and after estimating J_{SVD} , the Jacobian of the SVD of F using (6.16).

then, the 9×9 covariance matrix of the fundamental matrix F is given by:

$$\Lambda_F = J_{\text{SVD}} \begin{bmatrix} \Lambda_{\tilde{\mathbf{f}}} & \mathbf{0}_{8,1} \\ \mathbf{0}_{1,8} & 0 \end{bmatrix} J_{\text{SVD}}^\top \quad (6.24)$$

6.6.2 Uncertainty estimation in the essential matrix

We have seen in Chapter 2 that the relative translation, T , and rotation, R , are encoded in the essential matrix, E . This matrix is given by $E = [T]_\times R$. In that chapter, we showed that the fundamental matrix can be used in estimating this essential matrix, as [82]:

$$E = K_j^\top F K_i \quad (6.25)$$

where K is the camera calibration matrix. Given, Λ_F , the covariance of the fundamental matrix from (6.24), and using (6.10), then, the covariance of the essential matrix, E , Λ_E , can be computed as:

$$\Lambda_E = \frac{\partial (K_j^\top F K_i)}{\partial F} \Lambda_F \frac{\partial (K_j^\top F K_i)}{\partial F}^\top \quad (6.26)$$

The derivative of $K_j^\top F K_i$ with respect to F_{ij} , an element of F , is given by:

$$\frac{\partial (K_j^\top F K_i)}{\partial F} = K_j^\top \frac{\partial F}{\partial F} K_i \quad (6.27)$$

where $\frac{\partial F}{\partial F}$, the derivative of F , is a matrix whose all its entries are zeros, apart from from those in column j and row i , which are ones.

6.6.3 Uncertainty estimation in the rotation and translation

After estimating the uncertainty of the essential matrix, we move now to the estimation of the uncertainty in the rotation R , and the translation T . As stated in Section 2.9.2, given the essential matrix $E = U \text{diag}(1, 1, 0) V$ and making the first camera matrix $P_i = [I|0]$, then the four possible choices for the second camera matrix P_j are [82]:

- $P_j = [R_1 | +T]; P_j = [R_1 | -T];$
- $P_j = [R_2 | +T]; P_j = [R_2 | -T].$

where the rotation matrices $R_1 = U W V^\top$ and $R_2 = U W^\top V^\top$, the translation vector $T = U(0, 0, 1)^\top$ is the last column of U and $W = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$.

Estimation of the Covariance of the rotation matrix:

The uncertainty in the rotation matrix R_1 knowing Λ_E , the uncertainty of E , is given by:

$$\Lambda_{R_1} = \frac{\partial(UWV^\top)}{\partial E} \Lambda_E \frac{\partial(UWV^\top)^\top}{\partial E} \quad (6.28)$$

For the estimation of the derivative $\frac{\partial(UWV^\top)}{\partial E}$, we can use the technique presented in Section 6.6.1 in the part: Computation of the Jacobian of the SVD. The same technique is used to estimate the covariance of R_2 , where we use W^\top instead of W .

Estimation of the covariance of the translation vector:

The translation vector T is given by $u_3 = U(0, 0, 1)^\top$, the last column of U . Hence, its covariance Λ_T is then simply:

$$\Lambda_T = \frac{\partial(u_3)}{\partial E} \Lambda_E \frac{\partial(u_3)^\top}{\partial E} \quad (6.29)$$

Again, we can use the procedure presented in Section 6.6.1 in the part: Computation of the Jacobian of the SVD to estimate the derivative $\frac{\partial(u_3)}{\partial E}$.

6.7 Feature uncertainty propagation to the 3D reconstruction

Assume we have m views of a scene point \hat{X} , which maps to image points $\hat{x}_i = (\mathbf{x}^\top, 1)^\top = (u_i, v_i, 1)^\top$ via the camera matrices P_i , reconstruction is to recover the 3D space position of points \hat{X} such that $\hat{x}_i^\top = P_i \hat{X}$, for $i = 1, \dots, m$. These quantities are related by the projection function f , where:

$$\begin{aligned} f_u(R, T, \mathbf{x}_i, \mathbf{X}) &= u_i - \frac{r_{i1}^\top \mathbf{X} + t_{i1}}{r_{i3}^\top \mathbf{X} + t_{i3}} = 0 \\ f_v(R, T, \mathbf{x}_i, \mathbf{X}) &= v_i - \frac{r_{i2}^\top \mathbf{X} + t_{i2}}{r_{i3}^\top \mathbf{X} + t_{i3}} = 0 \end{aligned} \quad (6.30)$$

The camera matrices, P_i , are given by: $P_i = [R_i | T_i]$, where $R_i = [r_{i1}, r_{i2}, r_{i3}]$ and $T_i = [t_{i1}, t_{i2}, t_{i3}]$ are the rotation matrix and the translation vector respectively. Note that $\hat{X} = [\mathbf{X}, 1]$ is represented by homogeneous coordinates ($\mathbf{X} = [X, Y, Z] \in \mathbb{R}^3$). In a calibrated vision system, normalised camera matrices are used, where $P_1 = [I | 0]$, and set as a reference camera, and $P_2 = [R_2 | T_2]$. In this case, a system composed of four equations can be obtained from (6.30).

Let Λ_X be the covariance matrix of the scene point X , and $\Lambda_{In} = \text{diag}(\Lambda_{x_i}, \Lambda_{R_i}, \Lambda_{T_i})$ is the diagonal covariance matrix associated with the input parameters vector: $In = [x_i, R, T]^\top$, which is estimated using techniques presented in Section 6.6. Then, the following expression can be written to model the covariance propagation [35]:

$$J_X \Lambda_X J_X^\top = J_{In} \Lambda_{In} J_{In}^\top \quad (6.31)$$

where J_X and J_{In} are the Jacobian matrices of derivatives of f in (6.30) with respect to the 3D scene point X and the input parameters vector $In = [x_i, R, T]^\top$ respectively:

$$J_X = \begin{bmatrix} \frac{\partial f_{u_1}(R, T, x_1, X)}{\partial X} & \frac{\partial f_{u_1}(R, T, x_1, X)}{\partial Y} & \frac{\partial f_{u_1}(R, T, x_1, X)}{\partial Z} \\ \frac{\partial f_{v_1}(R, T, x_1, X)}{\partial X} & \frac{\partial f_{v_1}(R, T, x_1, X)}{\partial Y} & \frac{\partial f_{v_1}(R, T, x_1, X)}{\partial Z} \\ \frac{\partial f_{u_2}(R, T, x_2, X)}{\partial X} & \frac{\partial f_{u_2}(R, T, x_2, X)}{\partial Y} & \frac{\partial f_{u_2}(R, T, x_2, X)}{\partial Z} \\ \frac{\partial f_{v_2}(R, T, x_2, X)}{\partial X} & \frac{\partial f_{v_2}(R, T, x_2, X)}{\partial Y} & \frac{\partial f_{v_2}(R, T, x_2, X)}{\partial Z} \end{bmatrix}$$

$$J_{In} = \begin{bmatrix} J_{x_1} & 0 & J_{RT_1} & 0 \\ 0 & J_{x_2} & 0 & J_{RT_2} \end{bmatrix}$$

where:

$$J_{x_i} = \begin{bmatrix} \frac{\partial f_{u_1}(R, T, x_1, X)}{\partial u_i} & \frac{\partial f_{u_1}(R, T, x_1, X)}{\partial v_i} \\ \frac{\partial f_{v_1}(R, T, x_1, X)}{\partial u_i} & \frac{\partial f_{v_1}(R, T, x_1, X)}{\partial v_i} \end{bmatrix}$$

$$J_{RT_i} = \begin{bmatrix} \frac{\partial f_{u_i}}{\partial R_{i11}} & \frac{\partial f_{u_i}}{\partial R_{i12}} & \cdots & \frac{\partial f_{u_i}}{\partial R_{i33}} & \frac{\partial f_{u_i}}{\partial T_{i1}} & \cdots & \frac{\partial f_{u_i}}{\partial T_{i3}} \\ \frac{\partial f_{v_i}}{\partial R_{i11}} & \frac{\partial f_{v_i}}{\partial R_{i12}} & \cdots & \frac{\partial f_{v_i}}{\partial R_{i33}} & \frac{\partial f_{v_i}}{\partial T_{i1}} & \cdots & \frac{\partial f_{v_i}}{\partial T_{i3}} \end{bmatrix}$$

The output covariance matrix and, thus, the output uncertainties are given by:

$$\Lambda_X = J_X^\dagger (J_{In} \Lambda_{In} J_{In}^\top) J_X^{\top\dagger} \quad (6.32)$$

where $J_X^\dagger = (J_X^\top J_X)^{-1} J_X^\top$ is the pseudo inverse matrix of J_X .

Implementation of 3D scene point uncertainty estimation:

An extensive experimental validation has been conducted for the developed uncertainty propagation given above. Figure 6.6 shows the 3D scene points uncertainty patterns of a sequence of frames against their respective depths. It can be clearly seen from this figure that closer points have smaller uncertainties than relatively distant points from the camera. Therefore, tracked points over long sequence of images will have decreasing uncertainties as the vehicle approaches to them. Note that while these uncertainties are dependent on the 3D position, they are completely

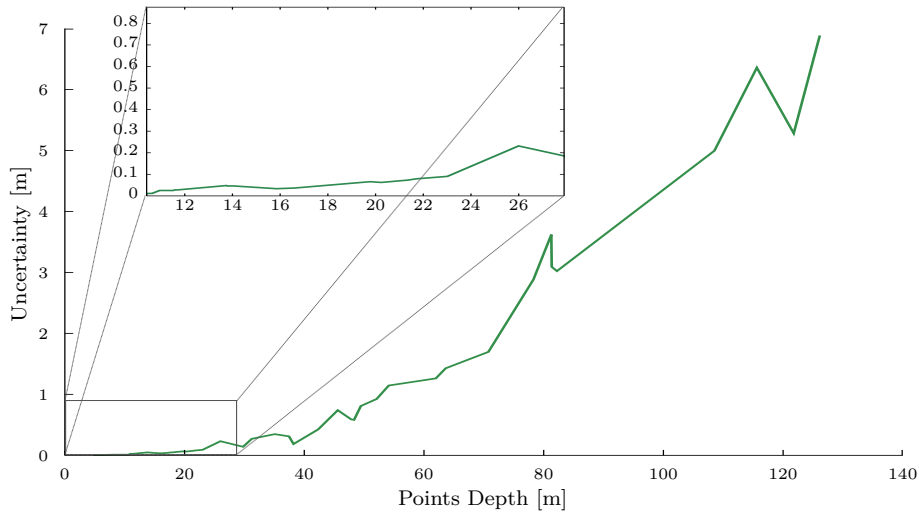


Fig. 6.6 3D points uncertainties against their depths.

independent of their images coordinates (u_i, v_i) . Indeed, selecting relatively closer points would be better for good robust motion estimation.

As a summary of Sections 6.6 and 6.7:

- Feature position uncertainty estimation is given in (5.10) for the SIFT extractor and in (5.9) for the Harris corner detector (Chapter 5);
- The propagation of this uncertainty to the rotation matrix is given in (6.28);
- The propagation of this uncertainty to the translation vector is given in (6.29);
- The propagated uncertainty to the reconstructed 3D points is given in (6.32).

6.8 Robust L_∞ motion estimation solution

After having estimated Λ_x , the uncertainties in features position from (5.9) and (5.10), and the uncertainties in the initial rotations and translations Λ_R and Λ_T from (6.28) and (6.29) respectively, robust L_∞ motion estimation is then performed. For each consecutive image pair, the whole algorithm can be depicted in the following operations:

- Estimating the 3D position for each point X_i using robust convex L_∞ triangulation and including the covariance matrices Λ_{x_i} , Λ_T and Λ_R (Section 6.8.2);
- Estimating the covariance matrix Λ_{X_i} for each recovered 3D scene point X_i using (6.32) (Section 6.7);

- Estimating the camera parameters using robust convex L_∞ resectioning and including the covariance matrices Λ_{x_i} and Λ_{X_i} (Section 6.8.3).

These operations are repeated until the L_∞ re-projection errors reach a satisfactory minimum.

6.8.1 Computational cost

It is known that the computational complexity of the Levenberg-Marquardt algorithm is: $\mathcal{O}((m+n)^3)$, where m is the number of cameras and n is the number of 3D scene points [82]. In our algorithm, at a given time step, we will be dealing with either the triangulation problem or the camera parameters recovering problem for which they are solved using a bisection algorithm for feasibility checks. The triangulation problem has a computational complexity of $\mathcal{O}(m^{1.5})$ and a memory requirement of $\mathcal{O}(m)$, where m is the number of cameras in which the triangulating point is visible [130, 135]. Similarly, the camera parameters recovering problem has a computational complexity of $\mathcal{O}(n^{1.5})$ and a memory requirement of $\mathcal{O}(n)$, where n is the number of scene points. Therefore, for one whole update of the proposed solution, when dealing with n scene points and m cameras, the computational complexity is given by: $\mathcal{O}(mn(\sqrt{m} + \sqrt{n}))$ and the memory requirement is $\mathcal{O}(\max(m, n))$ [130].

The number of iterations is crucial in this kind of problems, where a bisection search is performed. In our solution, defining the initial diameter of the second order cones plays a very important role in determining the number of the required iterations. Hence, the upper and the lower parameters of the bisection algorithm are chosen so the search area is reduced. A memorable search, based on the previous iteration parameters is performed, where these parameters are chosen in relation with the previous iteration results. This technique significantly reduces the number of iterations and hence the time consumption. This makes our solution comparable to the classical L_2 Bundle Adjustment (BA) in terms of time consumption and in the same time it recovers the global minimum of the cost functions, which is a valuable advantage over the classical L_2 BA.

6.8.2 Robust L_∞ triangulation with uncertain data

In order to provide an optimal solution to subsequent L_∞ robust motion estimation, a need for a robust and efficient optimal triangulation algorithm will be crucial. Our approach is based on robust convex optimisation with L_∞ norm, taking in consideration all sources of uncertainty. Our uncertain data are bounded, therefore, we will be looking for optimal solutions, which are feasible for any realisation of these data.

As described in Section 6.7, the function f in (6.30) relates all parameters of the triangulation between two camera positions P_i , $i = 1, \dots, m$, where m is the number of cameras. In this problem, the aim is to recover the value of X that minimises the maximum of the re-projection error across all images:

$$\varepsilon_i = d(\hat{x}_i, P_i \hat{X}), \quad i = 1, \dots, m \quad (6.33)$$

where d denotes image-space Euclidean distance between two points in the image plane, the measured and the projected points.

Given the camera matrices $P_i = [R_i | T_i]$, where $R_i = [r_{i1}, r_{i2}, r_{i3}]$, $T_i = [t_{i1}, t_{i2}, t_{i3}]$, $\hat{X} = [X, 1]$ and their corresponding image points $\hat{x}_i = (\mathbf{x}^\top, 1)^\top = (u_i, v_i, 1)^\top$; and by considering the uncertainties in the image points positions $\Delta x_i = (\Delta u_i, \Delta v_i)$, in the rotation matrices ΔR_i and in the translation vectors ΔT_i from covariance matrices Λ_{x_i} , Λ_{T_i} and Λ_{R_i} respectively; the L_2 norm of this re-projection error function is given by:

$$F_i(X) = \left\| (u_i + \Delta u_i) - \frac{(r_{i1}^\top + \Delta r_{i1}^\top)X + (t_{i1} + \Delta t_{i1})}{(r_{i3}^\top + \Delta r_{i3}^\top)X + (t_{i3} + \Delta t_{i3})}, (v_i + \Delta v_i) - \frac{(r_{i2}^\top + \Delta r_{i2}^\top)X + (t_{i2} + \Delta t_{i2})}{(r_{i3}^\top + \Delta r_{i3}^\top)X + (t_{i3} + \Delta t_{i3})} \right\|_2 \quad (6.34)$$

It is shown in [80] that this type of cost function clearly has multiple local minima. This creates a problem to iterative minimisation methods, such as the Levenberg-Marquardt (LM) algorithm, as they converge with high probability to a local minimum. To get around this issue, this problem is formulated within a quasi-convex optimisation framework by using the L_∞ norm instead. Then, the projection error will be then given by:

$$G(X) = \max_i F_i(X) \quad (6.35)$$

For a scene point X to be visible as image points x_i , it must obviously lie in front of all cameras P_i , $i = 1, \dots, m$. This implies the constraint $g(X, P_i) > 0$ for all i , where $g(X, P_i) = (r_{i3} + \Delta r_{i3})^\top X + (t_{i3} + \Delta t_{i3})$. Our optimisation problem is then given by:

$$\begin{aligned} \min_X \quad & G(X) \\ \text{subject to} \quad & g(X, P_i) > 0; \forall i = 1, \dots, m \end{aligned} \quad (6.36)$$

The m error residuals in (6.33) give the error vector $\varepsilon = (\varepsilon_1, \dots, \varepsilon_m)^\top$. The estimated scene point is then the vector X that minimises the norm of this error vector. Section 3.7.1 (Chapter 3, page 69) gives a detailed proof that the problem is a quasi-convex optimisation problem. This optimisation problem is solved using a

sequence of robust SOCP feasibility problems similar the problem given in (6.3). This robust SOCP is capable of dealing with the feature uncertainties and recovering a robust solution [116]. This leads toward using a bisection search to find the minimum value of δ , for which the optimisation problem is feasible.

The recovered 3D points using this robust L_∞ triangulation and their uncertainties are then used in the optimisation algorithm for recovering the camera parameters, which is described in the next section.

6.8.3 Robust L_∞ camera resectioning with uncertain data

Camera resectioning problem is detailed in Appendix A (Section A.2, page 281). Firstly, let us assume that n scene points \hat{X}_i are given, with their uncertainties ΔX_i , which are mapped to image points x_i with their uncertainties Δx_i , via the camera projection matrix P . This mapping is written as $\hat{x}_i = P\hat{X}_i$, for $i = 1, \dots, n$. According to Definition 8 (page 281), the problem of camera resectioning is the problem of recovering the 3×4 matrix P , such that $\hat{x}_i = P_i\hat{X}_i$, for all i , that minimises the maximum of the re-projection error across all points:

$$\varepsilon_i = d(\hat{x}_i, P\hat{X}_i), \quad i = 1, \dots, n \quad (6.37)$$

where d denotes again the image-space Euclidean distances between two points in the image plane, the measured and the projected points. By introducing the uncertainties in feature position $\Delta x_i = (\Delta u_i, \Delta v_i)$, and those in their 3D corresponding points ΔX_i ; the L_2 norm of this re-projection error function is given by:

$$F(X) = \left\| \left(u_i + \Delta u_i - \frac{p^1{}^\top(X_i + \Delta X_i)}{p^3{}^\top(X_i + \Delta X_i)}, v_i + \Delta v_i - \frac{p^2{}^\top(X_i + \Delta X_i)}{p^3{}^\top(X_i + \Delta X_i)} \right) \right\|_2 \quad (6.38)$$

where p^j denotes the j^{th} row vector of P .

Again, this cost function is highly non-linear and has multiple local minima, where iterative algorithms, such as Levenberg-Marquardt (LM), can easily get trapped in one of these local minima. Similarly to the triangulation problem, to get around these drawbacks and to recover robust and global solutions, the L_∞ re-projection error is used. The global minimum can be obtained by solving the following optimisation problem:

$$\begin{aligned} \min_X \quad & \max_i F_i(X) \\ \text{subject to} \quad & p^3{}^\top(X_i + \Delta X_i) > 0; \forall i = 1, \dots, n \end{aligned} \quad (6.39)$$

This is again a quasi-convex optimisation problem and can be solved using technique presented in Section A.2 (Appendix A, page 281).

The recovered camera parameters and their uncertainties are then used again in the subsequent robust convex L_∞ triangulation algorithm and so on as detailed in Section 6.8.

6.9 Robust scale estimation

H_∞ filter is adopted for frame-to-frame scale estimation in the solution presented in Chapter 4. Some experimental tests on real data revealed that more accurate estimates can be obtained when the robust least squares approach via the SOCP algorithm is used instead. This is illustrated in Figure 6.10. Therefore, in this chapter, investigating this approach within the robust convex optimisation framework is performed.

More precisely and as detailed Chapter 4, after estimating the camera motion parameters using robust L_∞ convex optimisation under uncertain data, ambiguities in the translation scale still occur. Assuming we have i 3D points \hat{X}_i , which maps to image points $\hat{x}_i = (u_i, v_i, 1)^\top$ via the normalised camera matrix $P = [R|T]$, then [46, 82]:

$$\lambda \hat{x}_i = [R|ST]\hat{X}_i \quad (6.40)$$

where λ is the unknown depth factor that takes into account the projection plane ambiguity. This leads to the problem of finding a solution S to the over determined set of equations given by:

$$\begin{aligned} (t_z u_i - t_x)S &= (r^1 - r^3 u_i)X_i \\ AS &= b \end{aligned} \quad (6.41)$$

where r^i denotes the i^{th} row vector of R and $T = [t_x, t_y, t_z]^\top$. Finding a solution S to this problem in the least squares sense (LS) means minimising the residual $\|\Delta b\|$, subject to $AS = b + \Delta b$ [54]. However, knowing that x_i , X_i , R and T are subject to deterministic perturbations, this solution is expected to exhibit very sensitive behaviour to these perturbations [53]. Therefore, thinking of using more robust estimator would be a valid option to investigate. Thus, in our implementation, robust least squares (RLS) technique is used instead [52]. Indeed, as presented in Section 3.8.2 (Chapter 3, page 74), robust least squares (RLS) solution computes the exact value of the optimal worst-case residuals, using again a convex second-order cone

programming (SOCP) of the general form:

$$\begin{aligned} \min_x \quad & f^\top x \\ \text{subject to} \quad & \|A_i x + b_i\|_2 \leq c_i^\top x + d_i, \quad i = 1, \dots, m. \\ & g_i^\top x = h_i, \quad i = 1, \dots, p. \end{aligned} \quad (6.42)$$

where vectors $x, f, c_i, g_i \in \mathbb{R}^n$, scalars $d_i, h_i \in \mathbb{R}$, matrix $A_i \in \mathbb{R}^{(n_i-1) \times n}$ and $b_i \in \mathbb{R}^{n_i-1}$. The norm $\|\cdot\|_2$ is the standard Euclidean norm $\|u\|_2 = (u^\top u)^{1/2}$. The SOCP that formulates this problem is [52]:

$$\begin{aligned} \min_x \quad & \lambda \\ \text{subject to} \quad & \|AS - b\|_2 \leq \lambda - \tau \\ & \left\| \begin{bmatrix} S \\ 1 \end{bmatrix} \right\|_2 \leq \tau \end{aligned} \quad (6.43)$$

The unique solution to this problem is then given by:

$$S = \begin{cases} (\mu I + A^\top A)^{-1} A^\top b & \text{if } \mu = (\lambda - \tau)/\tau > 0 \\ A^\ddagger & \text{else,} \end{cases} \quad (6.44)$$

where A^\ddagger is the pseudo-inverse matrix of A . The quantities (λ, τ) are the unique optimal solutions for problem (6.43).

6.10 Experimental results

This section presents an experimental evaluation of the proposed solution using robust convex optimisation. As explained previously, in this implementation, uncertainties in feature positions and their propagation to the rotations, to the translations and to the reconstructed 3D scene points have been taken into consideration. Comparison with classical bundle adjustment approach based on the Levenberg-Marquardt algorithm is given as well.

In order to test the proposed solution, data from three different environments are used (Section 1.6, Chapter 1, page 8). The first one consists of data collected in our laboratory, using a Pioneer P3-DX platform with a fully-calibrated looking forward camera. Ground-truth is collected using the OptiTrack motion-capture system, which provides absolute ground truth position information with millimetre accuracy at 100 Hz (Figure 6.7).



Fig. 6.7 Set-up used in our indoor experimental validations.

The second dataset is collected from a vehicle travelling in an urban environment with a forward-pointing calibrated camera mounted on the roof of this vehicle [68]. The third one is a collection of data gathered at a Mars/Moon analogue site at Devon Island, Nunavut [64].

As we can see, the three environments are completely different, so the technique would be tested in unbiased circumstances. The concurrent methods of the proposed solution are those which use iterative optimisation via L_2 norm, hence comparisons with bundle adjustment method based on the Levenberg-Marquardt algorithm on exactly the same data are given.

Thus, in this section, we show first the navigation results, illustrating the motion estimation using robust convex optimisation applied on a vehicle travelling in a variety of environments. Second, the robustness of the proposed solution is investigated, where the solution is tested under different error-level scenarios. A sequence of robust SOCP feasibility problems for the convexity task using SeDuMi toolbox [187] is employed along with the Yalmip toolbox [116] for uncertainties modelling.

6.10.1 Motion estimation

Motion estimation using robust convex optimisation algorithm was integrated into our implementation for each environment. Figure 6.8 shows the performance of both algorithms in each environment. Figure 6.8a shows errors of the indoor experiment, where the robot has performed two loops in the main arena in our laboratory. Figure 6.8b plots errors of a trajectory of more than 350 meters estimated through 200 key-frames from the urban environment, while Figure 6.8c shows errors of a travelled

distance of more than 650 meters recovered through 1000 key-frames taken from the Mars/Moon analogue site.

It can be seen, from the Euclidean distance errors, that robust convex optimisation approach is more accurate in all environments than the classical bundle adjustment using the Levenberg-Marquardt approach. Indeed, convex optimisation has shown its ability to ensure the global minimum in recovering the motion parameters in comparison to iterative methods, where a predefined termination criterion is set, which favours convergence to local minima.

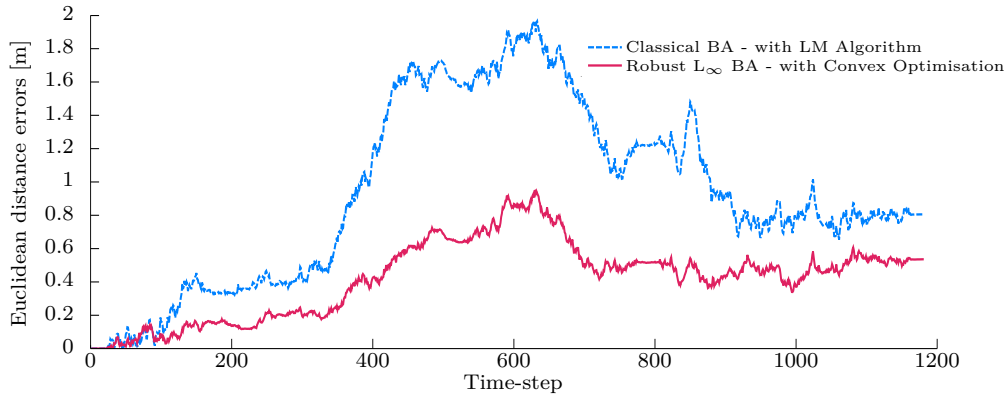
Notwithstanding of the convex optimisation properties, this can be explained as well by the fact that incorporating uncertainties made the optimisation problem more robust, leading to an efficient min-max optimisation in the presence of high level of noise.

In addition, monocular motion estimation using robust algorithm for the scale estimation problem ensures more consistent trajectories. Deployment of robust least squares algorithm under second-order cone programming (SOCP) along with incorporating uncertainties in both feature positions and in their corresponding 3D scene points, has demonstrated its ability to accurately and robustly compute the absolute scale. Figure 6.10 shows that more accuracy is obtained when robust least squares algorithm is used in comparison to results when using the H_∞ filter. This can be seen as well in the recovered trajectories in Figure 6.9. This figure plots the trajectory estimates aligned with their corresponding ground truth. Figure 6.9a plots the trajectory of the indoor experiment, while Figure 6.9b gives the trajectories estimates of a portion of the Moon/Mars analogue dataset. Clearly, more accuracy is provided when robust convex optimisation is used in comparison with classical bundle adjustment with LM algorithm. This is in accordance with the theory as the estimates should be globally optimal.

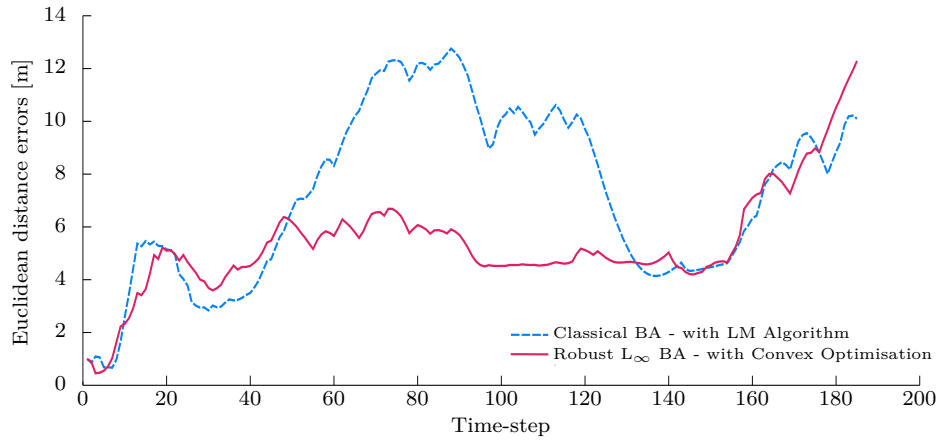
Achieved results are good, reaching errors smaller than 3% and normally bounded by 5 – 12% in terms of travelled error, defined as:

$$\text{Travelled error} = 100 \frac{\text{abs(error)}}{\text{Travelled distance}} \quad (6.45)$$

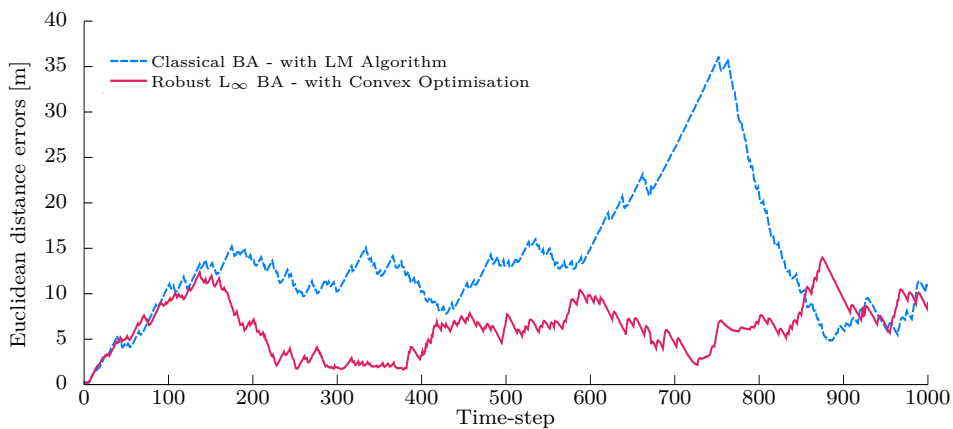
This also shows that our algorithm is suitable for estimating the motion of a vehicle travelling in different environments where high level of noises of unknown nature are likely to occur.



(a) Indoor environment

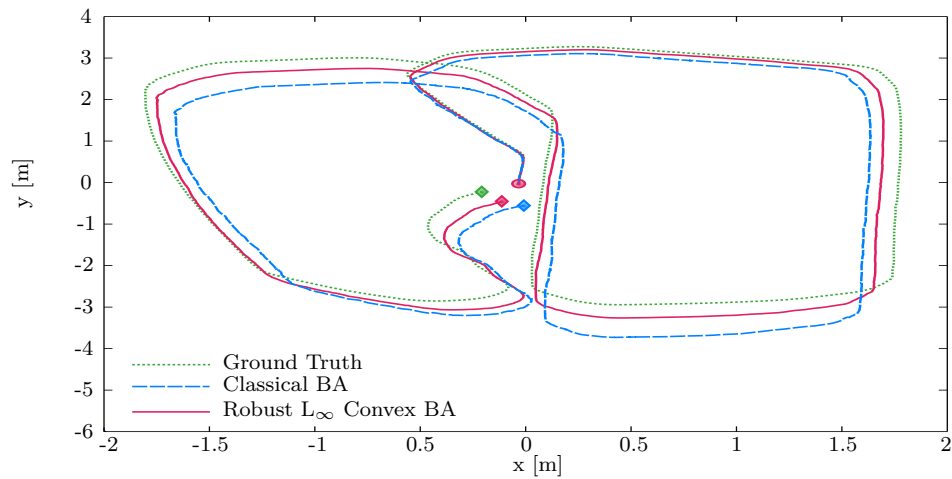


(b) Urban environment

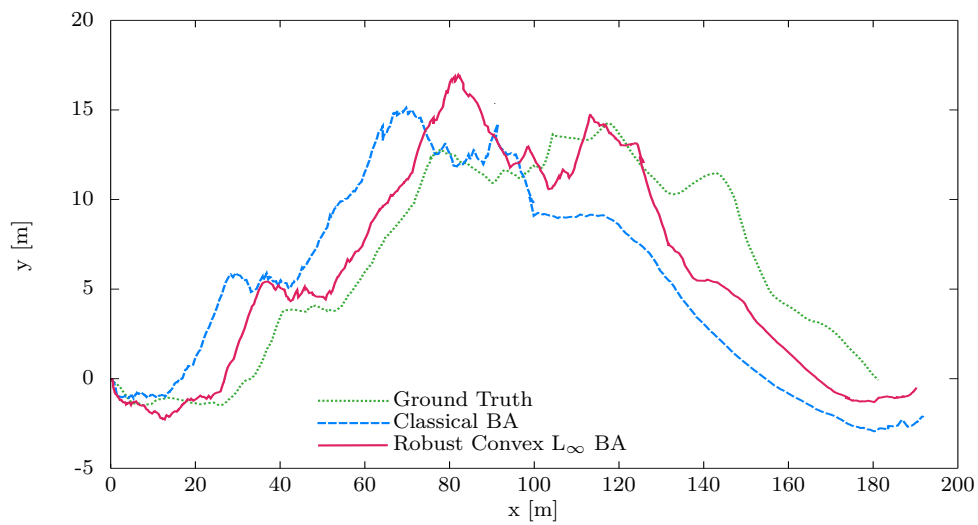


(c) Moon/Mars analogue environment

Fig. 6.8 Camera motion estimation errors.



(a) Indoor environment



(b) Moon/Mars analogue environment

Fig. 6.9 Comparison between the trajectory estimates using robust convex optimisation and the classical BA using LM algorithm.

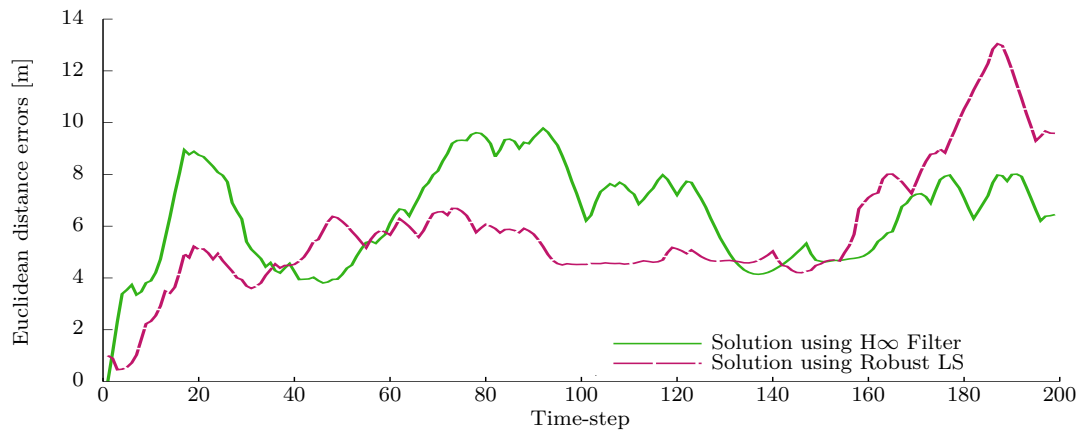


Fig. 6.10 Comparison between using the H_∞ Filter and the robust least squares algorithms in solving the scale ambiguity problem.

6.10.2 Motion estimation robustness

We want to highlight the robustness of the proposed solution and its behaviour and sensitivity to high level of noise. To do that, four scenarios were adopted in function to the noise level (Table 6.2). In fact, we take the extracted image points $x_i = (u_i, v_i)$ using the SIFT algorithm and then perturb them with varying levels of Gaussian noise Δx_i , where $\hat{x}_i = (\hat{u}_i, \hat{v}_i) = x_i + \Delta x_i$. These noisy image points \hat{x}_i are then subsequently used to estimate the camera motion and scene points X_i using the proposed solution (which we called here Robust CVX). The same noisy image points \hat{x}_i are used as well to estimate the camera motion using a L_∞ convex optimisation without taking in consideration the uncertainties (which we called Normal CVX). Therefore, four different scenarios can be distinguished in this experiment as illustrated in Table 6.2.

Table 6.2 The four scenarios for robustness investigation.

	Non Perturbed Image Points x_i	Perturbed Image Points \hat{x}_i
Robust convex optimisation (Robust CVX)	Scenario 1	Scenario 2
Normal convex optimisation (Normal CVX)	Scenario 3	Scenario 4

The estimated camera motions for the four scenarios are aligned with the ground truth and the Euclidean distance errors on the camera position are computed. A plot of these errors is shown in Figure 6.11.

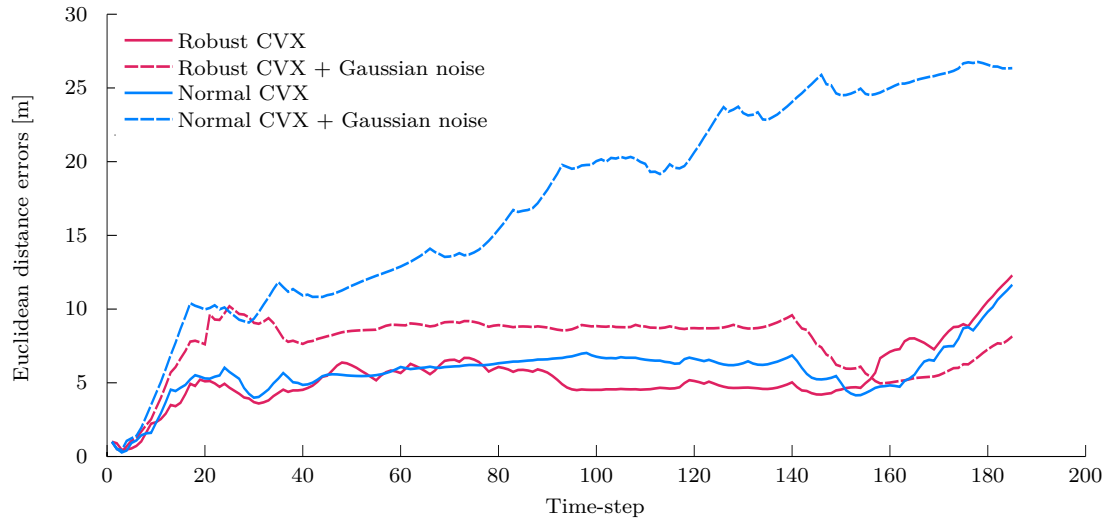


Fig. 6.11 Performance of the motion estimation algorithm as a function of noise level for the four different scenarios.

Table 6.3 Average back projection errors of the estimation of the 3D scene points and the camera parameters of the two real datasets under the four scenarios.

	Average back projection errors		
	Indoor site	Urban site	Moon/Mars Ana. site
Robust CVX	0.4985	0.7189	3.3107
Robust CVX +Gaussian noise	0.5102	0.7281	3.3204
Normal CVX	0.5215	0.7372	3.3401
Normal CVX +Gaussian noise	1.7253	2.0504	7.2978

From these results, we can learn that the proposed method, which takes into consideration the uncertainties of its inputs, performs remarkably better than the normal L_∞ convex optimisation as expected, since it encodes large intervals in its optimisation and models well the uncertainties. In order to assess the robustness in depth, back projection errors of the 3D scene points for the same scenarios are investigated as well. Figure 6.12 plots these re-projection errors of the recovered 3D scene points on the image plane. These errors depict the accuracy of both the 3D position of the scene points and the camera motion parameters. Similar pattern to Figure 6.11 can be noticed here as well. Typically, robust convex optimisation performs well, even when corrupted image points \hat{x}_i are used and in all environments (Figure 6.12 - solid purple graphs). On the other hand, using normal convex optimisation for motion estimation with corrupted image points \hat{x}_i would generate a significant divergence (Figure 6.12 - solid cyan graphs). Logically, both solutions provide similar performances with non-corrupted inputs x_i (dashed purple and cyan graphs). Indeed, it is clear that the uncertainty scheme always produces the best results.

6.11 Conclusions

In this chapter, a robust convex optimisation solution for monocular motion estimation systems has been presented. Including uncertainty estimation, based on the SIFT derivative approach with the developed propagations through the eight-point algorithm and the singular value decomposition SVD, to the rotations and the translations of the camera and also to the 3D reconstructed points via triangulation, have improved the global motion estimation. An experimental validation has been conducted and results are compared to a solution using classical bundle adjustment based on the Levenberg-Marquardt algorithm.

Although solutions to the motion estimation problem based on bundle adjustment with LM algorithm are eligible to provide accurate results, they impose limitations in the presence of a high level of noise that a system based on robust L_∞ norm is able to overcome. Through several experimental results, we show that the proposed technique, by including all sources of uncertainties, clearly outperforms these classical techniques, which use the Levenberg-Marquardt algorithm for motion and the least squares approach for absolute scale estimation.

Our second contribution, which follows on nicely from the first one, is to use the robust least squares algorithm capable of dealing with system uncertainties for frame to frame absolute scale estimation.

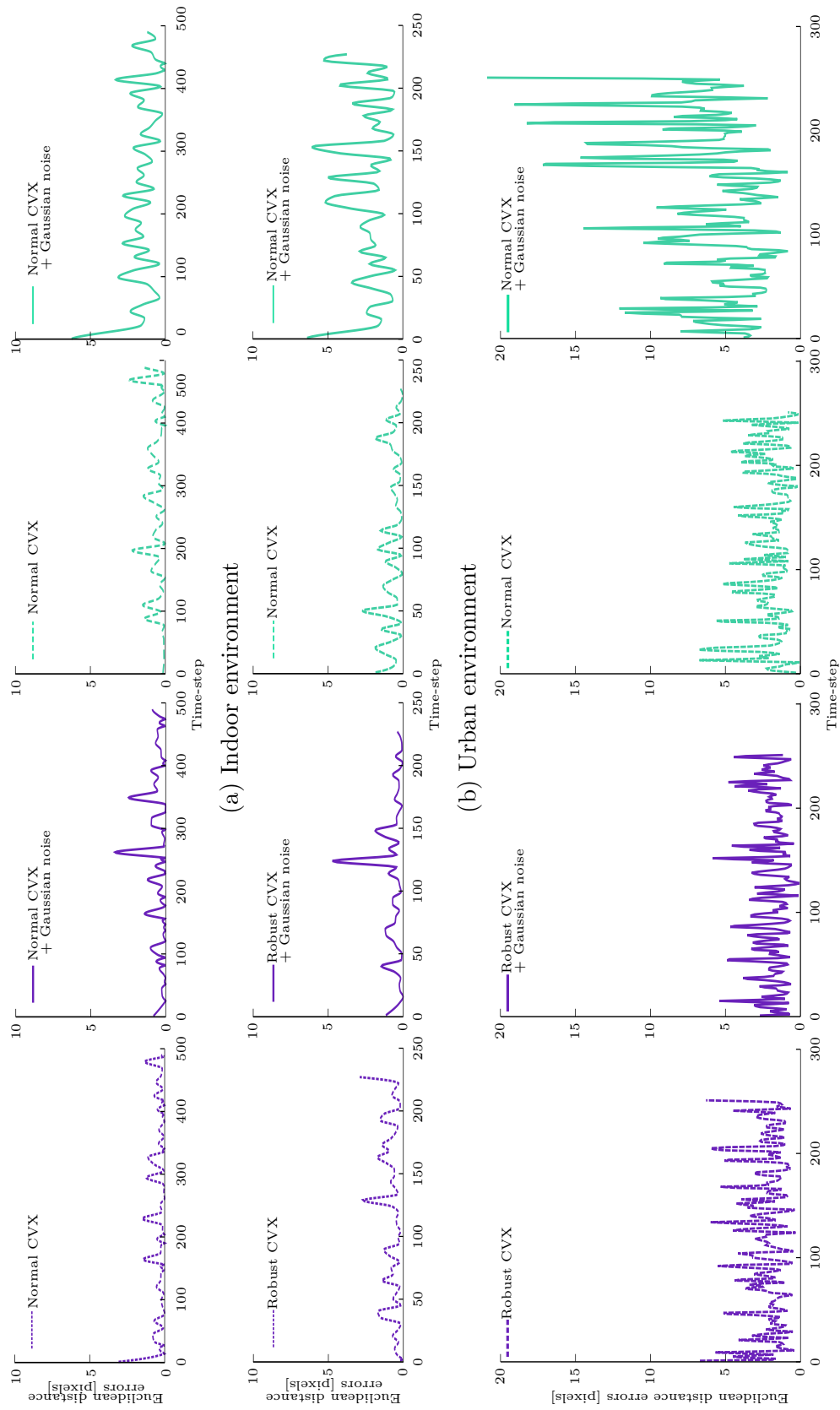


Fig. 6.12 Re-projection errors for the four different scenarios applied on the three environments.

Chapter 7

Real-Time Loop-Closure Detection

This chapter investigates the problem of recognising locations relying on their appearance. In computer vision, this problem is known as "Loop-Closure Detection". In this chapter, a fast and efficient appearance-based solution for visual loop-closure detection is proposed. This solution tries to extend the state-of-the-art by developing a probabilistic framework with the Gaussian mixture models (GMM) for appearance-based place recognition. The widely used techniques based on the bag-of-words image representation has shown some limitations, especially with the perceptual aliasing problem. In this work, however, two appearance-based techniques using local invariant and colour features are introduced. The first technique adopts the Bayes decision theory based on the Gaussian mixture model (GMM). The second technique is based on a combination of the GMM modelling with the KD-Tree data structure. Both solutions have been validated using monocular vision systems in several environments.

Loop-closure detection can be addressed as a problem of data association and matching, which intends to correctly associate the camera measurements obtained at a given time with the information already stored in a map. This data association allows the vehicle to correctly localise itself and consequently recognises in what part of the map it is.

7.1 Overview

The loop-closure detection is of critical interest in improving the robustness of autonomous navigation systems. After long navigation in unknown environments, detecting that the vehicle has returned to a previously visited position offers the opportunity to increase the estimation accuracy and consistency. Recognising previously mapped locations may also be relevant for a solution to the problem of global localisation. Thus, solving the loop-closure detection problem can significantly

improve the performance of the navigation algorithm, but at the same time, it also provides additional tasks to the navigation system.

In the case where the vehicle has no prior information about its position in a map, finding the best data association will solve the global localisation problem. Global localisation may itself be considered as a special case of the loop-closure detection, for which it is assumed that the robot is located in a known part of the map. In some situations, the vehicle could be in a location that has not yet known. In this case, the vehicle should be able as well to distinguish this location from the others, even in the presence of a strong resemblance to a place already visited.

The loop-closure solution is independent from the motion estimation. This means that the loop-closure solution has its own model of the environment where the vehicle is navigating, and the objective is to detect whenever the vehicle has returned to a previously mapped location. However, it is critical that the loop-closure algorithm is able to work in conjunction with any motion estimation algorithm to improve the global performance of the general navigation. This will be discussed in the next chapter.

The loop-closure problem is a data association problem whose solution brings the robustness to the motion estimation algorithm, allowing the detection of cycles in the trajectory without prior information on the position. One more important aim of the loop-closure detection consists of allowing the vehicle to restore its position after a failure or temporary obstruction of its sensors. Therefore, the outcome of the loop-closure detection is major for autonomous navigation systems, since the robustness of the motion estimation algorithms determines the quality of the navigation and, accordingly, the vehicle adaptability.

In this chapter, we propose robust methods for loop-closure detection based on vision systems. For this, we define a Bayesian filtering framework to estimate the probability that newly acquired images to come from an already visited place. This Bayesian filtering is working with Gaussian mixture modelling and KD-tree data structure.

In the literature, the well-known method for loop-closure detection is the bag-of-words (BoW) approach [180]. Even though this method has been showing good performance in loop-closure detection, it imposes some drawbacks. The main limitation is its dependency to an off-line learning while building the visual dictionaries. It is also considerably affected by the perceptual aliasing problem [216]. This usually happens when two distinct places are considered as the same place due to the quantisation process. This problem generates more missed loop-closures and more geometrical verifications [113].

Moreover, most methods presented in the literature are based on a single feature space to describe images, where the vast majority of them use the SIFT features [7, 13, 21, 44]. These features have impressive robustness qualities to the changes in the orientation and scale, but they suffer from limited robustness to illumination and affine changes. Fewer approaches in the literature use global signature features such as colour histograms [154, 201].

New methods have started to appear, in which they rely directly on matching features extracted from images [66, 113, 216]. In these methods, the loop-closure detection is performed by using the features themselves rather than their vector quantised representation. The main challenge for these techniques is how to cope with the search computational time. Fortunately, a data structure for storing the extracted features such as KD-tree would solve this problem. The search time in this data structure is logarithmic due to the time to descend from the root of the tree to the leaves.

7.2 Related Work

In the last decade, many appearance-based localisation and mapping solutions have been proposed. Although many solutions use the stereo vision systems [201, 219], many others make use of monocular configurations [8, 9, 43, 113]. The proposed technique belongs to this latter category. This approach is motivated by the bag-of-words methods, which were introduced to the computer vision by Sivic and Zisserman [180], and Nister and Stewenius [144]. However, we adopt the appearance-based technique using local image features themselves.

Before the introduction of the loop-closure concept to the computer vision community, previous techniques have been used for the localisation task, where the map is known and the vehicle is guaranteed to be in some location within this map. To use these techniques in the loop-closure detection problem, the algorithm must be able to distinguish views coming from unvisited locations. Obviously, this makes the problem significantly more complicated. The perceptual aliasing problem is the most challenging problem of these techniques, where different places can appear similar to the vehicle vision sensing system. Many techniques have been proposed to deal with this problem, in which some success is achieved. However, a complete and satisfactory approach has not been yet developed.

For image description, the BoW approach has gained more interests [140, 180]. This method has introduced new concepts, such as the visual vocabularies. This model processes an image as a collection of visual vocabularies called "Bag of Words", similarly to a text document (Figure 7.1) [180]. This model adopts the technique

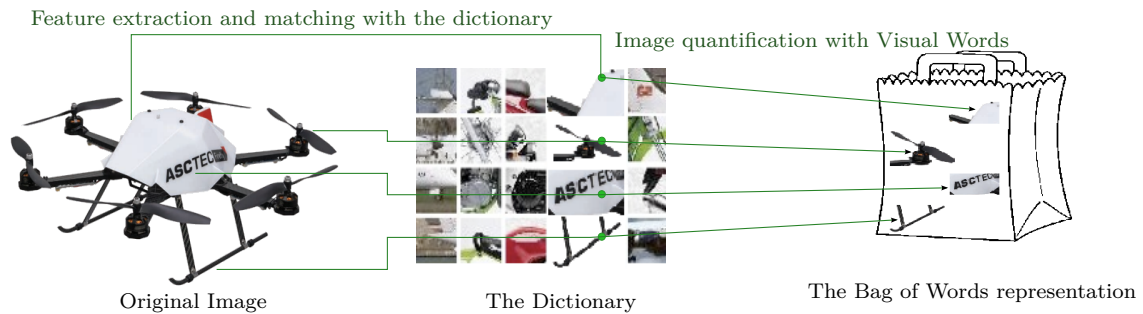


Fig. 7.1 Example of image representation with the "bag-of-words" concept. After visual words extraction from the original image (left), matching with words already existing in a dictionary is performed (middle). Image will be represented in function of the occurrence of these words in the dictionary (right).

used for text retrieval, which allows a fast search. However, this model ignores the image geometry, which limits its efficiency. Other proposed techniques in the literature try to cope with this problem by using the geometric information in a second step for verification [38].

Authors in [140] have proposed a solution that combines laser and vision systems. An incremental scheme with the BoW concept is used by Angeli et al. [8, 9] in estimating the loop-closure probabilities. This work was expanded with other local feature of colour histograms along with the SIFT features. Even though good performance is obtained using this technique, some drawbacks can be noticed, especially in dealing with perceptual aliasing problem. Schindler et al. proposed a more discriminative approach for building the visual words [169]. Cummins and Newman developed a new algorithm for fast appearance-based mapping (FAB-MAP), where a Chow-Liu tree is used for modelling the dependencies between the visual words [43].

Other techniques use the similarity matrices as an extension to the appearance-based methods [109, 176]. A similarity score is defined between selected images, then a square similarity matrix is computed, gathering the pair similarity between all images. Loop closures in this approach will appear in the off-diagonal entries of this matrix. As a development to this technique, Newman et al. [85, 86, 141] introduced an approach to deal with the perceptual aliasing problem by using the SVD of the similarity matrix, where the aim is to eliminate the effect of repetitive structures.

Other techniques use global descriptors, such as Gist descriptors [148]. Siagian and Itti presented in [175] a low-computational complexity and biologically-inspired scene classifier using Gist representation. In [112], Y. Liu and H. Zhang presented a method for visual loop-closure detection using Gabor-Gist descriptor by applying

the principal component analysis (PCA) technique to them. Similarly, Singh and Kosecka presented in [179] an approach for detecting loop-closures in a large sequence of omni-directional images from urban environments.

Clearly, some methods require a prior offline learning phase, in which a substantial amount of representative images of the environment need to be analysed and encoded to obtain a relevant model of the environment [44, 140]. The main drawback of these methods is the limitation of the robot navigation in areas not properly learned. Other approaches try to create a model for the environment where the robot is planned to navigate [31, 211]. However, these techniques prevent a direct adaptation of the navigation algorithms to different environments.

Another category of loop-closure detection method, apart from bag-of-words (BoW) or global descriptors, focuses on using local invariant features themselves. Authors in [100] proposed an approach which uses position-invariant robust features (PIRFs) to describe images. Zhang presented a method which uses a selective subset of visual features relying on the SIFT features. These feature in turn will be matched consecutively in several images [216]. The main drawback of this technique is the growing complexity with the number of images. To cope with this issue, Liu and Zhang presented a KD-tree-structure-based approach in appearance-based robot SLAM for a faster loop-closure detection technique [113]. In this work, they heavily rely on the efficiency of the tree structures for feature matching to achieve a real-time processing. However, in real applications where a vehicle is navigating for long distances, a linear increase of the number of features to deal with, while new images are acquired, can be a serious problem to the navigation algorithm. Therefore, relying on just the KD-tree data structure for features matching makes it really harder to ensure a real time processing, especially for faster moving vehicles.

The diversity of methods introduced in the literature illustrates how difficult the loop-closure detection problem and global localisation are. Having said that, it seems that detecting the loop-closure has been addressed in different ways and there is no global solution applicable in all cases.

In this chapter, new appearance-based techniques for loop-closure problem are presented. These solutions extend the previously presented methods by particularly addressing the perceptual aliasing problem with the Gaussian mixtures model (GMM) in combination with the KD-tree data structure. The proposed technique improves the computational time, where a significant reduction in the search time is achieved. Furthermore, to ensure the efficiency in different environments, two feature spaces are used to describe the images. These feature spaces are the SIFT features and the local colour histograms. In addition, an entirely GMM-based technique for loop-closure detection is presented first in this chapter.

7.3 Tools for loop-closure detection

The proposed solutions for the loop-closure detection problem are based on image classification approach. A newly acquired image is classified into the place it belongs. If this image comes from a known location, this is equivalent to a return of the vehicle to a previously visited place. In our approach, each location has to be modelled to distinctively describe any already visited location. This model is continuously and incrementally updated while the vehicle is navigating in the environment. More specifically, once a new image is acquired, the solution should correctly decide whether this image comes from a previously visited location, where its representation has already been modelled, or it characterises a new location.

As an overview of the proposed solution, newly acquired images are encoded first under the form of features. This is referred in our solutions as image description. The output of this image description stage is a number of feature vectors that describe each image. Then, this image is compared to the last processed image in order to avoid local loop-closures. This allows the vehicle to have sufficient motion before looking for loop-closures. Otherwise, the image will be ruled out. If the newly acquired image has not been ruled out, then its probability of closing the loop with previously seen images is estimated. This is performed within a Bayesian filtering framework. Prior to that, this image has to undergo the Gaussian mixture modelling (GMM), where the distribution of its feature vectors is estimated.

In fact, two distinct methods for loop-closure detection are proposed in the present chapter. The first one adopts the GMM modelling with the Bayes decision theory, while the second technique uses the GMM modelling with the KD-tree data structure. In the first technique, the outputs of the GMM modelling will serve as inputs to the Bayesian loop-closure detection module. In the second technique, however, the GMM modelling outputs are used to construct the KD-tree, which is used in turn for loop-closure detection.

The outputs of both solutions need to undergo through a further checking scheme to confirm or reject their decisions. This checking scheme consists of a multi-view geometry algorithm, which verifies the epipolar geometry constraint [82, 143]. This test approves whether effectively there is a common structure between the query image in question and the supposed place of origin, before confirming the loop-closure. It may occur in fact that the current image is very similar to a place in the model that it does not come from. By doing so, any ambiguity should be removed.

Finally, for both techniques, if a loop-closure is confirmed, the location to which the current image belongs is updated with the GMM parameters of the current image.

If the loop-closure is rejected, the GMM parameters will be used to create a new location in the model.

Therefore, introducing the basic concepts of the tools adopted in the proposed solutions is necessary. These tools consist of the Bayesian probability, the Gaussian mixture modelling (GMM) and the KD-Tree data structures.

7.3.1 Bayes decision theory for loop-closure

The adopted method in our work defines a probabilistic Bayesian filtering framework. The aim of this approach is to model the probability that a query image comes from one of the places already visited. This is considered as the loop-closure probability. In this chapter, we introduce techniques inspired of Bayes decision theory and we show how to employ them in the loop-closure detection task.

In a general classification problem, the inputs consist of a query pattern and the task is to correctly classify it into one of the existing c classes. The inputs are represented using specific description values, x_k , where $k = 1, \dots, l$. Then, these representations are collected in a feature vector $\mathbf{x} = [x_1, x_2, \dots, x_l]^T \in \mathbb{R}^l$.

Given $\mathbf{x} \in \mathbb{R}^l$ and the classes ω_i , $i = 1, \dots, c$, then the Bayes decision theory gives the probability that \mathbf{x} belongs to a specific class ω_i as:

$$P(\omega_i|\mathbf{x})P(\mathbf{x}) = P(\mathbf{x}|\omega_i)P(\omega_i) \quad (7.1)$$

and:

$$P(\mathbf{x}) = \sum_{i=1}^c P(\mathbf{x}|\omega_i)P(\omega_i) \quad (7.2)$$

where:

- $P(\mathbf{x}|\omega_i)$, $i = 1, \dots, c$ is the likelihood of ω_i with respect to \mathbf{x} ;
- $P(\omega_i|\mathbf{x})$ is the a posterior probability of the class ω_i given the value of \mathbf{x} ;
- $P(\mathbf{x})$ is the probability density function (pdf) of \mathbf{x} ;
- $P(\omega_i)$ is the a priori probability of class ω_i ; $i = 1, \dots, c$;

More precisely, given an input pattern with the feature vector $\mathbf{x} = [x_1, x_2, \dots, x_l]^T \in \mathbb{R}^l$, which is collected from some measurements, and given a number of possible classes c , that is, $\omega_1, \dots, \omega_c$, then the Bayes decision theory states that \mathbf{x} is assigned to the class ω_i if:

$$P(\omega_i|\mathbf{x}) > P(\omega_j|\mathbf{x}), \quad \forall j \neq i \quad (7.3)$$

Using (7.1) and given that $P(\mathbf{x})$ is positive, then (7.3) may be rewritten as:

$$P(\mathbf{x}|\omega_i)P(\omega_i) > P(\mathbf{x}|\omega_j)P(\omega_j), \quad \forall j \neq i \quad (7.4)$$

In some circumstances, the probability density function $P(\mathbf{x}|\omega_i)$ of the input query is not known. In this case, it has to be estimated before employing the Bayesian classifier. One of the most used methods to model this probability density function is the Gaussian mixture modelling (GMM) (Section 7.3.2). The unknown probability density function can be modelled as the sum of weighted Gaussian densities, as given in (7.5) in the following section. In the literature, the most used probability density function (pdf) for $P(\mathbf{x}|\omega_i)$, is the Gaussian pdf due to its mathematical tractability [192].

7.3.2 Gaussian mixture modelling (GMM)

Gaussian mixtures are used in the proposed solutions to model the keyframes (in our framework, keyframes refer to the selected images and considered as our classes). The mixture model in general is based on a probabilistic representation of the existing sub-populations within a larger population. This model assumes that all data are generated from a mixture of a finite number of Gaussian distributions with unknown parameters. Iterative expectation maximisation (EM) algorithm is used to estimate the GMM parameters from the training data.

Assuming that we have K component Gaussian densities, then the Gaussian mixture model is the weighted sum of these components [156, 191, 192] and given by:

$$P(\mathbf{x}|\lambda) = \sum_{i=1}^K w_i g(\mathbf{x}|\mu_i, \Sigma_i), \quad (7.5)$$

where \mathbf{x} is the data vector of dimension D (i.e. the measurements), w_i , $i = 1, \dots, K$, are the mixture weights, and $g(\mathbf{x}|\mu_i, \Sigma_i)$, $i = 1, \dots, K$, are the component Gaussian densities given by:

$$g(\mathbf{x}|\mu_i, \Sigma_i) = \frac{1}{(2\pi)^{\frac{D}{2}} |\Sigma_i|^{\frac{1}{2}}} e^{-\frac{1}{2}(\mathbf{x}-\mu_i)^\top \Sigma_i^{-1}(\mathbf{x}-\mu_i)}, \quad i = 1, \dots, K \quad (7.6)$$

where μ_i is the mean vector and Σ_i is the covariance matrix. Note that the sum of the mixture weights is one, i.e. $\sum_i^K w_i = 1$ [191, 192].

Therefore, the complete parameters of the Gaussian mixture model are the mean vector, the covariance matrices and the mixture weights from all components densities. These parameters are represented by:

$$\lambda_i = \{w_i, \mu_i, \Sigma_i\}, \quad i = 1, \dots, K \quad (7.7)$$

Due to their ability to characterise big class of sample distributions, the GMMs have attracted more interests, especially in biometric and speaker recognition systems [102, 156]. Generally, a GMM represents data distributions using a position, which is the mean vector, and an elliptical representation, which is the covariance matrix, and finally a nearest neighbour model.

These characteristics have motivated us in this work to employ the GMM representation in the loop-closure detection problem, where the system's inputs are the selected keyframes and the outcome is the classification of any new keyframe into previously modelled keyframes.

7.3.3 KD-tree Structure

KD-tree data structure is of great import in the proposed solution. In general, a KD-tree, or K-dimensional tree, is a data structure that can be used for storing and organising a number of points in a space with K dimensions. This data structure is also known as a multidimensional binary search tree, which is a generalisation of the simple binary trees [61, 136, 177].

KD-trees are very effective for the nearest neighbour search (NNS), in which every node is a k-dimensional point. Each node in these trees has two pointers; each one has a null value or points to another node in the tree. Therefore, a non-leaf node divides the space into two parts, known as half-spaces, forming two sub-trees. In other words, each non-terminal node contains two sons or successor nodes [61].

In one-dimensional search, a single key is used to define a node in the tree. All nodes in this tree with key values less than or equal to this key will be stored in the left-hand side, while nodes with larger key values will be in the right-hand side. In the k-dimensional search, each node is represented with k keys. In this tree, any one from these k keys can be used as the discriminator to position the sub-trees.

Appendix B (page 287) presents a detailed description of the KD-tree data structure, including the tree construction and the nearest neighbour search within these structures. A two-dimensional illustrative example is also given in this appendix. The KD-tree performance given this appendix has motivated us to adopt this data structure in the proposed solution. In our problem, we will be dealing with image features with high-dimensional descriptors. Therefore, these tree structures are more likely to suit us since they significantly reduce the search time to logarithmic. In addition, one more important property is their ability to provide an already sorted neighbours by traversing the tree just once. In the case of a query image, which is described with one of the two feature spaces, the KD-tree search provides the top nearest neighbours features among all images already modelled in the tree.

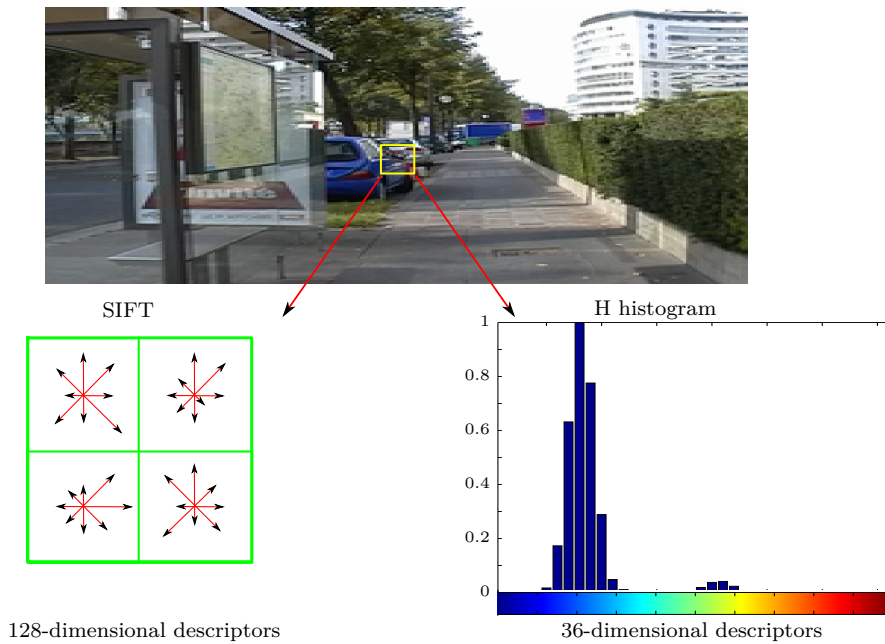


Fig. 7.2 A sample image from the outdoor environment showing the description of a window in the image using the SIFT and the colour descriptors.

7.3.4 Image features and their matching

The value of colour information quality is commonly agreed in upon computer vision community, due to its valuable ability to describe the world around us. However, most local feature spaces used in the literature are based on shape description only and attach less importance to the colour information. Describing images with local colour has received little consideration. Consequently, the vast majority of methods use only luminance and dismiss colour space. Therefore, considering augmenting the image description with colour space would be a valid option in our solution.

Colour features use histograms to represent the distribution of colours in a particular image. Basically, they count similar pixels and store them in appropriate bins in order to describe the number of pixels in each range of colours (or bin) independently. Local colour histograms capture information about distribution of colour in particular areas in the image.

Therefore, in our approach, each image is described using two feature spaces. The first one is the SIFT extractor and the second is the local colour histograms. Merging two complementary visual representations, such as the shape and the colour, would bring more efficiency and consistency to the solution in different environmental contexts.

In the colour description, images are decomposed into regularly spaced sub-areas (windows) of different sizes to improve scale invariance, then the normalised \mathbb{H}

histograms in the Hue Saturation Value (HSV) colour space for each sub-area are used as features [8, 207]. The two feature spaces are respectively detailed in Sections C.2.2 and C.2.3 (Appendix C, page 300). We incremented our solution with this colour feature space in order to ensure the robustness especially in less structured environments. For this colour descriptor, the histogram is divided into 36 bins. Figure 7.2, shows a sample image from an outdoor environment, where an image window is described using the SIFT and colour features. Employing a combination of the SIFT and the colour features would overcome the problem of perceptual aliasing that methods using BoW technique have.

7.3.5 Keyframe selection

The aim of the proposed solutions is to associate a query image to a location in the navigation environment. Due to the high frame rate in our datasets, any consecutive images will refer relatively to the same place since they were taken from very close viewpoints. Therefore, a distance threshold is applied to impose that the travelled distance between these views is considerable. This is referred in this framework as: *sufficient motion*.

Sufficient motion check means that the loop-closure detection will not be performed unless the vehicle has moved a certain distance, so newly captured images would be different to the last maintained images. In our experiments, in order to ensure that, a predefined number of frames will be always overcome before selecting any new keyframe.

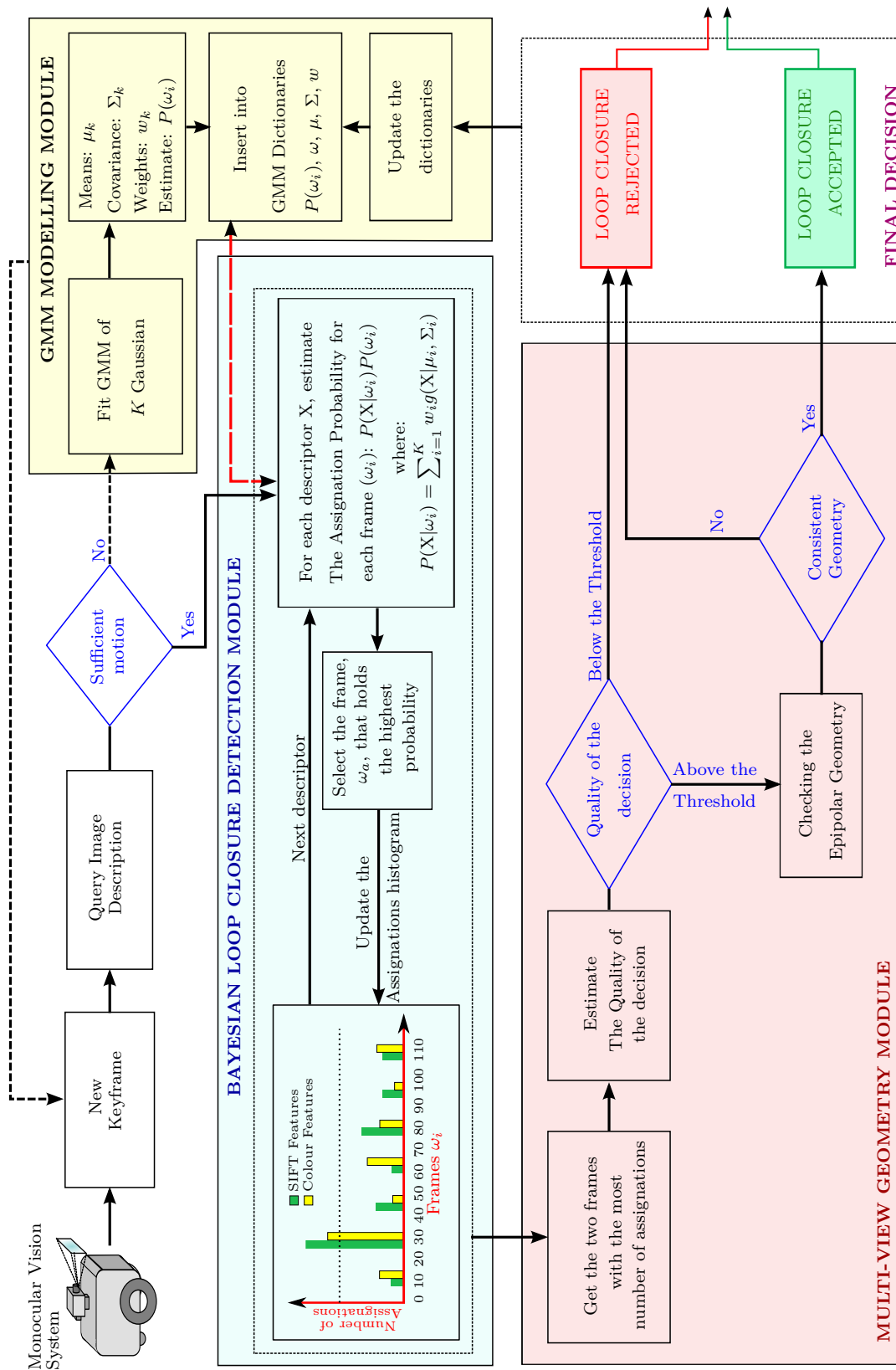


Fig. 7.3 Diagram depicting the main modules of the Gaussian mixture Bayes Solution for Loop-Closure Detection.

7.4 Gaussian mixture Bayes solution

In this section, the first solution for the loop-closure detection is presented. This solution uses the Gaussian mixture modelling (GMM) with Bayes decision theory. This solution aims to automatically classify a query image into previously seen images, whose GMM parameters are accumulated and stored into some GMM dictionaries, built up for each feature space. The construction of these dictionaries is done on-line in parallel with the image acquisition. This solution uses Bayes decision theory to classify the descriptors of a query image into previously seen images. Figure 7.3 shows a diagram depicting the main modules of this solution. Details about each module is given in the following sections.

Given a query image for loop-closure detection (Figure 7.3), after image description, a Gaussian mixture model will be fitted to their descriptors (Figure 7.3 - GMM modelling module). This is performed in the case where sufficient motion is not occurred (Section 7.3.5). Clearly, loop-closure detection is not performed over recently seen images as they always look similar to their neighbours. The GMM modelling outputs are the means μ_k , the covariances Σ_k and the mixture weights w_k , $k = 1, 2, \dots, K$, where K is the number of mixtures. These parameters are recovered using the expectation maximisation algorithm (Section 7.3.2). These parameters will be inserted into the GMM dictionaries μ , Σ and w , which will be used as inputs to the Bayesian loop-closure detection module (Figure 7.3 - Bayesian LC Detection module). The probability $P(\omega_i)$ of each selected keyframe is estimated in this module as well. As long as no sufficient motion is occurred, the GMM parameter $\lambda_i = \{w_i, \mu_i, \Sigma_i\}$ of the newly selected keyframes are estimated and inserted in the dictionaries and the probabilities $P(\omega_i), i = 1, \dots, c$, are updated accordingly, where c is the number of selected frames thus-far. This loop is shown in Figure 7.3 with the dashed black arrows.

In the case where sufficient motion is detected, for each descriptor in the query image, the probability of this descriptor to belong to each frame already seen is estimated. The frame with the maximum probability will be assigned to this descriptor. Then, the frame with the highest number of descriptors belonging to it defines a candidate for a loop-closure.

7.4.1 Bayesian loop-closure detection module

To formulate this module in mathematical way, suppose at time $t - 1$, a number of c keyframes have already been modelled and their parameters have already been inserted into the GMM dictionaries (Figure 7.3 yellow block at the top right). These

keyframes are denoted by ω_i , where $i = 1, 2, \dots, c$. Given a new keyframe I_t at time t , let us denote X_t^j as a descriptor extracted from this keyframe using the feature space j . Using (7.1) and (7.4), Bayes theorem gives the probability that this descriptor vector X_t^j belongs to one of the keyframes ω_i as (superscript j and subscript t in X_t^j are omitted hereafter for simplicity):

$$P(\omega_i|X) = \frac{P(X|\omega_i)P(\omega_i)}{P(X)} \quad (7.8)$$

where:

- $P(X|\omega_i)$ is the keyframe conditional probability density of X given ω_i ;
- $P(\omega_i)$ is the a priori probability of keyframe ω_i ;
- $P(\omega_i|X)$ is the a posterior probability of keyframe ω_i given X ;
- $P(X)$ is the probability density of X given as;

$$P(X) = \sum_{i=1}^c P(X|\omega_i)P(\omega_i) \quad (7.9)$$

According to Bayes theory given in (7.4), if $P(\omega_i|X) > P(\omega_j|X)$, for all keyframes such that $j \neq i$, then X will be assigned to the keyframe ω_i . Since $P(X)$ is the same for all keyframes, this becomes:

$$P(X|\omega_i)P(\omega_i) > P(X|\omega_j)P(\omega_j) \quad (7.10)$$

Let us call the probability $P(X|\omega_i)P(\omega_i)$, $i = 1, 2, \dots, c$, in (7.10) as the Assignment Probability. This probability is of great importance in the proposed solution. Each descriptor in the query image will be assigned to a frame ω_a , ($a = 1, \dots, c$), where a is the index of the frames that holds the highest assignment probability. Each descriptor will vote in a histogram for one frame. After processing all descriptors of the query image, the frame ω_n with the highest number of assignments in the histogram will be considered as a loop-closure candidate (Figure 7.3).

However, estimating the assignment probabilities $P(X|\omega_i)P(\omega_i)$, $i = 1, 2, \dots, c$ involves estimating first the unknown frame conditional probability densities $P(X|\omega_i)$, $i = 1, 2, \dots, c$. In our solution, these conditional probability densities are estimated using the Gaussian mixture modelling (GMM). Thus, the GMM modelling module is triggered (Figure 7.3). Details about this module is given in the following section.

Prior to that, let us detail this scenario from a different standpoint. Given at time step t a new query keyframe with M descriptors X_t^j . To check if this keyframe forms a loop-closure with one of the keyframes already seen, we estimate the assignment probability of each column of the matrix $T = \{X_{t_1}^j, X_{t_2}^j, \dots, X_{t_M}^j\}$,

(where T is of size $d \times M$, d is the dimension of the descriptor space j). That is, for each of the keyframes in the dictionaries, the GMM modelling module calculates $G_i^j(\mathbf{X}_t^j) = P(\mathbf{X}_t^j|\omega_i) = \sum_{i=1}^K w_i g(\mathbf{X}_t^j|\mu_i, \Sigma_i)$. Then, the assignation probability that \mathbf{X}_{t_m} comes from the keyframe ω_i is:

$$G_i^j(\mathbf{X}_{t_m})P(\omega_i), \quad m = 1, \dots, M \quad (7.11)$$

The probability $P(\omega_i)$ of the keyframe ω_i is estimated and updated continuously by GMM modelling module, which is the ratio between the total number of features in the keyframe ω_i and the total number of all features in all previously selected keyframes (Figure 7.3). Then, the keyframe with the maximum probability for each vector \mathbf{X}_{t_m} is assigned to this keyframe. This process is repeated for all M descriptors in the query image. Then, an M -element vector is constructed, containing the assignations to all keyframes. This vector is displayed as a histogram in Figure 7.3. The keyframe with the most assignations is declared as the winner at this stage and continues to the next stage which is the multi-view geometry module.

7.4.2 Gaussian mixture modelling (GMM) module

This section describes the functionality of the Gaussian mixture modelling (GMM) module. While the frames conditional probability density $P(\mathbf{X}|\omega_i)$ is supposed to be Gaussian, the feature distribution in a previously seen image is not identified. One way to solve this issue is to estimate this distribution using the Gaussian mixture modelling (Section 7.3.2). Therefore, to model the arbitrary probability density function, the sum of K weighted Gaussian densities is adopted:

$$P(\mathbf{X}|\omega_i) = G_i^j(\mathbf{X}) = \sum_{i=1}^K w_i g(\mathbf{X}|\mu_i, \Sigma_i), \quad (7.12)$$

where $g(\mathbf{X}|\mu_i, \Sigma_i)$ are the component Gaussian densities and given by:

$$g(\mathbf{X}|\mu_k, \Sigma_k) = \frac{1}{(2\pi)^{\frac{d}{2}} |\Sigma_k|^{\frac{1}{2}}} e^{-\frac{1}{2}(\mathbf{X}-\mu_k)^\top \Sigma_k^{-1}(\mathbf{X}-\mu_k)}, \quad (7.13)$$

where μ_k is the mean vector and Σ_k is the covariance matrix. The quantity $w_k, k = 1, \dots, K$, is the mixture weight where $\sum_k^K w_k = 1$, and d is the dimension of the descriptor space j .

The Gaussian mixture model is fitted to the descriptor matrix $T = \{\mathbf{X}_t^j\}$ using the expectation maximisation algorithm to estimate the parameters $\lambda_k = \{w_k, \mu_k, \Sigma_k\}$.

The covariance matrix Σ_k is assumed to have the form:

$$\Sigma_k = \sigma_k^2 I \quad (7.14)$$

where I is the identity matrix.

In the expectation step (\mathbb{E}), the probability that each descriptor $X_{t_m}^j$ belongs to each Gaussian distribution in the mixture is estimated as:

$$P(k|X_{t_m}^j, \lambda_k) = \frac{w_k g(X_{t_m}^j | \lambda_k)}{\sum_{k=1}^K w_k g(X_{t_m}^j | \lambda_k)} \quad (7.15)$$

Using the current parameters $\lambda_k = \{w_k, \mu_k, \Sigma_k\}$, this probability is computed for all descriptor $X_{t_m}^j$, $m = 1, \dots, M$, in the query image for each feature space j and also for all the mixture components $k = 1, \dots, K$.

On the other hand, in the maximisation step (\mathbb{M}), the parameters $\lambda_k = \{w_k, \mu_k, \Sigma_k\}$ are updated as follows:

The mixture weights:

$$w_k^+ = \frac{1}{M} \sum_{m=1}^M P(k|X_{t_m}^j, \lambda_k) \quad (7.16)$$

Note that M is the number of descriptors in the query image.

The means:

$$\mu_k^+ = \frac{\sum_{m=1}^M P(k|X_{t_m}^j, \lambda_k) X_{t_m}^j}{\sum_{m=1}^M P(k|X_{t_m}^j, \lambda_k)} \quad (7.17)$$

The variances (diagonal covariance):

$$\sigma_k^{2+} = \frac{\sum_{m=1}^M P(k|X_{t_m}^j, \lambda_k) \mathbf{X}_{t_m}^j}{\sum_{m=1}^M P(k|X_{t_m}^j, \lambda_k)} - \mu_k^{2+} \quad (7.18)$$

where σ_k^{2+} , $\mathbf{X}_{t_m}^j$ and μ_k^{2+} refer to the arbitrary elements of the vectors σ_k^{2+} , $\mathbf{X}_{t_m}^j$ and μ_k respectively.

The EM algorithm starts with initial values of $\lambda_k = \{w_k, \mu_k, \Sigma_k\}$ to estimate the updated model $\lambda_k^+ = \{w_k^+, \mu_k^+, \Sigma_k^+\}$, such that $P(X_{t_m}^j | \lambda_k^+) \geq P(X_{t_m}^j | \lambda_k)$. The updated model then serves for the next iteration. This process is repeated until convergence. This is reached by estimating the log-likelihood for each iteration and computing the difference between iterations. If this difference appears to stabilise, then the algorithm stops. This log-likelihood is estimated as:

$$l(\lambda_k) = \sum_{k=1}^K \log G_i^j(X_{t_m}^j) \quad (7.19)$$

In our solution, the initial values for the means μ_k are selected randomly, while those for the mixture weights w_k are chosen to be equal, so $w_k = \frac{1}{K}$. The initial values of the variances σ_k (diagonal entries of the covariance Σ_k) are chosen as the variances of our descriptor matrix $T = \{X_t^j\}$ rows. A regularisation value of 10^{-4} is added to these variances to ensure that the estimates are positive-definite.

Then, the outputs of the mixture models are the parameters: $w_k, \mu_k, \Sigma_k, k = 1, \dots, K$.

7.4.3 The multi-view geometry module

Before checking the multiple-view geometry constraint and in order to reject weak decisions, some thresholds are applied. This is similar to the technique used in [56] with some modifications (Figure 7.3). The quality of the decision is estimated using the following formula:

$$\text{Quality} = \frac{N_{\text{first winner}} - N_{\text{second winner}}}{M} \quad (7.20)$$

where N_x is the number of assignments to the keyframe x and M is the number of descriptors in the query image. The Bayesian loop-closure detection module's decision is only accepted if the quality and the number of assignments are above some thresholds.

As stated before, our solution uses two features spaces. Therefore, the given details about the estimation of the posterior to a specific feature space j apply identically to the other feature space. i.e., Equation (7.11) gives independently the probability that X_t^j in space j belongs to keyframe ω_i . A fusion scheme based on the intersection of the winners in each space is applied at this stage.

Then, a second step of further verification of the possible candidates is performed. In this step, a multiple-view geometry check is performed in order to remove outliers. This is performed by checking that the two images of the potential loop-closure ensure the epipolar geometry constraint. The geometric consistency test is completed using a RANSAC algorithm to fit an affine transformation between the query image and the candidate image. If the two images pass this test, this couple is declared as a loop-closure pair. Otherwise, the loop-closure will be rejected (Figure 7.3).

Finally, if a loop-closure is confirmed, the location to which the query image belongs is updated with the new GMM parameters. However, if the loop-closure is rejected, the GMM parameters will be used to create a new location in the model (Figure 7.3).

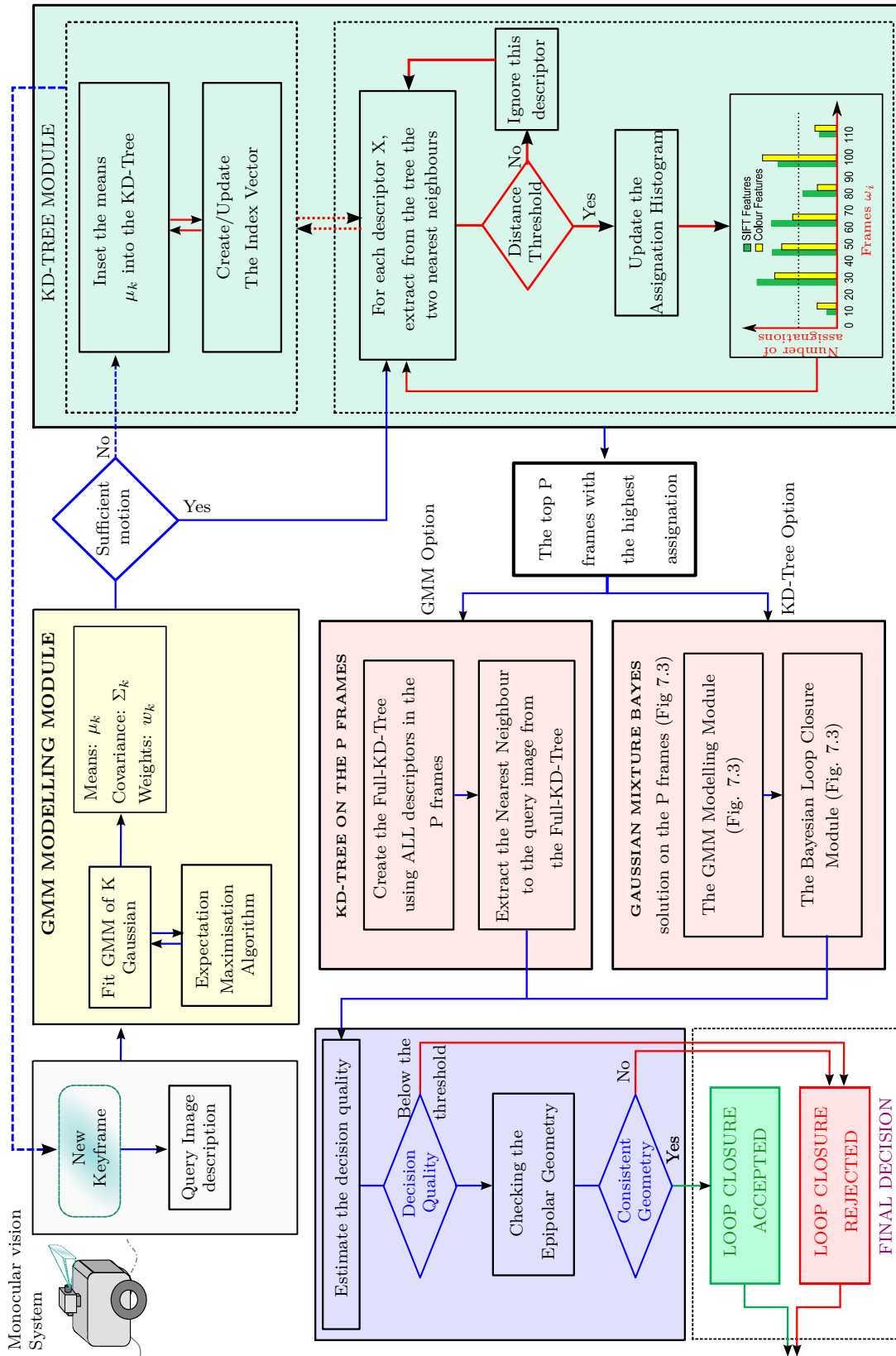


Fig. 7.4 Combination of GMM with KD-trees for Loop-Closure Detection.

7.5 The GMM KD-tree solution for loop-closure detection

In this section, the second solution for loop-closure detection is presented. This solution is based on a combination of the GMM with the KD-tree data structure. A diagram giving the main modules of this solution is depicted in Figure 7.4. This solution takes advantage of the robustness of the KD-tree data structure in the nearest neighbours search and the efficiency of the GMM representation.

In [113], Liu and Zhang presented a solution for loop-closure detection using a tree structure in appearance-based robot SLAM. In that work, authors used the KD-tree data structure for feature matching. The descriptors of a query keyframe are compared to all descriptors in all already selected keyframes via a continuously updated tree. Then, the keyframe that contains the most matched features with the query keyframe will be declared as a loop-closure after a second step of further verification. This solution offers a real time processing in the case where the number of the selected keyframes is relatively small. However, in real applications for vehicles navigating for long distances, where the occurrence of the loop-closure is not frequent, or for multi-vehicle cooperative navigation, a linear increase of the number of features to deal with while new keyframes are inserted makes the exploration of the tree considerably time consuming.

Our approach, however, does not compare the descriptors of a query keyframe to all the descriptors of all keyframes already selected. Instead, a Gaussian mixture is modelled for each new keyframe, and only the K means μ of these mixtures are inserted into the KD-tree, where K is the number of mixtures (Section 7.3.2). Therefore, the descriptors of a query keyframe are compared to just the means of the descriptors of all keyframes already selected (Figure 7.4). Hence, a significant reduction in the size of the tree is obtained. Consequently the search time for the nearest neighbours will be considerably reduced. This reduction in the tree size does not compromise the efficiency of the approach as shown in the experimental results of this chapter.

The proposed solution can be divided into three main stages. The first stage uses the KD-tree structure with the GMM representation to select only the most appropriate candidate frames for the loop-closure. The second stage looks for the loop-closure among the output frames of the first stage using a full representation this time. Finally, the third stage is a post-process for the geometric consistency check.

7.5.1 The first stage: Gaussian mixture model with KD-tree data structure

This stage is shown in Figure 7.4 through the yellow and the green blocs. Given a new keyframe for loop-closure detection, after image description (features extraction), a Gaussian mixture model will be fitted to their descriptors using the previously presented Expectation-Maximisation algorithm. The outputs of this module are the parameters $\lambda_k = \{w_k, \mu_k, \Sigma_k\}$, $k = 1, \dots, K$, for each selected keyframe, where K is the number of mixtures. These parameters will be used in subsequent stages.

First, we check whether sufficient motion is occurred. If it is not the case, this image will not be checked for loop-closure. Instead, its K GMM means μ_k will be inserted into a KD-tree. In this solution, this tree is referred as the GMM-KD-tree. In parallel, an index vector is created, which maps the means μ_k to their corresponding images, from which they are modelled. This is similar to the inverted index used in the bag-of-words (BoW) method, which associates keyframes and word frequencies for each visual word. As long as no sufficient motion is occurring, the GMM means μ_k of the newly selected keyframes are inserted in the GMM-KD-tree and the index vector is updated accordingly. This loop is shown in Figure 7.4 with the dashed blue arrows.

In the case where sufficient motion is detected, the query image will be first checked for potential loop-closure. Note that in this case, the GMM-KD-tree gathers the parameters of all previously selected keyframes. Let us assume that parameters of c keyframes are already modelled. In this case, the GMM-KD-tree will be have $K \times c$ nodes. Then, the next step is a nearest neighbours search (NNS) of the query image descriptors in the GMM-KD-tree. This search is done for each descriptor of the query image.

Let us assume that the query image is described with M descriptors X_t^j (as stated before, j stands for the descriptor space, either SIFT or colour features). To check if this keyframe forms a loop-closure pair with one of the keyframes already seen, we perform a nearest neighbours search (NNS) in the GMM-KD-tree of each column of the matrix $T = \{X_{t_1}^j, X_{t_2}^j, \dots, X_{t_M}^j\}$, (where T is of size $d \times M$, d is the dimension of the descriptor space j). This is performed in a logarithmic retrieval time as shown in Appendix B (Section B.3, page 287).

In this solution, we extract the first two nearest neighbours, denoted as Ψ_1 and Ψ_2 . The output Ψ_1 is accepted as the nearest neighbour to a descriptor query $X_{t_m}^j$ if:

$$\xi_1 \leq \kappa \cdot \xi_2,$$

where $\xi_u = \|X_{t_m}^j - \Psi_u\|$, $u = 1, 2$, is the distance between the descriptor query $X_{t_m}^j$ and the nearest neighbours Ψ_u , $u = 1, 2$. The quantity κ is a distance threshold. This means that when the ratio between the distance to the nearest neighbour and to the second nearest neighbour is less than the threshold, the output of this search is accepted. Otherwise, the descriptor query $X_{t_m}^j$ is ignored and supposed to have no neighbours.

In the case where Ψ_1 is accepted as a nearest neighbour of $X_{t_m}^j$, then a query is sent to the Index Vector in order to extract the frame ω_a from which the nearest neighbour Ψ_1 was modelled (dotted red arrows in Figure 7.4). The index a of the extracted frame ω_a will be assigned to the corresponding index in an assignation histogram (Figure 7.4). Therefore, any descriptor in the query image that passes the distance threshold will vote in this histogram for the frame ω_a from which it was modelled, accumulating to its score.

After processing all descriptors of the query image, the top P frames in the assignation histogram that have successfully passed the assignation threshold will continue to the next stage (Figure 7.4). This main purpose of the assignation threshold is to reject any frames with low number of assignations. If no frame passes this threshold, the "No Loop-Closure" decision will be taken at this early stage and no need to go further. Otherwise, this list of P candidates will serve as an input to the second stage. In order to ensure the efficiency and to avoid any possible ambiguity, this list is relatively large.

7.5.2 The second stage: Full representation stage

The blocs depicting this stage are shown in red in Figure 7.4. This stage is referred in our solution as the Full Representation Stage. This is because at this stage, the descriptors themselves of the P frames are used rather than using their GMM models. The P frames are considered in this stage as a "Short List" of loop-closure candidates. This, because we have narrowed the search space from c frames to just P frames, where c is the number of frames that have already been seen (representing the locations that have been visited by the vehicle). In our experiments, we have used an average value for P of 15. We can easily realise that P is by far smaller than c .

Therefore, the second stage consists of looking for a frame among the P frames that closes the loop with the query image. As we can see in Figure 7.4, there are two red blocs at this stage. These two blocs represent two options for solving the loop-closure detection at this stage. The two options have exactly the same inputs (the short list of P frames) and the same outputs (the loop-closure final candidate),

but the way they solve the problem is completely different. The designer can choose one option depending on the system parameters.

The first option adopts a typical KD-tree solution for loop-closure using the reduced database of P frames. The second option, on the other hand, adopts the Gaussian mixture Bayes solution, which is presented in Section 7.4, but on the reduced database of P frames.

- **Option one (Full representation technique with the KD-tree):** Firstly, a KD-tree of all the descriptors, this time, of the P frames is constructed. In this case, this tree is referred as the Full-KD-tree and composed of $\sum_{p=1}^P M_p$ nodes, where M_p is the number of descriptors in the frame ω_p , $p = 1, \dots, P$. Similarly to the first stage, an index vector, which maps each feature to the image where it was found, is also created.

Then, a search through this Full-KD-tree for the nearest neighbours is done for each descriptor in the query frame. Applying the same principles as in the first stage, the descriptors that pass the distance threshold will vote for a frame among the P frames. After processing all descriptors of the query image, the top two frames in the assignation histogram that have successfully passed the assignation threshold will continue to the third stage.

- **Option two (Full representation technique with the Gaussian mixture Bayes solution):** The second option adopts our first solution (The Gaussian mixture Bayes solution), presented in Section 7.4 (Figure 7.4), but applied on the P frames only. In Section 7.4 the Gaussian mixture Bayes solution is applied on all previously seen frames as a single solution. The number of these previously seen frames is referred in that solution as c . In this second solution (The GMM-KD-tree solution), we deploy the Gaussian mixture Bayes solution on just the P frames. One can realise that P is by far smaller than c , which increases the efficiency of the Gaussian mixture Bayes solution at this stage. Note that since the GMM modelling is already performed for all keyframes, there is no need to model again the P frames at this stage. Instead, these GMM parameters $\lambda_k = \{w_k, \mu_k, \Sigma_k\}$, $k = 1, \dots, K$, of all P frames are collected from the GMM module in the first stage (yellow bloc in Figure 7.4). Then, a Bayesian Loop-Closure Module will assign each descriptor in the query image to one of the P frames. Similarly to the first option, the top two frames in the assignation histogram that have successfully passed the assignation threshold will continue to the third stage.

7.5.3 The third stage: geometric consistency check

The output of both options in the second stage is two candidate frames for loop-closure detection. Before checking the multiple-view geometry constraint and in order to reject weak decisions, some thresholds are applied before accepting the winner. At this stage, we adopt the decision quality (7.20) (Blue bloc in Figure 7.4). The second stage's decision is accepted only if the quality and the number of assignments are above some thresholds.

The frame that passes the decision quality threshold will be confirmed or rejected by a further condition derived from the multiple-view geometry. The multiple-view geometry check is performed in order to remove outliers by verifying that the two images of the loop-closure ensure the epipolar geometry constraint. The geometric consistency test is completed using a RANSAC algorithm to fit an affine transformation between the query image and any candidate image. If the two images pass this test, this couple is declared as a loop-closure. Otherwise the loop-closure will be rejected (Figure 7.4).

Finally, and similarly to the first solution, if a loop-closure is confirmed, the location to which the current image belongs is updated with the GMM parameters of the current image. However, if the loop-closure is rejected, the GMM parameters will be used to create a new location in the model.

7.6 Computational complexity of the solution

The search complexity of the tree structure is decreased from linear to logarithmic (Appendix B, Section B.3, page 290). Suppose we are looking for a loop-closure in c previously visited locations in which each frame is described with an average of M features (M in general is between 1000 and 4000), then building a KD-tree has an $\mathcal{O}(Mc \log Mc)$ time complexity and an $\mathcal{O}(D \log nc)$ space complexity (D is the tree dimension). A search for m nearest neighbours costs closely to $\mathcal{O}(m \log Mc)$. However, by introducing Gaussian mixture modelling (GMM), instead of inserting M features for each frame, only K means μ_k , $k = 1, \dots, K$, mixtures are inserted into the KD-tree, where k is the number of mixtures (GMM dimension, in our solution 5 mixtures are used). Consequently, in comparison to methods that use the KD-tree only, the computational complexity would be significantly reduced by a factor of:

$$f = \frac{M \log Mc}{K \log Kc} \quad (7.21)$$

Note that K is significantly smaller than M . Then the computational cost of our solution becomes: $\mathcal{O}(Kc \log Kc)$ plus the cost of the GMM modelling. This latter modelling is done using the expectation maximisation algorithm, where for each frame with M features, the computational complexity is given by: $\mathcal{O}(MKD + MK)$ for the E-step and $\mathcal{O}(2MKd)$ for the M-step, where d is the feature dimension.

7.7 Experimental validation

Many experiments have been carried out in order to validate the proposed solutions to efficiently detect any loop-closure. Datasets from different environments, including outdoor and indoor conditions, are used. Each dataset is provided with the ground truth, indicating all the loop-closure occurrences. Similarly to [66], the evaluation has been conducted against the ground truth by counting the number of Correct Detection (CD), Correct Rejection (CR), Incorrect Detection (ID) and Incorrect Rejection (IR) for each sequence. A correct detection (CD) means that there is effectively a loop-closure and the solution has successfully detected it. Correct rejection (CR), on the other hand, means that there is no loop-closure and the solution has successfully stated that. Incorrect detection (ID) refers to the case where there is no loop-closure but the solution has detected a wrong one (false alarm). Finally, Incorrect rejection (IR), means that there effectively is a loop-closure, however the solution has wrongly rejected it or missed it.

The above criteria are used to compute the following three metrics:

- **Precision:** is the ratio between the number of correct detections (real loop-closures) and total number of the detected loop-closures:

$$\text{Precision} = \frac{\text{CD}}{\text{CD} + \text{ID}}$$

- **Recall:** is the ratio between the number of correct detections and the total number of loop-closures existing in the dataset:

$$\text{Recall} = \frac{\text{CD}}{\text{CD} + \text{IR}}$$

- **Accuracy:** is the rate of the correctly classified keyframes (Correct Detection (CD) and Correct Rejection (CR)):

$$\text{Accuracy} = \frac{\text{CD} + \text{CR}}{\text{CD} + \text{CR} + \text{ID} + \text{IR}}$$



Fig. 7.5 Example of keyframes forming a loop-closure in the indoor dataset. Top is image number 15, which closes a loop with image number 83 (bottom).

For keyframe description, the proposed solution adopts the SIFT detector, using 128-dimensional descriptors. In addition to the SIFT descriptors, images are also described with local colour histograms. In the latter feature space, images are decomposed into regularly-sized windows, and then the normalised \mathbb{H} histograms in the Hue Saturation Value (HSV) colour space for each sub-area are used as features. For this descriptor, the histogram is divided into 36 bins. Therefore, the dimension of colour descriptors is 36. Figure 7.2, shows a sample image from the outdoor data.

7.7.1 The Gaussian mixture Bayes solution

This first solution is applied on datasets from two different environments, outdoor and indoor environment. These datasets are obtained from Angeli et al. [8]. The indoor environment consists of 388 images of 240×192 pixels. The obtained results using the first solution for this dataset are summarised in Table 7.1. The 100% precision figure means that no incorrect loop-closure detection is found. The latter can significantly mislead the motion estimation. We can see that the accuracy of this solution is high, reaching 92%. Due to the severe epipolar geometry check, applied in our solution in the third stage, 28 correct loop-closures were missed out. This has lowered the recall rate down to just 87%. However, this solution has correctly rejected 155 incorrect loop-closures.

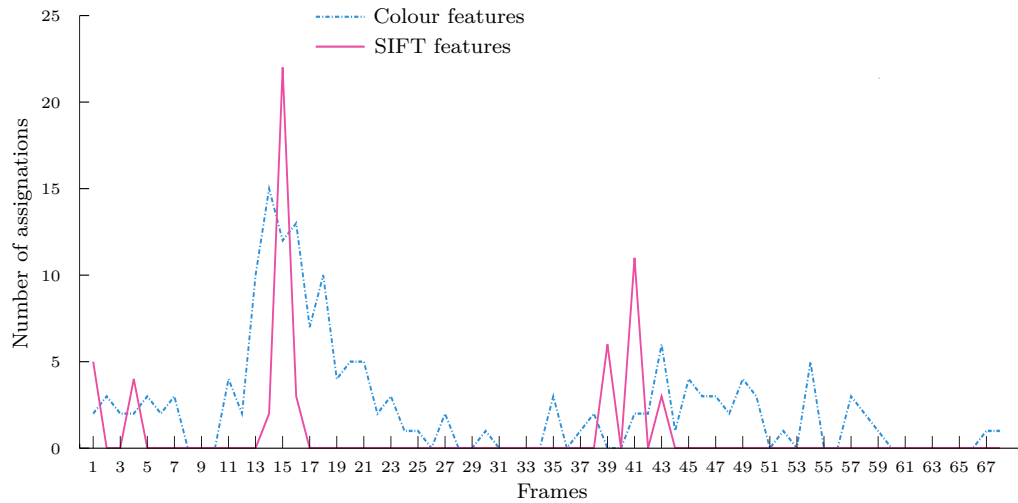


Fig. 7.6 Example of loop-closure detection using the Gaussian mixture modelling (GMM) with Bayesian decision theory in the indoor environment. The graphs plot the number of assignments for each frame of a query keyframe (image number 83) using two feature spaces, SIFT and Colour features.

Table 7.1 Performance in the indoor environment

	Correct	Incorrect
Detection	189	0
Rejection	155	28

Precision	Recall	Accuracy
100%	87%	92%

Figure 7.5 shows an example of keyframes forming a loop-closure pair in the indoor dataset. Figure 7.6 shows the robustness of the framework under changes in the environment. It is shown that this solution is able to recover loop-closures despite the changes in the scene and the camera rotation. The graphs in this figure show the assignments of all descriptors of an example image to all previously seen images for the two feature spaces used in this solution. Two peaks are recorded for the SIFT descriptor space, around images with indices 15 and 41. It can be clearly seen that colour descriptors, support and confirm the result obtained by the SIFT space around the image with index 15. However, it remarkably declines it around image with index 41, where few assignments are noticed. Therefore, only images around the index 15 can continue to the next stage.

The high precision rate, 100%, and accuracy rate, 92%, show the capability of this technique in indoor environments, where less structured locations are most likely.

A second experiment is conducted on the outdoor environment [8]. This dataset consists of hand-held camera travelling around an urban environment. A total of 1063 images, with a resolution of 240×192 pixels, were captured in this sequence. Obviously, this environment is more challenging than the indoor one, since it contains more images with big changes in appearance of the background throughout the city due to the pedestrians and traffic. Results of this experiment are summarised in Table 7.2.

Table 7.2 Performance in the outdoor environment

	Correct	Incorrect
Detection	538	0
Rejection	438	65

Precision	Recall	Accuracy
100%	89%	94%

Again our solution shows a 100% precision rate, indicating that no incorrect detections were made. Along with the 100% precision rate, higher recall and accuracy rates are obtained as well, reaching 89% and 94% respectively. Incorrect rejections are due to the same reasons mentioned in the previous experiment. However, regarding the size of the sequence, a relatively small number of incorrect rejections (IR) is recorded. This is due to the effect of perceptual aliasing, which is less present in this environment.

Figure 7.7 shows an example of keyframes forming a loop-closure in the outdoor environment. Figure 7.8 shows the corresponding results. The graphs in this figure show the assignments of all descriptors in the image with index 600 to all previously seen images. For the SIFT feature space, several peaks were recorded. Again, this is due to background similarities in this environment. Thus, high probabilities have distributed around many frames. Colour descriptors have managed to reduce the ambiguities, and kept only frames with indexes around 140. The final decision will be accepted or rejected according to the multiview geometry constraint.

7.7.2 The GMM KD-tree solution

In order to validate the second proposed solution, experiments were conducted on an indoor dataset collected from a hallway environment [216]. This dataset consists of 7.420 images (Figure 7.9). In this dataset, a mobile robot performs two loops in a hallway; therefore, loop-closures occur during the whole second loop. Due to the



Fig. 7.7 Example of two keyframes forming a loop-closure in the outdoor dataset. Top: image with index 140, which closes a loop with the image with index 600 (bottom)

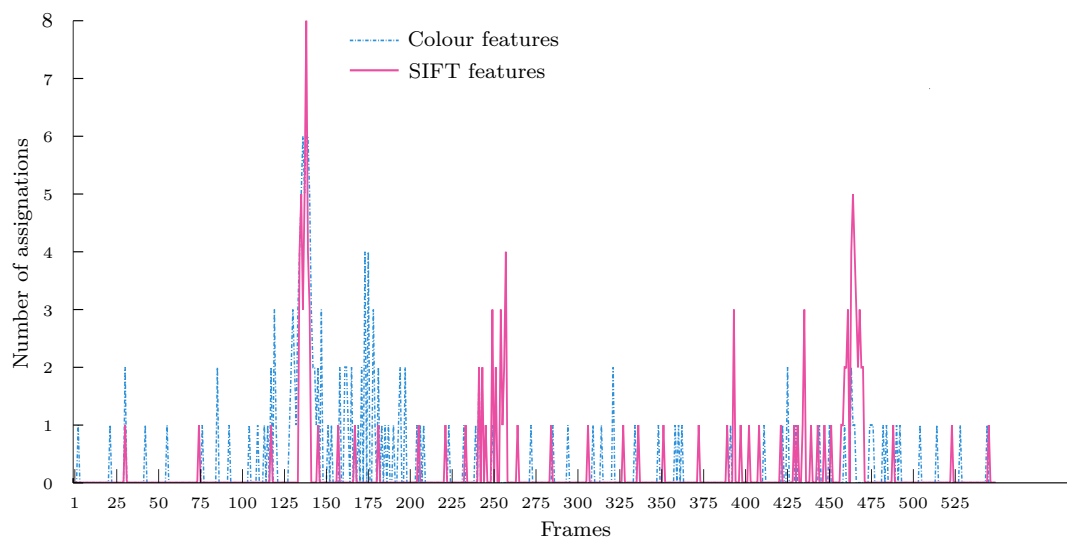


Fig. 7.8 Example of loop-closure detection using the Gaussian mixture modelling (GMM) with Bayesian decision theory in the outdoor environment. The graphs plot the number of assignments for each frame of a query keyframe (image number 600) using two feature spaces, SIFT and colour features.

high frame rate in this dataset, a keyframe selection mechanism has been employed, where steps of 100 images are taken after each keyframe selection.



Fig. 7.9 Example of loop-closure in hallway dataset

Table 7.3 summarises the performance of the proposed solution in this environment. Over all possible loop-closures, only two loops were rejected reaching a higher recall rate of 95%. The 100% precision rate is due the low frequency of false alarms, which cross the second stage of the solution. If any, the geometric consistency check in the third stage will reject any candidate that not fulfilling the multiple-view geometry constraint. This performance is obtained regardless the option we used in the second stage of the solution, where both options give similar results.

Table 7.3 Performance in the hallway environment

	Correct	Incorrect
Detection	35	0
Rejection	38	2

Precision	Recall	Accuracy
100%	95%	97%

These results outperform those obtained from state-of-the-art methods [8, 9, 113]. In addition, the key feature of the proposed solution is the computational time. As discussed, in Section 7.5.2 (page 179), the second stage of the solution can be conducted via two options (Figure 7.4). Even though the computational time for the whole algorithm with the first option offers a real-time processing, GMM option is relatively time consuming in comparison to the full KD-tree option.

Experiments show that the GMM classifies the query image in an average time of 14.05 ms against just 3.14 ms using the full KD-tree. This is due to the complexity of estimating the probability of each descriptor to belong to every single frame among

the P frames in the GMM option. Therefore, in terms of computational time, the solution with the full KD-tree option clearly outperforms the second option.

An analysis relative to the computational time of this solution using the full KD-tree option is shown in Table 7.4. This includes the GMM modelling run time for each new keyframe, the run time for building the GMM parameters tree and the time required for search and indexing. Note that the time for building the GMM KD-tree in stage one is taken into consideration only when a new frame is inserted.

Table 7.4 Computational time - Tree Construction, Search and Indexing (ms)

The Proposed solution				Solutions using all frames directly			
	Min	Max	Average		Min	Max	Average
GMM and Building the Trees				Building the Tree			
Fitting GMM	1.30	21.4	7.30	Building the whole Tree	2.1	31.2	16.2
Building the GMM Tree	0.02	0.12	0.08				
Building the full Tree	0.06	1.02	0.13				
Total	1.38	17.34	7.51		2.1	31.2	16.2
Search and Indexing				Search and Indexing			
In the GMM Tree	0.09	3.10	1.80	In the whole tree	0.55	128.4	10.3
In the full Tree	1.06	7.42	3.01				
Total	1.15	10.52	4.81		0.55	128.4	10.3

To evaluate the performance of the proposed solution in terms of computational complexity, we compare our figures to similar methods applied on the same dataset (The hallway data) by using all frames directly, like the one presented in [113]. In these techniques, the average time of building the tree is 16.2 ms. Since the time for this operation is counted only when a new keyframe is found. This operation is equivalent to the following operations in our solution:

- Fitting the GMM;
- Building the GMM KD-tree;
- Building the full KD-tree of the P frames.

The latter three operations are performed together in just 7.51 ms, which is less than half time required for techniques that use all frames. Note that our solution uses 5 mixtures and an average value of P of 15. Search and indexing take less than 5 ms in the proposed solution, while it exceeds 10 ms in techniques that search in all frames.

These figures show that our solution gets good results in relatively less time in comparison to techniques that use full representation.

7.7.3 Comparison between the first and the second solutions

In this section a comparison is given between the first solution using Gaussian mixture modelling (GMM) with Bayesian loop-closure detection and the second solution using

the GMM representation in combination with the KD-tree data structure. Both solutions were tested on the same outdoor data. We have selected this dataset due to its complexity, since it includes more images with more changes in the appearance around the urban environment such as the traffic and pedestrians.

In addition to its ability to robustly detect loop-closures, the distinctive feature of the second solution is its reduced computational complexity. In this solution, the first option using the Full-KD-tree is adopted in its third stage. This choice is justified by the fact that KD-tree data structure offers a significant reduction in the computational cost without compromising the search robustness.

Results from this experiment are summarised in Table 7.2. Testing the second solution on this dataset has revealed some interesting results. These results are shown in Table 7.5.

Table 7.5 Comparison of performance between the GMM-Bayesian and the GMM-KD-tree solution.

	Correct		Incorrect	
	Sol. 1	Sol. 2	Sol. 1	Sol. 2
Detection	538	520	0	0
Rejection	438	318	65	83

Precision		Recall		Accuracy	
Sol. 1	Sol. 2	Sol. 1	Sol. 2	Sol. 1	Sol. 2
100%	100%	89%	86%	94%	91%

- Sol. 1: The first solution using Gaussian mixture modelling (GMM) with Bayesian loop-closure detection.
- Sol. 2: The second solution using the GMM representation in combination with the KD-tree data structure.

The outdoor dataset gathers 1063 images with a resolution of 240×192 pixels, where a hand-held camera is navigating in urban environment making a two-loop trajectory. One can realise that this dataset generates too many potential loop-closure pairs. The first statement that can be drawn in this experiment is that both solutions have no incorrect loop-closure detections, leading to a 100% precision. We know that these false alarms (incorrect detections) have a significant impact on the subsequent motion estimation, as they mislead any localisation or motion estimation algorithm.

The correct detection figure in the GMM-KD-tree solution has dropped from 538 to 520 detections in comparison to the solution using GMM/Bayesian solution. This is due to the fact that the former solution uses the technique of the Short List in the

second stage. The size of this short list can be in some circumstances not sufficient to include all the potential loop-closure candidates. This leads to missing more correct loop-closures and decreases the recall and the accuracy rates in comparison to the first solution.

However, computational time is of great importance in this kind of problems. Real time applications require a rational execution cost. Note that any loop-closure solution should be able to work in conjunction with any motion estimation or localisation algorithms.

Analysis of the computation time for both solutions gives advantage to the GMM-KD-tree method. Both solutions perform the GMM modelling, which is executed in an average time of 7.30 milliseconds. In addition to the time required for GMM modelling, the GMM-KD-tree solution needs 0.08 milliseconds for each frame to build the first KD-tree and 1.80 milliseconds for the nearest neighbour search. However, the GMM-Bayesian solution is required to estimate the unknown keyframe conditional probability density $P(X|\omega_i)$ for each descriptor X , which requires an average time of 8.43 milliseconds. This operation significantly affects this solution in terms of computational time. Finally, the geometrical consistency check is performed by both solutions, thus this operation does not balance any solution.

The average total computational time for GMM-KD-tree method is about 0.3 seconds, while it exceeds 3.5 seconds in the GMM-Bayesian solution. These figures justify our adoption of the former method in the next chapter for the global motion estimation solution.

7.8 Conclusions

Two appearance-based loop-closure detection techniques have been presented in this chapter. Despite other techniques, which use BoW approach for image representation; in this work, local invariant features have been used along with colour features. These two feature spaces work in a cooperative scheme for loop-closure detection. The first solution uses the Gaussian mixture modelling (GMM) with Bayesian loop-closure detection. This solution aims to automatically classify a query keyframe into previously seen keyframes, whose GMM parameters are accumulated and stored into some GMM dictionaries, constructed for each feature space.

The second solution takes advantage of the robustness of the KD-trees in features matching and the efficiency of the GMM representation. The descriptors of a query keyframe are compared to all modelled descriptors in all already selected keyframes via a continuously updated KD-tree. Then, the keyframe that contains the most

matched features will be declared as a loop-closure with the query keyframe after a second step of further verification.

Experimental validation using datasets from different environments has been conducted. It is shown in our work that due to their efficiency and complementarity, a combination of KD-trees with the GMM would be an alternative solution for real-time loop-closure detection for mobile robots navigation with high recall rate.

Chapter 8

Robust L_∞ Convex Pose-graph Optimisation for Monocular Localisation Solution for UAVs

After developing a loop-closure detection solution, a correction solution for any potential position drift would be required. Indeed, in the present chapter, a robust L_∞ convex pose-graph optimisation solution for monocular motion estimation with loop closing is presented. Once a loop-closure is detected using the technique presented in Chapter 7, the convex pose-graph optimisation solution performs the correction of any drift occurred during the motion estimation. Most solutions proposed in the literature formulate the pose-graph optimisation as a least-squares problem, by minimising a cost function using iterative methods such as Gauss-Newton or Levenberg-Marquardt algorithms. However, with these algorithms and as we have seen previously, there is no guarantee to converge to a global minimum as they, with high probabilities, converge to a local minimum or even to an infeasible solution. The solution we propose in this chapter uses a new robust convex optimisation pose-graph technique, which efficiently corrects the UAV's pose after loop-closures detections. Uncertainty estimation using derivative method and its propagation through the multi-view geometry algorithms are included in the developed solution. The detection of visual loop-closures, in appearance-based navigation, is achieved using our innovative technique presented in the previous chapter.

8.1 Introduction

Unmanned and micro aerial vehicles will shortly be the first choice asset to be deployed in important missions, such as inspection, reconnaissance, surveillance

and search and rescue. The GPS and other satellite navigation systems may offer valuable assist for these aerial vehicle. However, urban- and indoor-environment operations present real handicaps to these navigation systems, where its availability is extremely limited or even does not exist at all. Inertial navigation systems (INS) or GPS-aided INS systems might be used in these situations. However, the eventual growth in the INS errors is prohibitive to these applications. Therefore, investigating other alternative, such as vision system has become a priority for many research programmes. Indeed, in this chapter, we address the problem of the UAVs localisation in unknown environments, using monocular visual systems as the only source of information. However, the inherent difficulties in UAVs localisation in unknown environments relying on visual sensing impose great research challenges, especially when big coverage areas from higher altitudes capability is required.

In the literature, many studies have been focussing on using vision as a perception sensor for UAVs navigation or even as a guidance tool for safely landing of aerial vehicles [41, 150, 165]. Visual navigation approach has been widely studied in the last years as an alternative navigation solution for autonomous aerial systems. It estimates the pose of moving UAVs using visual inputs only with single camera, stereo cameras or multi camera systems [82, 109, 204]. An INS-aided with stereo vision system may be used in a cooperative visual simultaneous localisation and mapping (VSLAM) design for multiple UAVs, in which the INS localisation errors are corrected in a combination with vision algorithms [137]. In addition, 3D texture mapping models for UAV applications can also be used [110].

In practice, and similarly to other navigation systems, errors in the vehicle position estimates for aerial visual navigation are continuously growing due to the integration of noisy measurements over time and imperfect computational techniques. Using relative information in the positioning process rather than the absolute measurements (like in the GPS) is another serious reason for this error propagation. This unavoidable drift in motion estimation, due also to inherent inaccuracy of the devices, needs to be corrected. Thus, providing additional correction tools would have a significant impact on the final estimates.

During the last few years, visual loop-closures have gained more attention and considered of as a powerful and practical tool for motion drift correction. Indeed, after long navigation into an unknown environment, detecting that the autonomous aerial vehicle has returned to a previously visited location offers the opportunity to correct the positioning residual drift and consequently increase the accuracy and the consistency of the vehicle motion estimates. To achieve this, many research studies have started investigating efficient techniques for visual loop-closure detection.

We have stated in the previous chapter that solving the problem of detecting loop-closure would considerably improve the localisation algorithm performance. On the other hand, it also induces additional computational charges. The previously presented technique for loop-closure detected is employed in this chapter. This technique relies on a combination of Gaussian mixture model (GMM) with the KD-tree data structure.

8.1.1 Pose-graph representation

Once a loop-closure is detected, indicating that the UAV has returned to a previously visited location; the challenge is how to ensure the creation of a consistent map despite of the drift. In the recent years, pose-graph optimisation has become a preferred technique for loop-closure correction, where relative constraints between poses are used, especially those imposed when a loop-closure is detected. In a pose-graph representation for the motion, each node in the graph represents a vehicle pose. Constraints between these poses are represented by the edges between the nodes (Figure 8.1). These constraints are defined from observations and depict a rigid body transformation between the poses. Obviously, these constraints are affected by the noise and drift (Figure 8.1). The main objective of the optimisation algorithm is then to recover the optimal configuration of the nodes that best satisfies the constraints (maximises the observation likelihood determined in the constraints) [149]. Therefore, this involves solving a large error minimisation problem. In other words, the principal aim is to jointly optimise the vehicle poses in order to minimise the errors dictated by the constraints (Figure 8.1).

The pose-graph representation was first proposed by Lu and Milios in 1997 [55]. However, this approach took many years to become a solution for error minimisation problems [75]. This work was followed by Gutmann and Konolige, who proposed a graph construction technique by including loop-closures constraints [76]. In the literature, optimisation techniques that recover the optimal poses given the constraints are usually called back-ends. In contrast, front-ends techniques recover the input data to obtain the constraints that are the basis for the optimisation.

8.1.2 Classical pose-graph optimisation

We have seen throughout this thesis that optimisation plays an important role in motion estimation. Over the last decade, many optimisation algorithms have been recommended for the pose-graph optimisation. These algorithms can be classified into two main categories:

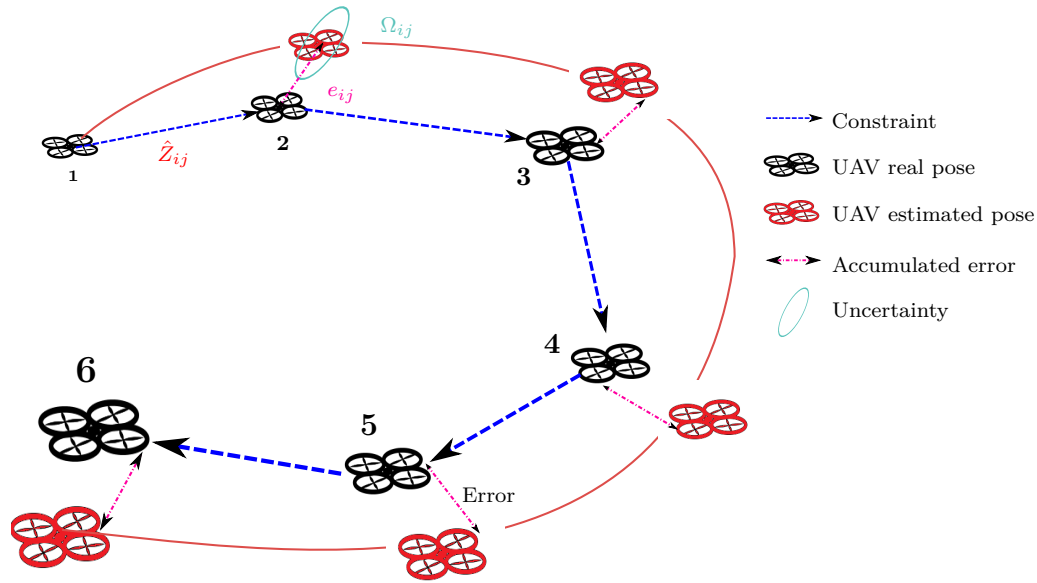


Fig. 8.1 A typical pose-graph representation. The UAV navigates around an unknown environment. Poses are connected through constraints (blue dashed arrows), which represent a rigid body transformation between poses.

- The first category employs linear approaches based on least squares minimisation. In this approach, singular value decomposition (SVD) is usually adopted, where an algebraic cost function is minimised. These methods have the advantage of offering a closed-form solution with a simple implementation. However, the quantity being minimised is not geometrically or statistically meaningful.
- The second category relies on iterative estimation techniques, where a non-linear cost function is minimised using iterative algorithms such as Levenberg-Marquardt, Gauss-Newton, gradient descent or conjugate gradient. The cost function here is geometrically interpretable and can statistically be optimal under an assumption of Gaussian noise. Commonly, this category is adopted as a solution for the pose-graph optimisation problems.

In order to have a deep sight on the second pose-graph optimisation category, let us consider the following example (Figure 8.1). Let $X = (X_1, \dots, X_t)$ be a vector representing the poses of a moving UAV. Let z_{ij} and Ω_{ij} represent respectively the measurement and the covariance between the nodes i and j . Let $\hat{z}_{ij}(X_i, X_j)$ encodes the measurement prediction given the poses X_i and X_j , which is the relative transformation between the two poses. Let e_{ij} be the error between the predicted observation \hat{z}_{ij} and the real observation z_{ij} [75].

The main aim of the optimisation problem is then to find the configuration of the nodes X^* that minimises the negative log likelihood $F(X)$ of all observations,

where the objective function $F(X)$ is defined as:

$$F(X) = \sum e_{ij}^\top \Omega_{ij} e_{ij} \quad (8.1)$$

This leads to solve the following optimisation problem:

$$X^* = \operatorname{argmin} F(X) \quad (8.2)$$

This optimisation problem is formulated as a non-linear least squares problem, where the error is the squared L_2 norm. The error function is approximated by its first order Taylor expansion around the current initial values \check{X} , which are selected usually by guess. This leads to solve a linear system of the following form:

$$A\Delta x = -b \quad (8.3)$$

where Δx is the increment of the system. Then, the solution X^* is obtained by adding the recovered increments to the initial values:

$$X^* = \check{X} + \Delta x \quad (8.4)$$

The algorithm iterates the linearisation step, the solution of (8.3) and the update of the equation (8.4). In each iteration, the previous solution is used as an initial guess. This procedure is repeated until a satisfactory convergence standard is achieved. Usually, until a predefined termination criterion is met. However, this non-linear problem has multiple minima, which pose a serious problem for iterative methods. Therefore, notwithstanding of their dependency on good initialisation guess, these algorithms present high probabilities of convergence to a local minimum or even an infeasible solution.

As a valid alternative and to get around these drawbacks, we present in this chapter a third approach to solve the pose-graph optimisation problem for visual navigation. This alternative employs convex optimisation using a more robust norm such as the L_∞ norm.

8.2 Convex optimisation formulation

In contrast to linear and iterative methods, convex optimisation ensures getting a single global minimum, and the cost function is geometrically meaningful. As we

have seen in Chapter 3, a convex optimisation problem has the form:

$$\begin{aligned} \min_x \quad & f_0(x) \\ \text{subject to} \quad & f_i(x) \leq 0 \text{ for } i = 1, \dots, m \\ & a_i^\top x = b_i \text{ for } i = 1, \dots, p \end{aligned} \quad (8.5)$$

It is the problem of finding an x that minimises $f_0(x)$ among all x that satisfy the constraints $f_i(x) \leq 0$ and $a_i^\top x = b_i$. The vector $x \in \mathbb{R}^n$ is the optimisation variable.

It is worth mentioning here that the particular convex optimisation problems where the objective function f_0 is linear and the constraints are of the form: $\|A_i x + b_i\| \leq c_i^\top x + d_i$ are called Second-Order Cone Programming (SOCP). Therefore, a SOCP is an optimisation problem of the form:

$$\begin{aligned} \min_x \quad & f^\top x \\ \text{subject to} \quad & \|A_i x + b_i\| \leq c_i^\top x + d_i, \text{ for } i = 1, \dots, m \\ & g_i^\top x = h_i \text{ for } i = 1, \dots, p \end{aligned} \quad (8.6)$$

where vectors $x, f, c_i, g_i \in \mathbb{R}^n$, scalars $d_i, h_i \in \mathbb{R}$, matrix $A_i \in \mathbb{R}^{(n_i-1) \times n}$ and $b_i \in \mathbb{R}^{n_i-1}$.

Our task in this chapter, is to formulate the pose-graph optimisation as a convex optimisation problem. Before detailing that, the L_∞ optimisation will be adopted in our framework. Therefore, the problem (8.6) is formulated as a min-max form:

$$\begin{aligned} \min \quad & \max \frac{\|A_i x + b_i\|}{c_i^\top x + d_i}, \quad \text{for } i = 1, \dots, m \\ \text{subject to} \quad & c_i^\top x + d_i > 0, \quad \text{for } i = 1, \dots, m \end{aligned} \quad (8.7)$$

As we have seen in Section 3.7.1 (Chapter 3, page 69), this problem may be transformed into an equivalent problem by incorporating a new variable δ :

$$\begin{aligned} \text{Find} \quad & \delta \\ \text{subject to} \quad & \|A_i x + b_i\| \leq \delta (c_i^\top x + d_i), \quad \text{for } i = 1, \dots, m \\ & c_i^\top x + d_i \geq 0, \quad \text{for } i = 1, \dots, m \end{aligned} \quad (8.8)$$

For a given value of $\delta \in \mathbb{R}$, this optimisation problem will become a sequence of SOCP feasibility problems. This leads toward using a bisection search to find a minimum value δ^* for which the optimisation problem is still feasible.

Thus, any problem that could be formulated as the one in (8.8) can be solved as a SOCP sequence.

8.3 Robust convex optimisation formulation

Before modelling the problem, we know that extracting feature points is the first step for the solution. Since the detected feature points have some uncertainty, regardless the nature of the detector [99, 215], we also propose to include these uncertainties in the pose-graph optimisation formulation. We then solve the problem using the robust L_∞ convex optimisation technique via the SOCP.

In addition to the uncertainties in feature positions, the solution takes into account the rotational and the translation uncertainties, which are estimated through the propagation of feature position uncertainties via the multiple view geometry algorithms, using the techniques presented in Chapter 7. The general form of this robust optimisation is given as:

$$\begin{aligned} \min_x \quad & \max_{\omega} f_0(x, \omega) \\ \text{subject to} \quad & f_i(x, \omega_i) \leq 0; \forall \omega_i \in \mathcal{W}, i = 1, \dots, m \end{aligned} \tag{8.9}$$

where ω is the uncertain variable, \mathcal{W} is the uncertainty set and x is the decision variable. Similarly to (8.5), this problem can be efficiently recast and solved using second-order cone programming (SOCP) [20].

8.4 Overview of the proposed solution

The overall pipeline of the proposed solution is described in Figure 8.2. The set-up of this solution consists of an unmanned aerial vehicle (UAV) equipped with a fully calibrated monocular vision system with known intrinsic parameters K , and capturing sequence of images as it moves. In a loop-closure scenario, the UAV navigates around a cycle and returns to an already visited location. Unfortunately and due to the drift, there will be an error between the final UAV's pose and its estimate. The task of the pose-graph optimisation solution is then to correct this drift error.

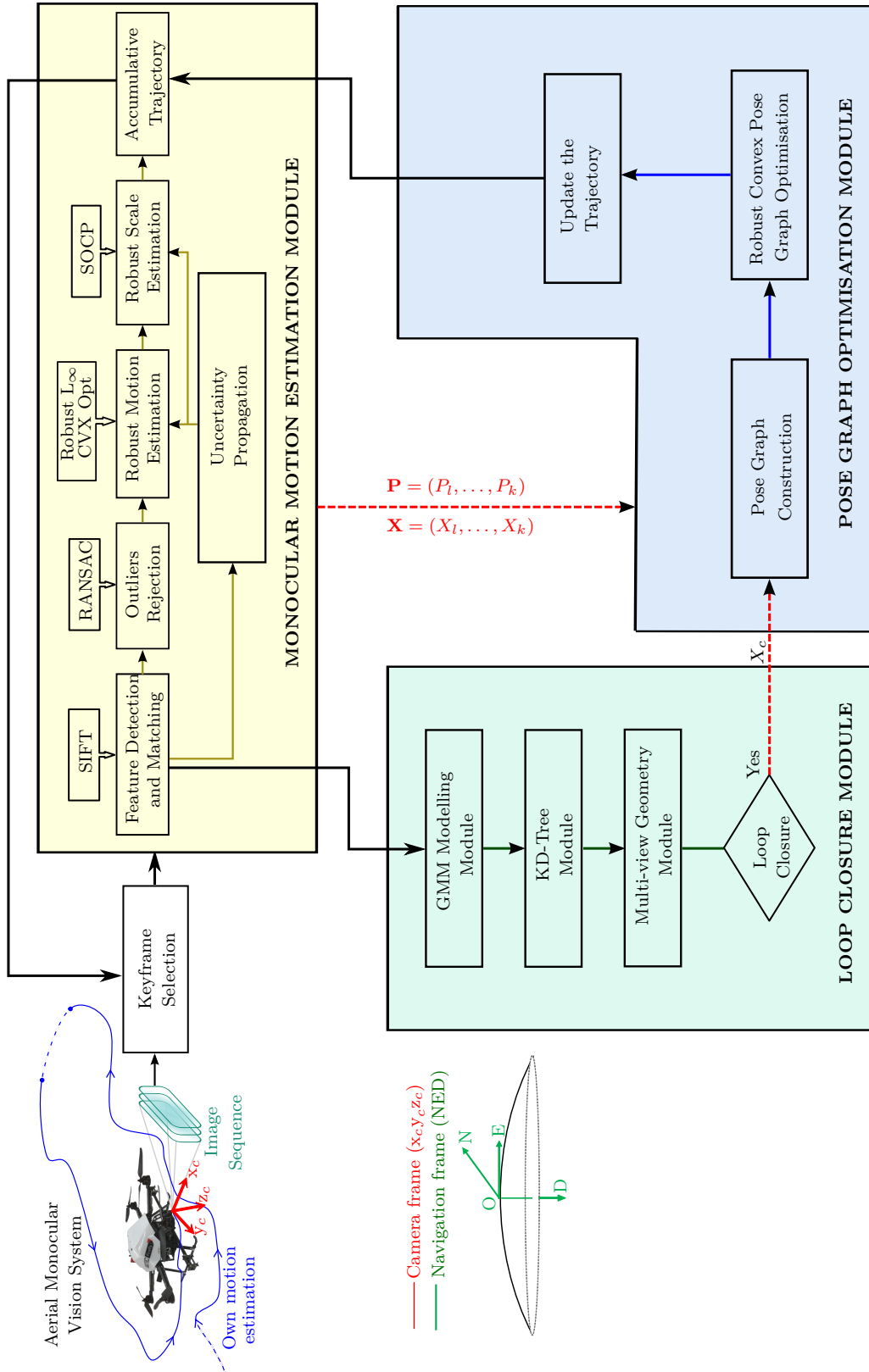


Fig. 8.2 A block diagram showing the main architecture of the proposed solution. The set-up consists of an UAV equipped with an on-board monocular vision system capturing sequence of images and estimating its own motion as it moves (Monocular Motion Estimation Module). When the UAV navigates around a cycle and returns to a previously visited position (loop-closure Module); the convex Pose-graph Optimisation Module will be then triggered to optimise the UAV poses around the loop.

As we can see from Figure 8.2, the whole solution can be divided into three main sub-tasks:

- First, we estimate the motion using a monocular visual navigation algorithm (The Monocular Motion Estimation Module).
- Second, we detect whenever a place is revisited by the UAV (loop-closure Module). The output of this module is a geometric constraint between the two camera poses, which typically represents to the same position.
- Third, the drift in the final pose estimate is corrected by distributing this drift error around the loop.

In more details, the main steps of the solution can be defined as follows:

- Acquisition of image sequences by the on-board monocular vision system.
- Keyframes selection based on the amount of the translation regarding the previous selected keyframe.
- Calling the Monocular Motion Estimation Module (MMEM) using the solution presented in Chapter 6. (the Monocular visual navigation algorithm).
- In parallel, a loop-closure Module (LCM) is launched. This module processes the data asynchronously as they arrive in order to detect any previously visited place. This module uses the solution presented in Chapter 7. (Bayes Decision Theory with Gaussian Mixture Model (GMM) in combination with the KD-tree data structure).
- If the output of the LCM is positive, which means that the vehicle has returned to an already visited location, the Robust Convex Pose-graph Optimisation Module (RCPOM) will be in turn triggered. Its ultimate aim is to optimise the vehicle poses around the loop and continuously the whole trajectory is updated accordingly.

8.4.1 Monocular motion estimation module

The Monocular Motion Estimation Module (MMEM) estimates the UAV's rotations and translations as it moves from visual inputs alone. This module adopts the previously presented solution in Chapter 6 on the on-board monocular vision system for aerial vehicles. This system is equipped with a fully calibrated camera with known intrinsic parameters K . The ultimate task of this module is to robustly estimate the UAV pose at each time step, relying only on the captured images and incorporating the system uncertainties. Hence, the main steps of this module are reminded here (Figure 8.2):

- Extraction of image feature points using the SIFT detector along with their uncertainties.
- Estimating the initial relative rotations R_i and translation t_i via the essential matrix.
- Estimation of the propagated uncertainties to R_i and t_i through the normalised 8-point algorithm and SVD.
- Estimation of the 3D scene points using convex optimisation along with their uncertainties.
- Optimising the motion using robust L_∞ convex optimisation taking into consideration all sources of uncertainty, using a sequence of camera resectioning /triangulation.
- Computing the unknown absolute scale ratio using robust least squares approach.

The outputs of this module at a given time step k are the UAV's absolute positions $P = (P_1, \dots, P_k)$, where $P_i = (x_i, y_i, z_i)^\top$, and the relative-pose estimates $X = (X_{12}, X_{23}, \dots, X_{k-1,k})$ between the selected key-frame pairs. These relative transformations are in fact the relative rotations and translations $(R_{k-1,k}, t_{k-1,k})$ between consecutive keyframes.

8.4.2 Loop-closure module

After long navigation using the monocular motion estimation module, the drift would affect the estimated positions. Even though drift during the exploration is unavoidable, it is important to keep it as small as possible. This drift is due to the integration of noisy measurements over time and to the imperfections and inherent inaccuracy of the devices. Consequently, complementary information to overcome these cumulative drift errors would become critical. As a great correction tool, the solution relies on visual loop-closures detection. This is modelled through position constraints given by the Loop Closure Module when the vehicle returns to a previously visited place. Recognising previously optimised locations would restore correct estimates and consequently allows generating consistent maps and reduces their uncertainty.

In this module, we adopt our solution for visual loop-closure detection, presented in Chapter 7. Mainly, the principal output of this module is the loop-closure constraint of the current pose of the vehicle. This pose constraint, denoted as $X_c = (R_c, t_c)$, indicates the relative rotation R_c and the relative translation t_c between the current

keyframe and the keyframe that closes the loop with it. This pose constraint is estimated using multiple view geometry algorithms. This constraint, in fact, evaluates the drift in motion estimation. In other words, this pose constraint tells us where the vehicle should be located (Figure 8.3).

8.5 Pose-graph optimisation module

Once a loop-closure is detected, the robust convex pose-graph optimisation module performs the correction of any drift occurred during the monocular motion estimation. In this section, we present the new convex pose-graph optimisation approach, which robustly corrects the rotation and the translation drift when a loop-closure is detected.

Let $X = (X_{12}, X_{23}, \dots, X_{ij}, \dots, X_{k-1,k})$ be the relative pose estimates and $P = (P_1, \dots, P_k)$ be the UAV's absolute positions estimated from the Monocular Motion Estimation Module, where k is the index of the current pose (Figure 8.3a). When a loop-closure is detected, assuming between nodes k and l , then the loop-closure constraint is estimated. This constraint defines the relative pose between the two keyframes of the loop-closure:

$$X_c = (R_c, t_c) \quad (8.10)$$

In our case $X_c = X_{kl} = (R_{kl}, t_{kl})$. The aim of the pose-graph optimisation is to find the optimal configuration of the UAV's positions $\hat{P} = (\hat{P}_1, \dots, \hat{P}_k) \in \mathbb{R}^{3n}$, that satisfies all the constraints, including the loop-closure constraint. This configuration would be obtained by an optimal distribution of the drift error over all relative constraints.

Most proposed solutions in the literature formulate this pose-graph optimisation as a least squares problem and solve it by minimising a cost function similar to one given in equation (8.2). Most of these solutions use iterative estimation methods for minimisation such as the Gauss-Newton or Levenberg-Marquardt algorithms [55, 75, 76, 186]. However, with these methods, there is no guarantee of convergence to the global minimum. Furthermore, they could lead to an infeasible solution. As such, these methods are also very dependent on good initialisation.

In contrast, our solution recovers the optimal position configuration by using convex optimisation through the adoption of a more robust norm such as the L_∞ norm. Contrarily to linear and iterative optimisation methods, convex optimisation guarantees the convergence to a single and global minimum.

8.5.1 Robust convex pose-graph optimisation formulation

In this section, we consider the scenario in which a loop-closure between the keyframes with indices k and l is confirmed by the Loop Closure Module (Figure 8.3a). In this case, the robust convex pose-graph optimisation module (RCPOM) extracts, from the monocular motion estimation module, all the UAV's absolute positions $P = (P_l, \dots, P_k)$ involved in this loop, along with their relative poses $X = (X_{l,l+1}, \dots, X_{k-1,k})$, from index k back to index l (Figure 8.3a). In addition to that, the loop-closure constraint $X_c = X_{kl}$ is also extracted from the Loop Closure Module (dashed red arrows in Figure 8.2). This constraint defines the relative pose between the loop-closure keyframes k and l . These poses will serve as inputs to the pose-graph optimisation module.

Among all the extracted absolute positions $P = (P_l, \dots, P_k)$, let us consider first any two consecutive positions P_i and P_j , and their relative pose $X_{ij} = (R_{ij}, t_{ij})$, where $l \leq i < j \leq k$. Let us now consider the two 3-element vectors p_{ij} and t_{ij} , where:

- $p_{ij} = (P_j - P_i)$ represents a vector linking the two absolute positions P_i and P_j ,
- and, the 3-element vector t_{ij} is the relative translation vector in the relative pose X_{ij} , which was estimated by the Monocular Motion Estimation Module.

The relative constraint between these two nodes will be defined as:

$$t_{ij} \equiv p_{ij} \quad (8.11)$$

The relative constraint in (8.11) means that vectors t_{ij} and p_{ij} have exactly the same orientation (Figure 8.3a). In our scenario of a loop-closure, this remains valid for all frames i and j , where $i, j = l, \dots, k$. However, due to error drift, this is not the case for the loop-closure constraint between k and l , where:

$$t_{kl} \not\equiv p_{kl} \quad (8.12)$$

This inequality (or drift error) can be expressed as an angular error θ_{kl} between the two vectors. This is illustrated in Figure 8.3b. It is clear that, due to the drift error, the node k should be moved into the node \hat{k} in order to satisfy the loop-closure constraint. To do that, we have to minimise the angle θ_{kl} . However, minimising θ_{kl} will imply changing all the graph's nodes around the loop. Thus, the job of the convex pose-graph optimisation module is to find the optimal nodes' configuration.

Note that each relative translation t_{ij} , including the loop-closure relative translation t_{kl} , will create a cone in \mathbb{R}^3 , with a vertex on the node j , an axis t_{ij} and an angle

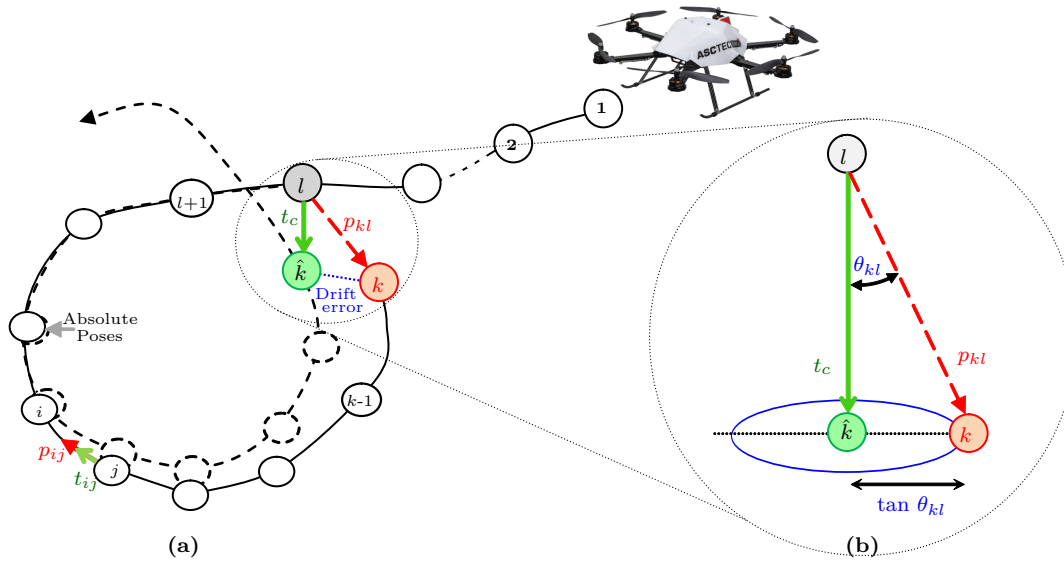


Fig. 8.3 Convex pose-graph optimisation problem. **(a)** Solid line shows the recovered UAV's trajectory before optimisation. Nodes represent the vehicle poses and edges represent constraints. After loop-closure detection (between nodes k (red) and l (gray)), the loop-closure relative pose constraint t_c (green) is estimated by the Loop-Closure Module, indicating that node k is supposed to be in \hat{k} . Clearly, the angular error between t_c and p_{kl} is considerable, while it is not between t_{ij} and p_{ij} . Convex pose-graph optimisation module corrects the trajectory (dashed line) by distributing the drift error (or the angular error) around the loop. **(b)** Depicts the geometry of the convex optimisation for angular error minimisation for one constraint.

determined by θ_{ij} (Figure 8.3b). The aim of the pose-graph optimisation module is then to distribute this angular error around the loop (dotted blue line). Hence, closing the loop as illustrated in Figure 8.3a (dashed line). In other words, the objective of the optimisation is to modify all the absolute positions P_i , in a way such that all angular errors (cones' angles) are as close to zero as possible. To do that, we consider all the relative constraints in (8.11) and the loop-closure constraint in (8.12) as measurements (constants), and the new vehicle's positions $\hat{P} = (\hat{P}_1, \dots, \hat{P}_k) \in \mathbb{R}^{3n}$ is our optimisation variable, where n is the number of nodes involved in the loop.

8.5.2 The optimisation problem formulation

To minimise the angular error θ_{kl} , one might instead minimise the tangent of this angle [84, 94, 101]. Then, the error residual associated with all t_{ij} is $\varepsilon_{ij} = \tan \theta_{ij}$, where $0 < \theta_{ij} < \frac{\pi}{2}$. These residual errors give the error vector:

$$\varepsilon = (\varepsilon_{l,l+1}, \dots, \varepsilon_{k-1,k}, \varepsilon_{k,l})^\top \quad (8.13)$$

In minimising the angular errors, the most important part of t_{ij} is its orientation, which indicates the direction between nodes. Therefore, using unit vectors divided by its norm is more appropriate. The optimal configuration \hat{P} is then the one that minimises the norm of this error vector.

In our solution, we adopt the L_∞ norm. Thus, the cost function is defined as:

$$F(x) = \|\varepsilon\|_\infty = \max_{i,j} |\varepsilon_{ij}| = \max_{i,j} \tan \theta_{ij} \quad (8.14)$$

This can be formulated as the min-max optimisation problem:

$$\text{Find } \min_{\hat{P}} \max_{i,j} \max_{i,j} \tan \theta_{ij} \quad (8.15)$$

This means that we perform a min-max optimisation over all the cones of the graph. As detailed in Chapter 3 (Section 3.7, page 67), this may be reformulated as the problem:

$$\begin{aligned} \text{Find } & \min_{\hat{P}, \delta} \delta \\ \text{subject to } & \tan \theta_{ij} \leq \delta, \forall i, j \end{aligned} \quad (8.16)$$

In order to solve this problem as a SOCP sequence, it has to be formulated as the problem in (8.8). To do that, let us reformulate $\tan \theta_{ij}$. Note that the dot product of the two vectors t_{ij} and p_{ij} that form the angle θ_{ij} is given as: $t_{ij}^\top \cdot p_{ij} = \|t_{ij}\| \|p_{ij}\| \cos \theta_{ij}$, and their cross product's length is $\|t_{ij} \times p_{ij}\| = \|t_{ij}\| \|p_{ij}\| \sin \theta_{ij}$. Therefore, dividing the cross product's length by the dot product yields:

$$\tan \theta_{ij} = \frac{\|t_{ij} \times p_{ij}\|_2}{t_{ij}^\top \cdot p_{ij}} \quad (8.17)$$

$$\tan \theta_{ij} = \frac{\|t_{ij} \times (\hat{P}_i - \hat{P}_j)\|_2}{t_{ij}^\top \cdot (\hat{P}_i - \hat{P}_j)} \quad (8.18)$$

$$\tan \theta_{ij} = \frac{\|[t_{ij}]_\times (\hat{P}_i - \hat{P}_j)\|_2}{t_{ij}^\top (\hat{P}_i - \hat{P}_j)} \quad (8.19)$$

Thus, the optimisation problem in (8.16) may be rewritten as:

$$\begin{aligned} \text{Find } & \min_{\hat{P}, \delta} \delta \\ \text{subject to } & \|[t_{ij}]_\times (\hat{P}_i - \hat{P}_j)\|_2 \leq \delta (t_{ij}^\top (\hat{P}_i - \hat{P}_j)), \forall i, j \\ & (t_{ij}^\top (\hat{P}_i - \hat{P}_j)) > 0, \forall i, j \end{aligned} \quad (8.20)$$

This falls exactly under the desired SOCP form given in (8.8), with $A_i = [t_{ij}]_\times$.

The problem in (8.20) is only solvable up to a translation and a scale. We use the constraint $\hat{P}_1 = P_1$ to remove the translation ambiguity. More importantly, to deal with scale ambiguity, we exploit the known initial absolute positions before optimisation P_i , so: $(\hat{P}_i - \hat{P}_j) \leq (P_i - P_j)$. Hence, the optimisation problem (8.20) will be rewritten as:

$$\begin{aligned}
& \text{Find} && \min_{\hat{P}, \delta} \delta \\
& \text{subject to} && \| [t_{ij}]_{\times} (\hat{P}_i - \hat{P}_j) \|_2 \leq \delta (t_{ij}^{\top} (\hat{P}_i - \hat{P}_j)), \forall i, j \\
& && (t_{ij}^{\top} (\hat{P}_i - \hat{P}_j)) > 0, \forall i, j \\
& && (\hat{P}_i - \hat{P}_j) \leq (P_i - P_j), \forall i, j
\end{aligned} \tag{8.21}$$

The optimisation problem in (8.21) is again as the desired form (8.8), which can be solved via a sequence of SOCP feasibility problems as described in Chapter 3 (Section 3.7, page 67).

8.5.3 Robust Convex pose-graph optimisation formulation

In the previous section, the convex optimisation problem is formulated with the assumption that image keypoints have been perfectly extracted with no uncertainties. However, as we have seen in Chapter 5, the detected feature points, regardless of the feature detector, have some uncertainty in their positions.

In the scenario of a loop-closure between nodes k and l , the inputs to the pose-graph optimisation are the relative poses $X = (X_{l,l+1}, \dots, X_{k-1,k})$, where $X_{ij} = (R_{ij}, t_{ij})$ and the loop-closure relative pose $X_c = X_{kl} = (R_{kl}, t_{kl})$. Therefore, since image keypoint correspondences are used to estimate t_{ij} , the uncertainties in these keypoint positions must have already propagated the relative translations t_{ij} . Let $\Delta_{t_{ij}}$ be the uncertainty in t_{ij} , which is estimated through the propagation of feature position uncertainties via the multiple view geometry algorithms as detailed in Chapter 6, then the optimisation problem in (8.20) becomes:

$$\begin{aligned}
& \text{Find} && \min_{\hat{P}, \delta} \delta \\
& \text{subject to} && \| ([t_{ij}]_{\times} + \Delta_{t_{ij}}) (\hat{P}_i - \hat{P}_j) \|_2 \leq \delta ((t_{ij} + \Delta_{t_{ij}})^{\top} (\hat{P}_i - \hat{P}_j)), \forall i, j \\
& && ((t_{ij} + \Delta_{t_{ij}})^{\top} (\hat{P}_i - \hat{P}_j)) > 0, \forall i, j
\end{aligned} \tag{8.22}$$

This again is of the desired form in (8.8), with $A_i = [t_{ij}]_{\times} + \Delta_{t_{ij}}$, which can be solved using a sequence of robust SOCP.

8.5.4 Computational complexity

In this section, we compare the L_∞ convex pose-graph optimisation complexity with the classical Levenberg-Marquardt algorithm. The computational complexity for the monocular motion estimation module and the loop-closure module are given in Chapter 6 (Section 6.8) and in Chapter 7 (Section 7.6) respectively.

For the pose-graph optimisation module, in the classical technique, the Levenberg-Marquardt algorithm has a cubic complexity in the number of parameters: $\mathcal{O}(N^3)$ per iteration (N is the number of the poses in the graph) [82]. Convex pose-graph optimisation, however, is solved by a bisection algorithm [94, 101]. The convex optimisation problems given in (8.21) and (8.22) are solved at each step by a feasibility check. This leads to solve in total N second-order cone feasibility problems. Therefore, this problem has a computational complexity of no more than $\mathcal{O}(\sqrt{N})$ and a memory requirement of $\mathcal{O}(N)$ [135].

This discussion concerns the computational complexity per iteration. In fact, the number of iterations required for convergence is extremely important. Similarly to the previous solution, the upper and the lower parameters of the bisection algorithm are chosen according to the previous optimisation parameters when a global solution was found. This technique reduces the search area and consequently fewer iterations are required for convergence.

8.6 Experimental validation

This section discusses the experimental evaluations of the proposed solution. Comparison with iterative methods based on the Levenberg-Marquardt algorithm is given. We compare our convex optimisation with the state-of-the-art solutions using the open-source implementation in [186], in which pose-graph optimisation problem is formulated as a non-linear least-squares minimisation problem and solved iteratively using the Levenberg-Marquardt algorithm.

Experiments are performed using an AscTec Firefly MAV platform (Section 1.6, Chapter 1, page 8), with a fully-calibrated forward looking camera. Implementation of these techniques is conducted using real-world data in both indoor and outdoor environments. Indoor experiments are held in our laboratory as shown in Figure 8.4. Ground-truth in the indoor experiment is collected from an OptiTrack motion-capture system, which provides absolute ground truth position information with millimetre accuracy at 100 Hz. Implementations to generate the results shown in this section are based on a sequence of robust SOCP feasibility problem, in which SeDuMi toolbox [187] is used the convexity task and Yalmip [116] toolbox is employed for

the uncertainty modelling. The estimated UAV motions are aligned with the ground truth and the Euclidean distance errors on UAV position are computed.

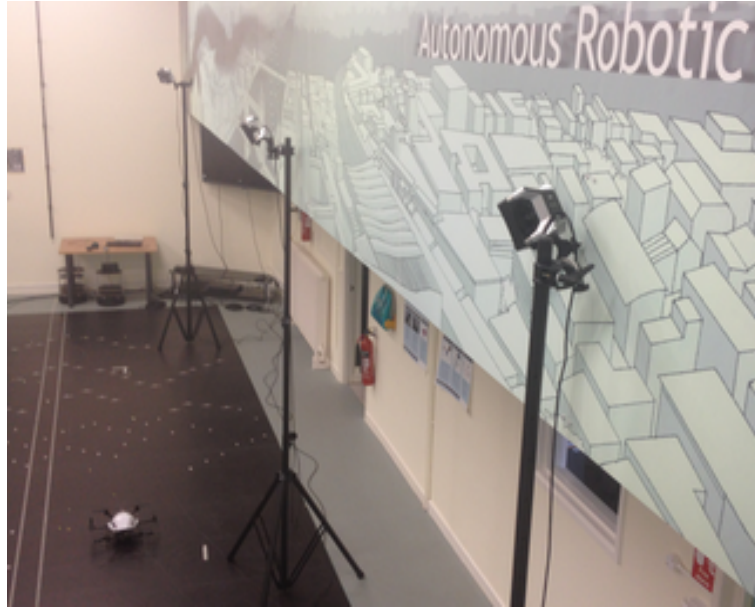


Fig. 8.4 Set-up used in the indoor experiments.

As the global solution is modular, any alternative algorithm could be used in any module among the three modules of the solution (Figure 8.2). Therefore, and in order to independently compare the performance of each module, two comparison scenarios were retained as shown in Table 8.1. These scenarios are established to assess the performance of the convex pose-graph optimisation method without the influence of the chosen loop-closure method. In the first scenario, we compare the convex pose-graph optimisation (denoted hereafter CVX) with the iterative Levenberg-Marquardt pose-graph optimisation (denoted hereafter LM) and for loop-closure detection; the same classical bag-of-words (denoted hereafter BoW) method is used. The second scenario is similar to the first one but we use the GMM/KD-Tree method for loop-closure detection instead.

Table 8.1 The two comparison scenarios

	Pose-graph optimisation method		Loop-closure detection method
Scenario 1	CVX	+	BoW
	LM	+	
Scenario 2	CVX	+	GMM/KD-Tree
	LM	+	

Investigation on the effect of using a particular loop-closure detection method on the global performance of the solution is conducted as well in this section. The robust

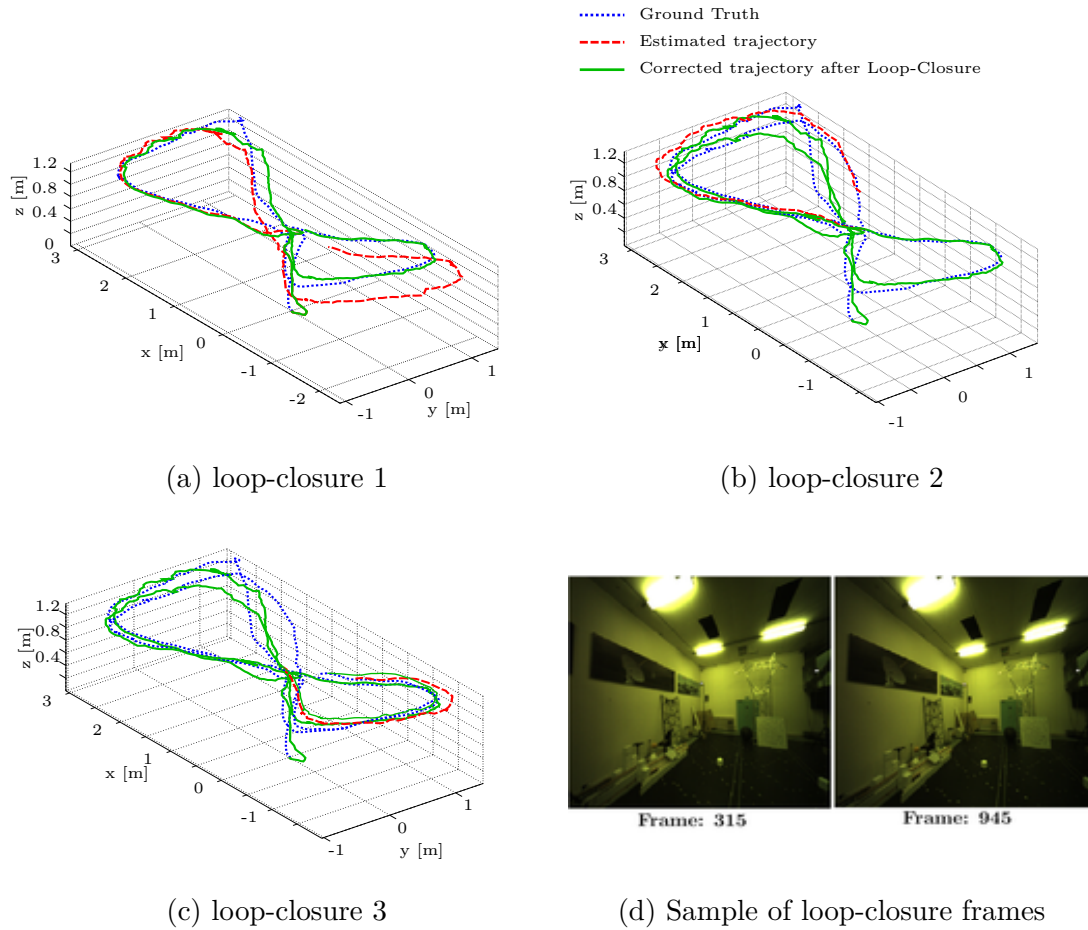


Fig. 8.5 Convex pose-graph optimisation results in the indoor experiment. Dotted blue lines show the ground truth. Dashed red lines show the UAV motion estimation before convex pose-graph optimisation and solid green lines illustrate the corrected estimates after loop-closure detection. (a) shows results when the vehicle closes the first loop. (b) and (c) depict results after detecting the second and the third loop-closures. (d) shows a sample of loop-closure frames.

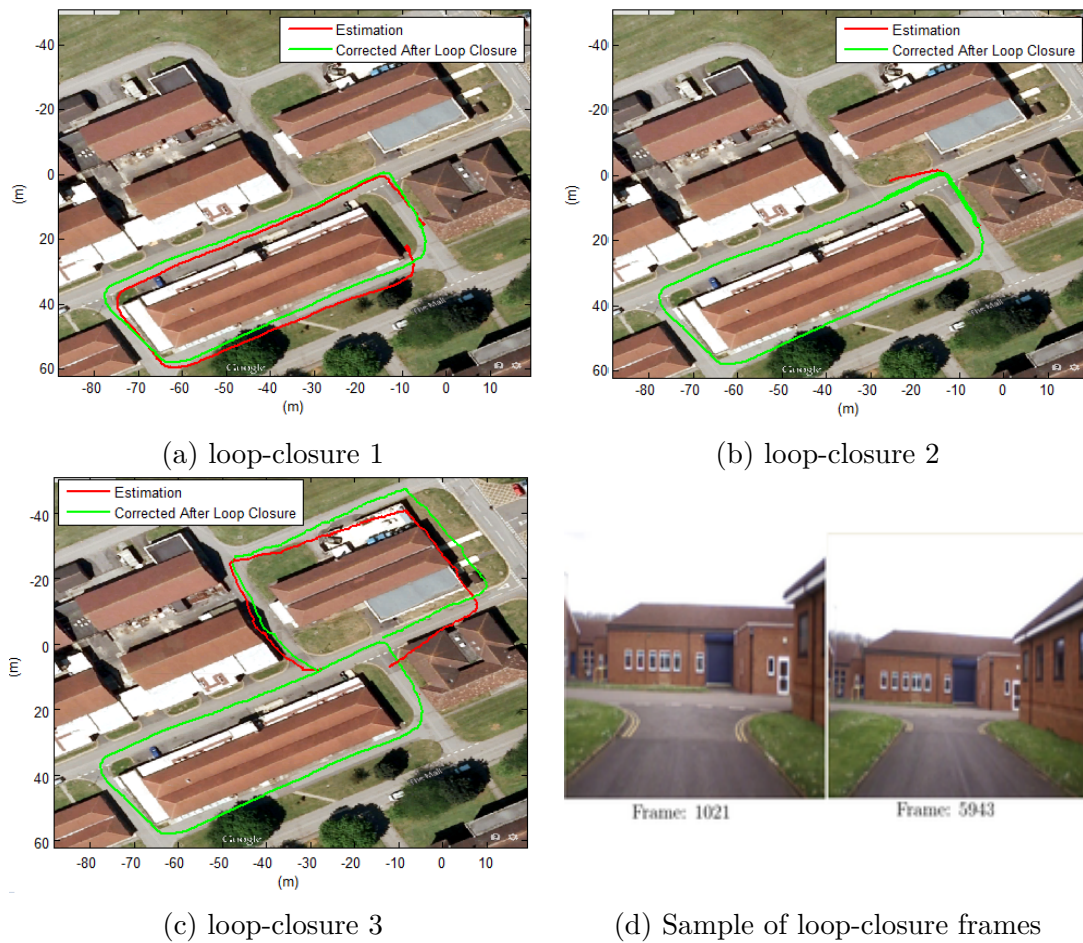


Fig. 8.6 Convex pose-graph optimisation results in the outdoor experiment. Red lines show the estimated trajectories before convex pose-graph optimisation. Green lines illustrate their corrections after loop-closures detection.

convex pose-graph optimisation performance, in which uncertainties in the relative translations are incorporated, are also compared to the normal convex pose-graph optimisation.

In order to illustrate the final output of the solution, let us first consider Figure 8.5 and Figure 8.6. These plots show the outcome of the convex pose-graph optimisation process after loop-closures detection. Trajectories before loop-closure are shown in dashed red lines. For every loop-closure detection, and before resuming monocular motion estimation, convex pose-graph optimisation is performed on all frames included in this particular loop as shown in Figure 8.5a to Figure 8.6c. Clearly, robust convex pose-graph optimisation is accurately and effectively able to correct any drift during navigation in both indoor and outdoor environments. These figures confirm that the solution suits the multiple loop-closures as well.

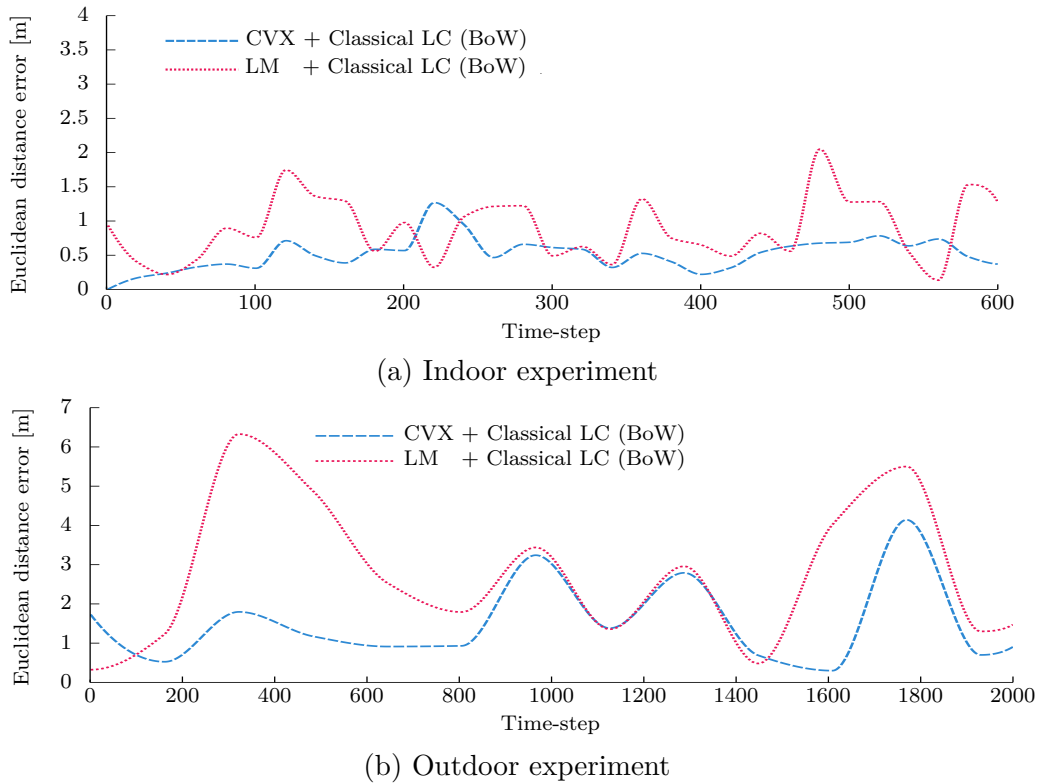


Fig. 8.7 Comparison of convex pose-graph optimisation with the classical LM method using the same classical loop-closure detection technique (BoW).

8.6.1 Assessment of pose-graph optimisation with BoW method (Scenario 1)

In this experiment, we investigate the performance of the proposed convex pose-graph optimisation (CVX) against the classical method using the Levenberg-Marquardt (LM) algorithm, where for both methods the classical Bag-of-Word (BoW) approach is employed. This experiment is conducted in indoor and outdoor environments as shown in Figure 8.7. Results obtained using convex optimisation substantially outperform those with the LM algorithm for both environments. The accuracy of the proposed convex optimisation is in accordance with the theory, as the estimates should be globally optimal. Due to its landscape nature, higher errors are still noticed after pose-graph optimisation in outdoor experiment especially for the LM method.

8.6.2 Assessment of pose-graph optimisation with GMM/KD-tree method (Scenario 2)

Similarly to the previous setup, in this experiment we use the loop-closure detection method based on Gaussian mixture modelling (GMM) with Bayesian theory and

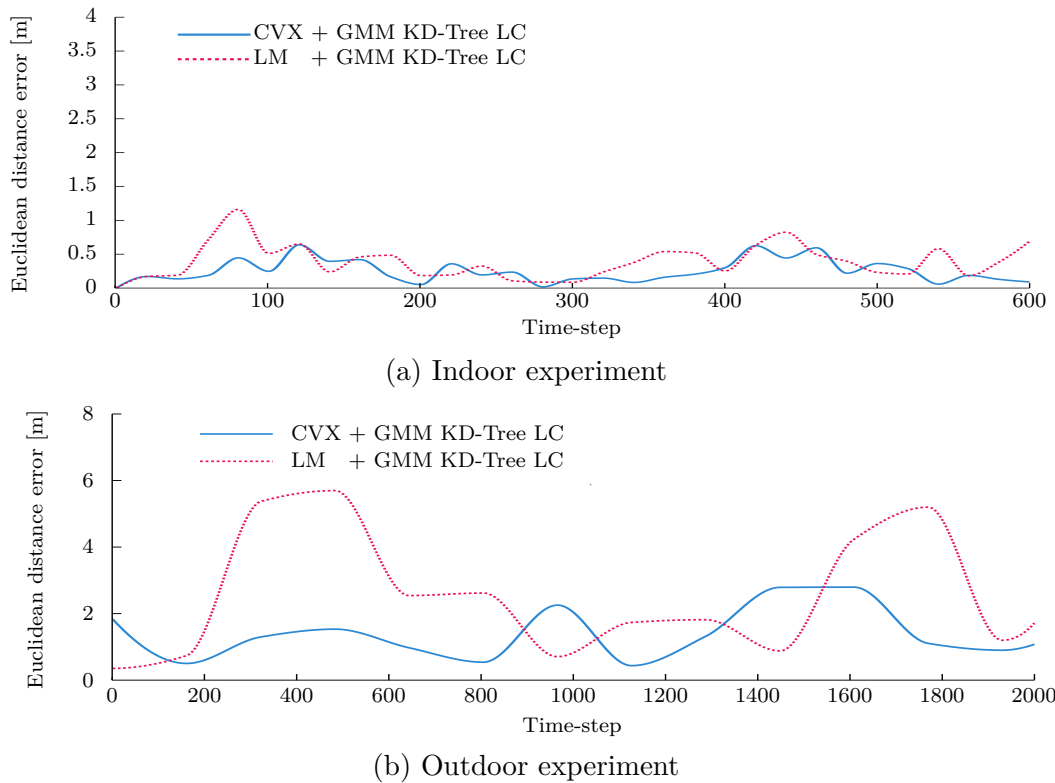


Fig. 8.8 Comparison of convex pose-graph optimisation with classical LM method using the same GMM/KD-Tree method for loop-closure detection.

KD-tree structure to test the convex pose-graph optimisation. Results are shown in Figure 8.8. In this scenario, we want to assess the performance of the convex pose-graph optimisation with the method of loop-closure based on GMM/KD-Tree. Similarly to the previous experiment, convex pose-graph optimisation outperforms the traditional LM optimisation technique. Even though the improvement is not considerable in the indoor experiment, More significant improvement can be seen in the outdoor experiment. The RMS error in indoor environment with convex method does not exceed 0.28 metres, while it is about 0.59 metres for the LM method. For the outdoor experiment, the RMS of Euclidean distance errors have dropped from 3.35 metres when using the LM method to just 1.15 metres with convex optimisation.

It can be seen, from the Euclidean distance errors, and regardless the employed loop-closure technique, that convex optimisation approach is more accurate in all environments than the classical techniques using Levenberg-Marquardt approach. Indeed, convex optimisation with L_∞ norm has shown its capability of ensuring the global minimum in recovering the motion parameters in comparison to iterative least

square based methods, where a predefined termination criterion is set, which favours convergence to local minima.

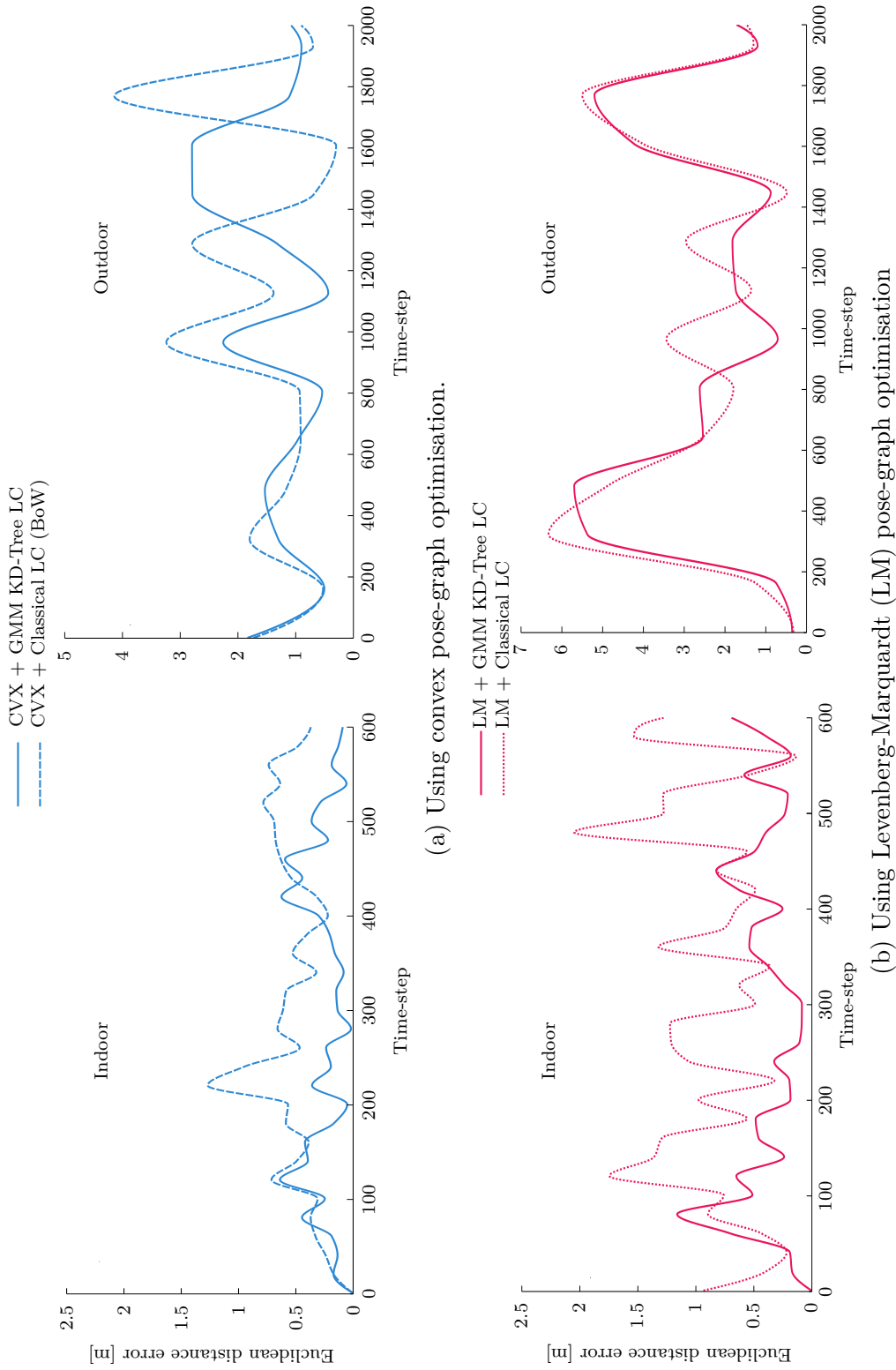


Fig. 8.9 Effects of loop-closure detection method on the global trajectories estimates. (a) and (b) compare GMM/KD-Tree (solid lines) to classical BoW technique (dotted line). (a) Convex optimisation is used (blue lines). (b) Levenberg-Marquardt (LM) technique is used (red lines).

8.6.3 Effect of loop-closure detection method

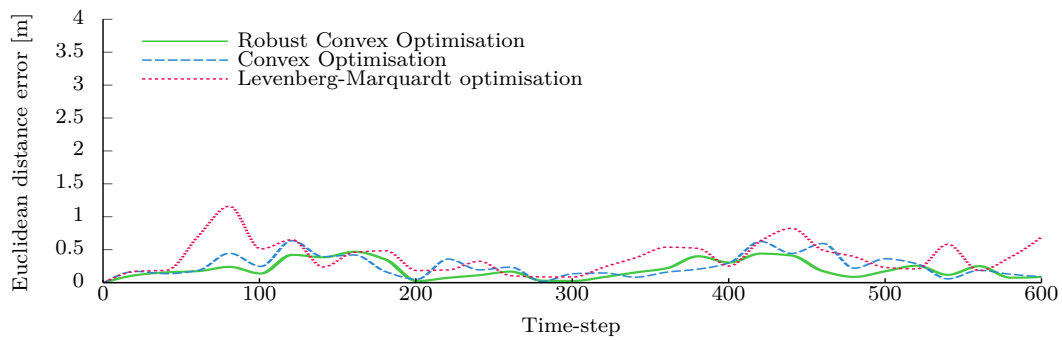
We have assessed so far the performance of the convex pose-graph optimisation regardless the employed loop-closure technique. We want to investigate, in this section, to what extent using a particular loop-closure detection method can affect the global performance of the solution. Let us consider Figure 8.9, where both pose-graph optimisation techniques (convex optimisation and Levenberg-Marquardt optimisation) are tested under the two loop-closure detection techniques (the GMM/KD-tree and the classical BoW). The top row in this figure shows the Euclidean distance errors using convex pose-graph optimisation with GMM/KD-tree (solid blue lines) and the classical BoW (dashed blue lines) in both indoors and outdoors environments. These figures show that camera positions can be accurately estimated using the GMM/KD-tree for loop-closure detection; even this improvement is not very considerable in outdoor environment. Bottom row in the same figure illustrates the effects of GMM/KD-tree for loop-closure with the Levenberg-Marquardt pose-graph optimisation. Similar pattern is noticed here as well. From these results, we can learn that in addition to the convex optimisation properties, in which solutions are guaranteed to be globally optimal, using robust and accurate methods for loop-closure detection, can improve the global performance of the solution.

8.6.4 Robust convex pose-graph optimisation

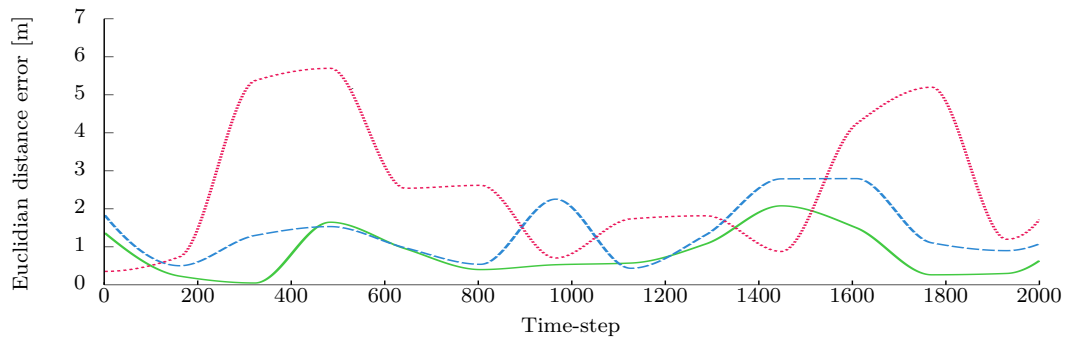
In this section, we investigate the performance of the solution when uncertainties are incorporated as given in (8.22). These uncertainties are originally estimated from image feature's imperfect positions, due to deterministic perturbations, and then propagated through the multiple view geometry algorithms to the relative translations.

Figure 8.10 illustrates the outcome of the proposed solution when uncertainties are included. Even though the improvement in indoor experiment is only reasonable (the average error of 0.20 metres when including uncertainties against 0.28 metres when using normal convex optimisation), it demonstrates the robustness of the algorithm (Figure 8.10a solid green line). This improvement can be remarkably seen in outdoor experiment, in which the average error has dropped even more when uncertainties are taken into consideration, reaching a value of 0.72 metres (was 1.15 metres when using normal convex optimisation) (Figure 8.10b solid green line).

From these results, we can learn that taking the uncertainties into consideration in the proposed method leads to robust and more accurate estimations than the normal convex optimisation as expected, since it encodes large intervals in its formulation and models well the uncertainties.



(a) Indoor experiment



(b) Outdoor experiment

Fig. 8.10 Robust convex pose-graph optimisation. Solid green lines plot Euclidean distance errors after performing robust convex pose-graph optimisation. Dashed blue lines show errors from normal convex optimisation and dotted red lines show results from classical LM method.

8.7 Conclusions

In this chapter, a novel robust convex pose-graph optimisation solution for UAVs monocular motion estimation systems is presented. Through a variety of experimental validations conducted on real-world data, from indoor and outdoor environments and comparison to state-of-the-art methods, using convex optimisation in pose-graph problems has proven its efficiency in motion estimation correction after loop-closure detections.

Furthermore, including uncertainty estimations, based on the derivative approach and their propagation through the multiple view geometry algorithms, have contributed in the improvement of the global motion estimation. The unavoidable drift in motion estimation due to imperfect computational tools and to inherent inaccuracy of the devices would be robustly and accurately corrected using robust convex optimisation. In addition, the proposed solution is suitable to multiple loop-closures circumstances.

Chapter 9

Robust L_∞ Cooperative Motion Estimation

In this chapter, a system for real-time cooperative monocular visual motion estimation with multiple Unmanned Aerial Vehicles (UAVs) is proposed. Distributing the system across a network of vehicles allows for efficient processing in terms of both computational time and estimation accuracy. The resulting global cooperative motion estimation employs state-of-the-art approaches for optimisation, individual motion estimation and registration. Three-view geometry algorithms are developed within a convex optimisation framework on-board the monocular vision systems of each vehicle. In the presented novel distributed cooperative strategy, a visual loop-closure module is deployed to detect any simultaneously overlapping fields of view of two or more vehicles. A positive feedback from the latter module triggers the collaborative motion estimation algorithm between any vehicles involved in this loop-closure. This scenario creates a flexible stereo set-up which jointly optimises the motion estimates of all vehicles, in a cooperative scheme. Prior to that, vehicle-to-vehicle relative pose estimates are recovered with a novel robust registration solution in a global optimisation framework. Furthermore, as a complementary solution, a robust non-linear H_∞ filter is designed to fuse measurements from the vehicles' on-board inertial sensors with the visual estimates. The proposed cooperative navigation solution has been validated on real-world data, using two UAVs equipped with monocular vision systems.

9.1 Introduction

9.1.1 Motivation

The ability to accurately estimate the position of a vehicle in its navigation environment is a critical prerequisite for any successful deployment of autonomous systems. Localisation, the process of resolving the position and orientation of a vehicle within the operating environment, is of particular interest over the course of a mission. In multi-vehicle systems, each vehicle may rely on its own positioning capabilities for self-localisation. In such systems, sharing a vehicle's sensing capabilities provides the ability to significantly improve the localisation accuracy of each platform within the collective. This navigation architecture is known as cooperative navigation. Vehicle-to-vehicle relative pose measurements and a vehicle's own motion estimate could lend supports to the joint estimation of the motion of a group of vehicles.

A team of collaborating vehicles leads to a solution distributed temporally and spatially. The resulting collaborative strategy exploits parallel and redundant mechanisms to gain increased robustness and efficiency. Every vehicle within a cooperative navigation approach has the ability to improve its localisation using data provided by other vehicles [11]. Collaborating autonomous vehicles finds use in various applications such as environmental mapping, search and rescue, and aerial surveillance [16]. To successfully accomplish their missions, vehicles participating in such applications require accurate estimates of their localisation.

Each vehicle is equipped with its own navigation system and estimates its own localisation independently. However, if a fleet of vehicles is deployed, a cooperative strategy would potentially bring greater efficiency to the system. Furthermore, localisation accuracy will also be much improved, besides additional gains in efficiency with regards to vehicles performing their own specific tasks [134].

9.1.2 Visual sensing

There is common agreement that satellite navigation systems, such as GPS, are dependable as navigation aids in large-scale integrated systems. This is due their being easy to implement and efficient in most circumstances. However, in critical situations, they are required to operate in tough environments where navigation maps are unavailable and GPS signals may be denied, their use is inappropriate. Urban, indoor, or underground locations during natural disasters, for instance, present big handicaps to these satellite-based navigation systems, whereby the satellite signals are either intermittent or not available at all. Therefore, alternative positioning systems are required.

From a practical standpoint, many aspects need to be taken into consideration in the choice of sensors to be fit on-board unmanned aerial vehicles (UAVs). In urban and indoor environments, small-sized and lightweight platforms are most likely to fulfil the mission requirements. On the other hand, power autonomy and payload restrictions impose further constraints. In the case of laser scanners for example, their weight and large power consumption rule them out of contention. Inertial navigation system (INS), on the other hand, can be used in such situations. Nevertheless, accumulation of INS errors is prohibitive their application. Uncertainty, if let uncorrected, can grow without bound.

Consequently, as we have seen in the previous chapters, investigating alternative solutions, such as visual systems, has become the focus of a number of research programmes. Indeed, vision systems appear better suited for UAVs and attract significant research interests due to their light weight, low power consumption, low cost and high fidelity of the information they provide. For instance, a fleet of UAVs equipped with high resolution cameras can be deployed in operations where each UAV relies solely on its visual sensing. However, this raises the problem of estimating the vehicle's position in real-time.

As detailed in Chapter 8, in vision-aided autonomous navigation systems, platforms are usually equipped with stereo systems formed of two cameras. These systems exploit the known distance between the two cameras, usually called the baseline, to remove any ambiguity in motion estimation. These systems, however, present an important drawback when their baseline is relatively small with regard to the distance from the scenery. In reality, for aerial systems, the scene into consideration has to be observed from a sufficiently large baseline, which is non-practical for UAVs. Therefore, observing scenes from much higher altitudes in comparison to the camera's baseline reduces a stereo set-up to almost a bearing-only sensor, suffering similar depth estimation problems as in the monocular case (using a single camera). Such considerations spur more researchers to focus on monocular systems.

Indeed, monocular vision systems have become an indispensable solution for autonomous aerial navigation systems, since they are practical and offer cheap and compact installations. However, one of the most challenging issues of the monocular visual odometry is the scale ambiguity due to the projective effects. In a monocular cooperative navigation framework, where multiple vehicles are navigating within a common environment, each vehicle constructs its own map 3D points for motion estimation purposes. Clearly, this raises the necessity of studying the relationship between these points (relative orientation and translation). One may see this problem as working out the transformation between two different systems. This problem is known in computer vision as the registration problem [125, 193].

The registration problem can be addressed in several contexts: from estimating the absolute pose given three dimensional measurements to solving the hand-eye calibration problem. In our cooperative navigation framework, we are interested in recovering the relative transformation between two UAV cloudreconstructions. Specifically, suppose we are given a number of points from a scene measured in two different coordinate systems. This would create two point patterns (sets of points). Our aim, then, is to find the optimal transformation parameters (rotation and translation) that transform the first pattern into the second. This allows estimating the relative rotation and translation between the two UAVs, which is used in turn in estimating accurate 3D scene points for our cooperative navigation.

9.1.3 Related work

Tremendous interest has been directed to unmanned mobile systems operating in dynamic and complex environments over recent decades. A relatively small portion of that has been directed toward cooperative systems. Among these studies, the vast majority of cooperative navigation applications have been implemented using range sensors, i.e. sonar, lidar and laser [74]. Indeed, ground wheeled robots navigating on two dimensional planes have greatly dominated the autonomous navigation systems literature. Limited interest has been shown toward visual cooperative navigation, especially on autonomous aerial platforms. Relying on cameras as a primary sensing tool is still an emerging research area. Interestingly, most studies in this area have focused on building 3D maps [164, 170]. These approaches are usually designated as visual simultaneous localisation and mapping (VSLAM)[213].

Most of the visual systems have focused solely on stereopsis vision, where a set-up with two cameras with a fixed relative transformation is designed. However, as mentioned before, the stereo configuration in aerial navigation systems has evidenced some limitations, notably when the scene under consideration is relatively distant from the platform. This would inhibit its ability of recovering the real scene depths, which is the main purpose of the stereopsis configuration. However, less consideration has been given to monocular vision systems. Indeed, such bearing-only sensors present immense challenges, especially of unconstrained (6 DoF) motion estimation for aerial vehicles [40, 125].

The authors in [197] deployed an extended Kalman filter (EKF) to fuse monocular motion estimates from two distinct cameras, where their algorithm is implemented on ground vehicles. In [4], an EKF is again used as well to recover the relative pose between two UAVs equipped with a monocular vision system. Filtering is again

adopted in [205], where a cooperative aerial/ground robotic system supplied with stereo cameras is proposed.

An adaptive cooperative visual localisation solution based on a stereo vision system is proposed in [138]. In the latter, a robust non-linear H_∞ filter is designed in a cooperative VSLAM (C-VSLAM) framework. Both aerial and ground platforms were deployed separately to validate this approach. In [220], a vision-based SLAM with multiple hand-held cameras for dynamic environments is presented. Some assumptions related to covering the same scene at the start and the simultaneous processing of images from different cameras make this approach impractical. A maximum a posteriori estimator is used in [139], where an approach to distribute the data processing between vehicles is adopted. In [45], a decentralised approach for multi-robot SLAM is presented, in which each platform maintains a local map that combines local information shared by other robots into a global augmented map. In the cooperative visual simultaneous localisation and mapping (C-VSLAM) framework, some studies have dealt with system uncertainty during the different stages of cooperation [134]. These studies challenge the localisation performances within their growing uncertainties. Information theory and entropy minimisation have also previously been used in cooperative SLAM [30, 157].

The filtering option, rather than optimisation, has drawn more attention among computer vision researchers. However, these sorts of solutions are highly sensitive to outliers and the lower bound for map accuracy, due to errors introduced during the filter linearisation process, which produces inconsistent estimates.

Our work, however, deals with the cooperative navigation problem from a completely different perspective. Global optimisation by means of convex and quasi-convex optimisation and branch-and-bound algorithms is adopted for our registration-based navigation. In addition to global optimisation, and in order to overcome EKF non-representative models, a non-linear H_∞ filter is adopted. The latter filter is specifically designed for the relative pose estimation between the UAVs.

The remainder of this chapter is structured as follows: Section 9.2 provides an overview of global optimisation. Section 9.3 details the general cooperative motion estimation solution pipeline. Section 9.4 presents the solution adopted for the three-view geometry motion estimation. Section 9.5 explains our registration solution. Section 9.6 describes the implementation design for the non-linear H_∞ filter for relative pose estimation. Experimental validation is provided in Section 9.7, followed by Section 9.8 giving the main conclusions of this work.

9.2 Global optimisation

In this section, a brief introduction reminder to the principal concepts of global optimisation, including convex optimisation [25] and branch-and-bound algorithms, [104] is given. These two techniques are the main tools used for the proposed solution in this chapter. Convex optimisation is presented in detail in Chapter 3, including quasi-convex optimisation (Section 3.5, page 64) and the second-order cone programming (SOCP) (Section 3.7, page 67).

In computer vision, certain constrained optimisation problems cannot be solved directly. This is due to the non-convexity of either their objective functions or their constraints. Therefore, convex optimisation cannot be deployed in such cases.

Conveniently, some algorithms, such as branch and bound algorithms, are able to overcome this issue and provide globally optimal solutions. Branch and bound algorithms are iterative methods for finding global optima in non-convex problems.

These algorithms employ a cleverly structured search of the space of all feasible solutions. As presented in Section 3.9 (page 75), this space is continuously and repeatedly partitioned (Branch) into narrower and narrower subsets while a lower bound (Bound) is estimated for the cost of the solutions within all subsets [107]. That is why this method is also called progressive separation and evaluation [107]. A known feasible solution is always kept for reference the least costly so far. Obviously, after each partitioning, the subsets that cost more than the known least costly solution are discarded from further partitioning. This partitioning policy continues until a feasible solution is found, such that its cost is no greater than the bound of all subsets. This provides satisfactory ε -suboptimal solutions. Figure 3.12 (Chapter 3, page 76) presents a detailed example of the branch and bound algorithm used to solve this problem

A practical way to solve a more complicated problem is to solve a related simpler problem, and hope the latter's solution can be a solution to the original problem. This is the core principle of branch and bound algorithms. However, instead of replacing the original problem with one single problem, a set of problems that bound the original problem is used.

Suppose a tree structure is used to visualise this principle. At the beginning, the tree is represented with one single node (root-node) which represents the original problem. Problems that replace this original problem in the bounding set are pointed to by branches that create more nodes (hence the term 'Branch'). Usually, a convex relaxation is performed resulting in a new convex problem. If the optimal solutions of the newly created problems (nodes) are not feasible in the original problem or they cost more, new branching is performed, creating more leaf-nodes. At any

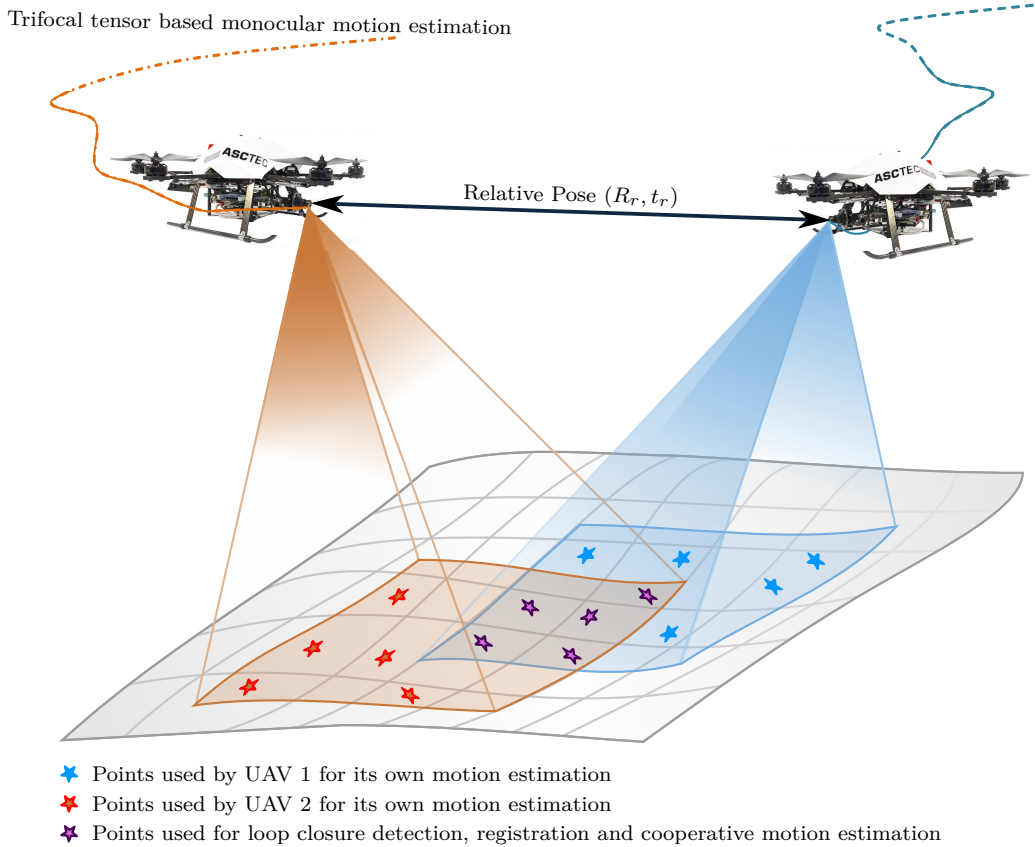


Fig. 9.1 The set-up used in the cooperative motion estimation. Each UAV is equipped with monocular visual system. In a loop-closure scenario, the UAVs construct a flexible stereo set-up. This set-up is used to reconstruct the 3D scene points which are in turn used in cooperative navigation.

intermediate stage, the least costly solution \hat{x} is retained. Any leaf-node of the tree whose bound \tilde{x} costs less than \hat{x} will be kept in the tree; otherwise it will be discarded from any further consideration [107].

In view of its robust performance, this technique is used to solve our registration-based navigation problem in this chapter.

9.3 Overview of the proposed solution

The overall aim in this work is to extend the state-of-the-art for autonomous cooperative navigation systems. Our solution employs new optimisation techniques that guarantee the global minimum, such as convex optimisation. The general set-up of the solution is illustrated in Figure 9.1. The architecture consists of two UAVs equipped with fully calibrated monocular systems capturing sequences of images

and estimating their own motion as they move. The solution rests on three main sub-solutions:

- a three-view-geometry-based solution for monocular motion estimation using convex optimisation,
- global optimisation for the registration problem between the UAVs,
- and a non-linear H_∞ -filter-based solution for the relative pose estimation problem.

The overall pipeline of the proposed solution is described in Figure 9.2. The whole solution can be divided into three main sub-tasks. We first, independently, estimate the motion of each vehicle using its on-board monocular visual systems. Unlike most solutions presented in the literature, our approach adopts the three-view geometry for this task (Figure 9.2). As proven in the literature, this approach ensures more consistent and stable solutions over the two-view geometry approach [83, 173, 174, 184, 206]. Moreover, estimating the trifocal tensor in our approach is performed through convex optimisation, wherein getting a single global solution is guaranteed. This is to overcome the linear and iterative methods problems, which present a high probability of getting trapped in a local minimum or reaching infeasible solutions, even for low noise levels [25, 80, 94, 101, 116].

In a loop-closure scenario, the two UAVs simultaneously fly over the same location (Figure 9.1). In other words, the UAVs have a relatively significant overlap in their fields of view. A positive feedback from the loop-closure detection module (Figure 9.2) triggers the cooperative motion estimation between any UAVs involved in this loop-closure.

The cooperative motion estimation starts by first estimating the relative pose between vehicles. This relative pose is of great importance to our solution. Recovering a consistent full 6 DoF relative pose between two vehicles in metric units and in real time will allow the creation of a flexible stereo vision set-up. At this point, one can use the stereo vision's advantages in order to compensate for the monocular vision's drawbacks. This way one can exploit the power of both monocular and stereo vision while avoiding their drawbacks.

It is known that using solely monocular systems makes the motion estimation challenging due to the absolute scale ambiguity caused by projective effects. A measured point in a particular image can represent the projection of an infinite number of 3D scene points in the world. Consequently, the estimated translation vectors from frame-to-frame image point correspondences suffer from scale ambiguity. Some monocular solutions in the literature have implicitly solved this scale ambiguity. Linear approaches in combination with some optimisation tools are also used to recover the global scale factor.

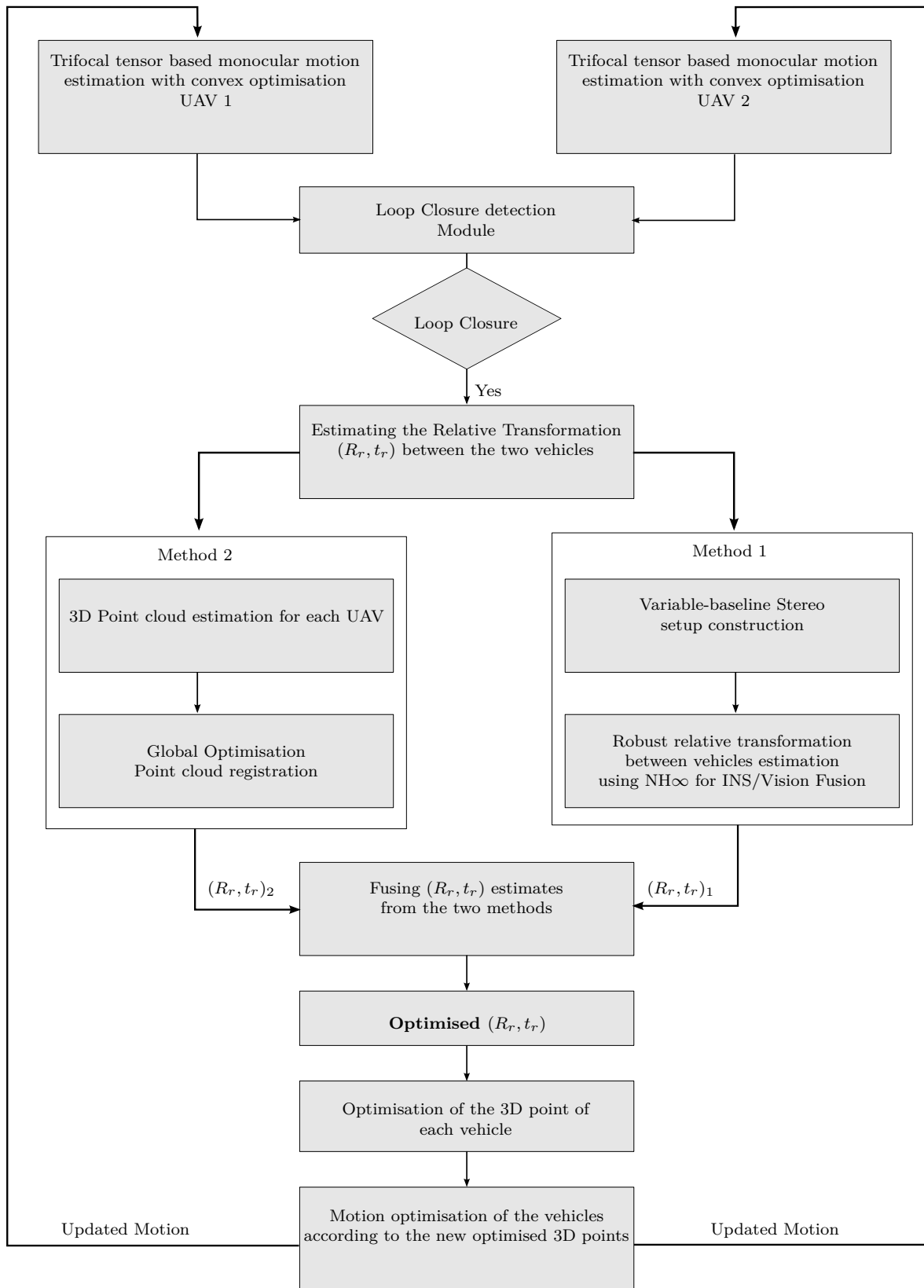


Fig. 9.2 A block diagram showing the main architecture of the proposed solution. The set-up consists of two UAVs equipped with monocular vision systems capturing sequences of images and estimating their own motion as they move using three-view geometry. If commonly-viewed scenery is detected by the loop-closure module, the cooperative motion estimation module is triggered to optimise the UAVs poses.

On the other hand, stereo visual odometry takes advantage of the known baseline to directly remove any scale ambiguity. This baseline provides two viewpoints distinct enough to recover the scene. However, it suffers from substantial lack of accuracy when the distance to the 3D scene points is much larger than the baseline, which is the case for most aerial platforms since it is impractical to install a stereo rig with a large baseline on one UAV. The baseline in fact defines the relative rotation and translation between the left and the right cameras in the stereo rig.

Our ultimate aim is to use two monocular systems (on-board each cooperative vehicle) and construct a stereo rig with a sufficiently large baseline between the two monocular systems, forming a variable baseline stereo set-up (Figure 9.1). This would solve our problem with regard to scale ambiguity. The challenge then become how to accurately estimate this variable baseline (relative pose) in real time. Since this relative pose has great impact on the subsequent cooperative motion estimation, our strategy relies on a complementary configuration in estimating it. Two separate techniques are used for computing a consistent estimate of this entity. The two estimates are then fused, aiming to increase robustness, stability and consistency.

For the first technique, we rely on the point cloud registration problem. The second technique, on the other hand, fuses inertial measurements from the on-board inertial measurement units (IMUs) with the vision estimates. After constructing the stereo vision set-up, 3D scene points can more accurately be estimated, capitalising on the stereo vision's advantages. Note that the monocular motion estimation in each vehicle relies heavily on their self-estimated 3D scene points in order to remove the frame-to-frame translation ambiguity [23, 54]. Thus, any uncertainty with regards to the self-estimated 3D scene points will propagate to the final motion, inducing unstable estimates. Therefore, optimised ambiguity-free 3D scene points estimated through the stereo vision would certainly improve the individual motion estimates of each vehicle.

Details about each block of the proposed solution in Figure 9.2 are given in the following sections.

9.4 Three-view geometry motion estimation

This section presents the algorithm for our motion estimation method for each vehicle, which is based on the robust computation of trifocal tensors using image features. Our motion estimation solution, described in Figure 9.2, assumes a fully calibrated monocular system with known intrinsic parameters K . Using a vehicle equipped with a single camera capturing sequences of images, the final goal is to estimate the camera pose at each time step relying only on these images. Thus, for

each consecutive image triad, the whole algorithm is summarised in the following operations:

- Extraction of image feature points using SIFT detector,
- Estimating the trifocal tensor between the triad (previous, current, next) frames, using feature point correspondences through convex optimisation,
- Extracting the camera matrices from the trifocal tensor,
- Estimating the initial relative rotations and translations $(R_i, t_i$ and $R_j, t_j)$ between the current-to-previous and the current-to-next frame pairs respectively,
- Estimation of the 3D scene points using convex optimisation for each frame pair (current-to-previous and current-to-next),
- Optimising the motion between each frame pair using robust L_∞ convex optimisation,
- Computing the unknown absolute scale ratio of each frame pair using robust least squares via SOCP.

9.4.1 Trifocal tensor estimation using convex optimisation

Three-view geometry has several advantages over algorithms that use two views only. Adding a third image provides a good level of stabilisation to the subsequent solutions [83, 174, 206]. More importantly for our work, the introduction of a third view has the ability to stabilise 3D point estimation algorithms from 2D images. We have seen in Chapter 2 (Section (2.10), page 42) that the trifocal tensor is the algebraic representation of the three-view geometry. Indeed, the trifocal tensor exploits better the information available in three views in comparison to the fundamental matrix in two views [174]. Thus, in our work, an investigation on the adoption of the three-view geometry on the motion estimation task is conducted. Moreover, a new approach for estimating the trifocal tensor parameters via convex optimisation is proposed as well in this work.

In our monocular system, the multiple view geometry is created by distributing frame sequences over time. Correspondences between three frames are used to estimate the trifocal tensor from which, along with the camera calibration parameters, the motion parameters are estimated.

First, let the 3×4 camera matrices for three views be $P = [I|0]$, $P' = [a_1 \ a_2 \ a_3 \ a_4] = [A|a_4]$ and $P'' = [b_1 \ b_2 \ b_3 \ b_4] = [B|b_4]$, where I is the identity matrix, A and B are 3×3 matrices and the vectors a_i and b_i are the i^{th} columns of the respective camera matrices for $i = 1, \dots, 4$. Then, the quantity we wish to find is

the trifocal tensor \mathcal{T} , which is parametrised as [82] (Figure 2.8, Chapter 2, page 43):

$$\mathcal{T}_i = a_i b_4^\top - a_4 b_i^\top, i = 1, \dots, 3 \quad (9.1)$$

In this equation, the set of three matrices $\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3$ constitute the $3 \times 3 \times 3$ trifocal tensor, where \mathcal{T}_i is 3×3 with rank 2. Note that the relationship between the three cameras is completely described by $\mathcal{T}_1, \mathcal{T}_2$ **and** \mathcal{T}_3 .

Let $\mathbf{x} = (x_1, x_2, x_3)$, $\mathbf{x}' = (x'_1, x'_2, x'_3)$ and $\mathbf{x}'' = (x''_1, x''_2, x''_3)$ be corresponding image points in the three views respectively. The relationship between these three image point correspondences $\mathbf{x} \leftrightarrow \mathbf{x}' \leftrightarrow \mathbf{x}''$ is given by the point-point-point relation [82, 174]:

$$[\mathbf{x}']_\times \left(\sum_{i=1}^3 x_i \mathcal{T}_i \right) [\mathbf{x}'']_\times = 0_{3 \times 3} \quad (9.2)$$

In the case of the line correspondence $l \leftrightarrow l' \leftrightarrow l''$ in the three images, then, it can be verified that (9.1) is equivalent to:

$$l_i = l'^\top \mathcal{T}_i l'' \quad , i = 1, \dots, 3 \quad (9.3)$$

where $l = (l_1, l_2, l_3)$. Using the tensor notation, the above relations can be equivalently written as:

$$\mathcal{T}_i^{jk} = a_i^j b_4^k - b_4^j b_i^k \quad (9.4)$$

$$l_i = l'_j l''_k \mathcal{T}_i^{jk} \quad (9.5)$$

The trifocal tensor has 27 elements; however, it represents an epipolar geometry that has only 18 degrees of freedom. Clearly, there are some dependencies between the tensor parameters. These dependencies, in fact define the internal constraints that must be satisfied by any estimated tensor. Otherwise, the tensor would not consistently encode any meaningful geometry [82, 83, 174, 206]. The rank constraint is the most trivial constraint. Specifically, each \mathcal{T}_i must be of rank 2.

To estimate the trifocal tensor parameters, 26 equations are required. Each triplet of corresponding points provides four independent linear equations. Accordingly, 7 points are required to compute the trifocal tensor linearly. One can use linear algorithms or normalised linear algorithms (when input data are normalised), to estimate this tensor as discussed in [82]. This solution minimises an algebraic error using the least-square approach via the singular value decomposition (SVD). Then the iterative Levenberg-Marquardt (LM) algorithm is used to optimise the trifocal tensor. However, in a major drawback, the quantity being minimised in the first

part of the algorithm is not geometrically or statistically meaningful. Secondly, the iterative part using the LM algorithm is prone to be affected by local minima. Moreover, the success of this approach is heavily dependent on good initialisation.

As an alternative, our solution adopts convex optimisation, which guarantees getting the global minimum by minimising the geometric image-plane distance. In addition to getting the global minimum, another advantage is that the cost function is geometrically meaningful.

Suppose we have three views of a set of scene points X_i . These points are imaged as x_i , x'_i and x''_i in the first, second and third view respectively. The corresponding point sets $x_i \leftrightarrow x'_i \leftrightarrow x''_i$ may then be related by the trifocal as given in (9.2). Moreover, using (9.3), a point x_i in the first view can be transferred to the third view using the line l'_i passing through the point x'_i in the second view as:

$$\tilde{x}''_i{}^k = x_i l'_i{}^j \mathcal{T}_i{}^j \quad (9.6)$$

This is known in the trifocal tensor framework as point transfer. Specifically, if the transferred point is $\tilde{x}''_i = (\tilde{x}''_i{}^1, \tilde{x}''_i{}^2, \tilde{x}''_i{}^3)$, then each coordinate $\tilde{x}''_i{}^k$ is given by:

$$\tilde{x}''_i{}^k = x_i^u l'_i{}^j \mathcal{T}_u{}^{jk} \quad (9.7)$$

Our aim is to find the parameters of the trifocal tensor $\mathcal{T} = [\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3]$ that satisfy this point transfer along with its constraints using convex optimisation. Our optimisation variable is then:

$$\mathbf{x} = (\mathcal{T}_1^{11}, \dots, \mathcal{T}_1^{33}, \mathcal{T}_2^{11}, \dots, \mathcal{T}_2^{33}, \mathcal{T}_3^{11}, \dots, \mathcal{T}_3^{33}) \in \mathbb{R}^{27} \quad (9.8)$$

Suppose that x_i , x'_i and x''_i are scaled such that their last coordinate is equal to one. To estimate this trifocal tensor by the proposed method, the geometric distance error is used:

$$\varepsilon_i = d(x''_i, \tilde{x}''_i) = d(x''_i, x_i l'_i{}^j \mathcal{T}_i) \quad (9.9)$$

This equation defines the Euclidean distance error between the measured point x''_i and the transferred point \tilde{x}''_i . This error is due to measurement errors in the image points. Each triad of corresponding points ($x_i \leftrightarrow x'_i \leftrightarrow x''_i$) will then have an error residual associated with its noise. Our aim is then to recover the trifocal tensor \mathcal{T} which minimises the maximum of this error across all image point correspondences. To evaluate these residual errors, let us first estimate each coordinate $\tilde{x}''_i{}^k = x_i^u l'_i{}^j \mathcal{T}_u{}^{jk}$ of the transferred point. Suppose $x_i = (x_i^1, x_i^2, 1)^\top$, $l'_i = (l'_i{}^1, l'_i{}^2, l'_i{}^3)^\top$ and $\tilde{x}''_i = (\tilde{x}''_i{}^1, \tilde{x}''_i{}^2, \tilde{x}''_i{}^3)$, then, using (9.7), the first coordinate of the transferred point to the

third view is given by:

$$\tilde{x}_i''1 = \mathbf{x}_i^\top \begin{bmatrix} l_i^1 \mathcal{T}_1^{11} + l_i^2 \mathcal{T}_1^{21} + l_i^3 \mathcal{T}_1^{31} \\ l_i^1 \mathcal{T}_2^{11} + l_i^2 \mathcal{T}_2^{21} + l_i^3 \mathcal{T}_2^{31} \\ l_i^1 \mathcal{T}_3^{11} + l_i^2 \mathcal{T}_3^{21} + l_i^3 \mathcal{T}_3^{31} \end{bmatrix} \quad (9.10)$$

In order to simplify this transfer equation, we introduce the notation \mathcal{T}^{vw} . This new column vector $\mathcal{T}^{vw} = (\mathcal{T}_1^{vw}, \mathcal{T}_2^{vw}, \mathcal{T}_3^{vw})^\top$ is formed by taking the $(v, w)^{th}$ element from $(\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3)$. Then, (9.10) can be reformulated as:

$$\tilde{x}_i''1 = \mathbf{x}_i^\top \left(l_i^1 \mathcal{T}^{11} + l_i^2 \mathcal{T}^{21} + l_i^3 \mathcal{T}^{31} \right) \quad (9.11)$$

Similarly, the second and the third coordinates of the point transferred to the third view are given by:

$$\tilde{x}_i''2 = \mathbf{x}_i^\top \left(l_i^1 \mathcal{T}^{12} + l_i^2 \mathcal{T}^{22} + l_i^3 \mathcal{T}^{32} \right) \quad (9.12)$$

$$\tilde{x}_i''3 = \mathbf{x}_i^\top \left(l_i^1 \mathcal{T}^{13} + l_i^2 \mathcal{T}^{23} + l_i^3 \mathcal{T}^{33} \right) \quad (9.13)$$

The transferred point is scaled such that its last coordinate is equal to one:

$$\hat{\mathbf{x}}_i'' = \left(\frac{\tilde{x}_i''1}{\tilde{x}_i''3}, \frac{\tilde{x}_i''2}{\tilde{x}_i''3} \right)^\top \quad (9.14)$$

Then, the i^{th} error residual given in (9.9) is rewritten as:

$$\begin{aligned} \varepsilon_i &= d(\mathbf{x}_i'', \mathbf{x}_i l_i^j \mathcal{T}_i^j) = \sqrt{\left(\frac{\tilde{x}_i''1}{\tilde{x}_i''3} - x_i''1 \right)^2 + \left(\frac{\tilde{x}_i''2}{\tilde{x}_i''3} - x_i''2 \right)^2} \\ &= \sqrt{\frac{(x_i''1 - x_i''1 \tilde{x}_i''3)^2 + (x_i''2 - x_i''2 \tilde{x}_i''3)^2}{(\tilde{x}_i''3)^2}} \\ &= \sqrt{\frac{f_{i1}(\mathbf{x})^2 + f_{i2}(\mathbf{x})^2}{f_{i3}(\mathbf{x})^2}} \end{aligned} \quad (9.15)$$

where:

$$\begin{aligned} f_{i1} &= \mathbf{x}_i^\top \left[(\mathcal{T}^{11} l_i^1 + \mathcal{T}^{21} l_i^2 + \mathcal{T}^{31} l_i^3) - x_i''1 (\mathcal{T}^{13} l_i^1 + \mathcal{T}^{23} l_i^2 + \mathcal{T}^{33} l_i^3) \right] \\ f_{i2} &= \mathbf{x}_i^\top \left[(\mathcal{T}^{12} l_i^1 + \mathcal{T}^{22} l_i^2 + \mathcal{T}^{32} l_i^3) - x_i''1 (\mathcal{T}^{13} l_i^1 + \mathcal{T}^{23} l_i^2 + \mathcal{T}^{33} l_i^3) \right] \\ f_{i3} &= x_i''1 (\mathcal{T}^{13} l_i^1 + \mathcal{T}^{23} l_i^2 + \mathcal{T}^{33} l_i^3) \end{aligned} \quad (9.16)$$

Note that f_{i1} , f_{i2} and f_{i3} are all affine functions of the optimisation variable \mathbf{x} given in (9.8). Given a set of corresponding image points $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i \leftrightarrow \mathbf{x}''_i$ for $i = 1, \dots, m$, then the m error residuals generate the error vector:

$$\varepsilon = (\varepsilon_1, \dots, \varepsilon_m) \quad (9.17)$$

The estimated trifocal tensor is then the tensor \mathcal{T} that minimises the norm of this vector. In our solution, the L_∞ norm is adopted; therefore, the cost function will be $\|\varepsilon\|_\infty = \max_i |\varepsilon_i| = \max_i |d(\mathbf{x}''_i, \mathbf{x}_i l_i^j \mathcal{T}_i)|$ or, equivalently:

$$f_0(\mathbf{x}) = \max_{i=1, \dots, m} f_i(\mathbf{x}) = \max_{i=1, \dots, m} d(\mathbf{x}''_i, \mathbf{x}_i l_i^j \mathcal{T}_i) = \max_{i=1, \dots, m} \left(\frac{f_{i1}(\mathbf{x})^2 + f_{i2}(\mathbf{x})^2}{f_{i3}(\mathbf{x})^2} \right) \quad (9.18)$$

Thus, the optimisation is given by:

$$\begin{aligned} \min_{\mathbf{x}} \quad & \max_{i=1, \dots, m} \left(\frac{f_{i1}(\mathbf{x})^2 + f_{i2}(\mathbf{x})^2}{f_{i3}(\mathbf{x})^2} \right) \\ \text{subject to} \quad & f_{i3}(\mathbf{x}) > 0, \quad \text{for } i = 1, \dots, m. \end{aligned} \quad (9.19)$$

Since f_{i1} , f_{i2} and f_{i3} are all affine functions of \mathbf{x} , according to Theorem 8 (Chapter 3, page 70), $f_i(\mathbf{x}) = \frac{f_{i1}(\mathbf{x})^2 + f_{i2}(\mathbf{x})^2}{f_{i3}(\mathbf{x})^2}$ is quasi-convex function. In addition, according to Theorem 7 (Chapter 3, page 61), the function $f_0 = \max_{i=1, \dots, m} \left(\frac{f_{i1}(\mathbf{x})^2 + f_{i2}(\mathbf{x})^2}{f_{i3}(\mathbf{x})^2} \right)$ is also quasi-convex since the pointwise maximum is also convex. The constraint function has a convex domain $\{\mathbf{x} | f_{i3}(\mathbf{x}) > 0, \text{ for } i = 1, \dots, m\}$, therefore problem (9.19) is a quasi-convex optimisation problem [80, 94, 101].

Therefore, this problem can be solved using a sequence of SOCP feasibility problems as described in Section 3.7 (page 67), where the aim is to recover the optimal tensor \mathcal{T}_{opt} that minimises the maximum error.

The next step is to adapt this estimated tensor to satisfy all required constraints [82, 173]. As a first stage, the epipoles e' and e'' in the second and the third views respectively are extracted from the estimated trifocal tensor \mathcal{T} [82]. Then, the remaining parameters of the camera matrices $P'[A|a_4]$ and $P''[B|b_4]$ are recovered. The estimated camera matrices are then used to adapt the geometrically valid tensor.

From (9.4), one can see that once the epipoles are $e'^v = a_4^v$ and $e''^w = b_4^w$, the trifocal tensor can be recovered linearly by rectifying the remaining entries a_4^v and b_4^w [82, 174, 206]. This relationship can be parametrised linearly as $t = Ea$, where t is a vector containing the tensor parameters, E is a matrix representing the linear relationship in (9.4) and a is the vector of the remaining entries a_4^v and b_4^w . This is performed by minimising $\|AEa\|$ where $\|AE\| = 1$. The matrix A is constructed

over the transferred image point using the estimated trifocal tensor. The recovered tensor then satisfies all constraints as shown in [82, 173].

9.4.2 Camera motion estimation from the trifocal tensor

After estimating a valid trifocal tensor, the final step is to extract the camera matrices P' and P'' , from which the relative translation and rotation between the first-and-second views and the first-and-third views are estimated. The first projection matrix can be assumed canonical. Hence, the set of projection matrices for three views can be written as [82]:

$$\begin{aligned} P &= [I|0] \\ P' &= [[\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3]e''|e'] \\ P'' &= [(e''e''^\top - I)[\mathcal{T}_1^\top, \mathcal{T}_2^\top, \mathcal{T}_3^\top]e'|e''] \end{aligned} \tag{9.20}$$

It noteworthy that in the proposed solution, while the UAV is navigating, the triad views are configured such that the first view is located between the second and third views. In two-view geometry, before the camera acquires a new frame, the current-frame camera matrix is set to $P = [I|0]$, and motion between the current and the newly acquired frame is extracted from new camera matrix P' . In our three-view geometry set-up, the previous frame is included in the motion estimation process. Before acquiring a new frame, the current-frame camera matrix is set to $P = [I|0]$, and the trifocal tensor is estimated such that the first frame is the current frame, the second frame is the previous frame and the third frame is the newly acquired frame.

One may notice that in this solution, the relative motion in each consecutive frame pair is estimated twice. An averaging procedure between the two estimates is then performed in order to get a final optimised motion. This certainly, as it will be shown in the experimental validation section, would provide increased accuracy and stability to the motion estimates.

After having robustly estimated the motion of the camera using three-view geometry, ambiguities in the translation scale still occur. Unlike in the stereo scheme, the monocular visual motion estimates both the relative motion and the 3D structure up to an unknown scale. This absolute scale cannot be estimated unless information about the real world is provided. Our solution in this chapter employs the technique presented in Section 6.9 (page 148), where the robust least squares (RLS) solution is used [52] as well. The RLS algorithm is deployed using again a convex second-order cone programming (SOCP).

One might notice that the 3D scene points X_i are involved in estimating the scale in our monocular motion estimation. In fact, our experimental studies showed that these scene points have a significant influence over the final motion estimates. The greater the number of accurately estimated 3D points, the better the estimate of the motion will be. This has motivated us to exploit two monocular systems and build a stereo rig, where more ambiguity-free 3D scene points can be estimated through stereo vision, as shown in the following sections.

9.5 Registration problem

In a loop-closure scenario, the proposed solution employs a flexible stereo rig formed by two UAVs involved in this overlap. The ultimate aim in this section is to estimate the relative pose between the two vehicles, in order to use this stereo set-up in the cooperative motion estimation.

Prior to that, each vehicle had to construct its own 3D scene points using its on-board monocular vision system. These construction are based on pairs of frames acquired over time. These 3D scene points are referred to us as a point cloud. In the loop-closure scenario, the two vehicles are able to construct approximately the same scenery. Given the 3D scene structures (point clouds) we further align them to each other and then estimate the geometry between the 3D scene structures of each camera.

If one performs 3D scene reconstruction, as seen from each camera, and then estimates the relative transformation between them, we speak of registration problem. The registration problem is akin to finding the transformation between two coordinate systems [6, 82, 193]. Many solutions have been proposed to this problem. However, the well-known was presented by Horn et al. [87]. A similar solution with some modifications was proposed by Umeyama in [202]. Both solutions propose a closed-form approach that recovers the Euclidean transformation in a least-squares minimisation framework. However, as concluded in [95], noise in both point sets can significantly affect the final estimate in this approach. To overcome this problem, another well known algorithm called the iterative closest point (ICP) algorithm is presented in [19]. One major drawback of this algorithm is its dependency on a good initial transformation in order to converge to the global minimum. Diverse solutions have been proposed since then, however, they are still prone to being affected by local minima.

9.5.1 Registration using least squares estimation

In this section, a brief illustration of the registration method with least-squares minimisation is given [87, 202]. Given two corresponding point sets $X^i = (x_1^i, x_2^i, \dots, x_n^i)$ and $X^j = (x_1^j, x_2^j, \dots, x_n^j)$, the aim is to find the best transformation that maps one set onto the other. The optimisation function with respect to the rotation matrix R and the translation vector t is defined as:

$$e^2(R, t) = \frac{1}{n} \sum_{k=1}^n \|x_k^j - (Rx_k^i + t)\|^2 \quad (9.21)$$

Here, e^2 is the mean squared error. If UDV^\top is the singular value decomposition of the covariance between the two point sets, then the optimal rotation matrix R and the translation vector t are recovered as [202]:

$$R = USV^\top \quad (9.22)$$

$$t = \mu_{x^j} - R\mu_{x^i} \quad (9.23)$$

where μ_{x^i} and μ_{x^j} are the centroids of point sets X^i and X^j respectively. Note that S must be chosen as:

$$S = \begin{cases} I & \text{if } \det(U)\det(V) = 1 \\ \text{diag}(1, 1, \dots, 1, -1) & \text{if } \det(U)\det(V) = -1 \end{cases} \quad (9.24)$$

9.5.2 Overview of the proposed solution for the registration

In this section, an alternative approach to the registration problem using global optimisation is presented. Note that this problem can be formulated as finding the transformation between two or more coordinate systems using points expressed in each system [87].

Let X be 3D scene points estimated from two camera viewpoints C_i and C_j . Suppose X^i and X^j are their respective coordinates seen from camera C_i and C_j . The main aim is to recover the Euclidean transformation $T = [R \ t]$, where $R = [r^1 \ r^2 \ r^3]^\top$ and $t = [t_x \ t_y \ t_z]^\top$ are the rotation matrix and the translation vector respectively. We wish to recover a transformation such that the mapping of the points X^i stands as close as possible to its corresponding points in X^j . Hence, the registration problem is then defined as follows: given a set of 3D scene point correspondences $X_k^j \leftrightarrow X_k^i$ for $k = 1, \dots, m$, where m is the number of the correspondences, find T such that for all k :

$$X_k^j = TX_k^i, \quad k = 1, \dots, m \quad (9.25)$$

The quantity we wish to find is the transformation T which is parametrised as:

$$T = [R|t] = \begin{bmatrix} r_1 & r_2 & r_3 & | & t_x \\ r_4 & r_4 & r_6 & | & t_y \\ r_7 & r_8 & r_9 & | & t_z \end{bmatrix} = \begin{bmatrix} r^1 & | & t_x \\ r^2 & | & t_y \\ r^3 & | & t_z \end{bmatrix} \quad (9.26)$$

Thus:

$$X^j = \begin{bmatrix} r_1 & r_2 & r_3 & | & t_x \\ r_4 & r_4 & r_6 & | & t_y \\ r_7 & r_8 & r_9 & | & t_z \\ 0 & 0 & 0 & | & 1 \end{bmatrix} X^i \quad (9.27)$$

The optimisation variable is then:

$$\mathbf{x} = (r_1, r_2, r_3, r_4, r_5, r_6, r_7, r_8, r_9, t_x, t_y, t_z)^\top \in \mathbb{R}^{12} \quad (9.28)$$

Due to the imperfect estimation of the 3D scene point positions caused by measurement noise, after transformation, each pair of the corresponding scene points $X_k^i \leftrightarrow X_k^j$ defines an error residual. We define the 3D Euclidean distance from a target point to a transformed one as: $d(X_k^j, TX_k^i)$, the associated error residual as:

$$\varepsilon_k = d(X_k^j, TX_k^i), k = 1, \dots, m \quad (9.29)$$

Given $X_k^i = [x_k^i \ y_k^i \ z_k^i \ 1]^\top$ and their correspondence $X_k^j = [x_k^j \ y_k^j \ z_k^j \ 1]^\top$, then the residual error for the k^{th} correspondence is parametrised as:

$$\begin{aligned} F_k(\mathbf{x}) &= \varepsilon_i = d(X_k^j, TX_k^i), k = 1, \dots, m \\ &= \sqrt{[(r^1 X_k^i + t_x) - x_k^j]^2 + [(r^2 X_k^i + t_y) - y_k^j]^2 + [(r^3 X_k^i + t_z) - z_k^j]^2} \\ &= \sqrt{f^1(\mathbf{x})_k^2 + f^2(\mathbf{x})_k^2 + f^3(\mathbf{x})_k^2} \end{aligned} \quad (9.30)$$

Given m correspondences, the error vector is then defined as: $\varepsilon = [\varepsilon_1, \dots, \varepsilon_m]^\top$. Note that $f^1(\mathbf{x})_k$, $f^2(\mathbf{x})_k$ and $f^3(\mathbf{x})_k$ are affine functions over the optimisation variable \mathbf{x} given in (9.28). The L_2 norm of this error function is given by:

$$\|\varepsilon\|_2 = \sqrt{\sum_{k=1}^m \varepsilon_k^2} = \sqrt{\sum_{k=1}^m d(X_k^j, TX_k^i)^2}$$

Then:

$$F_k(\mathbf{x}) = \|(r^1 X_k^i + t_x) - x_k^j, (r^2 X_k^i + t_y) - y_k^j, (r^3 X_k^i + t_z) - z_k^j\|_2 \quad (9.31)$$

This optimisation problem involves estimating the rotation matrix R . It is known that any rotation matrix must satisfy the orthogonality constraint ($R^\top R = I$). This means that $R \in SO(3)$ (Chapter 2, Section 2.2, page 24). Using the L_∞ norm instead, the optimisation is then given by:

$$\begin{aligned} \min_{\mathbf{x}} \quad & \max_{k=1, \dots, m} \left(f^1(\mathbf{x})_k^2 + f^2(\mathbf{x})_k^2 + f^3(\mathbf{x})_k^2 \right) \\ \text{subject to} \quad & R \in SO(3). \end{aligned} \quad (9.32)$$

In this case (9.32) becomes non-convex due to the orthogonality constraints $R^\top R = I$. Therefore, standard convex optimisation tools cannot be used here. Instead, global optimisation through the branch and bound algorithm is adopted [107]. Referring to (9.26), one may rewrite the constraint in (9.32) as:

$$\begin{aligned} r_1^2 + r_2^2 + r_3^2 = 1 & \quad , \quad r_1 r_4 + r_2 r_5 + r_3 r_6 = 0 \\ r_4^2 + r_5^2 + r_6^2 = 1 & \quad , \quad r_1 r_7 + r_2 r_8 + r_3 r_9 = 0 \\ r_7^2 + r_8^2 + r_9^2 = 1 & \quad , \quad r_4 r_7 + r_5 r_8 + r_6 r_9 = 0 \end{aligned} \quad (9.33)$$

Note that these newly created constraints include bilinear and quadratic terms. It is essential to define the bounds of these entries before using the branch and bound algorithm. As shown in Section 9.2, the next step is to perform a linear relaxation for the problem. To do that, all bilinear and quadratic terms are replaced with new linear variables.

Prior to that, given the lower and the upper bounds r_i^L , r_i^U , r_j^L and r_j^U of r_i and r_j , where $i = 1, \dots, 9$, $j = 1, \dots, 9$ according to (9.33), then the following inequalities holds [116, 127, 160]:

$$\begin{aligned} (r_i^U - r_i)(r_j^U - r_j) \geq 0 & \quad , \quad (r_i - r_i^L)(r_j - r_j^L) \geq 0, \\ (r_i^U - r_i)(r_j - r_j^L) \geq 0 & \quad , \quad (r_i - r_i^L)(r_j^U - r_j) \geq 0. \end{aligned} \quad (9.34)$$

The linear relaxation of the terms in (9.33) is obtained by replacing the terms $r_i r_j$ with the new variables w_{ij} , $i = 1, \dots, 9$, $j = 1, \dots, 9$, where $w_{ij} = r_i r_j$, ($w_{ii} = r_i^2$). Then, in order to make the relaxation as tight as possible, the inequalities in (9.34) are added to the optimisation problem to relate the new linear variables w_{ij} with the original non-linear terms [116, 127, 160]. The inequalities in (9.34) yield

[116, 127, 160]:

$$\begin{aligned}
 r_i r_j &\geq \max \begin{cases} r_i r_j^U + r_i^U r_j - r_i^U r_j^U \\ r_i r_j^L + r_i^L r_j - r_i^L r_j^L \end{cases} = \max \begin{cases} \varphi_1^u \\ \varphi_2^u \end{cases} = \varphi^u \\
 r_i r_j &\leq \min \begin{cases} r_i r_j^L + r_i^L r_j - r_i^L r_j^L \\ r_i r_j^U + r_i^U r_j - r_i^U r_j^U \end{cases} = \min \begin{cases} \varphi_1^l \\ \varphi_2^l \end{cases} = \varphi^l
 \end{aligned} \tag{9.35}$$

It is well-known that φ^u and φ^l are the convex and the concave envelopes of $r_i r_j$ [116, 127, 160]. Thus, using (9.34) and (9.35), we get:

$$\begin{aligned}
 w_{ij} - \varphi_1^u &\geq 0 & , & & -w_{ij} + \varphi_1^l &\geq 0 \\
 w_{ij} - \varphi_2^u &\geq 0 & , & & -w_{ij} + \varphi_2^l &\geq 0
 \end{aligned} \tag{9.36}$$

Since φ^u and φ^l are the convex and the concave envelopes of $r_i r_j$, then the convexity of the constraints in (9.36) is guaranteed. Thus, the new, relaxed problem can be summarised as:

$$\begin{aligned}
 \min_{\mathbf{x}} \quad & \max_{k=1, \dots, m} \left(f^1(\mathbf{x})_k^2 + f^2(\mathbf{x})_k^2 + f^3(\mathbf{x})_k^2 \right) \\
 \text{subject to} \quad & w_{11} + w_{22} + w_{33} = 1, w_{14} + w_{25} + w_{36} = 0, \\
 & w_{44} + w_{55} + w_{66} = 1, w_{17} + w_{28} + w_{39} = 0, \\
 & w_{77} + w_{88} + w_{99} = 1, w_{47} + w_{58} + w_{69} = 0, \\
 & w_{ij} - \varphi_1^u \geq 0, -w_{ij} + \varphi_1^l \geq 0, \\
 & w_{ij} - \varphi_2^u \geq 0, -w_{ij} + \varphi_2^l \geq 0
 \end{aligned} \tag{9.37}$$

where the new optimisation variable is:

$$\hat{\mathbf{x}} = (r_1, \dots, r_9, t_x, t_y, t_z, w_{11}, \dots, w_{99})^\top \tag{9.38}$$

One may see that the original non-convex problem (9.32) has been reformulated into a new convex optimisation problem. The solution of the problem (9.37) gives a lower bound on the global minimum of the original problem. Figure 3.12 (Chapter 3, page 76) presents a detailed example of the branch and bound algorithm used to solve this problem.

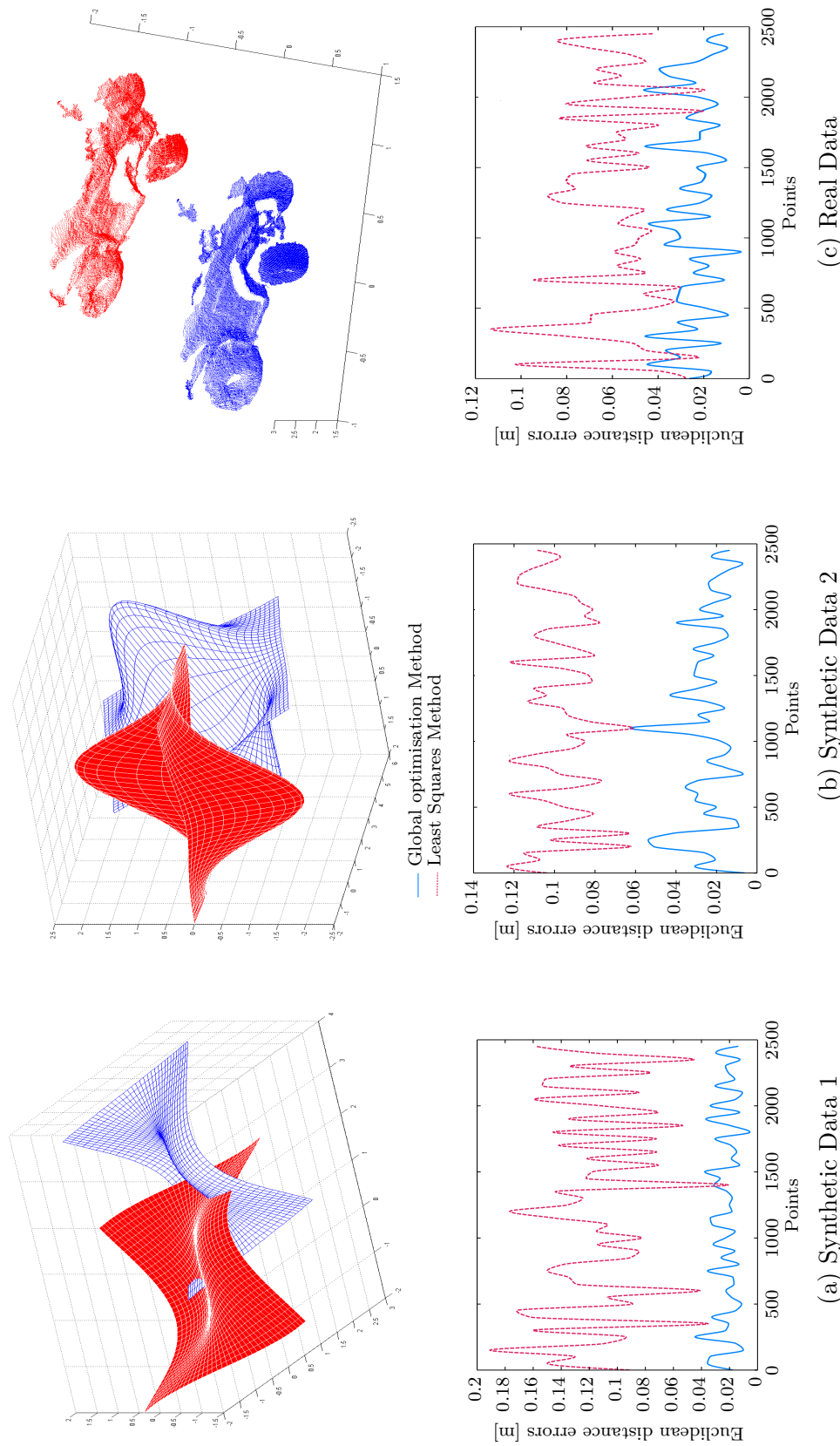


Fig. 9.3 Top: Samples of synthetic and real data used for the experimental validation. Red point sets represent the original sets and blue ones represent their corresponding sets. Bottom: The corresponding results in the experimental validation. The graphs all show the number of matched points on the horizontal axis, and residual error on the vertical axis. Red dashed curves plot errors using the classical least-squares method while blue solid curve illustrates the results when the global optimisation technique through the branch and bound algorithm is used. One may see that the global optimisation residual errors are substantially smaller than those from least-squares method.

9.5.3 Implementation

In order to evaluate and validate its capability before deploying it in the main cooperative motion estimation problem, implementation of the proposed algorithm is performed on both synthetic and real data. Comparison with state-of-the-art methods based on least squares estimation is provided as well. Our first aim was to investigate the robustness and efficiency of the proposed solution in the presence of noise.

The branch and bound algorithm detailed in Section 9.2 and the relaxation approach given in the previous section are implemented. Synthetic data consists of a series of point sets (point clouds) having known positions, and then transformed with known transformations (Figure 9.3). Subsequently, the transformed point sets are corrupted with varying levels of Gaussian noise: up to 0.01 metres of Gaussian noise was then added to the transformed point sets. Real data, on the other hand, were collected using an RGB-D sensor (MS Kinect) using the point cloud library (PCL) for 3D mapping a wheeled vehicle (Figure 9.3). Given the original point set and the target set, our aim is to then robustly recover the best transformation that maps one set onto the other in the presence of high levels of noise.

The implementation used to generate the results shown in this section uses a sequence of SOCP feasibility problems for the convexity task using the SeDuMi toolbox [187] and Yalmip [116] toolbox for modelling.

Figure 9.3 shows samples of synthetic and real data used for the experimental validation. The graphs in this figure show the outcomes of the proposed solution in comparison to the least-squares technique. One can see that global optimisation clearly outperforms methods based on least squares minimisation. This pattern is valid for both synthetic and real data. These figures are in accordance with the theory and confirm that the global optimisation solution is a superior solution for the registration problem, especially on noisy data.

9.6 Filtering based approach for relative pose estimation

The second technique used in our solution to estimate the relative pose between the two vehicles is based on a filtering approach. This technique recovers the relative pose between the vehicles in order to create a flexible stereo rig. Inspired by the work presented in [4], in addition to the monocular vision system, each UAV is equipped with an inertial measurement unit (IMU). The relative pose is estimated by fusing

metric measurements from the IMUs with visual estimates from the vision system. The latter is referred to as the visual algorithm.

When the loop-closure detection module detects an overlapping field of view for two or more UAVs, the visual algorithm estimates the relative pose between the two vehicles. This is similar to monocular systems, where a multi-view geometry is created by distributing frame sequences over time. Naturally, the estimated relative translation is then scaled by a factor of λ , which is less optimal. Inertial systems, on the other hand, provide metric measurements, but only for the acceleration. Hence, a fusion mechanism that combines noisy measurements from both the vision and the inertial systems is necessary. This fusion system is represented by the following non-linear model:

$$\begin{cases} \dot{x}(t) &= f(x(t), t) \\ y(t) &= h(x(t), t) \end{cases} \quad (9.39)$$

where x is the state vector which contains the quantities we want to estimate. These quantities are essentially the relative rotation R_r , the relative translation t_r , and the translation scale λ .

To estimate these quantities, an extended Kalman filter (EKF) is used in [4], obtaining encouraging results. In our implementation, however, robustness is imperative in the subsequent cooperative motion estimation tasks. The estimated relative pose between the two UAVs is used to construct a stereo rig, which is a key device in achieving global cooperative motion. Therefore, more care is taken in estimating this pose, especially in the presence of high levels of noise.

At a specific time step, the key measurement of the visual algorithm (the vision system) consists of image points (features) extracted and matched between images taken by the first and the second vehicle. As we have seen in the previous chapter, due to the way they are extracted, image point location accuracy is heavily dependent on the variation in intensity within their neighbourhoods, causing high levels of noise with unknown mean and variance. Measurements from inertial systems consist of the triaxial acceleration and the angular rates; again, these measurements are laden with noise of undefined types. One may formulate the system in (9.39) as a non-linear discrete-time state transition equation:

$$\begin{cases} x_k &= f(x_{k-1}, w_{k-1}) \\ y_k &= h(x_k, v_k) \end{cases} \quad (9.40)$$

where x_k is the state vector at time step k , containing the estimated relative pose, w_k is some additive process noise, y_k is the observation, and v_k is some additive observation noise.

The objective of the filtering technique is to estimate x_k using available observations y_k . The Kalman filter makes certain assumptions about the noise statistics that cannot always be valid. The process and observation noise (w_k and v_k respectively) is assumed to be uncorrelated, zero-mean Gaussian with perfectly known covariances (Q_k and R_k respectively). In addition, system model matrices should be specified with high accuracy. Unfortunately, these assumptions limit the EKF's application especially in aerial systems, where models and/or noise descriptions are typically not fully known.

The state and observation model in (9.40) may be rewritten as:

$$\begin{aligned} x_{k+1} &= F_k x_k + G_k w_k + \Omega_k + \Delta_1(\tilde{x}_{k|k}) + \Delta_2(\tilde{x}_{k|k})w_k \\ y_k &= H_k x_k + v_k + \Psi_k + \Delta_3(\tilde{x}_{k|k}) \end{aligned} \quad (9.41)$$

where F_k , G_k and H_k are matrices estimated using the Jacobian of f and h w.r.t. x_k and w_k . More importantly, terms Δ_i model higher orders of the Taylor series expansion, which are norm-bounded, as in $\|\Delta_i\| \leq \delta_i$. The error state vector \tilde{x} is defined as the difference of an estimate \hat{x} from its real value x , i.e. $\tilde{x}_k = x_{k|k} - \hat{x}_{k|k}$.

The extended Kalman filter (EKF) makes use of this model (9.41); though neglecting higher order terms of the Taylor series Δ_i , presuming they are seldom necessary. However, a number of studies have concluded that in certain real-time applications, the EKF is affected by linearisation problems [14, 51]. Usually, non-linear models are weakly approximated by the Taylor series expansion. Higher order terms of the Taylor expansion then become necessary. Some alternative approaches have been proposed to cope with this problem, such as the particle filter (PF) or unscented Kalman filter (UKF). However, computational requirements impair their suitability for real-time aerial navigation solutions.

Therefore, it is crucial to design a filter that can handle the modelling errors and noise uncertainty, while also minimising the worst-case estimation error rather than the covariance of the estimation error. State estimators that can tolerate such uncertainty are called robust. Applying more robust estimator such as the non-linear H_∞ filter would appear to be logical next step. This filter is based on the min-max estimation problem, drawing an important difference from the Kalman filter: the former is optimal in terms of minimising the L_∞ -norm between ranges of disturbances.

9.6.1 Non-linear Filter design

The discrete-time linear H_∞ filter is detailed in Section 4.7.2 (Chapter 5, page 90), where the model is given by:

$$\begin{aligned} x_{k+1} &= F_k x_k + w_k \\ y_k &= H_k x_k + v_k \end{aligned} \quad (9.42)$$

where w_k and v_k are noise terms with an unknown distribution law or deterministic.

Before detailing the non-linear H_∞ filter, let us give a brief insight into linear filtering problems. Two main differences between the Kalman and the H_∞ filters can be distinguished. First, the H_∞ filter is optimal in terms of minimising the L_∞ norm of the gain between a set of disturbance inputs and the estimation error. On the other hand, the Kalman filter minimises the mean square gain between the disturbances and the estimation error. The second core difference is that the minimum mean square estimate of the Kalman filter commutes with linear operations. However, the minimal L_∞ -norm estimate does not possess this property and the H_∞ optimal estimator depends on the plant output being estimated.

The non-linear H_∞ filter has been a subject of research in many studies [3, 14, 51]. It tries to robustly estimate the state while satisfying the H_∞ performance criterion. The system given in (9.41) can be rewritten in the following form:

$$\begin{aligned} x_{k+1} &= F_k x_k + G_k w_k + \Omega_k + \Pi_k \\ y_k &= H_k x_k + v_k + \Psi_k + \Sigma_k \end{aligned} \quad (9.43)$$

where

$$\begin{aligned} \Pi_k &= \Delta_1(\tilde{x}_{k|k}) + \Delta_2(\tilde{x}_{k|k})w_k \\ \Sigma_k &= \Delta_3(\tilde{x}_{k|k}) \end{aligned} \quad (9.44)$$

These inputs must satisfy the following norm bounds:

$$\begin{aligned} \|\Pi_k\|_2^2 &\leq \delta_1^2 \|\tilde{x}_{k|k}\|_2^2 + \delta_2^2 \|w_k\|_2^2 \\ \|\Sigma_k\|_2^2 &\leq \delta_3^2 \|\tilde{x}_{k|k}\|_2^2 \end{aligned} \quad (9.45)$$

An alternative way to approach the non-linear model in (9.43) is with the additional terms Π_k and Σ_k , given as [51]:

$$\begin{aligned} x_{k+1} &= F_k x_k + G_k c_w + \Omega_k \\ y_k &= H_k x_k + c_v v_k + \Psi_k \end{aligned} \quad (9.46)$$

where $c_w^2 = 1 - \gamma^2 \delta_3^2$ and $c_v^2 = c_w^2 (1 + \delta_1^2)^{-1}$. One may notice that the terms Π_k and Ψ_k are omitted and replaced with the terms c_w and c_v . This formulation gives the non-linear H_∞ a similar structure to the EKF.

9.6.2 State representation

After presenting the general framework of the deployed non-linear H_∞ , in this section details about process, measurement and error models are given. The non-linear H_∞ filter is mainly used to estimate the relative rotation R_r and the relative translation t_r of the two UAVs involved in a loop-closure. The rotation is represented by either the rotation matrix R_r or quaternions \bar{q}_r . The Hamilton notation is used to represent a quaternion q , where $\bar{q}_r = [q \quad \mathbf{q}^\top]^\top$, q and \mathbf{q}^\top are the real and the imaginary parts respectively [105]. A quaternion representation of a vector $\mathbf{p} \in \mathbb{R}^3$ is $\bar{p}_r = [0 \quad \mathbf{p}^\top]^\top$. The key idea in this approach is a fusion between [4]:

- The drift-free pose estimation from the vision system, but with a scale ambiguity (the visual algorithm),
- And, the metric measurements of the accelerometers and gyros in both vehicles (inertial system).

These two measurements create a residual during the measurement update. This residual will induce a state correction, which updates the scale as well. The filter state includes the vision estimates, which contains the relative rotation R_r , the relative translation t_r and the scale λ . In order to relate the inertial estimates to those from the vision system (R_r , t_r and λ), angular velocities $\boldsymbol{\omega}_1$, $\boldsymbol{\omega}_2$ as well as the linear acceleration \mathbf{a}_1 , \mathbf{a}_2 of the first and second vehicles are included as in the state. In addition, and in order to relate the IMU's reading to the relative pose t_r , the relative velocity v_r is included in the state vector as well. This yields the following 23-component state:

$$x = [\bar{q}_r^\top, \boldsymbol{\omega}_1^\top, \boldsymbol{\omega}_2^\top, t_r^\top, v_r^\top, \mathbf{a}_1^\top, \mathbf{a}_2^\top, \lambda]^\top \quad (9.47)$$

The linear accelerations \mathbf{a}_1 , \mathbf{a}_2 and the angular velocities $\boldsymbol{\omega}_1$, $\boldsymbol{\omega}_2$ are modelled as a random walk. Time step k is omitted for simplicity. The scale factor λ has no

time-dependent dynamics. Hence, the state equations are given as:

$$\begin{aligned}
\dot{\bar{q}}_r &= 0.5(\bar{q}_r \otimes \bar{\omega}_2 - \bar{\omega}_1 \otimes \bar{q}_r), \\
\dot{\boldsymbol{\omega}}_1 &= n_{\omega_1}, \quad \dot{\boldsymbol{\omega}}_2 = n_{\omega_2}, \\
\dot{t}_r &= v_r - [\boldsymbol{\omega}_1]_\times t_r, \\
\dot{v}_r &= R_r \mathbf{a}_2 - \mathbf{a}_1 - [\boldsymbol{\omega}_1]_\times v_r, \\
\dot{\mathbf{a}}_1 &= n_{a_1}, \quad \dot{\mathbf{a}}_2 = n_{a_2}, \\
\dot{\lambda} &= n_\lambda,
\end{aligned} \tag{9.48}$$

where the operator \otimes stands for quaternion multiplication, $[\boldsymbol{\omega}_i]_\times$ denotes a skew-symmetric matrix of $\boldsymbol{\omega}_i$, and n represents the noise.

9.6.3 Error State Representation

In the state representation, the quaternions are used as attitude description. It is common that in such a case the error and its covariance are not represented in terms of an arithmetic difference but with the aid of an error quaternion. This will increase the numerical stability and handles the quaternion in its minimal representation. In the following, $\hat{\cdot}$ represents the estimated state. A preceding Δ or δ represents respectively the error state for additive error or multiplicative error. The 22-element error state \tilde{x} is defined as:

$$\tilde{x} = [\delta\boldsymbol{\theta}_r^\top, \Delta\boldsymbol{\omega}_1^\top, \Delta\boldsymbol{\omega}_2^\top, \Delta t_r^\top, \Delta v_r^\top, \Delta\mathbf{a}_1^\top, \Delta\mathbf{a}_2^\top, \Delta\lambda]^\top \tag{9.49}$$

where the difference of an estimate \hat{x} from its real value is x , i.e. $\tilde{x} = x - \hat{x}$.

Since the relative rotation associated with the error quaternion $\delta\bar{q}_r$ can be assumed to be very small, the small angular approximation is used [198]:

$$\delta\bar{q}_r \approx \begin{bmatrix} 1 \\ \frac{1}{2}\delta\boldsymbol{\theta}_r \end{bmatrix} \tag{9.50}$$

where $\delta\boldsymbol{\theta}_r \in \mathbb{R}^3$ is the attitude angular error vector. For the remaining state vector entries, an additive error is used and represented with Δ .

Note that the multiplicative error is used to represent the error state for the relative rotation expressed in quaternions. Thus, a small quaternion rotation is taken

as an error representation instead of quaternion differences:

$$\begin{aligned}\bar{q}_r &= \hat{q}_r \otimes \delta\boldsymbol{\theta}_r \\ \delta\dot{\boldsymbol{\theta}}_r &= \hat{q}_r^* \otimes (\dot{\bar{q}}_r - \dot{\hat{q}}_r \otimes \delta\boldsymbol{\theta}_r)\end{aligned}\quad (9.51)$$

Hence, the error state for the relative rotation is given by:

$$\delta\dot{\boldsymbol{\theta}}_r = -[\boldsymbol{\omega}_2]_{\times} \delta\boldsymbol{\theta}_r - R_r^{\top} \Delta\boldsymbol{\omega}_1 + \Delta\boldsymbol{\omega}_2 \quad (9.52)$$

An additive error model is used for the relative translation $\dot{t}_r = \dot{\hat{t}}_r + \Delta\dot{t}_r$ and for the relative velocity $\dot{v}_r = \dot{\hat{v}}_r + \Delta\dot{v}_r$. Then, the error states for these quantities are given as [4]:

$$\begin{aligned}\Delta\dot{v}_r &= [\hat{t}_r]_{\times} \Delta\boldsymbol{\omega}_1 - [\boldsymbol{\omega}_1]_{\times} \Delta v_r \\ \Delta\dot{v}_r &= -\hat{R}_r [\hat{\mathbf{a}}_2]_{\times} \delta\boldsymbol{\theta}_r + [\hat{v}_r]_{\times} \Delta\boldsymbol{\omega}_1 - [\boldsymbol{\omega}_1]_{\times} \Delta v_r - \Delta\mathbf{a}_1 + R_r \Delta\mathbf{a}_2\end{aligned}\quad (9.53)$$

The remaining error states remain unchanged, hence:

$$\Delta\dot{\boldsymbol{\omega}}_i = n_{\boldsymbol{\omega}_i}, \quad \Delta\dot{\mathbf{a}}_i = n_{\mathbf{a}_i}, \quad \Delta\dot{\lambda} = 0, \quad i = 1, 2. \quad (9.54)$$

This can be summarised in the linearised continuous-time error state equation:

$$\dot{\tilde{x}} = F_c \tilde{x} + G_c \mathbf{n} \quad (9.55)$$

with \mathbf{n} being the noise vector $\mathbf{n} = [n_{\boldsymbol{\omega}_1}^{\top} \ n_{\boldsymbol{\omega}_2}^{\top} \ n_{\mathbf{a}_1}^{\top} \ n_{\mathbf{a}_2}^{\top}]^{\top}$. Then, the new state covariance matrix is given as:

$$P_{k+1|k} = F P_{k|k} F^{\top} + Q \quad (9.56)$$

9.6.4 State Prediction

Using the equations in (9.48), the state prediction is performed as follow:

$$\begin{aligned}\hat{q}_{r_{k+1}} &= \hat{q}_{r_k} + 0.5 \cdot \Delta t \cdot (\hat{q}_{r_k} \otimes \hat{\boldsymbol{\omega}}_{2k} - \hat{\boldsymbol{\omega}}_{1k} \otimes \hat{q}_{r_k}), \\ \hat{t}_{r_{k+1}} &= \hat{t}_{r_k} + (\hat{v}_{r_k} - [\hat{\boldsymbol{\omega}}_{1k}]_{\times} \cdot \hat{t}_{r_k}) \cdot \Delta t, \\ \hat{v}_{r_{k+1}} &= \hat{v}_{r_k} + (\hat{R}_{r_k} \cdot \hat{\mathbf{a}}_{2k} - \hat{\mathbf{a}}_{1k} - [\hat{\boldsymbol{\omega}}_{1k}]_{\times} \hat{v}_{r_k}) \cdot \Delta t\end{aligned}\quad (9.57)$$

The remaining states, including the angular velocities $\boldsymbol{\omega}_1$ and $\boldsymbol{\omega}_2$, the linear accelerations \mathbf{a}_1 and \mathbf{a}_2 , and the scale λ , remain unchanged during state prediction.

9.6.5 Measurements

The visual algorithm:

As stated earlier, using the on-board monocular vision system of each UAV one can get the relative rotation R_r (or \bar{q}_r) between the two vehicles and the scaled relative translation t_r with a factor λ . The multiplicative error model is used again for the relative rotation \bar{q}_r :

$$\delta\bar{q}_r = \hat{q}_r^* \otimes \bar{q}_r = \begin{bmatrix} q \\ \delta\mathbf{q} \end{bmatrix} \approx \begin{bmatrix} 1 \\ \frac{1}{2}\delta\boldsymbol{\theta}_r \end{bmatrix} \Rightarrow \tilde{z}_{\bar{q}_r} = \delta\boldsymbol{\theta}_r = H_{\bar{q}_r}\tilde{x} \quad (9.58)$$

where $H_{\bar{q}_r}$ represents the relative rotation measurement matrix, which relates the error state vector the measurement residuals.

An additive error model is used for the relative translation; therefore, one can pose the estimated measurement as:

$$\begin{aligned} \tilde{z}_{t_r} &= \lambda t_r - \hat{\lambda}\hat{t}_r \\ \tilde{z}_{t_r} &= \hat{t}_r\Delta\lambda + \hat{\lambda}\Delta t_r \end{aligned} \quad (9.59)$$

which can be linearised to

$$\tilde{z}_{t_r} = H_{t_r}\tilde{x} \quad (9.60)$$

where H_{t_r} is the relative translation measurement matrix, computed through the Jacobian $\frac{\partial\tilde{z}_v}{\partial\tilde{x}}$ of the vision measurements $\tilde{z}_v = [\tilde{z}_{\bar{q}_r}^\top \tilde{z}_{t_r}^\top]^\top$ with respect to \tilde{x} . Then, one may write:

$$\tilde{z}_v = \begin{bmatrix} H_{\bar{q}_r} \\ H_{t_r} \end{bmatrix} \tilde{x} = H_v\tilde{x} \quad (9.61)$$

where $\begin{bmatrix} H_{\bar{q}_r} \\ H_{t_r} \end{bmatrix} = H_v$ represents the measurement matrix for the vision algorithm.

Inertial system:

Measurements from the on-board inertial system in each vehicle consist of the angular rates $\boldsymbol{\omega}_1$, $\boldsymbol{\omega}_2$ as well as the linear accelerations \mathbf{a}_1 , \mathbf{a}_2 of the first and second vehicles

respectively. Therefore:

$$\tilde{z}_{IMU} = \begin{bmatrix} \boldsymbol{\omega}_1 - \hat{\boldsymbol{\omega}}_1 \\ \mathbf{a}_1 - \hat{\mathbf{a}}_1 \\ \boldsymbol{\omega}_2 - \hat{\boldsymbol{\omega}}_2 \\ \mathbf{a}_2 - \hat{\mathbf{a}}_2 \end{bmatrix} \tilde{x} = H_{IMU} \tilde{x} \quad (9.62)$$

We obtain the IMU measurement matrices H_{IMU} , by computing the Jacobian $\frac{\partial \tilde{z}_{IMU}}{\partial \tilde{x}}$ with respect to \tilde{x} .

Then:

$$\tilde{z} = \begin{bmatrix} \tilde{z}_v \\ \tilde{z}_{IMU} \end{bmatrix} = \begin{bmatrix} H_v \\ H_{IMU} \end{bmatrix} \tilde{x} = H \tilde{x} \quad (9.63)$$

9.6.6 Filter Update

After estimating the 18×22 measurement matrix H and given the measurement covariance matrix $R_{18 \times 18}$, one can update the estimates by:

- Computing the residual: $r_{18 \times 1} = z - \hat{z}$,
- Computing the innovation: $S = HPH^\top + R$,
- Computing the gain: $K = PH^\top S^{-1}$,
- Computing the correction: $\hat{\hat{x}} = Kr$.

From the correction $\hat{\hat{x}}$, one can compute the updated state variables in the state vector x . Since a multiplicative error model is used for the relative rotation expressed in quaternion \bar{q}_r , the correction is computed as a small rotation \bar{q}_{r_+} . From (9.58), one can see that $\delta \mathbf{q}_{r_+} = 0.5\delta \boldsymbol{\theta}_r$, hence:

$$\bar{q}_{r_+} = \left[\sqrt{1 - \mathbf{q}_{r_+}^\top \mathbf{q}_{r_+}} \quad \mathbf{q}_{r_+}^\top \right]^\top \quad (9.64)$$

This corrective quaternion must be of unit length. If it is not the case, re-normalisation should be performed. Then,

$$\bar{q}_{k+1|k+1} = \bar{q}_{k+1|k} \otimes \bar{q}_{r_+} \quad (9.65)$$

For the remaining states, an additive correction is simply performed:

$$x_{k+1|k+1} = x_{k+1|k} + \hat{\hat{x}} \quad (9.66)$$

Finally, the error state covariance is updated as [51]:

$$\begin{aligned} P_{k+1|k+1} &= P_{k+1|k} - P_{k+1|k} \begin{bmatrix} -L^\top & H^\top \end{bmatrix} \Lambda^{-1} \begin{bmatrix} -L \\ H \end{bmatrix} P_{k+1|k} \\ \Lambda &= \begin{bmatrix} LP_{k+1|k}L^\top - \gamma^2 I & -L^\top P_{k+1|k}H^\top \\ -HP_{k+1|k}L^\top & HP_{k+1|k}H^\top + R \end{bmatrix} \end{aligned} \quad (9.67)$$

where $\gamma^2 = \frac{1}{\theta}$. Note that θ is the one given in Equation (4.35) (page 93).

9.7 Experimental validation

This section discusses the experimental evaluation of the proposed solution. Implementation of the proposed techniques is conducted using real-world data. Experiments were performed in a laboratory setting as shown in Figure 9.4, using two AscTec MAV platforms (Firefly and Pelican) with fully-calibrated forward-looking cameras capturing images at 20 Hz (Section 1.6, Chapter 1, page 8). In addition, each platform is equipped with an inertial measurement unit (IMU) providing data at 100 Hz. A wireless network is established to synchronise the time between the on-board computers in each platform and the ground truth system using the network time protocol (NTP). The ground truth is collected from an OptiTrack motion-capture system that provides absolute position information with millimetre accuracy at 100 Hz.

Implementations used to generate the results shown in this section use a sequence of SOCP feasibility problems for the convexity task with SeDuMi [187] and Yalmip [116] toolboxes for the modelling and relaxation tasks. The estimated motions are aligned with the ground truth and the Euclidean distance errors of the UAV position are computed.

Real experiments are performed within an airspace of about $15 \times 8 \times 10 \text{ m}^3$. During each flight, each vehicle estimates its own motion using its on-board monocular vision system. A loop-closure detection module is launched in the background whose objective is to detect any significant overlap between the fields of view of each vehicle [24]. This is done from a central ground station in communication with each vehicle. As soon as a loop-closure is detected, a stereo rig between the two vehicles is constructed. More importantly, the corresponding image points that are employed to detect this loop-closure are also used to recover relative rotation and translation of the two vehicles. This saves a considerable amount of processing time.

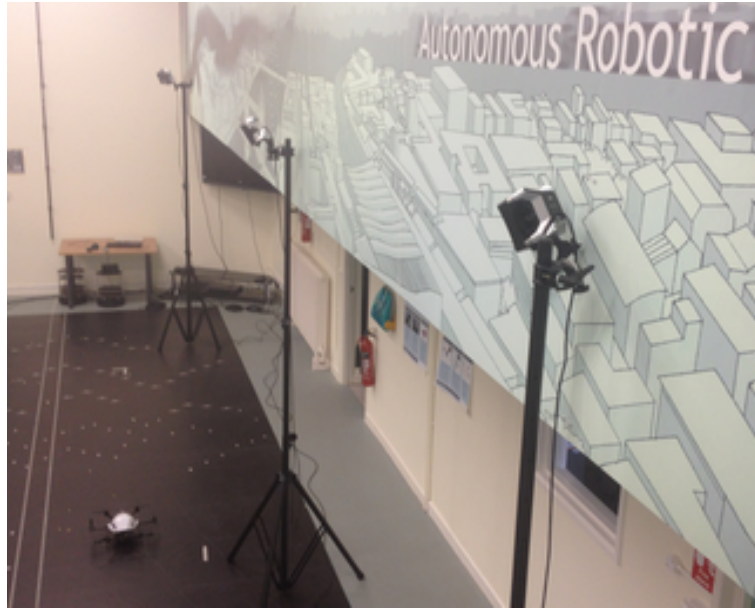


Fig. 9.4 Set-up used for the experimental validation.

In our experiment, planned paths were established for each vehicle for each flight maintaining a baseline of around two metres.

Once a stereo rig is constructed, accurate 3D points are generated, allowing significant correction of each vehicle's motion. This is due to the fact that 3D scene points are essential for recovering the scale in monocular motion estimation systems. The recovered relative pose of the vehicles is also used to correct each vehicle's position in the global frame. Figure 9.5 shows samples of images used to detect a loop closure. In this figure, each column corresponds to images captured by one vehicle (left column for vehicle one and right column for vehicle two). Matched points between top and bottom images in the same column are used to independently estimate the motion of each vehicle. Matched points between the two vehicles are shown in the images at the top. In our implementation the SIFT feature extractor is used, with the nearest neighbour search technique in the matching task.

9.7.1 Motion estimation for each UAV

As described in Section 9.4, three-view geometry through the trifocal tensor is used to recover the camera matrices from which the motion is estimated. The proposed algorithm for trifocal tensor estimation was tested separately using real data. The error is represented in terms of residual Euclidean distance, representing the root-mean-squared (RMS) distance between the measured points and the transferred points obtained via the estimated trifocal tensor \mathcal{T} .

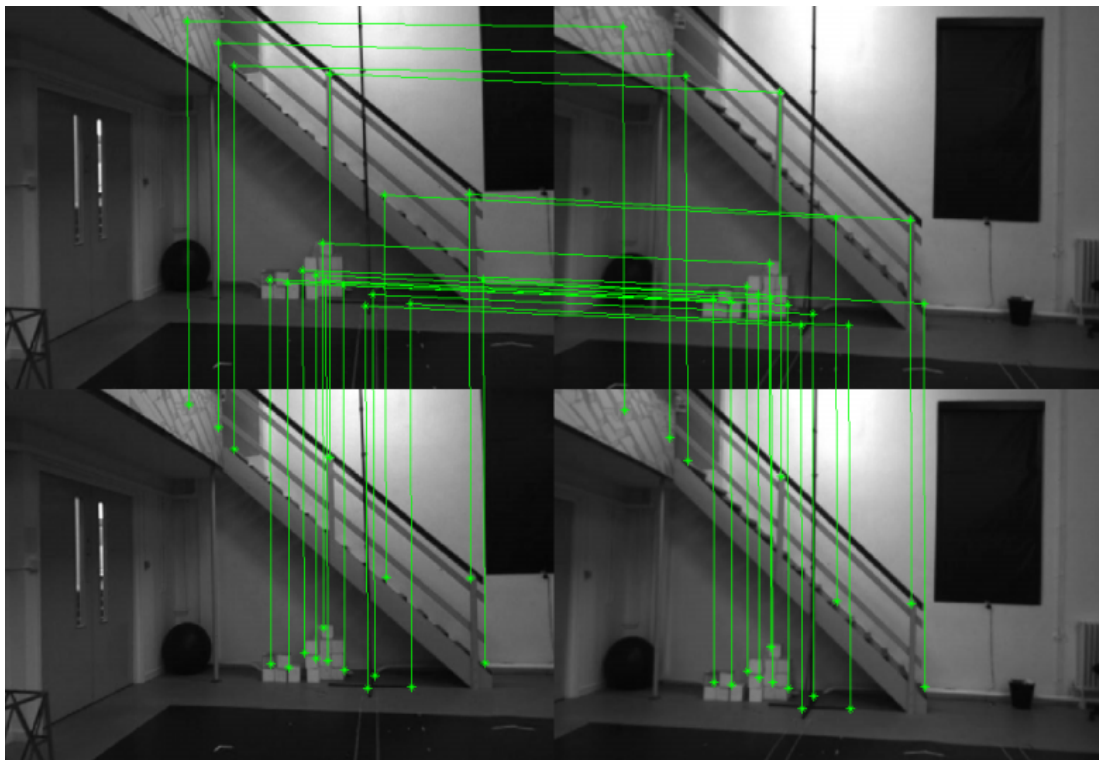


Fig. 9.5 Sample of images used to detect a loop-closure, and to then construct the stereo rig.

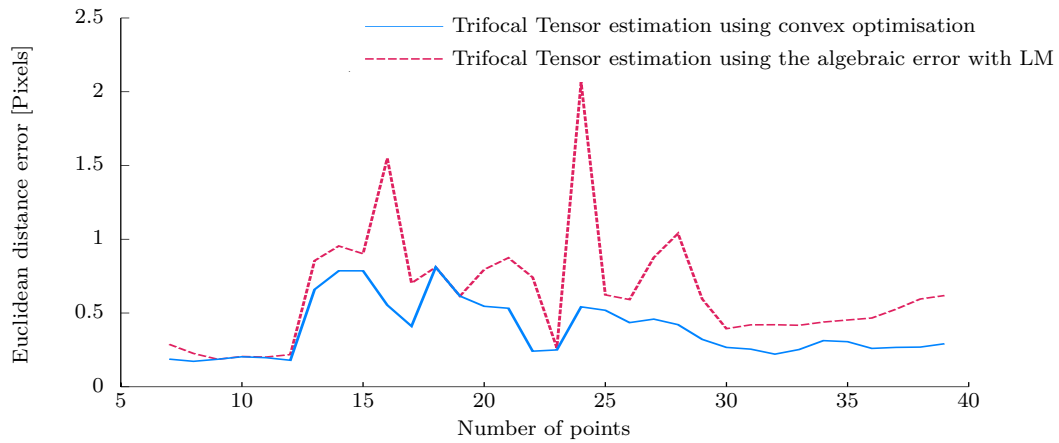
The proposed algorithm was tested on images with different illumination patterns from two different environments, namely, indoor and outdoor locations. Graphs shown in Figure 9.6 represent the points used along the horizontal axis and residual error on the vertical axis. A comparison with state-of-the-art iterative methods based on the Levenberg-Marquardt algorithm is given as well. Results obtained using the proposed solution based on convex optimisation are plotted in solid blue curves. For comparison purposes, the algebraic minimisation algorithm using the iterative Levenberg-Marquardt (LM) algorithm is implemented as well [82]. As one can see, the proposed algorithm performs very favourably in both environments compared to the algebraic minimisation algorithm.

Errors on the LM-based algorithm plots confirm its susceptibility to converging to local minima while the convex optimisation algorithm ensures global minima. The results obtained indicate that convex optimisation performs very well in comparison to the LM iterative algorithm. The proposed algorithm has two advantages over the LM iterative algorithm. First, the cost function which is minimised, i.e. the Euclidean distance between the measured and the transferred points, is geometrically meaningful. Second, the cost function is convex; therefore, it is ensured to get the global minimum.

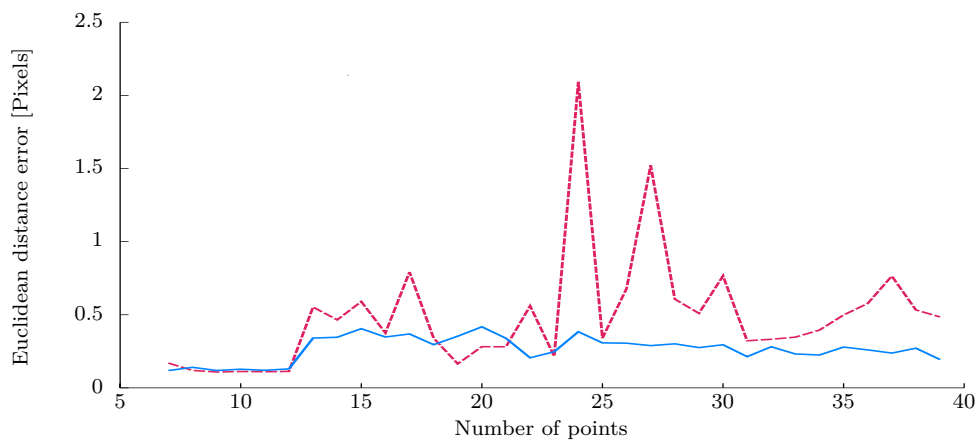
The recovered trifocal tensor is used to extract the camera matrices, from which the motion of each vehicle is estimated. Convex optimisation for triangulation is then used to optimise the estimated motion parameters in a convex bundle adjustment framework. Following this, 3D scene points are used to remove the scale ambiguity. Finally, the estimated camera positions are aligned with the ground truth camera positions and an RMS error for camera position is computed. Results from two different experiments with different trajectories are shown in this section. A plot of the error as a function of time is shown in Figure 9.7 for each experiment. In addition, Figure 9.8 shows plots of the estimated trajectory of each vehicle. These plots show that the vehicle positions can be estimated relatively well when three-view geometry is used.

From these results, one can learn that the trifocal tensor method performs noticeably better than the essential matrix with two-view geometry. This is as expected, since the trifocal tensor encodes information about the relative scale of the translations. This confirms that three-view geometry estimates are more accurate and more consistent. In fact, greater stability is achieved when including a third view.

Figure 9.8 compares the estimated trajectories of each vehicle using three- and two-view geometry. One may notice the improvement in the motion estimates when three-view geometry is used.

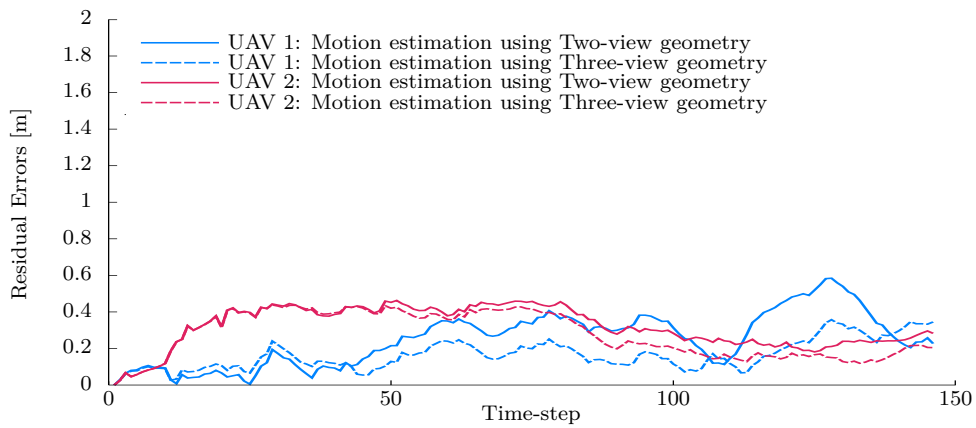


(a) Indoor environment

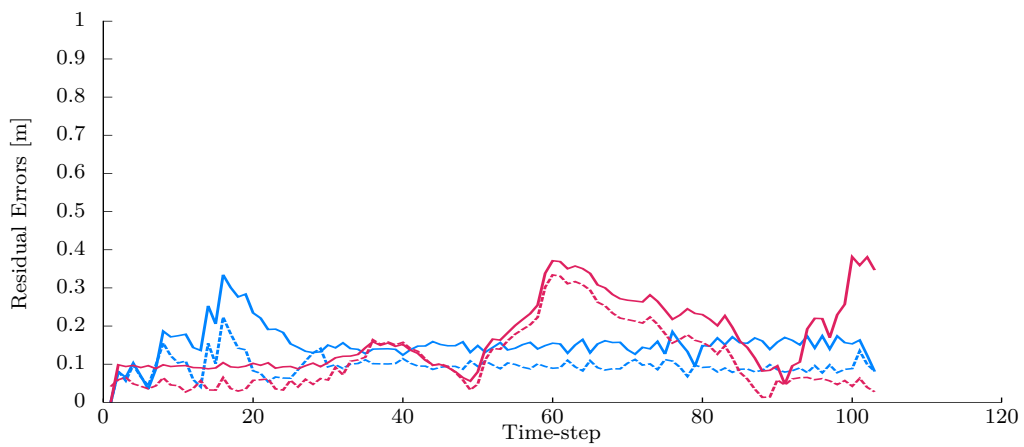


(b) Outdoor environment

Fig. 9.6 Trifocal tensor computation results using the convex optimisation technique and the algebraic error minimisation with iterative LM optimisation in two different experiments. The proposed algorithm was tested on images with different illumination patterns from two different environments, namely, indoor and outdoor locations. Residual error is plotted as a function of the number of points. The blue solid line is the convex optimisation algorithm, and the red dashed line is the algebraic minimisation algorithm using the iterative Levenberg-Marquardt (LM) algorithm. The results with the convex optimisation technique are better, with the residual errors being substantially less than in the algebraic error minimisation with iterative LM optimisation.

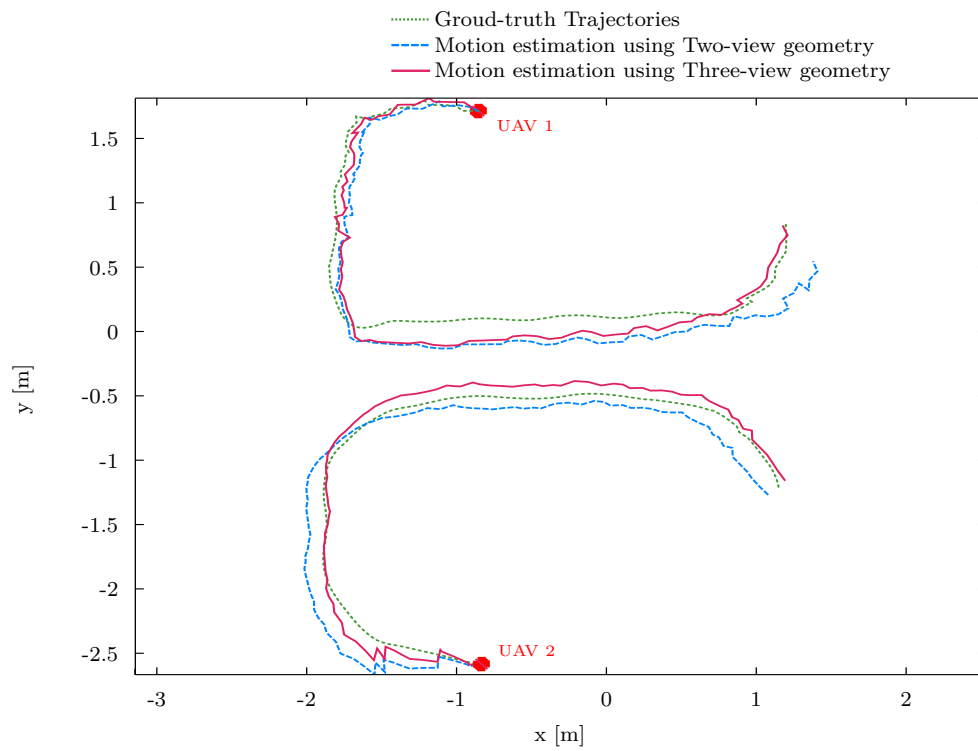


(a) Experiment 1

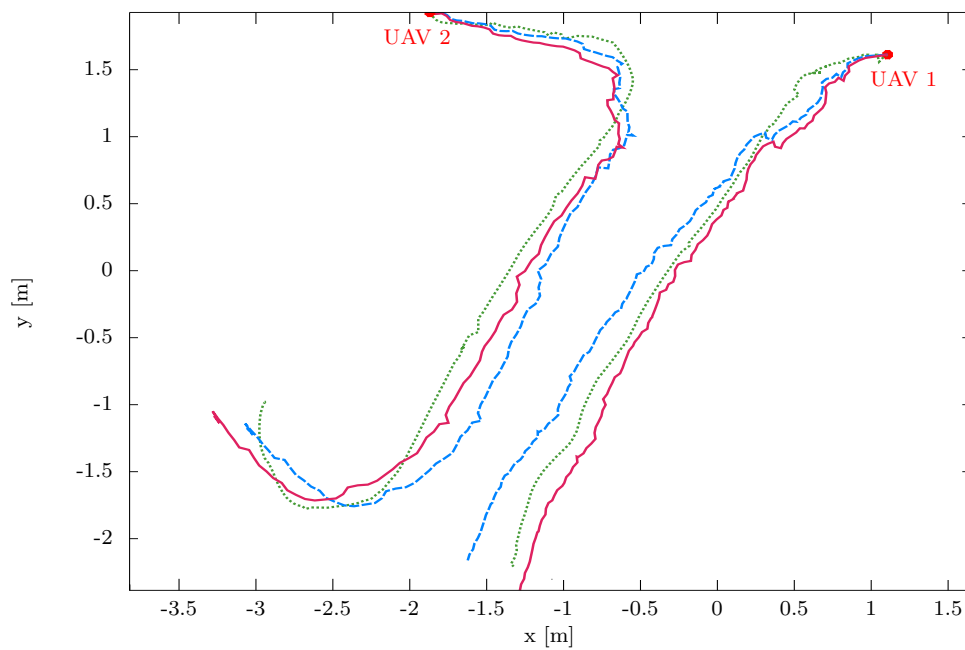


(b) Experiment 2

Fig. 9.7 Performance of motion estimation. The vertical axis gives the errors of each vehicle position. The trajectories estimated using the trifocal tensor are more accurate for both vehicles.



(a) Experiment 1



(b) Experiment 2

Fig. 9.8 The estimated trajectories using two- and three-view geometry plotted against the ground truth for the two experiments.

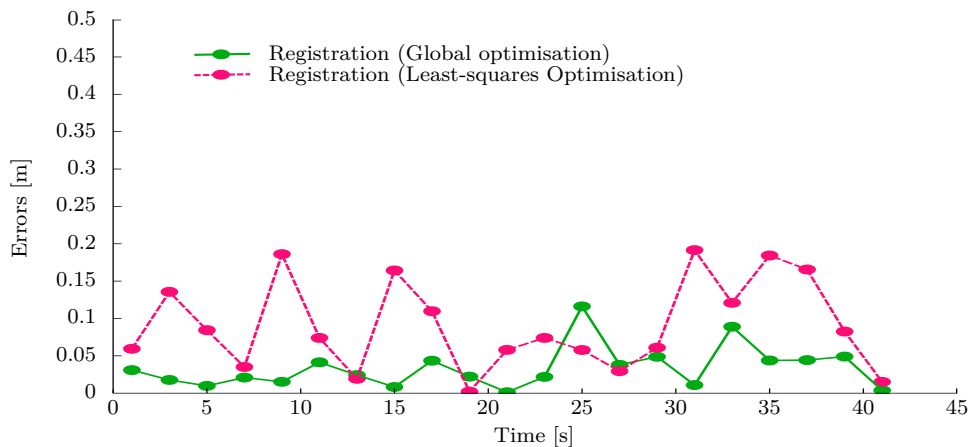


Fig. 9.9 The graph shows the results of the relative pose estimation using the registration solution via the global optimisation for the first experiment. Residual error is plotted as a function of time along with the loop closure occurrences. The dashed line is the classical ICP via the least squares minimisation. Residual errors from global optimisation are substantially smaller than those for the algorithm using the least squares method. The algorithm performs well all through the loop closure duration.

9.7.2 Relative pose estimation

The two techniques used to estimate the relative optimisation are set off once a loop-closure is detected. Each technique provides its own uncorrelated estimates. These estimates are then fused, in the interest of more robustness, stability and consistency.

Registration using global optimisation:

In this section we describe the results obtained when using the registration approach. This technique relies on global optimisation through the branch and bound algorithm to recover the relative rotation and the translation of the two vehicles. Registration is performed between 3D points constructed by each vehicle. Each vehicle is equipped with a monocular vision system capturing images at 20 Hz. For comparison purposes, implementation of the classical ICP algorithm is performed as well. Residual errors are shown as a function of time along with the loop closure occurrences. The relative translation errors are estimated in terms of the Euclidean 3D residual error. The plots all show time along the horizontal axis and residual error along the vertical axis.

Figure 9.9 and Figure 9.10 compare the classical ICP algorithm with the proposed technique using global optimisation. Through the residual errors, one can see that the branch and bound algorithm is able to provide consistent minimisations in

comparison to the least-squares-based algorithm. The former algorithm performs well all through the loop closure duration. In these plots, the dashed lines represent the classical ICP via the least squares minimisation. Residual errors from global optimisation are substantially smaller than the algorithm using the least squares method.

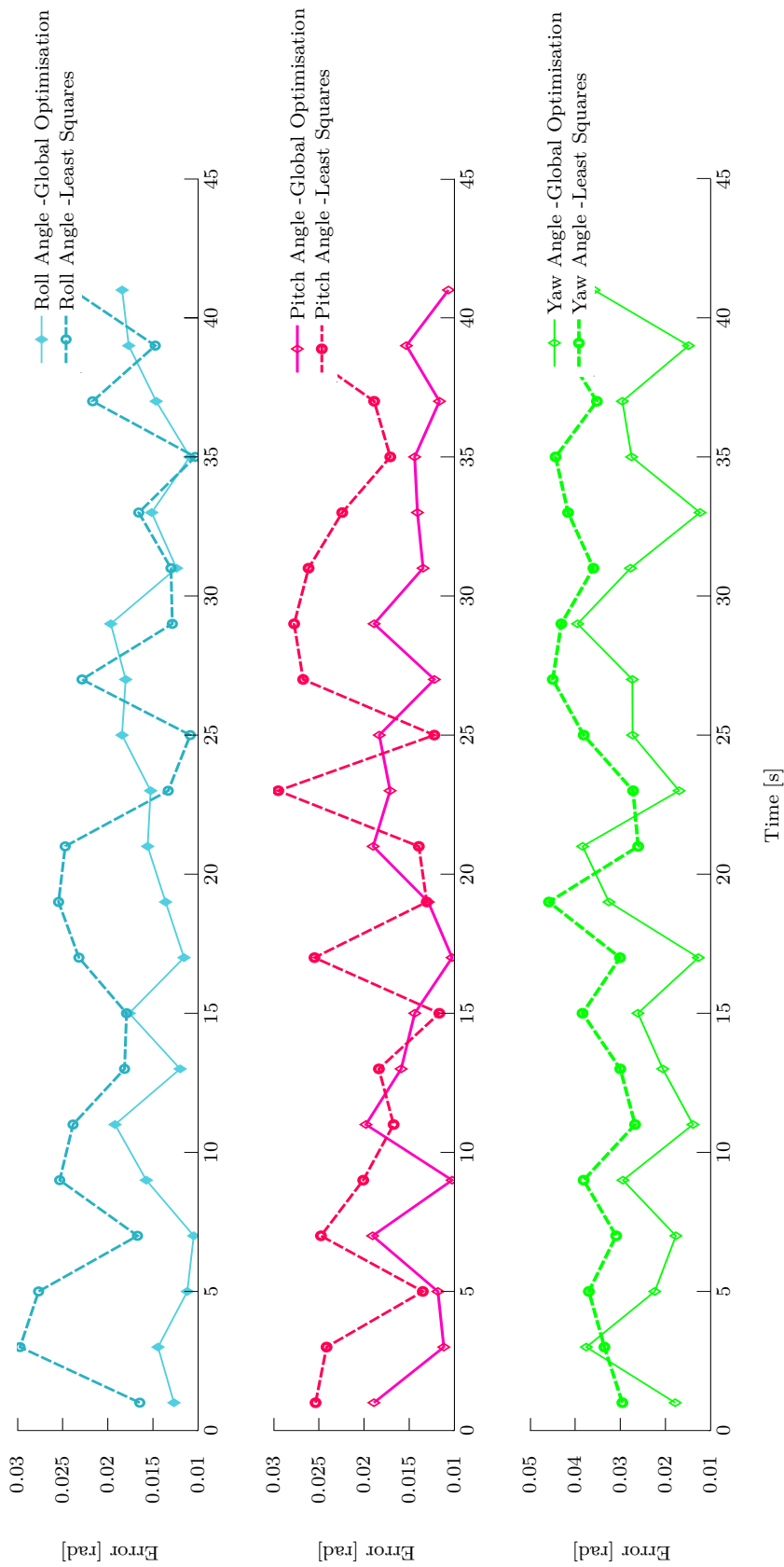


Fig. 9.10 The graphs show the experimental results for the relative rotation of the vehicles using the registration solution with global optimisation. Angular residual error (in radian) is plotted as a function of time along with the loop closure occurrences. Dashed lines represent the classical ICP via least squares minimisation. Again, residual errors from global optimisation are significantly smaller than for the algorithm using the least squares method. Top: Roll angles. Middle: Pitch angles. Bottom: Yaw angles.

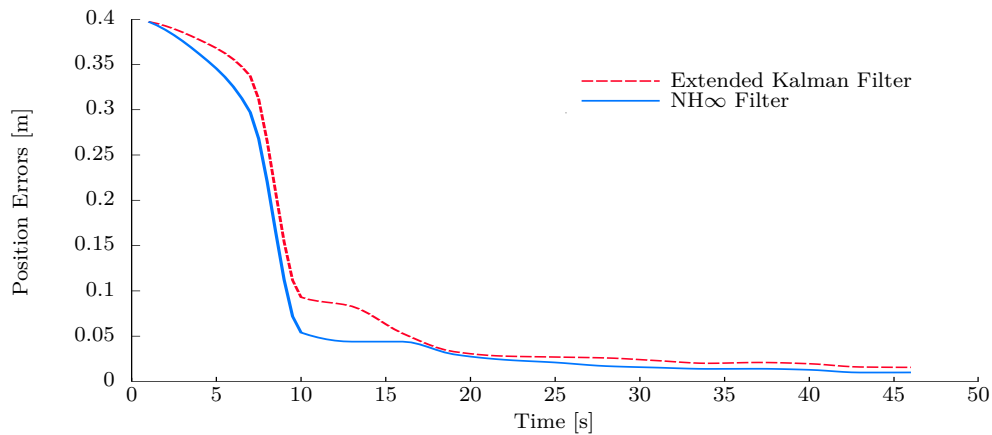


Fig. 9.11 The graph shows the results of relative pose estimation using the non-linear H_∞ . Residual errors are plotted as a function of time along with the loop closure occurrences. The dashed line plots residual errors using the extended Kalman filter.

Relative pose estimation using non-linear H_∞ filter:

The non-linear H_∞ filter presented in Section 9.6 is implemented to recover the relative translation and rotation of the vehicles involved in a loop-closure. To evaluate the filter performance, an extended Kalman filter (EKF) is implemented as well [4], and comparisons with the proposed non-linear H_∞ filter are drawn. Each vehicle is equipped with an IMU capturing data at 100 Hz. One can realise that the NH_∞ filter estimates are relatively more frequent than those of the registration technique, which provides relative pose estimates at a frequency of 20 Hz. Thus, synchronisation is of great importance in our solution.

Figure 9.11 and Figure 9.12 illustrate the results for both the translation and rotation. Errors on these plots are given in terms of residual error, which is the distance between the estimated relative pose and the true pose given by the Optitrack system. Again, these plots show residual error behaviour over time. Figure 9.11 shows the relative translation errors using the proposed NH_∞ filter and the standard EKF filter. Even though both filters converge relatively quickly (after about 7 seconds), residual errors from the non-linear H_∞ are consistently smaller than the EKF errors. From these plots, it is clear that the estimates given by the NH_∞ filter are more accurate than those given by the EKF. Indeed, one can clearly see the NH_∞ filter's robustness and consistency. The NH_∞ filter is not constrained by any assumptions about the system or the noise. This has made this filter more robust against any type of disturbances. Furthermore, the inclusion of higher order terms of the Taylor expansion has demonstrated its effectiveness on real aerial navigation, which is highly non-linear. However, the main problems with the non-linear H_∞

filter are related to the computational cost, in which slightly more time is required to perform the min-max minimisation. In addition, the non-linear H_∞ filter requires more tuning parameters (including the P , Q and R covariance matrices), in which the filter is more sensitive. However, the guaranteed robustness of the non-linear H_∞ is more important in our implementation.

A similar pattern is evident in the relative rotation. Figure 9.12 plots errors in the estimated relative rotations shown through the roll, pitch and yaw angles. One may notice that the relative rotation has converged relatively quickly in comparison to the translation. This is because the rotation is independent from the scale ambiguity in the monocular vision system. The rotations estimated from the vision system are more consistent in comparison to the translations. Less stable figures are observed for the yaw angles are noticed. This is due to the fact that most motion has occurred along the axis handling this angle.

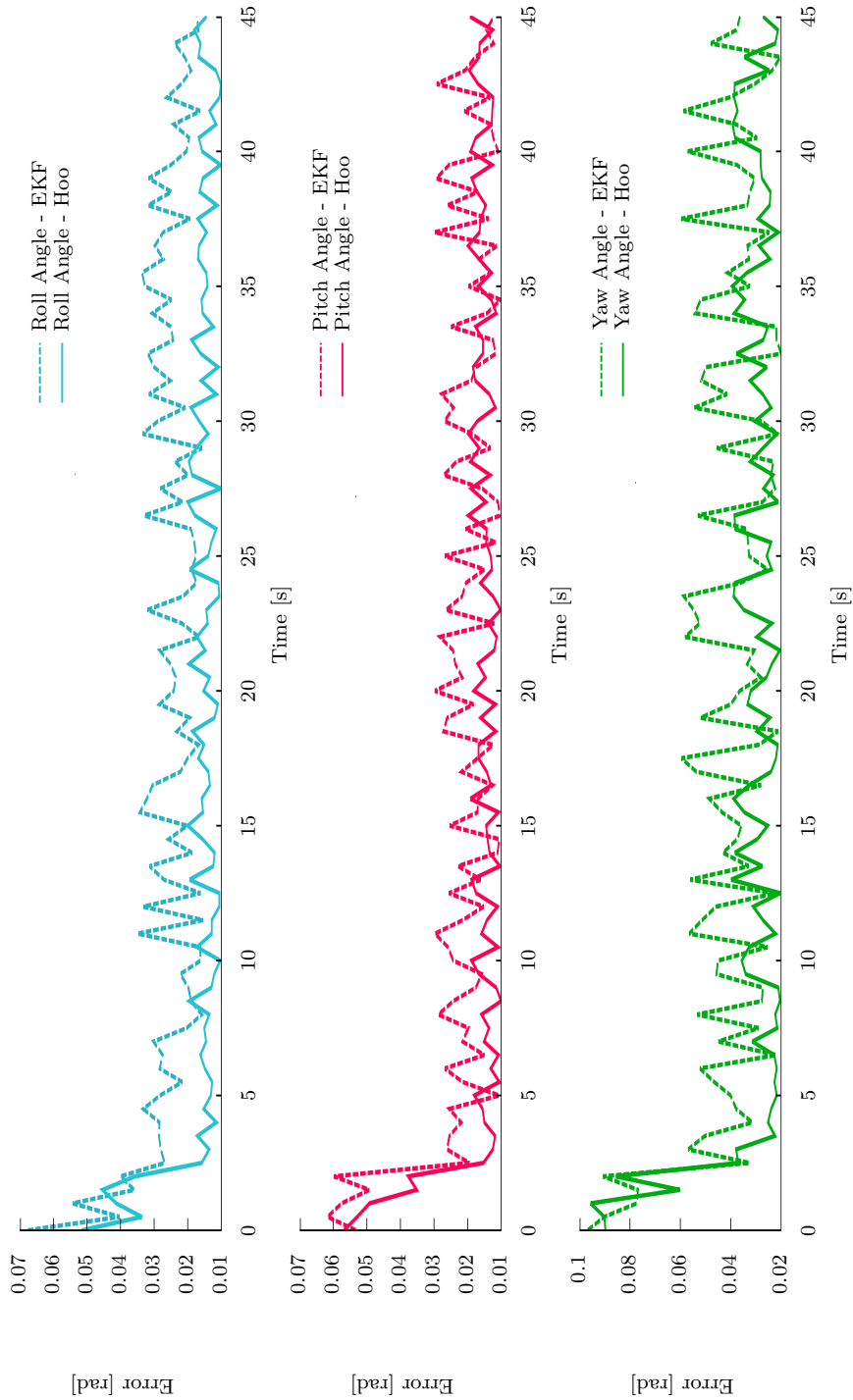


Fig. 9.12 The graphs show the experimental results for the relative rotation of the vehicles using the non-linear Hoo filter. Angular residual error (in radians) is plotted as a function of time along with the loop-closure occurrences. Dashed lines illustrate results with the extended Kalman filter (EKF). Again, residual errors from non-linear Hoo Filter are comparatively smaller than for the standard EKF filter. Top: Roll angles. Middle: Pitch angles. Bottom: Yaw angles.

Table 9.1 summarises the RMS, the minimum and the maximum errors from the conducted experiments. In each experiment, different trajectories were covered by each vehicle, involving different relative translations and rotations.

Table 9.1 RMS, maximum and minimum errors for the relative pose estimation using NH_{∞} and EKF

		Relative Translation [m]			Relative Rotation [rad]		
		X	Y	Z	Roll	Pitch	Yaw
RMS	NH_{∞}	0.080	0.105	0.097	0.030	0.025	0.060
	EKF	0.095	0.112	0.104	0.033	0.026	0.063
Max	NH_{∞}	0.205	0.430	0.290	0.100	0.120	0.180
	EKF	0.195	0.453	0.223	0.123	0.102	0.167
Min	NH_{∞}	0.019	0.017	0.020	0.018	0.014	0.028
	EKF	0.028	0.023	0.028	0.016	0.021	0.034

Fusing the registration and the NH_{∞} filter estimates:

After estimating the relative pose between the two vehicles using the two methods (registration and the NH_{∞} filter), a fusion algorithm based on averaging the estimates from the two techniques is used to recover an optimised estimate that will be used on the stereo rig. Figure 9.13 combines the estimates from each method. Obviously, estimates from the NH_{∞} before its stabilisation are discarded. Dots on the solid green line indicate estimation time with the registration technique. Clearly, the estimates of the NH_{∞} are more frequent than those of the registration technique. This is due to the difference in data acquisition rates between the IMUs and the vision systems.

Significantly, this graph illustrates the utility of fusing the estimates from the two techniques. Prior to the NH_{∞} filter stabilisation, estimates from the registration are used exclusively. This is, in fact, the main purpose of using two techniques in estimating the relative pose. Our experiments revealed that the NH_{∞} requires some time for its stabilisation when it can provide consistent estimations. This period is required to correct the absolute translation scale which is estimated from the vision algorithm. Meantime, the registration method provides absolute estimations and does not rely on the previous estimates to provide new accurate estimations. Thus, in the fusion algorithm, at the early stage of the loop-closure, larger coefficients are credited to the registration method than those for the NH_{∞} filter. These coefficients are gradually augmented for the latter technique, allowing more contribution to its estimates along with its stabilisation. After its stabilisation, even though more accurate estimates are obtained from the NH_{∞} filter, the registration technique

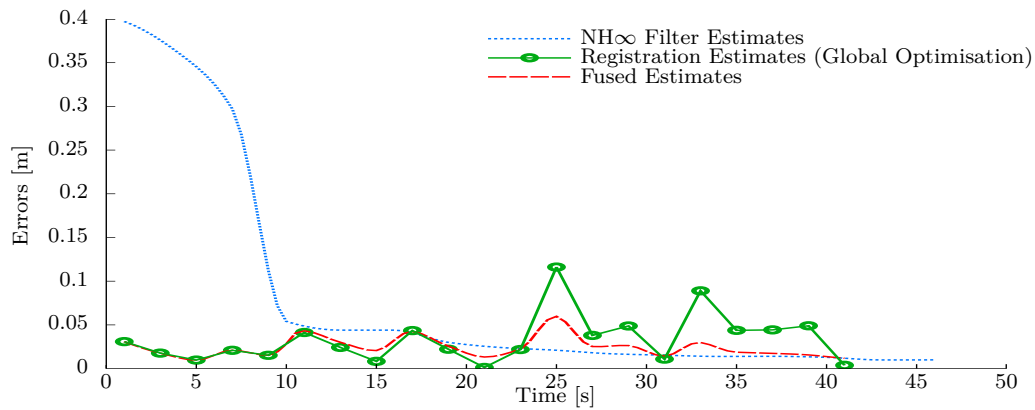


Fig. 9.13 The graph shows the estimation error of the relative translation using the non-linear H_∞ filter, the registration technique and the fused estimates. Residual errors are plotted as a function of time along with the loop-closure occurrences. The dotted blue line plot result from the non-linear H_∞ filter. Dots on the green solid curves indicate the estimation time for the registration technique. The dashed red line shows errors after fusing the estimations from the non-linear H_∞ filter and the registration technique.

continues doing well, with more consistent figures being recorded. This suggests that using two different techniques can increase the stability of the global cooperative motion estimation solution. Similar pattern is recorded also for the relative rotation, where the best performance alternates between the two techniques, as illustrated in Figure 9.10 and Figure 9.12.

Cooperative motion estimation:

As described in Section 9.3, after estimating the optimal relative transformation between any two vehicles involved in a loop-closure, a stereo vision set-up is constructed. By capitalising on the stereo vision's advantages, more accurate 3D scene points can be estimated. This results in improving the accuracy of the individual motion estimates of each vehicle. In addition, the recovered relative pose between the vehicles is also used to correct each vehicle's position in the global navigation frame.

Figure 9.14 and Figure 9.15 show results following cooperative motion estimation for two separate experiments. The first two plots on each figure record error evolution over time for each vehicle. The loop-closure occurs around key-frames 30 and 55 in the first and the second experiment respectively. The RMS position error was 0.15 m before the loop-closure detection. This RMS error has substantially dropped to 0.06 m after the cooperative motion is performed. The bottom plots on Figure

9.14 and Figure 9.15 illustrate each vehicle's trajectory before and after cooperative motion estimation, aligned with the ground truth. The highlighted portions indicate the loop-closure occurrences.

As soon as a loop-closure is detected, matched image features are used to construct joint 3D scenery, or more precisely, a joint map. In addition to the motion estimation, the proposed solution for the registration allows merging the maps constructed by each vehicle to build a global single map. This global map will implicitly contribute to improvement of the motion. By implication, this global map is continuously constructed and optimised, leading to more accurate motion estimates for vehicles navigating within this map.

9.8 Conclusions

In this chapter, a system for cooperative monocular visual motion estimation with multiple aerial vehicles is proposed. The distributed system between vehicles allows efficient processing in terms of both computational time and estimation accuracy. The global cooperative motion estimation employs state-of-the-art approaches for optimisation, individual motion estimation and registration. Three-view geometry algorithms in a convex optimisation framework are deployed on-board the monocular vision system for each vehicle. In addition, vehicle-to-vehicle relative pose estimation is performed with a novel robust registration solution in a global optimisation framework. Complementary to the relative pose, a robust non-linear H_∞ solution is also designed to fuse measurements from the UAVs' on-board inertial sensors with the visual estimates.

Results on real-world data demonstrate the effectiveness and the efficiency of the proposed solution in a vision-only motion estimation framework. Future work will focus on the potential use of a decentralised architecture, where communication becomes a challenging task. Even if it is still an immature technology, investigating the inclusion of the received signal strength indication (RSSI) technique and the light-based communication technique for estimating the relative pose between vehicles would be interesting options.

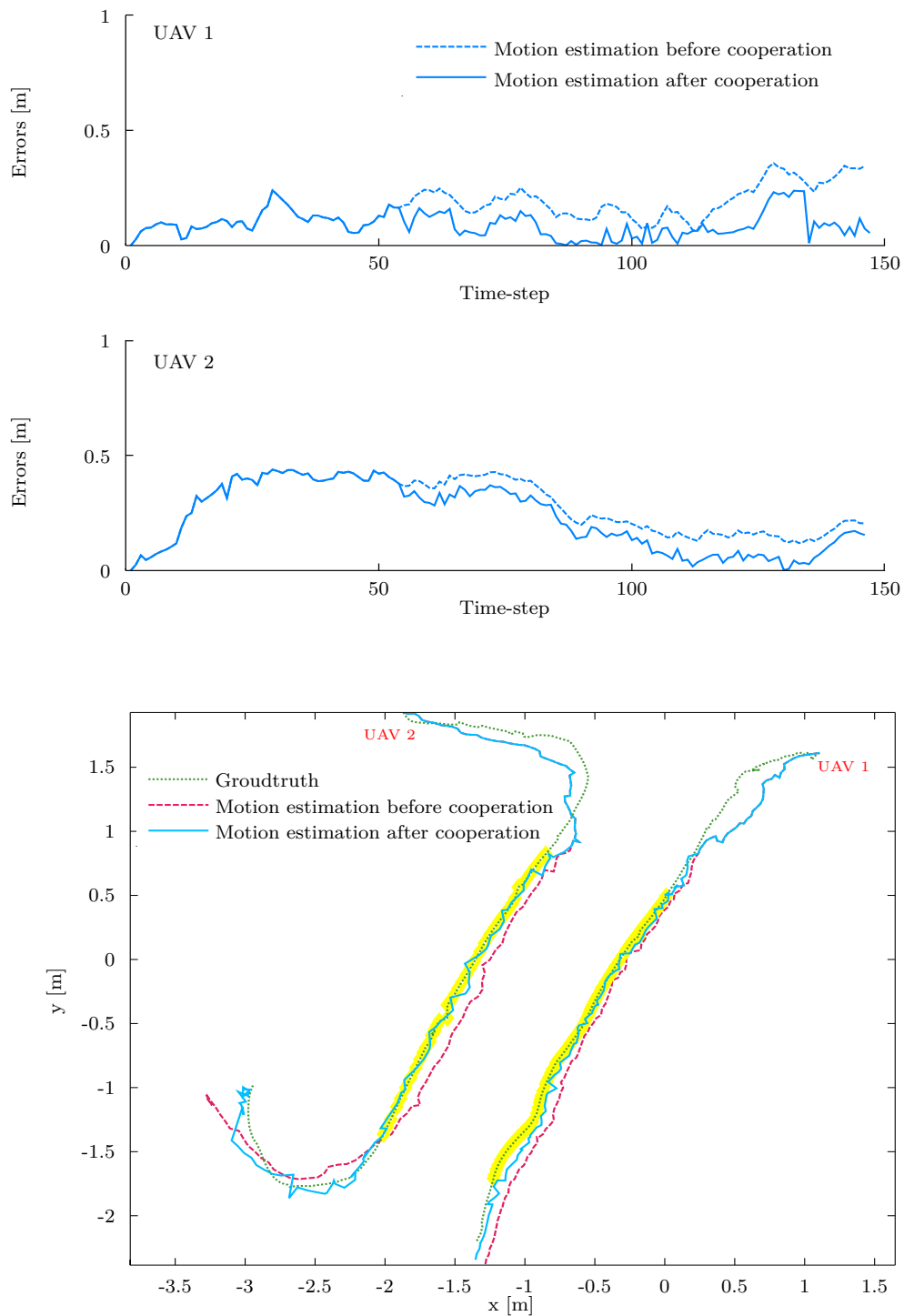


Fig. 9.14 Experiment 1: Top two graphs show the error evolution over time before and after loop-closure detection for each vehicle. The estimated trajectories, aligned with the ground truth, are shown on the bottom plot. The highlighted portions indicate the loop-closure occurrences.

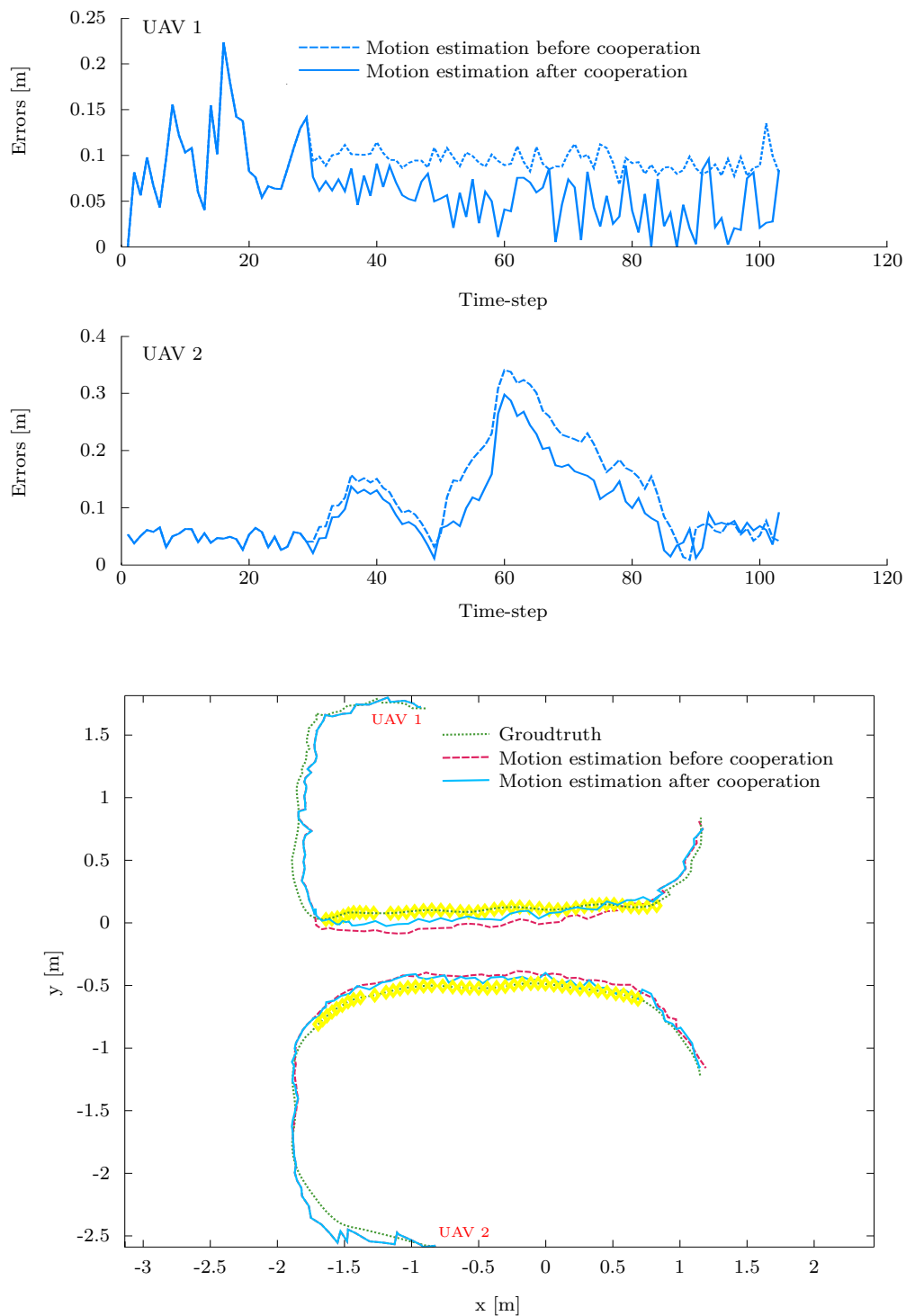


Fig. 9.15 Experiment 2: Top two graphs show the error evolution over time before and after loop-closure detection for each vehicle. The estimated trajectories, aligned with the ground truth, are shown on the bottom plot. The highlighted portions indicate the loop-closure occurrences.

Chapter 10

Discussion and Conclusion

10.1 Overview

Optimisation is of great importance in computer vision, as many fundamental geometrical tasks may be solved by formulating them as optimisation problems. Until recently, linear methods based on the singular value decomposition and iterative estimation algorithms based on the minimisation of an L_2 norm were typically adopted to solve vision problems.

In this thesis, we investigated the deployment of convex optimisation techniques to address the motion estimation problems in computer vision. We have seen that when minimising the maximal error, the optimisation functions exhibit several significant convexity properties, including quasi-convexity. Indeed, the L_∞ norm exhibits many attractive properties for optimisation in computer vision problems. Modelling the motion estimation problem via this norm facilitates getting globally optimal solutions.

10.2 Summary and discussion of contributions

The first contribution in this thesis is presented in Chapter 4. The L_∞ norm and the H_∞ filter are coupled to convex optimisation. In addition to the H_∞ filter, the recursive least squares technique is presented as well. The use of these techniques is motivated by their ability to cope with noisy data and to provide optimal solutions. We have shown in this chapter that although solutions based on least squares techniques are able to provide good results, they also impose limitations that a solution based on the L_∞ norm is able to overcome.

The second contribution is presented in Chapter 5. This solution investigates the integration of information from all three RGB channels of colour images. In most

vision applications, colour images are first converted to grey-level images, leading to a serious loss of information. In our solution, however, each channel is processed separately. Then a fusion mechanism via the the covariance intersection filter is employed. This filter combines information from all three RGB channels, which results in a decrease in measurement errors, leading to more accurate estimates of the fundamental matrix.

Due to the way they are extracted, image feature location accuracy is heavily dependent on the intensity variation within their neighbourhoods, from which their uncertainties are estimated. We showed through this solution that regardless of the algorithm used to extract them, feature positions always contain some uncertainty. Indeed, the novelty in our approach consists of fusing feature localisation uncertainties in each RGB channel to robustly estimate the motion of a monocular vision system via the fundamental matrix. Over a series of experiments in different environments, we showed that including feature uncertainties from all three RGB channels leads to better estimates of the fundamental matrix, and consequently better estimates of the motion parameters.

In Chapter 6, a robust convex optimisation solution for monocular motion estimation systems is presented. This solution employs the uncertainty estimation techniques presented in Chapter 5. Estimating the system uncertainty without evaluating its propagation may not be sufficient in some navigation systems. In this chapter we showed the improvement in the global motion estimates when the system uncertainties and their propagation to the relative rotations and translations, and to the 3D scene points are incorporated. Rather than using the H_∞ filter to solve the scale ambiguity problem for the monocular system, we set up a robust least squares algorithm based on the SOCP approach capable of handling system uncertainties. Experimental evaluations showed that robust convex optimisation with the L_∞ norm for uncertain data and robust least squares clearly outperform the classical methods based on least squares and the Levenberg-Marquardt algorithm.

In practice, for any navigation system, errors in position estimates grow continuously due to the integration of noisy measurements over time and to the inherent inaccuracy of the devices. This unavoidable drift in motion estimation needs to be corrected. Thus, providing additional correction mechanisms ought to have a major impact on the final estimate of the navigation solution. Thus, following a long period of navigation in an unknown environment, detecting that the vehicle has returned to a previously visited location offers the opportunity to correct and to increase the accuracy and consistency of a vehicle's motion estimate.

In the computer vision community this is known as detecting loop-closures. Consequently, in Chapter 7 a novel appearance-based technique for visual loop-

closure detection is presented. The widely used techniques based on the Bag-of-Words image representation have shown some limitations, especially in connection to the perceptual aliasing problem. The proposed solution uses both local invariant and colour features with a combination of Gaussian mixture modelling (GMM) and the KD-tree data structure. In doing so, this solution takes advantage of the robustness of the KD-tree structure and the efficiency of the Gaussian mixture modelling representation. Experimental validations using real data from indoor and outdoor environments showed that due to their efficiency and complementarity, the combination of the KD-tree structure and the GMM could be a plausible alternative in real-time loop-closure detection for mobile robot navigation.

Upon developing a loop-closure detection technique, a means of corrections for any potential drift is required. Hence, in Chapter 8, a new robust convex pose-graph optimisation solution for UAV monocular motion estimation systems is presented. Once a loop-closure is detected, using the technique presented in Chapter 7, the convex pose-graph optimisation solution performs a correction for any drift incurred during the monocular motion estimation.

The pose-graph formulation is an intuitive way to address the pose estimation problem. Most methods in the literature utilise standard approaches, like the Gauss-Newton or Levenberg-Marquardt algorithms. However, with these methods there is no guarantee of convergence to the global minimum. Furthermore, they may lead to an infeasible solution. As such, these methods are also greatly dependent on good initialisation. Also while the aforementioned iterative methods, the proposed solution is able to recover the optimal positions configuration using convex optimisation. Moreover, uncertainty estimates and their propagation through multiple view geometry algorithms are included in this solution as well. Through a variety of experimental validations on real-world data, from indoor and outdoor environments, and comparison to state-of-the-art methods, using convex optimisation in pose-graph problems has proven its efficiency in motion estimation correction after loop-closure detections.

After developing robust solutions for visual navigation systems, in which an autonomous vehicle can estimate its own localisation independently, a need for cooperative solutions arises. To round out the work in this thesis, a system for cooperative monocular visual motion estimation with multiple aerial vehicles is proposed in Chapter 9. The distributed system between vehicles allows efficient processing in both computational time and estimations accuracy. The global cooperative motion estimation introduced employs state-of-the-art approaches for optimisation, individual motion estimation and registration. Three-view geometry algorithms in a convex optimisation framework are deployed on board the monocular vision system for each

vehicle. In addition, vehicle-to-vehicle relative pose estimation is performed with a novel robust registration solution in a global optimisation framework. In parallel, and as a complementary solution for the relative pose, a robust non-linear H_∞ solution is designed to fuse measurements from the UAVs' on-board inertial sensors with the visual estimates. Results on real world data demonstrate the effectiveness and efficiency of the proposed solution in a vision-only motion estimation framework.

Through this thesis, we have provided insight, discussion and experiments about deploying convex optimisation in motion estimation. From a practical point of view, we conclude that it is a relatively computationally inexpensive approach to achieving optimality in motion estimation problems. In contrast to most previous work dealing with motion estimation, we have presented in this thesis a framework for estimating globally optimal solutions. For a number of computer vision problems, we have shown both theoretically and experimentally that convex optimisation is indeed a tractable approach. In conclusion, we believe that the use of convex optimisation methods certainly has a place in visual motion estimation.

10.3 Future work

In this thesis, we proposed several solutions to the robust visual motion estimation problem. However, to comprehensively solve this problem, there is still a significant amount of research left to be done. We consider that the solutions developed and presented here are a next fundamental building block towards achieving this goal. The proposed navigation solutions ensure a level of stability and modular design. Achieving a totally autonomous vision-based systems is still some way off. However, mass-production in parallel with maturing technology, especially for long-life batteries and heightened payload capabilities, can cause this to happen in the near future.

Taking a broader view, we believe that by using the proposed solutions in applications where vision is the only perceptual means of navigation, it will be possible in the near future to deploy these systems in operations like search and rescue, and aerial surveillance. Widening use of micro aerial vehicles (MAVs) on which compact vision installations can be mounted spurs further interest in robust vision systems requiring global solutions. The growing demand for MAVs in urban and indoor environments for domestic missions, such as item delivery or emergency medication supply is noticed.

Multi-camera installations are a potential research topic in which developing multi-camera rigs and their calibration for absolute motion estimation with many open challenges. Investigating visual motion estimation by tracking moving objects in

highly dynamic scenes also opens up new challenges. Future works may also include real-time implementation using graphics processing unit (GPU) programming. In particular, GPU programming may reduce the computation time required to solve the motion of multi-camera systems. RGB-D (Kinect-style) cameras provide real-time colour and dense depth data through active sensing, combining the strengths of passive cameras with laser range-finders. An emergent research community is moving towards using this affordable equipment as a real-time device for robot perception. Equally, future work may include a scenario of using an RGB-D sensor in an autonomous micro vehicle (MAV) for navigation, where real-time multi-view processing rates are sought.

In cooperative navigation, a potential use of a decentralised architectures could improve coordination in a fleet of MAVs in urban environments. Even if it is still an immature technology, investigating the inclusion of the received signal strength indication (RSSI) technique and light-based communication techniques to estimate the relative pose between vehicles would be valid options to investigate as well. Using Li-Fi technology rather than radio frequency signals which are intolerant to disturbances may improve decentralised cooperative motion estimation.

Appendix A

Convex Optimisation and Multiple View Geometry

This appendix presents a review of some multiple-view geometry problems that can be solved using convex optimisation. Techniques introduced in Chapter 3 and 2 are used extensively throughout this thesis. Particular attention is given to problems of triangulation estimation, camera resectioning and homography. These tasks were implemented using the second-order cone programming (SOCP) [94] on benchmark datasets, familiar to the computer vision community. Results are then compared to traditional methods such as the linear approach by applying the Direct Linear Transformation (DLT) [82] and the L_2 minimisation using the Levenberg-Marquardt algorithm. For comparison, the L_∞ projection error and Root Mean Squares (RMS) are used to illustrate the performance of convex optimisation.

A.1 Triangulation problem

The problem of estimating a 3D scene point's positions when seen from multiple cameras is known as the triangulation problem. Given the projection matrices, the position of the 3D scene points can be estimated from their measured image point positions in two or more views. Figure A.1 illustrates this problem.

Definition 7. The triangulation problem is defined as follow: Given a set of image points \hat{x}_i and camera matrices P_i for $i = 1, \dots, m$, find the scene point $\hat{X} = (X, Y, Z, 1)^\top$ such that $\hat{x}_i = P_i \hat{X}$ for all i .

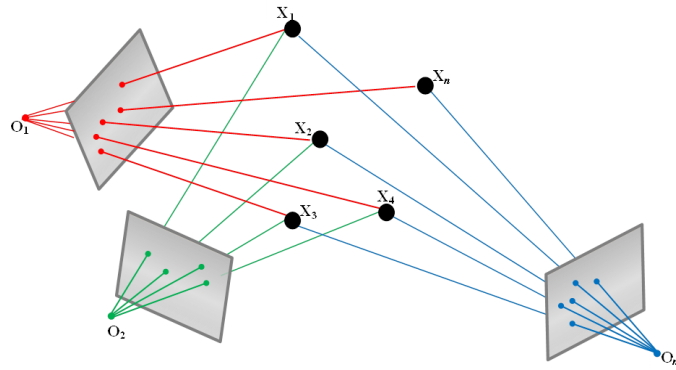


Fig. A.1 The triangulation problem. Each point is seen from two or more cameras. The aim is to estimate the position of the 3D scene points X_i .

This problem is usually solved using linear techniques such as the direct linear transformation (DLT) [82], where an algebraic cost function is minimised. It can also be solved using iterative minimisation techniques such as the Levenberg-Marquardt algorithm, in which a geometric non-linear cost function is iteratively minimised. In this section, we show how the triangulation problem can be formulated as a quasi-convex optimisation problem.

Let us start first by setting out the general framework for this problem. The last coordinate of the image points is equal to one, so $\hat{x}_i = (x_i, y_i, 1)$. The variable that we wish to estimate in this problem is the 4-element \hat{X} , where $\hat{X} = (X, Y, Z, 1)^\top$. The optimisation variable is then $x = (X, Y, Z)^\top \in \mathbb{R}^3$.

Let P_i , for $i = 1, \dots, m$ be the 3×4 camera matrices and \hat{x}_i , the corresponding measured image points of scene X , where $\hat{x}_i = P_i \hat{X}$. Then $P_i \hat{X} = (p_i^1 \hat{X}, p_i^2 \hat{X}, p_i^3 \hat{X})$. Here the 3×4 camera matrices P_i are given as:

$$P_i = \begin{bmatrix} p_i^1 \\ p_i^2 \\ p_i^3 \end{bmatrix} = \begin{bmatrix} p_{i1} & p_{i2} & p_{i3} & p_{i4} \\ p_{i5} & p_{i6} & p_{i7} & p_{i8} \\ p_{i9} & p_{i10} & p_{i11} & p_{i12} \end{bmatrix}$$

The vector p_i^j is the j^{th} row of the i^{th} camera matrix P_i . Then the optimisation problem can be formulated as:

$$\begin{aligned} \min_x \quad & \max_i d(\hat{x}_i, P_i \hat{X}), \quad \text{for } i = 1, \dots, m \\ \text{subject to} \quad & p_i^3 \hat{X} > 0, \quad \text{for } i = 1, \dots, m \end{aligned} \tag{A.1}$$

where $d(\hat{x}_i, P_i \hat{X})$ is the distance between the projected 3D scene points $P_i \hat{X}$ and the measured image points \hat{x}_i . Note that the quantity $p_i^{3\top} \hat{X}$, in the constraint function, represents the depth of a point in an image, which implies that the points must lie in front of the cameras. This optimisation entails that we are looking for the position of X that minimises the biggest error between its projections on the m images and the corresponding image points. Therefore, the residual error ε_i on the i^{th} image can be written as:

$$\begin{aligned}
 \varepsilon_i &= d(\hat{x}_i, P_i \hat{X}) \\
 &= \sqrt{\left(\frac{p_i^{1\top} \hat{X}}{p_i^{3\top} \hat{X}} - x_i\right)^2 + \left(\frac{p_i^{2\top} \hat{X}}{p_i^{3\top} \hat{X}} - y_i\right)^2} \\
 &= \sqrt{\frac{\left(p_i^{1\top} \hat{X} - x_i p_i^{3\top} \hat{X}\right)^2 + \left(p_i^{2\top} \hat{X} - y_i p_i^{3\top} \hat{X}\right)^2}{\left(p_i^{3\top} \hat{X}\right)^2}} \\
 &= \sqrt{\frac{f_{i1}(x)^2 + f_{i2}(x)^2}{f_{i3}(x)^2}}
 \end{aligned} \tag{A.2}$$

The point \hat{X} is seen from m cameras, this gives m error residuals: $\varepsilon = (\varepsilon_1, \varepsilon_2, \dots, \varepsilon_m)^\top$. Therefore, the optimal 3D scene point \hat{X} is the one that minimises the norm of the error residuals vector: $\|\varepsilon\|$.

Most classical methods use the L_2 norm for the cost function $\|\varepsilon\|_2$ or, equivalently, $\|\varepsilon\|_2^2 = \sum_{i=1}^m \varepsilon_i^2 = \sum_{i=1}^m d(\hat{x}_i, P_i \hat{X})^2$. This optimisation problem is shown in the literature to have multiple local minima [80, 82, 94, 101]. Therefore, iterative methods can easily get trapped in one of these local minima instead of ending up in the global minimum.

To get around this problem, the L_∞ norm is used instead. Optimising the L_∞ norm of ε leads to the cost function: $\|\varepsilon\|_\infty = \max_i |\varepsilon_i| = \max_i |d(\hat{x}_i, P_i \hat{X})|$. The goal of the optimisation problem now is to minimise the maximum error between the projected points and the measured image points, hence:

$$\begin{aligned}
 \min_x \quad & f_0(x) = \max_{i=1, \dots, m} f_i(x) = \max_{i=1, \dots, m} \left(\frac{f_{i1}(x)^2 + f_{i2}(x)^2}{f_{i3}(x)^2} \right) \\
 \text{subject to} \quad & f_{i3}(x) > 0, \quad \text{for } i = 1, \dots, m.
 \end{aligned} \tag{A.3}$$

From A.2, one can see that f_{i1}, f_{i2}, f_{i3} are all affine functions of the optimisation variable x . Then, according to Theorem 8 (page 70), $f_i(x) = \frac{f_{i1}(x)^2 + f_{i2}(x)^2}{f_{i3}(x)^2}$ is quasi-convex function. In addition, according to theorem 7 (Chapter 3, page 61), the function

$f_0 = \max_{i=1, \dots, m} \left(\frac{f_{i1}(x)^2 + f_{i2}(x)^2}{f_{i3}(x)^2} \right)$ is also quasi-convex since the pointwise maximum is also convex. The constraint function has a convex domain $\{x\} \mid f_{i3}(x) > 0, \text{ for } i = 1, \dots, m$, therefore problem (A.3) is a quasi-convex optimisation problem [94, 101].

To solve this optimisation problem, suppose δ is an upper bound for the objective function $f_0(x) = \max_{i=1, \dots, m} f_i(x)$. Obviously, δ is also an upper bound for each of the functions $f_i(x)$. Then,

$$\begin{aligned} f_i(x) &\leq \delta \\ \frac{f_{i1}(x)^2 + f_{i2}(x)^2}{f_{i3}(x)^2} &\leq \delta \\ \frac{\sqrt{f_{i1}(x)^2 + f_{i2}(x)^2}}{f_{i3}(x)} &\leq \delta \end{aligned} \tag{A.4}$$

Since $f_{i3}(x) > 0$ for $i = 1, \dots, m$, and by using the norm $\|u\|_2 = (u^\top u)^{\frac{1}{2}}$, then the function in A.4 may be reformulated as:

$$\|f_{i1}(x), f_{i2}(x)\|_2 \leq \delta f_{i3}(x), \quad i = 1, \dots, m. \tag{A.5}$$

Thus, the quasi-convex optimisation problem in (A.3) can be rewritten as:

$$\begin{aligned} \min_{\delta, x} \quad & \delta \\ \text{subject to} \quad & \|f_{i1}(x), f_{i2}(x)\|_2 \leq \delta f_{i3}(x), \quad i = 1, \dots, m. \end{aligned} \tag{A.6}$$

This problem can be solved using a sequence of SOCP feasibility problems via the bisection algorithm (Algorithm 1, page 65) as shown in Section 3.7.1. Essentially, for a given value of $\delta \in \mathbb{R}$ in each iteration of the bisection algorithm, the problem (A.6) is solved by checking its feasibility. The convexity frame is maintained due to the intersection of convex cones. If the SOCP problem (A.6) is feasible, then there must exist a more optimal solution $\delta^* \leq \delta$. However, if the SOCP is infeasible, then the optimal solution must be greater than δ ($\delta^* > \delta$). This leads toward using a bisection search to find the minimum value of δ for which the optimisation problem is feasible [82, 94]. Therefore the L_∞ norm of ε is defined to be the maximum of $(\varepsilon_1, \varepsilon_2, \dots, \varepsilon_m)$.

A.1.1 Experiments

To show the performance of quasi-convex optimisation with the L_∞ norm in the triangulation problem, we conducted some experimental tests on real data. The publicly available image sequence of the corridor dataset of the Oxford vision group [203] is the one used. This sequence consists of 11 real images. Errors were estimated using two methods. The first one is root mean squares (RMS):

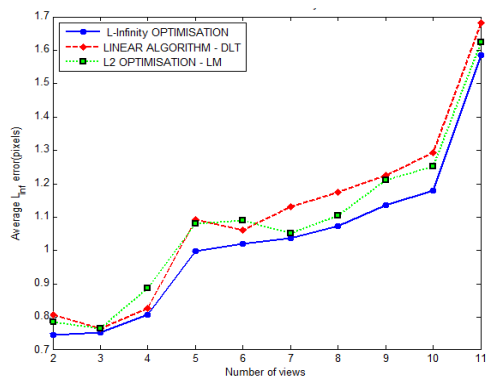
$$\text{RMS Error} = \sqrt{\frac{1}{m} \sum_{i=1}^m \varepsilon_i^2} \quad (\text{A.7})$$

and the second one is the L_∞ error:

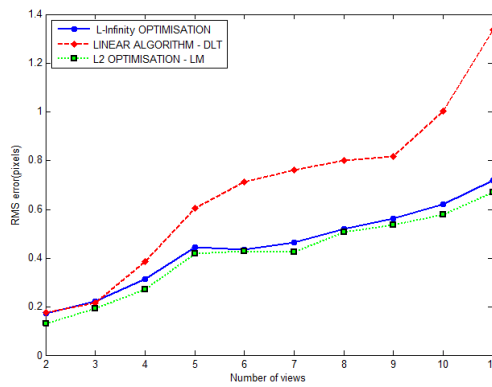
$$L_\infty \text{ Error} = \max_{i=1, \dots, m} |\varepsilon_i| \quad (\text{A.8})$$

For comparison purposes three optimisation methods were implemented in this experiment as shown in Figure A.2. The first is the quasi-convex optimisation method with the L_∞ norm as described in the present section. The second method is a linear approach applying the direct linear transformation (DLT) algorithm, which is presented in [82]. The third approach employs an iterative minimisation technique by using the Levenberg-Marquardt algorithm. The method adopted here is introduced in [94] and uses the SeDuMi toolbox [187] for the convex optimisation problem via the second-order cone programming SOCP.

Investigation of the results obtained shows varying performance according to the method used to evaluate the errors. Figure A.2a and Figure A.2b show the Euclidean distance errors in the image plane space against the number of views used to estimate the 3D scene point position. When the L_∞ projection error is used (Figure A.2a), the L_∞ method clearly outperforms both the DLT method and the L_2 -based iterative method. This is in conformity with the theory in terms of global optimality of the L_∞ . The L_2 method seems to perform better than the DLT method as well. When a comparison is performed using the RMS errors (Figure A.2b), both the L_∞ and the L_2 -based iterative methods clearly outperform the DLT method. Figure A.2c and Figure A.2d show respectively, the image points in the last view of the sequence and the 3D scene points recovered.



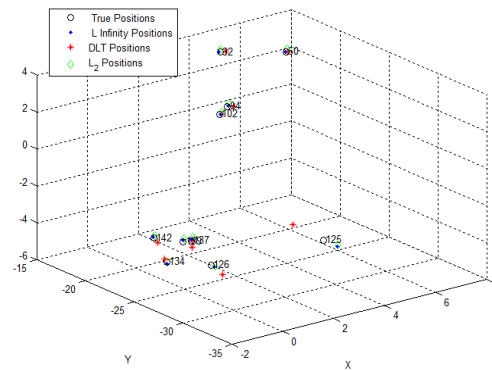
(a)



(b)



(c)



(d)

Fig. A.2 The RMS and L_∞ errors of the triangulation for the corridor dataset.

A.2 Camera resectioning

Camera resectioning is the process of finding the camera projection matrix P given 3D scene points and their corresponding image points. The camera projection matrix P represents the relationship between 3D scene points X_i and their image projections x_i (Figure A.3). The objective is to find a projection matrix P such that $x_i \simeq PX_i$.

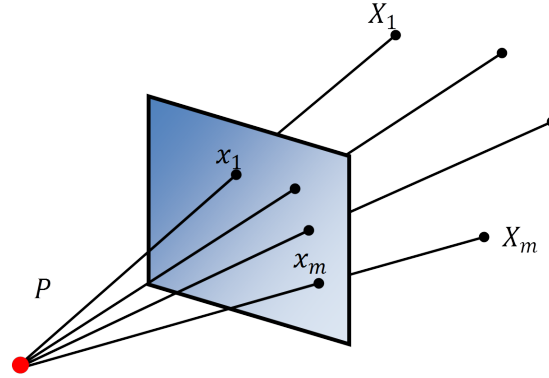


Fig. A.3 Illustration of the problem of camera resectioning. The problem here is to find the 3×4 camera matrix P that projects 3D scene points X_i to images points x_i where $x_i \simeq PX_i$ for $i = 1, \dots, m$.

This problem is taken into consideration in our work due to its importance in recovering the camera rotations and translations, and also for intrinsic calibration parameters. This task requires correspondence between the 3D scene points $\hat{X} = [X, 1]^\top$ and their image points $\hat{x} = [x, 1]^\top = [x_i, y_i, 1]^\top$ for $i = 1, \dots, m$. The parameters of the 3×4 projection matrix P are:

$$P = \begin{bmatrix} p^1 \\ p^2 \\ p^3 \end{bmatrix} = \begin{bmatrix} p_1 & p_2 & p_3 & p_4 \\ p_5 & p_6 & p_7 & p_8 \\ p_9 & p_{10} & p_{11} & 1 \end{bmatrix}$$

Note that because the projection matrix P can only be recovered up to a scale factor, its last parameter has been set to one. The optimisation variable is then $x = (p_1, \dots, p_{11}) \in \mathbb{R}^{11}$.

Definition 8. The camera resectioning problem is defined as follow. Given a set of correspondences $\hat{X}_i \leftrightarrow \hat{x}_i$, for $i = 1, \dots, m$, find the 3×4 camera matrix P such that $\hat{x}_i = P_i \hat{X}_i$ for all i .

As in the triangulation problem, let $d(\hat{x}_i, P\hat{X}_i)$ be the distance between the projected 3D scene points via the camera P and the measured image points. Then the optimisation problem is:

$$\begin{aligned} \min_{\mathbf{x}} \quad & \max_i d(\hat{x}_i, P\hat{X}_i), \quad \text{for } i = 1, \dots, m. \\ \text{subject to} \quad & p^{3\top} \hat{X}_i > 0, \quad \text{for } i = 1, \dots, m. \end{aligned} \quad (\text{A.9})$$

Let the i^{th} error residual be $\varepsilon_i = d(\hat{x}_i, P\hat{X}_i)$, where $i = 1, \dots, m$. Note that m is the number of image points. Thus we have m residual errors $\varepsilon = (\varepsilon_1, \varepsilon_2, \dots, \varepsilon_m)$. Similarly to the triangulation problem, this problem is generally solved using linear or iterative optimisation techniques. In our work, we formulate it as a quasi-convex optimisation problem and we try to solve it using a sequence of SOCP feasibility problems. The optimisation here tries to minimise the maximum error between the projected points and the measured image points. Thus:

$$\begin{aligned} \min_{\mathbf{x}} \quad & f_0(\mathbf{x}) = \max_{i=1, \dots, m} f_i(\mathbf{x}) = \max_{i=1, \dots, m} \left(\frac{f_{i1}(\mathbf{x})^2 + f_{i2}(\mathbf{x})^2}{f_{i3}(\mathbf{x})^2} \right) \\ \text{subject to} \quad & f_{i3}(\mathbf{x}) > 0, \quad \text{for } i = 1, \dots, m. \end{aligned} \quad (\text{A.10})$$

where $f_{i1}(\mathbf{x}) = p^{1\top} \hat{X}_i - x_i p^{3\top} \hat{X}_i$, $f_{i2}(\mathbf{x}) = p^{2\top} \hat{X}_i - y_i p^{3\top} \hat{X}_i$ and $f_{i3}(\mathbf{x}) = p^{3\top} \hat{X}_i$. The recovered camera matrix is then the matrix P that minimises the norm of this error vector $\varepsilon = (\varepsilon_1, \varepsilon_2, \dots, \varepsilon_m)$. Again, according to Theorem 8 (page 70) and Theorem 7 (page 61) in Chapter 3, the optimisation problem A.10 is quasi-convex. Hence, a sequence of feasibility problems can be used.

A.2.1 Experiments

To show the deployment of quasi-convex optimisation and the L_∞ norm for solving the camera resectioning problem, we used data from the corridor sequence [203]. This dataset has 11 frames, so we can recover 11 camera matrices P_j . In each frame, correspondences between 3D scene points \hat{X}_i and image points \hat{x}_i are recovered. We then used these point correspondences, $\hat{X}_i \leftrightarrow \hat{x}_i$, to estimate the camera matrix P_j for each view.

In this problem, we compare the convex optimisation performance of the L_∞ re-projection error norm and the RMS errors. Figure A.4 shows that the projection errors using the RMS errors norm are relatively better (errors vary between 0.42

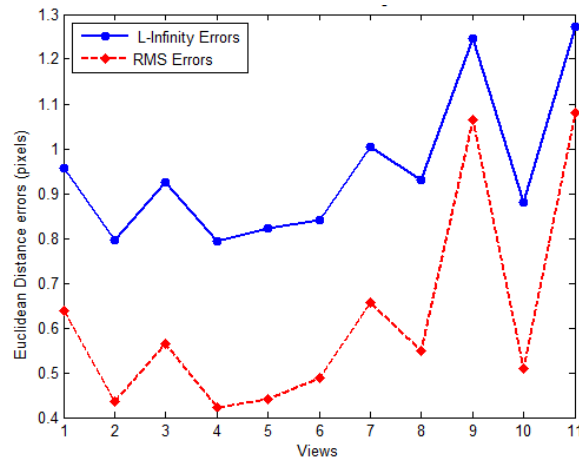


Fig. A.4 Camera resectioning problem - The RMS errors and L_∞ errors for the 11 camera matrices estimation.

and 1.1 pixels) than those with the L_∞ re-projection error norm where errors were between 0.8 and 1.2 pixels.

A.3 Homography estimation

In the homography problem we wish to estimate the eight unknown parameters of a transformation matrix \mathcal{H} , called the Homography matrix, based on known planar point correspondences (Figure A.5) [82, 94]. Thus, if we have planar 3D scene points \hat{X}_k projected to image I as \hat{x}_k^i and to image J as \hat{x}_k^j , then the correspondences $\hat{x}_k^i \leftrightarrow \hat{x}_k^j$ are related by the 3×3 homography matrix \mathcal{H} , where $\hat{x}_k^j \simeq \mathcal{H}\hat{x}_k^i$, for $k = 1, \dots, m$. Note that the vectors \hat{x}_k^i and $\mathcal{H}\hat{x}_k^j$ are only equal up to a scale factor (Figure A.5). Therefore, our aim is to find the parameters of \mathcal{H} :

$$H = \begin{bmatrix} h^1 \\ h^2 \\ h^3 \end{bmatrix} = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & 1 \end{bmatrix}$$

Note again that because the homography matrix \mathcal{H} can only be recovered up to a scale factor, its last parameter has been set to one. Hence, the optimisation variable will be then $x = (h_1, h_2, \dots, h_8)^\top \in \mathbb{R}^8$.

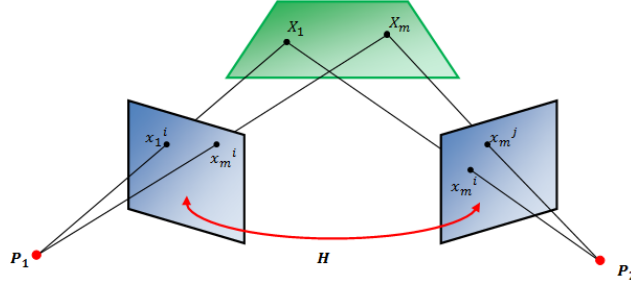


Fig. A.5 Homography Estimation. The task is to find 3×3 Homography matrix \mathcal{H} such that $\hat{x}_k^j \simeq \mathcal{H}\hat{x}_k^i$ for $k = 1, \dots, m$.

Definition 9. The homography estimation problem is defined as follows: given a set of point correspondences $\hat{x}_k^i \leftrightarrow \hat{x}_k^j$ for $k = 1, \dots, m$, find \mathcal{H} such that $\hat{x}_k^j \simeq \mathcal{H}\hat{x}_k^i$ for all k .

Similarly to the triangulation and camera resectioning problems, let $d(\hat{x}_k^j, \mathcal{H}\hat{x}_k^i)$ be the distance between the projected 3D scene points and the measured image points. Then, the optimisation problem can be formulated as:

$$\begin{aligned} \min_{\mathbf{x}} \quad & \max_i d(\hat{x}_k^j, \mathcal{H}\hat{x}_k^i), \quad \text{for } k = 1, \dots, m. \\ \text{subject to} \quad & h^{3\top} \hat{x}_k^i > 0, \quad \text{for } k = 1, \dots, m. \end{aligned} \quad (\text{A.11})$$

Let the k^{th} error residual be $\varepsilon_k = d(\hat{x}_k^j, \mathcal{H}\hat{x}_k^i)$ where $k = 1, \dots, m$ and $\hat{x}_k^i = (x_k^i, y_k^i, 1)$ (similarly $\hat{x}_k^j = (x_k^j, y_k^j, 1)$). Note that m is the number of image points (Figure A.5). We therefore have then m residual errors $\varepsilon = (\varepsilon_1, \varepsilon_2, \dots, \varepsilon_m)$. The estimated homography is the matrix \mathcal{H} that minimises the norm of this error vector.

As in previous problems, this optimisation problem is generally solved using linear or iterative optimisation techniques. In our work, we try to solve it using a sequence of SOCP feasibility problems after formulating it as a quasi-convex optimisation problem. Thus:

$$\begin{aligned} \min_{\mathbf{x}} \quad & f_0(\mathbf{x}) = \max_{k=1, \dots, m} f_k(\mathbf{x}) = \max_{k=1, \dots, m} d(\hat{x}_k^j, \mathcal{H}\hat{x}_k^i) = \max_{k=1, \dots, m} \left(\frac{f_{k1}(\mathbf{x})^2 + f_{k2}(\mathbf{x})^2}{f_{k3}(\mathbf{x})^2} \right) \\ \text{subject to} \quad & f_{k3}(\mathbf{x}) > 0, \quad \text{for } k = 1, \dots, m. \end{aligned} \quad (\text{A.12})$$

where $f_{k1}(\mathbf{x}) = h^{1\top} \hat{x}_k^i - x_k^j h^{3\top} \hat{x}_k^i$, $f_{k2}(\mathbf{x}) = h^{2\top} \hat{x}_k^i - y_k^j h^{3\top} \hat{x}_k^i$ and $f_{k3}(\mathbf{x}) = h^{3\top} \hat{x}_k^i$.

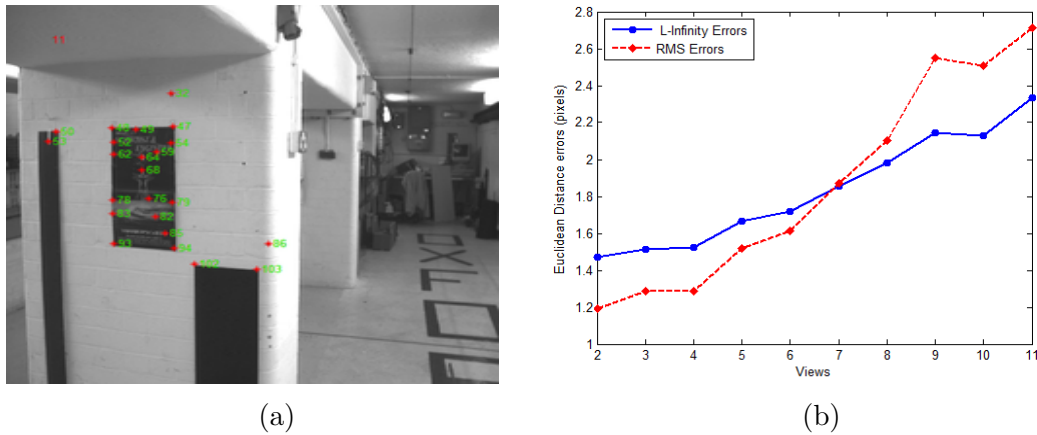


Fig. A.6 The RMS and L_∞ errors for the 10 homographies estimated using convex optimisation. Left image shows 23 planar image points used to estimate homographies between consecutive pairs in the corridor dataset.

A.3.1 Experiments

The corridor sequence is again used in this problem exploiting from 23 points on the left frontal wall (Figure A.6a). For each pair of consecutive frames (I_i and I_{i+1}), correspondences between image points $\{\hat{x}_k^i \leftrightarrow \hat{x}_k^{i+1}\}$ are recovered. These image points are images of 3D scene points that lie on the left frontal wall. These image points $\{\hat{x}_k^i\}$ and $\{\hat{x}_k^{i+1}\}$ are used to then estimate the homographies between the corresponding frames. The corridor dataset includes 11 frames, which means that we have 10 homographies to estimate. In this problem, we compared the convex optimisation performance between the L_∞ -reprojection error norm and the RMS errors. As shown in Figure A.6b, the L_∞ method performs better when adopting the L_∞ error while using more image correspondences.

Appendix B

The KD-tree Data Structure

In order to have a clear idea about the KD-tree data structure, let us consider the following example. Suppose that we want to build a search tree out of some points $\mathfrak{E} = (a, \dots, i)$, where the aim is to recover the nearest neighbour point to a query point p as illustrated in Figure B.1.

B.1 Construction of a KD-tree:

A way of constructing the KD-tree is illustrated in Figure B.2. In this case, we have two dimensions x and y , and the set of points $\mathfrak{E} = (a, \dots, i)$. First, we sort the points in each dimension, and then we divide the points perpendicular to the axis with widest spread. In two dimensions, we can split the plane into two regions by drawing a line across one axis. In three dimensions, we could partition space into two regions by drawing a plane, then taking the regions above and below the plane as the two half-regions. When working with KD-trees, one often uses the term *splitting hyper-plane* to refer to the object that splits space in half. This technique is known as *binary space partitioning* (since each step splits space into two regions).

In our two-dimensional illustrative example, the x axis holds the widest spread as shown in Figure B.2a. Then, we sort again the points in one side and split them into two sub-sets as shown in Figure B.2b. We recursively build a KD-tree in each half-space by selecting the axis with the widest point spread and splitting the data through it, as shown in Figure B.2b and Figure B.2c. If we continue this construction to completion, our resulting KD-tree will look like the tree in Figure B.2d. As we can see in this figure, each node in this tree has a *splitting axis*, a *splitting value*

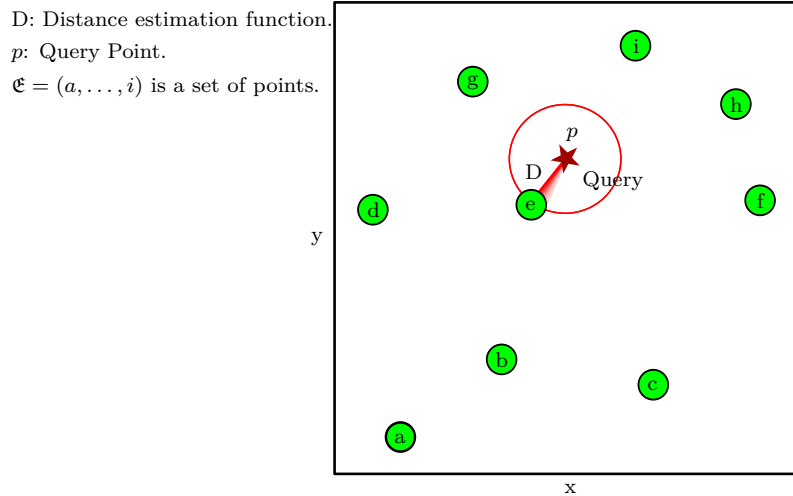


Fig. B.1 Nearest Neighbour Search illustration. The red star refers to the query point, and the green circles refer to the set of points.

(S_1, \dots, S_8) , and a left and right sub-tree. If the left and right children are null, a node will represent a point.

B.2 Nearest-neighbour search in KD-trees

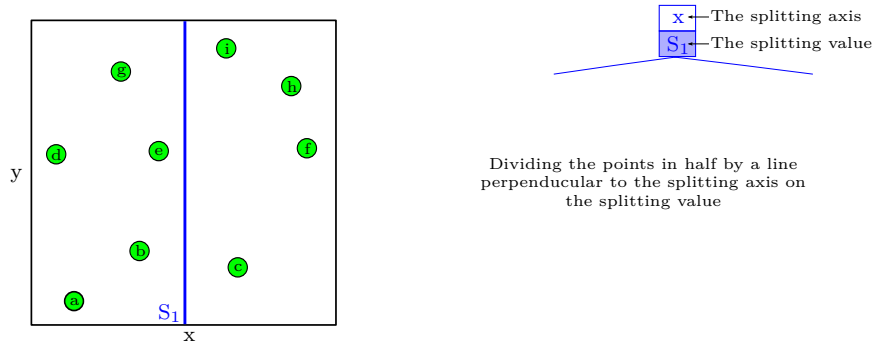
This structure has been widely used in the computer vision community. The most interesting operation on the KD-trees is the Nearest-Neighbour Search (NNS). KD-tree structures are known for their abilities to reduce the search time from linear to logarithmic.

For the nearest neighbour search, where a distance estimation function D is given (also known as dissimilarity measure), the aim is to extract the nearest elements to the query point p in a KD-tree B containing a collection of points (Figure B.1). If the nearest neighbour to p is Q , then the mathematical formulation of this constraint is given as [17]:

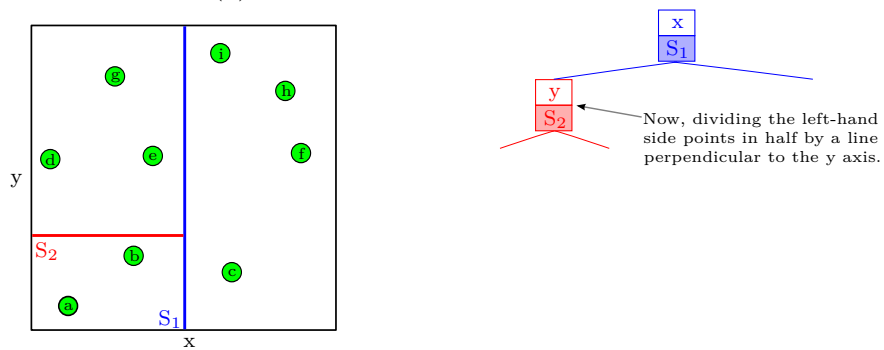
$$(\forall R \in B, \{R \neq Q\} \Rightarrow [D(R, p) \geq D(Q, p)]) \quad (\text{B.1})$$

This constraint can be applied as well for the m nearest neighbours to p .

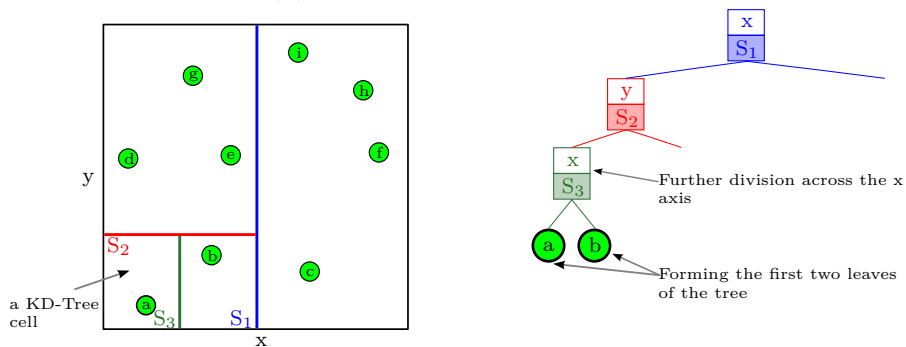
To illustrate the Nearest-Neighbour Search (NNS) in a KD-tree, let us consider again our example of the set of point $\mathfrak{E} = (a, \dots, i)$ and the constructed KD-tree shown in Figure B.2. Now, our scenario is as follow: given a KD-tree and a point in



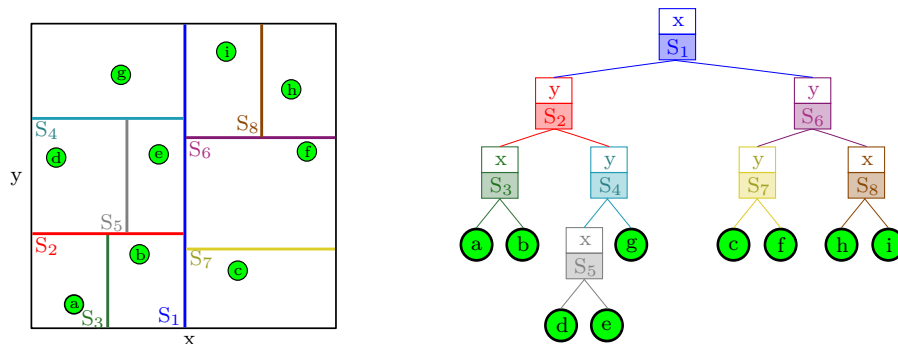
(a) First division across the x axis.



(b) Second division across the y axis.



(c) Further division.



(d) The final KD-tree after recursively constructing it for the two sets of points.

Fig. B.2 Illustration of a KD-tree construction.

space p (called the Query Point), which point in the KD-tree is closest to p ? (The point in the data set closest to the query point is called its nearest neighbour).

In our scenario, which is depicted in Figure B.1, the query point p is marked as a red star. The first operation to be performed is a recursive search to find the point in the same cell as the query point. To do that, let us consider the Figure B.3a. By checking the splitting value S_1 and the splitting axis x in the root of the tree with the query point, we can conclude that the cell that contains the query point must be in the left-hand side sub-tree (shaded in yellow). Applying the same principle to left-hand side sub-tree, by checking the splitting values S_2 and S_4 , we can understand that the query point belongs to cell where the point (g) is (shaded in cyan in Figure B.3b). The point (g) will be selected as the nearest neighbour at this moment.

This first point is not necessarily the nearest neighbour, but at least we know that any potential nearer neighbour must lie closer, and so must lie within the red circle centred on the query point and passing through the point (g) (Figure B.3b). Although in this example this region is a circle, in three dimensions it would be a sphere, and in general it is called the candidate hyper-sphere.

We now back up to the parent of the current node (point (g)), and we check whether it is possible for a closer solution to that so far found to exist in this parent's other sub-trees. This is done by checking the splitting value S_5 (Figure B.3c). Point (e) is found to be closer than the nearest neighbour so far (g) . Then, obviously the point (e) will be declared as the new nearest neighbour. Point (d) and the sub-tree through the splitting value S_3 (points (a) and (b)) will be discarded since they have greater values than the splitting values S_5 and S_2 respectively.

Then, we recursively check the right sub-tree of the splitting value S_1 , and we check whether each node is closer than the nearest neighbour we have so far. Figure B.3d illustrates the final search results. Blue nodes represent points that have been checked for potential nearer neighbours. Grey points characterise the discarded nodes.

B.3 Computational complexity

The average running times in a tree of n nodes is $\mathcal{O}(\log n)$ for the insertion and deletion tasks, and for the nearest neighbour search as well [17, 113]. Storage for the KD-tree is $\mathcal{O}(n)$. These performances overcome by far the known existing search algorithms [113]. That's why KD-tree data structure have been widely used in applications where computational time is very important. In vision systems,

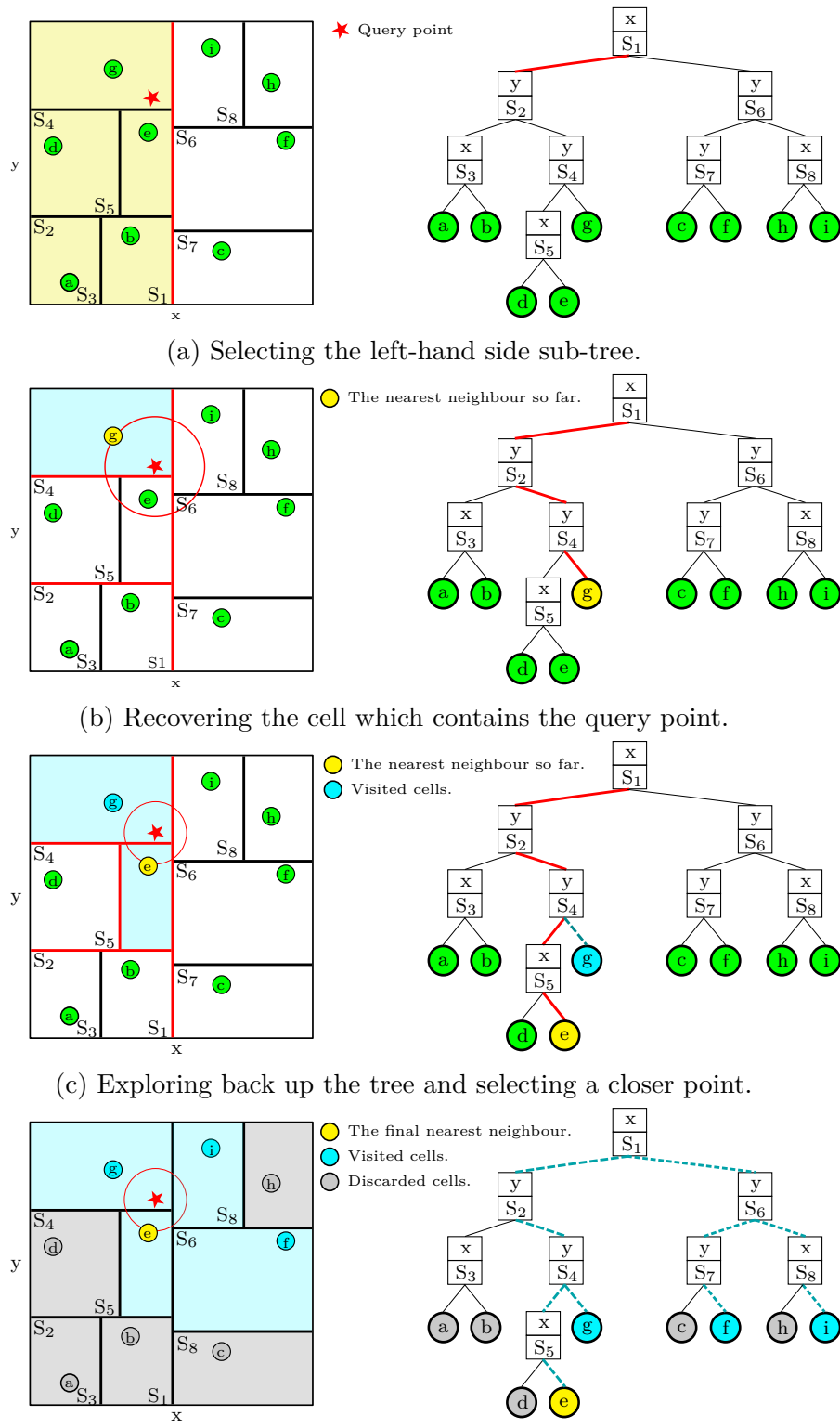


Fig. B.3 Illustration of Nearest Neighbour Search in a KD-tree.

these structures are started to broadly extent for efficient nearest neighbour search [136, 177].

Appendix C

The Fundamental Matrix and Image Interest Point Extraction

C.1 The Fundamental matrix

The fundamental matrix, F , is an algebraic representation of epipolar geometry in two-view geometry.

Given two views with camera matrices P_i and P_j , a pair of matching image points $x'_j \leftrightarrow x'_i$ must satisfy: $x'_j{}^\top F x'_i = 0$. It is easy to see that the essential matrix is the specialisation of the fundamental matrix to the case of calibrated cameras. That is, when the calibration matrix K is known. Historically, the essential matrix was introduced to the computer vision community by [117] before the fundamental matrix [82], but commonly the latter matrix is estimated first and then the essential matrix is deduced from it using (2.43) (Chapter 2, page 42).

The geometric interpretation of the fundamental matrix is illustrated in Figure C.1. First, note that $e_i \in \mathbb{R}^3$ is a point where the baseline crosses the image plane. This point is called the epipole (e_j in the second image plane in Figure C.1). In this figure, x'_i and x'_j are the image points of a 3D scene point X . One may notice that point x'_i may be mapped onto a line l_j in the second view. This important line is called *the epipolar line* since it intersects the epipole:

$$x'_j{}^\top F x'_i = x'_j{}^\top l_j = 0 \quad (\text{C.1})$$

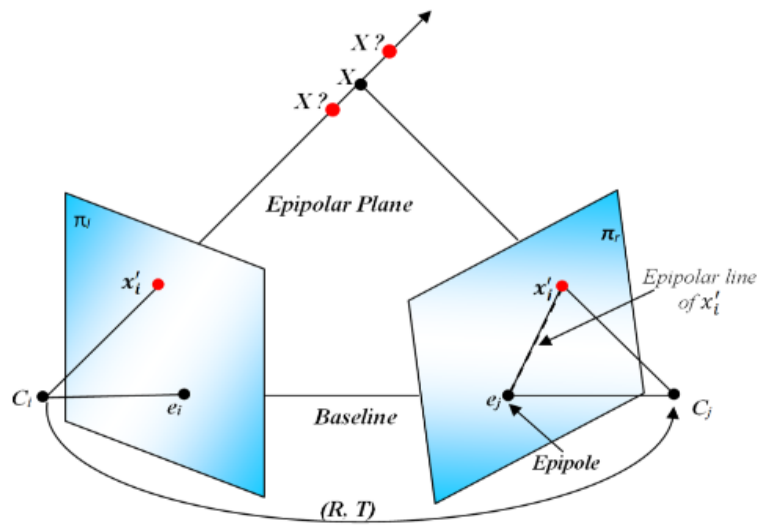


Fig. C.1 The epipolar geometry

Since the search area is reduced to a line, this epipolar geometry property will improve the matching task between image features. Figure C.2 shows an example of image point matching using epipolar lines. From (C.1) we can see that the mapping of a point via the matrix F to an epipolar line is given by:

$$Fx'_i = l_j \quad (\text{C.2})$$

Estimating F

In general, the fundamental matrix is recovered from point correspondences between images [42, 81, 121, 129, 217]. Since it is defined to a scale factor, it can be estimated from no more than eight correspondences [189]. Similarly to the essential matrix in the calibrated case, the fundamental matrix can be estimated using linear techniques only. Let us consider the epipolar constraint:

$$x'_j{}^\top Fx'_i = (x'_j, y'_j, z'_j)^\top \begin{bmatrix} f_1 & f_2 & f_3 \\ f_4 & f_5 & f_6 \\ f_7 & f_8 & f_9 \end{bmatrix} (x'_i, y'_i, z'_i) = 0 \quad (\text{C.3})$$

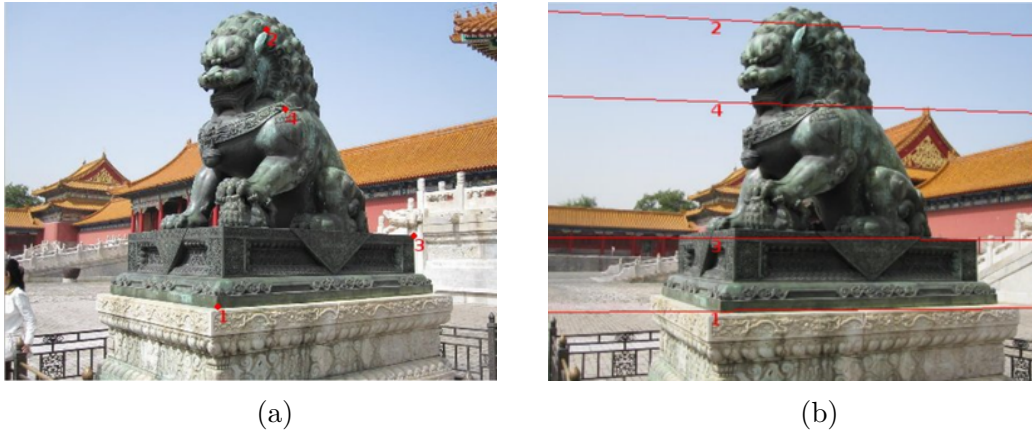


Fig. C.2 Example of image point correspondences using the epipolar constraints.

Let the vector $\mathbf{f} \in \mathbb{R}^9$ contain the elements of the fundamental matrix:

$$\mathbf{f} = (f_1, f_2, f_3, f_4, f_5, f_6, f_7, f_8, f_9)^\top \in \mathbb{R}^9 \quad (\text{C.4})$$

The epipolar geometry constraint (C.3) can be formulated as the inner product of \mathbf{A} and \mathbf{f} . This leads to the linear equation in the entries of \mathbf{f} :

$$\mathbf{A}\mathbf{f} = 0 \quad (\text{C.5})$$

where $\mathbf{A} \in \mathbb{R}^{n \times 9}$ can be constructed from n image points correspondences. At least eight such correspondences are required to solve for \mathbf{f} to a scale. Similar techniques to those used in estimating the essential matrix can be used to solve (C.5) for the vector \mathbf{f} . Therefore, the SVD can be used, where the solution is the smallest singular value of \mathbf{A} . Matrix \mathbf{F} is constrained to have a rank of 2. To enforce that we can decompose it using SVD and then put the third non-zero singular value to zero, i.e. $\sigma_3 = 0$.

However, we again incur false correspondences between image points, also known in computer vision as outliers. The latter could significantly affect the accuracy of the estimate of the \mathbf{F} matrix. This is due to some noisy measurements and to the inaccuracy inherent in matching algorithms. Thus, the problem now is how to solve this parameter estimation in the presence of outliers? One way to get rid of false correspondences and recover a dominant estimate of \mathbf{F} is by using an iterative outlier removal algorithm such as RanSaC (RANdom SAmple and Consensus)[58] (Algorithm 2). The output of the RanSaC algorithm is a dominant \mathbf{F} matrix and set of inliers of image point correspondences.

Algorithm 2 RanSaC algorithm for fundamental matrix estimation.

Objective: Reject correspondence outliers and estimate a dominant fundamental matrix, F .

Output: Dominant F with a set of consistent image point matches (inliers).

Given:

- Initial set of n image point matches $(x_i^k \leftrightarrow x_j^k)$, $k = 1, \dots, n$ ($n \geq 7$);
- Error threshold ε (to decide whether a point is an inlier or not);
- Maximum number of iterations i_{max} .

$i := 1$;

Stop = false;

while Stop is false and $i < i_{max}$ **do**

$i := i + 1$;

- Select m random image points out of n initial image point matches ($m = 7$ or $m = 8$);
- Compute the fundamental matrix from the m randomly selected image points;
- Classify (x'_i, x'_j) inliers/outliers among n points as a function of the distance of x'_j to epipolar line associated with x'_i with precision ε ;

if the number of inliers is larger than the most coherent one so far **then**

keep it and re-estimate F using this set;

Stop = true;

end if

end while

In this thesis, we will be investigating the introduction of optimisation techniques in motion estimation for computer vision tasks. Given that the motion estimation steps rely heavily on the estimation of this matrix, a premium need to be placed on recovering its parameters. Indeed, optimally estimating the fundamental matrix is a hard task since the data point locations are always corrupted with noise and the correspondences are spoilt by outliers [42]. RANSAC [58] is a well-known robust statistics solution for these types of problems. Unfortunately, though RANSAC and similar solutions are able to detect outliers, the inaccuracy in the image point location is still not estimated. In reality due to the noise, the measured points x'_i and x'_j do not satisfy the epipolar constraint $x'^{\top}_j F x'_i = 0$, therefore, a non-linear optimisation solution could be necessary. The relative objective function for this task is usually defined as:

$$\begin{aligned} \min_F \quad & \min_{\hat{x}} \left[d_1(x'_i, \hat{x}'_i)^2 + d_2(x'_j, \hat{x}'_j)^2 \right] \\ \text{subject to} \quad & \hat{x}'^{\top}_j F \hat{x}'_i = 0 \end{aligned} \quad (\text{C.6})$$

In this optimisation problem, we try to find \hat{x}'_i and \hat{x}'_j that satisfy this constraint and minimise the distances to the measured points x'_i and x'_j (Algorithm 3)

The notation $d(*)$ stands for the Euclidean distance and \hat{x}'_i and \hat{x}'_j are the true correspondences that satisfy the constraint $\hat{x}'^{\top}_j F \hat{x}'_i = 0$ [82]. This optimisation problem can be solved using the Levenberg-Marquardt (LM) algorithm. LM may, however, easily get trapped in a local minimum. In addition, it is sensitive to initial estimates. We will discuss this issue in the coming chapters.

Algorithm 3 Algorithm for fundamental matrix estimation with Ransac.

Objective: Estimate an accurate fundamental matrix F using image points.

Extract image points in each image, I and J ;

for $i := 1$ to number of image points in image I **do**

Find corresponding image points in image J ;

end for

- Perform outlier removal using RanSaC (Algorithm 2);
 - Perform non-linear optimisation by minimising the objective function given in (C.6).
-

Algorithm 4 The eight-point algorithm.

Given: a number of corresponding image points ($\mathbf{x}_i^k \leftrightarrow \mathbf{x}_j^k$), $k = 1, \dots, n$ ($n \geq 8$).

Objective: Find (R, T) with $R \in SO(3)$ and $T \in \mathbb{R}^3$ which solves:

$$\mathbf{x}_j^{k\top} [T]_{\times} R \mathbf{x}_i^k = \mathbf{x}_j^{k\top} E \mathbf{x}_i^k = 0, \quad k = 1, \dots, n$$

$k := 1$;

while ($k \leq n$) **do**

 Build the matrix $A \in \mathbb{R}^{9 \times n}$ as in (2.29).

$k = k + 1$;

end while

- Solve equation (2.36), $A\mathbf{e} = 0$, by computing the SVD of $A = U\Sigma V^{\top}$. Then, \mathbf{e} is the 9th column of V .
 - Reorganise \mathbf{e} into a square 3×3 matrix E .
 - Compute the SVD of the recovered $E = U_E \Sigma_E V_E^{\top}$, where $U_E, V_E \in SO(3)$, $\Sigma_E = \text{diag}(\sigma_1, \sigma_2, \sigma_3)$ and $\sigma_1 \geq \sigma_2 \geq \sigma_3$.
 - Compute the projection of E onto the essential space as $E = U_E \text{diag}(1, 1, 0) V_E^{\top}$.
 - Extract R and T from E using Theorem 3 (Chapter 2, Section 2.9.2, page 41), taking into consideration the positive depth constraint.
-

C.2 Image interest point extraction

In visual autonomous navigation systems, image points (or usually called *features*) are of great interest in estimating the camera translations and the rotations. In fact, for any computer vision problem, recorded images are the main source of information. Indeed, extracting these image points is the first step for any algorithm. Matching and tracking these image points is not less important than the extraction task, especially for problems of registration and recognition [106]. Having images in hand as inputs, the first thing to look for is points that can be recognised in other images. Therefore, interest points like corners and edges seem to be valid options. These image points are commonly called image features in computer vision. Other names such as keypoints, interest points, corners, control points and edges are used as well. Other tasks in computer vision apart from motion estimation such as object classification and recognition, use image features in their algorithms [27, 119, 218] as well.

The notion of feature space is important in the problem of feature extraction. It defines the nature of the information being extracted for a particular feature extraction algorithm. This varies from pixel values to other properties such as corners, edges or line intersections [27]. Large patches in images can also be used as image features and can be matched using area-based techniques. This method is

usually used with images with less texture or with poor content, i.e. medical images [27, 124, 214]. The reader may consult a rich survey on feature extraction in [218].

Many algorithms have been proposed in the literature during the last few decades to improve interest point extraction from images. One common way is to use a corner detection technique [158, 182]. One of the pioneer methods is the feature tracking technique which is known as the Lucas-Kanade iterative registration algorithm [120] and Kanade-Lucas-Tomasi (KLT) tracker [196]. Detectors based on intensity include the Harris feature detector, which is the most famous algorithm in computer vision. More advanced approaches used in computer vision are SURF (speeded up robust features) [15] and SIFT (scale invariant feature transform) [119]. These techniques are better at handling change in scale and other geometric transformations such as affine rotation. In our work, we used three detectors. The first one is the Harris feature detector, the second one is the SIFT detector; and the third one is local colour histograms. These detectors are detailed in the following sections.

C.2.1 Harris feature detector

The Harris corner detector is the most popular algorithm in computer vision. For detection, this algorithm relies on the second order derivative matrix, also known as the second moment matrix, constructed from intensity values. This detector is known as a gradient-based detector. The idea of extracting features is simple but powerful. This detector looks for pixel intensity variation over a region of an image. If this variation is greater than a predefined threshold, t , the area is declared as a feature. This technique was first used by Moravec in [132], then it was improved upon and presented by Harris and Stephen [79]. The new version uses a local gradient distribution represented as a second moment matrix A (Equation C.8). A feature is defined as an edge, a corner or a clear flat region according to the eigenvalues of A . Let λ_1 and λ_2 be the eigenvalues of matrix A , then:

- If both eigenvalues λ_1 and λ_2 are small, the area is flat. This means there is no important variation in the gradient in both directions;
- If one of the eigenvalues is high and the other one is low, then area refers to an edge. This describes a significant variation in just one direction;
- If both eigenvalues λ_1 and λ_2 are high, then the area is a corner. This describes a significant variation in both directions.

One of the major improvement proposed by Harris to complement Moravec's previous work [132] is the introduction of the corner-less function C_m :

$$C_m = \det(A) - k \cdot \text{tr}^2(A) \quad (\text{C.7})$$

Matrix A is the second moment matrix and tr is the trace. The matrix A is given by:

$$A = \sum_l \sum_m w(, m) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \quad (\text{C.8})$$

where I_i is the derivative in the direction i ; w is a weighted Gaussian windows of size $l \times m$; k is a constant chosen heuristically, usually between 0.04 and 0.15 [28].

Harris corner detector is considered a robust algorithm, especially in terms of invariant detection for translation and rotation. In comparison to its competitors in terms of time consumption, the Harris corner detector is a lot better. Correspondences from Harris features between two images are recovered by looking for a maximum correlation within windows surrounding each feature. Features with strong correlation with each other in both directions are considered as matching. This technique in general is good for matching. However, in some situations it is not robust enough [118]. Its drawbacks are especially in matching images where scale and/or rotation is the main transformation.

One solution to get rid of this matching problem is to use information about the surroundings of the features. Not just intensity information but other geometric information that describes the location or the area where the feature is detected. The SIFT and SURF extractors fall into this category, where a descriptor strategy is proposed.

C.2.2 Scale invariant feature transform (SIFT)

New algorithms that are not only able to detect points in an image but also interest regions [119] were developed to tackle problems with the Harris corner detector, such as invariance to scale and rotation. These regions of interest represent in general areas that are brighter or darker than the surroundings [215]. Popular region detectors are SIFT [119] and SURF [15]. In this thesis, we employed the SIFT feature detector due to their useful properties related to robustness to variation in scale, partial illumination, affine transformations and to some additive noise. Commonly, SIFT features are called *keypoints* in the SIFT framework.

The SIFT algorithm is described through five main steps:

- **Scale-space extrema detection:** First, the scaling is performed via smoothing the images using Gaussian masks resulting in a pyramid of the image. Subtracting adjacent smoothed images will result in what is called: Difference of Gaussian (DoG) pyramids. Each pixel in the current scale is compared with the upper and lower scales and with its neighbours (for example a region of 3×3 pixels provides comparison with 26 pixels which, i.e. 9 in the lower scale, 9 in the upper scale and the 8 surrounding pixels), and then the extrema value is estimated.
- **Accurate keypoint localisation:** The previous step produces too many keypoints candidates. Keypoints smaller than a threshold value are rejected. This procedure ensures discarding keypoints that are poorly localised along an edge or with low contrast.
- **Orientation assignment:** Each keypoint is associated to an orientation based on the orientation of pixel at its scale and the gradient magnitude. In total, 36 bins covering 360 degrees are used and represented via an orientation histogram. The highest peak of the histogram is taken along with any peak above 80% of the winning bin. Therefore, the outcomes of this step are keypoints with the same location and scale, but different directions. This contributes to stability matching.
- **Keypoint descriptors:** First, orientation histograms are created on a 4×4 pixel neighbourhood with 8 bins each for each keypoint. This makes descriptor vectors $\delta \in \mathbb{R}^{128}$ of 128 elements containing information about the keypoint neighbourhood orientation and gradient magnitude.

Keypoints in the SIFT framework are matched by identifying their nearest descriptor neighbours. Let us consider two images I and J . Image I is described with n_i keypoints with descriptors $\delta_I^i \in \mathbb{R}^{128}$, where $i = 1, \dots, n_i$. Similarly, image J is described with n_j keypoints with descriptors $\delta_J^j \in \mathbb{R}^{128}$, where $j = 1, \dots, n_j$. The main steps to match a feature δ_I^k , where $1 \leq k \leq n_i$, from image I to a feature in image J are:

- Compute the Euclidean distance between descriptor δ_I^k to all descriptors δ_J^j where $j = 1, \dots, n_j$:

$$\mathbf{d}^k = (d_1^k, d_2^k, \dots, d_{n_j}^k) \quad (\text{C.9})$$

$$d_j^k = \|\delta_I^k - \delta_J^j\| = \sqrt{(\delta_I^k - \delta_J^j)^\top (\delta_I^k - \delta_J^j)} \text{ for } j = 1, \dots, n_j \quad (\text{C.10})$$

- Arranging the vector \mathbf{d}^k into a vector ξ in ascending order:

$$\xi = (\xi_1, \xi_2, \dots, \xi_{n_j}) \quad (\text{C.11})$$

where $\xi_1 = \min(\mathbf{d}^k)$ and $\xi_{n_j} = \max(\mathbf{d}^k)$.

- The feature with descriptor δ_I^k is matched to the feature with descriptor δ_J^l if $\xi_1 \leq \kappa \cdot \xi_2$, $\xi_1 = \min(\mathbf{d}^k)$ and $\xi_1 = \|\delta_I^k - \delta_J^l\|$, where κ is the matching threshold. This means that when the ratio between the Euclidean distance to the closest neighbour and to the second closest neighbour is less than the threshold, a correct match is obtained.

Many studies have been conducted on local shape descriptors. Mikolajczyk and Schmid in [128] summarised their extensive study with the conclusion that the SIFT descriptors perform best and outperform all similar descriptors. Indeed, the SIFT detector is robust to scale and rotation changes. However, it is only partially robust to illumination variations, since the correspondences are based on the structure of the image gradient and not on the brightness of the keypoints. This fact induced other studies to look for alternatives, especially in less-structured locations. One of these is colour features.

C.2.3 Colour features

The value of colour information quality is commonly agreed in upon computer vision community, due to its valuable ability to describe the world around us. However, most local feature spaces in the literature are based on shape description only and attach less importance to colour information [207]. Describing images with local colour has received little consideration. Consequently, the vast majority of methods use only luminance and dismiss colour space. Therefore, considering augmenting the image description with colour space would be a valid option.

Early research programmes focused on methods that are invariant to illumination changes, where objects are described by colour histograms [12, 57, 63, 70, 71]. Ongoing researches still investigate indexing methods that are both invariant to illumination changes and shading using lighting geometry. Some techniques tried to concatenate local feature descriptors, like the SIFT descriptor, with colour descriptors, which result in a new extended feature space [128, 207].

Colour features use histograms to represent the distribution of colours in a particular image. Basically, they count similar pixels and store them in appropriate bins in order to describe the number of pixels in each range of colours (or bin)

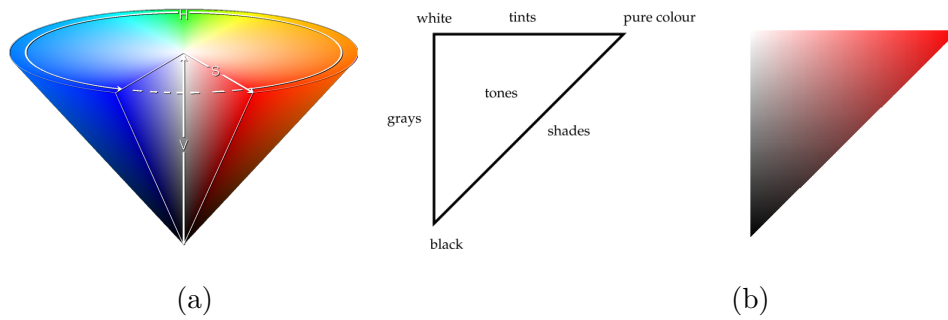
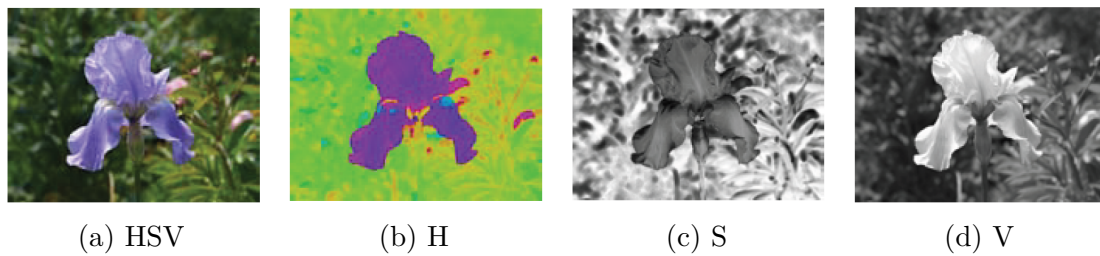


Fig. C.3 Hue Saturation Value (HSV). The colour wheel at the top includes all the pure colours. The angle in this wheel is represented by the H parameter [93].



(a) HSV

(b) H

(c) S

(d) V

Fig. C.4 Example of a colour image and its HSV representation [190].

independently. Local colour histograms capture information about distribution of colour in particular areas in the image.

Colours are commonly represented with the RGB model. An additional system to characterise colours is in terms of the Hue Saturation Value (HSV) model. More techniques have started to incorporate local colour histograms using this colour space. This space is used in our work as well. Therefore, it is worthwhile to present some details about it. Hue Saturation Value (HSV) is a colour model that describes colours (hue or tint) in terms of their shade (saturation or amount of gray) and their brightness (value or luminance). Figure C.3 shows a representation of the HSV model. Hue (H) is an angle from 0 degrees to 360 degrees. Figure C.3 gives a representation of the Hue Saturation Value (HSV) model as a colour cone. Figure C.4 shows a colour image with its HSV representation, in which the saturation S is encoded using a gray scale (saturated shown as darker).

Figure C.5 shows an example of the SIFT descriptor and the colour descriptor for a local patch in a colour image. Clearly, the SIFT descriptor deals with the orientation of the edges. The Hue descriptor on the bottom line describes the hue and the saturation at each position represented as vectors. As we mentioned before, the hue defines the vector angle and the saturation defines its length.

Similarly to most descriptor-based approaches, matching in colour descriptor space is usually based on the nearest neighbour in the descriptor space. Commonly descriptors are compared with the Euclidean distance [128][111].

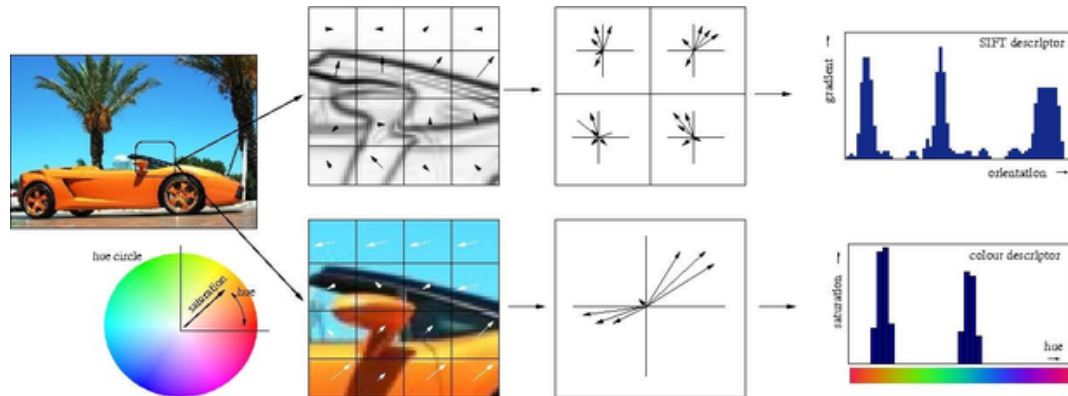


Fig. C.5 SIFT and colour descriptors. The same patch is described using SIFT descriptor space (top row) and colour descriptor space using the HSV model (bottom row). Image source [207].

C.3 Feature position uncertainties

We have seen that feature extraction is of great importance in computer vision where most applications rely on them, such as pose estimation and object detection. More works have been conducted on feature stability and its geometric and photometric robustness. State-of-the-art techniques extract accurate and robust features that are invariant to scale and rotation changes. However, less attention is given to their location errors, which are commonly assumed to be uniform or insignificant [215]. Before starting the subsequent tasks such as motion estimation, more care should be taken over the position of these features. Questions should be raised about whether their locations are correctly estimated. If they are not, how far are they from the correct positions? And how can one evaluate or estimate these errors.

It is known that most feature extraction techniques rely heavily on the variation in intensity. Therefore, there must be a relationship between their location accuracy and the variation in intensity within their neighbourhood. In fact, the problem of feature uncertainties has already been addressed outside of the computer vision community, such as in microscopy[126], astronomy [163] and photogrammetrics [195]. In such domains, feature position precision is paramount and the error sources are considered to originate from the image formation and feature extraction process.

In computer vision community, some studies have dealt with this subject [26, 77, 90, 99, 151, 185, 212]. However, most researchers avoid uncertainty due to the added complexity in constructing the robust optimisation model and to the lack of knowledge of the nature of these uncertainties, especially their propagation. On the contrary, in our work, feature uncertainties and their propagation through multiple view geometry algorithms are explicitly taken into consideration and are included in our optimisation framework. Details on uncertainty estimation approaches and their propagation are given in the following chapters.

References

- [1] OptiTrack: Motion Capture Systems. URL <http://www.optitrack.com/>.
- [2] Two-view motion analysis: a unified algorithm. *Journal of the Optical Society of America. A, Optics and image science*, 3(9):1492–1500, 1986. ISSN 1084-7529.
- [3] Nemra Abdelkrim, N. Aouf, A Tsourdos, and Brian White. Robust nonlinear filtering for INS/GPS UAV localization. In *Control and Automation, 2008 16th Mediterranean Conference on*, pages 695–702, 2008.
- [4] Markus W. Achtelik, Stephan Weiss, Margarita Chli, Frank Dellaert, and Roland Siegwart. Collaborative stereo. In *IEEE International Conference on Intelligent Robots and Systems*, pages 2242–2248, 2011.
- [5] A.H. Sayed, A. Garulli, V.H. Nascimento, and S. Chandrasekaran. Exponentially-weighted iterative solutions for worst-case Parameter estimation. In *Proceedings of the Fifth IEEE Med. Conference on Control and Systems*, Cyprus, 1997.
- [6] Soren W and others Slama, Chester C ; Theurer, Charles ; Henriksen. *Manual of photogrammetry*. American Society of photogrammetry, ed. 4 edition, 1980.
- [7] Henrik Andreasson. Localization for Mobile Robots using Panoramic Vision , Local Features and Particle Filter. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE*, number April, pages 3348– 3353, 2005. ISBN 078038914X.
- [8] a. Angeli, D. Filliat, S. Doncieux, and J.-a. Meyer. Fast and Incremental Method for Loop-Closure Detection Using Bags of Visual Words. *IEEE Transactions on Robotics*, 24(5):1027–1037, October 2008. ISSN 1552-3098. doi: 10.1109/TRO.2008.2004514.
- [9] Adrien Angeli, Jean-arcady Meyer, and David Filliat. Real-Time Visual Loop-Closure Detection. In *International Conference on Robotics and Automation*, pages 1842–1847, 2008. ISBN 9781424416479.
- [10] Andrew Zisserman Antonio Criminisi, Ian Reid. A plane measuring device. *Image and Vision Computing*, 17(8):625–634, 1999.
- [11] Alexander Bahr, Matthew R. Walter, and John J. Leonard. Consistent cooperative localization. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 3415–3422, 2009.
- [12] D.H. Ballard. Generalizing the Hough transform to detect arbitrary shapes, 1981. ISSN 00313203.

-
- [13] T.D. Barfoot. Online visual motion estimation using FastSLAM with SIFT features. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 579–585. Ieee, 2005. ISBN 0-7803-8912-3. doi: 10.1109/IROS.2005.1545444.
- [14] T. Basar and P. Bernhard. H infinity optimal Control and Related Minimax Design Problems: A Dynamic Game Approach. *IEEE Transactions on Automatic Control*, 41(9), 1996.
- [15] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. *Computer Vision?ECCV 2006*, 2006.
- [16] Randal W. Beard, Timothy W. McLain, Derek B. Nelson, Derek Kingston, and David Johanson. Decentralized cooperative aerial surveillance using fixed-wing miniature UAVs. *Proceedings of the IEEE*, 94(7):1306–1323, 2006. ISSN 00189219.
- [17] Jon Louis Bentley. Multidimensional binary search trees used for associative searching, 1975.
- [18] Dimitris Bertsimas, DB Brown, and Constantine Caramanis. Theory and applications of robust optimization. *SIAM review*, pages 1–34, 2007.
- [19] Paul Besl and Neil McKay. A Method for Registration of 3-D Shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992. ISSN 0162-8828. doi: 10.1109/34.121791.
- [20] O. Boni, A. Ben-Tal, and A. Nemirovski. Robust solutions to conic quadratic problems and their applications. *Optimization and Engineering*, 9(1):1–18, May 2007. ISSN 1389-4420.
- [21] O Booij, B Terwijn, Z Zivkovic, and B Kr. Navigation using an appearance based topological map. In *IEEE International Conference on Robotics and Automation*, number April, pages 10–14, 2007. ISBN 1424406021.
- [22] M. Bosse, P. Newman, J. Leonard, M. Soika, W. Feiten, and S. Teller. An Atlas framework for scalable mapping. *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, 2, 2003.
- [23] M. Boulekchour and N. Aouf. L_∞ Norm Based Solution for Visual Odometry. In *CAIP13*, pages II:185–192, York, UK., 2013.
- [24] Mohammed Boulekchour and Nabil Aouf. Efficient real-time loop closure detection using GMM and tree structure. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2014)*, number Iros, pages 4944–4949, Chicago, USA, September 2014. Ieee. ISBN 978-1-4799-6934-0. doi: 10.1109/IROS.2014.6943265.
- [25] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004. ISBN 0521833787.
- [26] Michael J Brooks, Wojciech Chojnacki, Darren Gawley, Anton Van Den Hengel, and Ô Æ. What Value Covariance Information in Estimating Vision Parameters ? In *In proceedings IEEE international conference on computer vision*, 2001.

- [27] Lisa Gottesfeld Brown. A survey of image registration techniques, 1992. ISSN 03600300.
- [28] N. D B Bruce and Pierre Kornprobst. Harris corners in the real world: A principled selection criterion for interest points based on ecological statistics. In *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2009*, pages 2160–2167, 2009.
- [29] Martin Bujnak, Zuzana Kukelova, and Tomas Pajdla. A general solution to the P4P problem for camera with unknown focal length. In *26th IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2008.
- [30] W. Burgard, D. Fox, and S. Thrun. Active mobile robot localization by entropy minimization. *Proceedings Second EUROMICRO Workshop on Advanced Mobile Robots*, 1997.
- [31] W. Burgard, O. Brock, and C. Stachniss. Mapping Large Loops with a Single Hand-Held Camera. In *2007 Robotics: Science and Systems Conference*, pages 297–304. MIT Press, 2007.
- [32] S. Chandrasekaran, G. H. Golub, M. Gu, and A. H. Sayed. Parameter Estimation in the Presence of Bounded Data Uncertainties. *SIAM Journal on Matrix Analysis and Applications*, 19(1):235–252, 1998.
- [33] Shivkumar Chandrasekaran, Gene H Golub, Ming Gu, and Ali H Sayed. Efficient Algorithms for Least Squares Type Problems with Bounded Uncertainties. In *Proceedings of the second international workshop on Recent advances in total least squares techniques and errors-in-variables modeling*, pages 171–180, 1997.
- [34] R. Chatila and J. Laumond. Position referencing and consistent world modeling for mobile robots. *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, 2, 1985.
- [35] Jiechun Chen, Zhenliang Ding, and Feng Yuan. Theoretical Uncertainty Evaluation of Stereo Reconstruction. *2008 2nd International Conference on Bioinformatics and Biomedical Engineering*, (2):2378–2381, May 2008. doi: 10.1109/ICBBE.2008.927.
- [36] G Chesi. Estimation of the camera pose from image point correspondences through the essential matrix and convex optimization. In *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pages 35–40, 2009. doi: 10.1109/ROBOT.2009.5152224.
- [37] Wojciech Chojnacki, Michael J. Brooks, and Anton Vanel. On the fitting of surfaces to data with covariances. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1294–1303, 2000.
- [38] R. Cipolla. Vision-based Global Localization Using a Visual Vocabulary. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 4230–4235. IEEE, 2005. ISBN 0-7803-8914-X. doi: 10.1109/ROBOT.2005.1570770.

- [39] Javier Civera, Andrew J Davison, and J M Martínez Montiel. Unified inverse depth parametrization for monocular slam. In *In Proceedings of Robotics: Science and Systems*, 2006.
- [40] Marco Cagnetti, Paolo Stegagno, Antonio Franchi, Giuseppe Oriolo, and Heinrich H. Bühlhoff. 3-D mutual localization with anonymous bearing measurements. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 791–798, 2012.
- [41] Peter Corke, Pavan Sikka, and Jonathan Roberts. Height Estimation for an Autonomous Helicopter. *Lecture Notes Contr. Inform. ScisNotes Contr. Inform. Scis*, 271:101–110, 2001.
- [42] Gabriella Csurka, Cyril Zeller, and Olivier D Faugeras. Characterizing the Uncertainty of the Fundamental Matrix. *Computer Vision and Image Understanding*, pages 1–39, 2004.
- [43] M. Cummins and P. Newman. FAB-MAP: Probabilistic Localization and Mapping in the Space of Appearance, 2008. ISSN 0278-3649.
- [44] Mark Cummins and Paul Newman. Probabilistic Appearance Based Navigation and Loop Closing. In *IEEE International Conference on Robotics and Automation (ICRA'07)*, number April, pages 10–14, 2007. ISBN 1424406021.
- [45] Alexander Cunningham, Vadim Indelman, and Frank Dellaert. DDF-SAM 2.0: Consistent distributed smoothing and mapping. In *2013 IEEE International Conference on Robotics and Automation*, pages 5220–5227. Ieee, May 2013. ISBN 978-1-4673-5643-5. doi: 10.1109/ICRA.2013.6631323.
- [46] A J Davison. Real-time simultaneous localisation and mapping with a single camera. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 1403–1410 vol.2, 2003. doi: 10.1109/ICCV.2003.1238654.
- [47] Andrew J. Davison, Ian D. Reid, Nicholas D. Molton, and Olivier Stasse. MonoSLAM: Real-time single camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1052–1067, 2007.
- [48] F. Dellaert, S.M. Seitz, C.E. Thorpe, and S. Thrun. Structure from motion without correspondence. *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No.PR00662)*, 2, 2000. ISSN 1063-6919.
- [49] G Di Leo, C Liguori, and A Paolillo. Covariance Propagation for the Uncertainty Estimation in Stereo Vision. *IEEE Transactions on Instrumentation and Measurement*, 60(5):1664–1673, May 2011. ISSN 0018-9456.
- [50] A. Di Leo, G. and Liguori, C. and Paolillo. Propagation of uncertainty through stereo triangulation. In *Instrumentation and Measurement Technology Conference (I2MTC), 2010 IEEE*, pages 12–17, 2010.
- [51] G.A. Einicke and L.B. White. Robust extended Kalman filtering. *IEEE Transactions on Signal Processing*, 47(9), 1999. ISSN 1053-587X. doi: 10.1109/78.782219.

- [52] Laurent El Ghaoui and Hervé Lebret. Robust Solutions to Least-Squares Problems with Uncertain Data. *SIAM Journal on Matrix Analysis and Applications*, 18(4):1035–1064, October 1997. ISSN 0895-4798.
- [53] Lars Elden. Perturbation Theory for the Least Squares Problem with Linear Equality Constraints. 17(3):338–350, 1980.
- [54] Isaac Esteban, Leo Dorst, and Judith Dijk. Closed form solution for the scale ambiguity problem in monocular visual odometry. In *Proceedings of the Third international conference on Intelligent robotics and applications - Volume Part I*, ICIRA'10, pages 665–679, Berlin, Heidelberg, 2010. Springer-Verlag. ISBN 3-642-16583-4, 978-3-642-16583-2.
- [55] F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. *Autonom. Robots*, 4:333–349, 1997.
- [56] David Filliat. A visual bag of words method for interactive qualitative localization and mapping. (April):10–14, 2007.
- [57] G Finlayson, B Schiele, and J Crowley. Comprehensive colour image normalization. *Computer Vision?ECCV'98*, 1:475–490, 1998.
- [58] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981. ISSN 0001-0782.
- [59] D. Franken and A. Hupper. Improved fast covariance intersection for distributed data fusion. In *2005 7th International Conference on Information Fusion*, page 7 pp. Ieee, 2005. ISBN 0-7803-9286-8. doi: 10.1109/ICIF.2005.1591849.
- [60] F Fraundorfer and D Scaramuzza. Visual Odometry : Part II: Matching, Robustness, Optimization, and Applications. *Robotics Automation Magazine, IEEE*, 19(2):78–90, 2012. ISSN 1070-9932. doi: 10.1109/MRA.2012.2182810.
- [61] Jerome H. Freidman, Jon Louis Bentley, and Raphael Ari Finkel. An Algorithm for Finding Best Matches in Logarithmic Expected Time, 1977. ISSN 00983500.
- [62] Jorge Fuentes-Pacheco, José Ruiz-Ascencio, and Juan Manuel Rendón-Mancha. Visual simultaneous localization and mapping: a survey, 2012.
- [63] Brian V. Funt and Graham D. Finlayson. Color constant color indexing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(5):522–529, 1995.
- [64] Enright J Barfoot T D Furgale P T, Carle P. The Devon Island Rover Navigation Dataset. *International Journal of Robotics Research*, Manuscript, 2012.
- [65] Virginie Gabrel. Recent Advances in Robust Optimization : An Overview Theory of Robust Optimization. pages 1–25, 2013.
- [66] Emilio Garcia-fidalgo and Alberto Ortiz. A Solution for Bayesian Visual Loop Closure Detection Based on Local Invariant Features. Technical report, 2013.

- [67] A Geiger, P Lenz, C Stiller, and R Urtasun. Vision meets robotics: The KITTI dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013. URL <http://ijr.sagepub.com/cgi/doi/10.1177/0278364913491297>.
- [68] Andreas Geiger, Martin Roser, and Raquel Urtasun. Efficient Large-Scale Stereo Matching. In *Asian Conference on Computer Vision (ACCV)*, 2010.
- [69] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets Robotics : The KITTI Dataset. *International Journal of Robotics Research (IJRR)*, (October):1–6, 2011.
- [70] Jan Mark Geusebroek, Rein Van Den Boomgaard, Arnold W M Smeulders, and Hugo Geerts. Color invariance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(12):1338–1350, 2001.
- [71] Theo Gevers and Arnold W.M. Smeulders. Color-based object recognition, 1999. ISSN 00313203.
- [72] V.M. Govindu. Combining two-view constraints for motion estimation. *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, 2, 2001. ISSN 1063-6919.
- [73] R F Graf. *Modern Dictionary of Electronics*[. Electronics & Electrical. Newnes [Imprint], 1999. ISBN 9780750698665.
- [74] Dorota A. Grejner-Brzezinska, Charles K. Toth, Hongxing Sun, Xiankun Wang, and Chris Rizos. A robust solution to high-accuracy geolocation: Quadruple integration of GPS, IMU, pseudolite, and terrestrial laser scanning. *IEEE Transactions on Instrumentation and Measurement*, 60(11):3694–3708, 2011.
- [75] W Grisetti, G., Kummerle, R., Stachniss, C., & Burgard. A tutorial on graph-based SLAM. *Intelligent Transportation Systems Magazine, IEEE*, pages 31–43, 2010.
- [76] Jens-steffen Gutmann, Kurt Konolige, S R I International, and Menlo Park. Incremental Mapping of Large Cyclic Environments. In *IEEE Int. Symp. Computational Intelligence in (CIRA)*, Monterey, CA., 1999.
- [77] Andreas Haja, Bernd Jähne, and Steffen Abraham. Localization accuracy of region detectors. In *26th IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2008.
- [78] Robert M Haralick. Propagating Covariance in Propagating Covariance in Computer Vision. In *In proceedings of the Theoretical Foundations of Computer Vision, TFCV on Performance Characterisation in Computer Vision*, number September, 1998.
- [79] Chris Harris and Mike Stephens. A combined corner and edge detector. In *In Proc. of Fourth Alvey Vision Conference*, pages 147–151, 1988.
- [80] R Hartley and F Schaffalitzky. L infin; minimization in geometric reconstruction problems. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 1, pages I-504–I-509 Vol.1, 2004. doi: 10.1109/CVPR.2004.1315073.

- [81] R.I. Hartley. In defence of the 8-point algorithm. *IEEE Trans. Patt. Anal. Mach. Intell.*, 97(6):580–593, 1967. doi: 10.1109/ICCV.1995.466816.
- [82] R.I. Hartley and A Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.
- [83] Richard Hartley. Lines and points in three views and the trifocal tensor. *International Journal of Computer Vision*, 22(2):125–140, 1997.
- [84] Richard Hartley and Fredrik Kahl. Optimal algorithms in multiview geometry. *Computer Vision?ACCV 2007*, pages 13–34, 2007.
- [85] K Ho and P Newman. Multiple map intersection detection using visual appearance. *International Conference on Computational*, 2005.
- [86] Kin Leong Ho and Paul Newman. Loop closure detection in SLAM by combining visual and spatial appearance. *Robotics and Autonomous Systems*, 54(9):740–749, 2006.
- [87] Berthold K. P. Horn, Hugh M. Hilden, and Shahriar Negahdaripour. Closed-form solution of absolute orientation using orthonormal matrices. *Journal of the Optical Society of America A*, 5(7):1127–1135, July 1988. ISSN 1084-7529. doi: 10.1364/JOSAA.5.001127.
- [88] Berthold K.P. Horn and Brian G. Schunck. Determining optical flow, 1981.
- [89] O Huang, T.S.; Faugeras. Some properties of the E matrix in two-view motion estimation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 11(12):1310–1312, 1989.
- [90] N Imran, S.A. and Aouf. A Robust Two Signature Descriptor with Orientation Bins and Colour Codes for Enhanced Feature Matching. In *Systems, Man, and Cybernetics (SMC), 2013 IEEE International Conference on*, pages 3031–3035, 2013.
- [91] Jcgm. Evaluation of measurement data ? Guide to the expression of uncertainty in measurement. Technical report, 2008.
- [92] Jean-Yves Bouguet. Complete Camera Calibration Toolbox for Matlab. URL http://www.vision.caltech.edu/bouguetj/calib_doc/index.html.
- [93] John W. Shipman. Introduction to color theory, 2012.
- [94] F Kahl. Multiple view geometry and the L_∞ -norm. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 2, pages 1002–1009 Vol. 2, 2005. doi: 10.1109/ICCV.2005.163.
- [95] K. Kanatani. Unbiased estimation and statistical analysis of 3-D rigid motion from two views. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(1):37–50, 1993. ISSN 01628828. doi: 10.1109/34.184773.
- [96] Kenichi Kanatani. *Statistical Optimization for Geometric Computation: Theory and Practice*. 1996.

- [97] Kenichi Kanatani. Optimal Fundamental Matrix Computation: Algorithm and Reliability Analysis. In *Proc. 6th Symp. Sensing via Image Inf.*, number June, pages 291—298, 2000.
- [98] Kenichi Kanatani. Statistical optimization for geometric fitting: Theoretical accuracy bound and high order error analysis. *International Journal of Computer Vision*, 80(2):167–188, 2008.
- [99] Y. Kanazawa and K. Kanatani. Do we really have to consider covariance matrices for image features? *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, 2:301–306, 2001.
- [100] Aram Kawewong, Noppharit Tongprasit, Sirinart Tungruamsub, and Osamu Hasegawa. Online and Incremental Appearance-based SLAM in Highly Dynamic Environments. *The International Journal of Robotics Research*.
- [101] Qifa Ke and T Kanade. Quasiconvex Optimization for Robust Geometric Reconstruction. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(10):1834–1847, 2007. ISSN 0162-8828. doi: 10.1109/TPAMI.2007.1083.
- [102] Mohamed Sadek Kemouche and Nabil Aouf. A GMM approximation with merge and split for nonlinear non-Gaussian tracking. *2010 13th International Conference on Information Fusion*, pages 1–6, 2010.
- [103] Georg Klein and David Murray. Parallel Tracking and Mapping for Small AR Workspaces. In *IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 1–10. Ieee, November 2007. ISBN 978-1-4244-1749-0. doi: 10.1109/ISMAR.2007.4538852. URL <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4538852>.
- [104] R.E. Kolman, B. and Beck. *Elementary Linear Programming with Applications*. cademic Press, 1980.
- [105] Jack. B Kuipers. *Quaternions and Rotation Sequences: A Primer With Applications to Orbits, Aerospace, and Virtual Reality*. Princeton University Press, 2002.
- [106] Iago Landesa-Vázquez, Francisco Parada-Loira, and José L. Alba-Castro. Fast real-time multiclass traffic sign detection based on novel shape and texture descriptors. In *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, pages 1388–1395, 2010.
- [107] E. L. Lawler and D. E. Wood. *Branch-and-Bound Methods: A Survey*, 1966. ISSN 0030-364X.
- [108] Peter Leedan, Yoram; Meer. Heteroscedastic Regression in Computer Vision: Problems with Bilinear Constraint. *International Journal of Computer Vision*, 37(2):127–150, 2000.
- [109] A Levin and R Szeliski. Visual odometry and map correlation. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 1, pages I–611–I–618 Vol.1, 2004. doi: 10.1109/CVPR.2004.1315088.

- [110] Xiaodong Li, Nabil Aouf, and Abdelkrim Nemra. 3D mapping based VSLAM for UAVs. In *2012 20th Mediterranean Conference on Control & Automation (MED)*, pages 348–352, Barcelona, Spain., July 2012. Ieee. ISBN 978-1-4673-2531-8. doi: 10.1109/MED.2012.6265662.
- [111] Haibin Ling and Kazunori Okada. Diffusion distance for histogram comparison. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 246–253, 2006.
- [112] Yang Liu and Hong Zhang. Visual Loop Closure Detection with a Compact Image Descriptor. In *International Conference on Intelligent Robots and Systems.*, pages 1051–1056, 2012. ISBN 9781467317368.
- [113] Liu Yang and Zhang Hong. Indexing visual features: Real-time loop closure detection using a tree structure. In *2012 IEEE International Conference on Robotics and Automation*, pages 3613–3618, Saint Paul, MN., May 2012. Ieee. ISBN 978-1-4673-1405-3. doi: 10.1109/ICRA.2012.6224741.
- [114] Lennart Ljung. *System Identification: Theory for User*, volume 11 of *Prentice-Hall information and system sciences series*. Prentice Hall, 1987.
- [115] J. Lobo and J. Dias. Relative Pose Calibration Between Visual and Inertial Sensors, 2007.
- [116] J Löfberg. Automatic robust convex programming. *Optimization methods and software*, 27(00):115–129, 2012.
- [117] H C Longuet-Higgins. Readings in computer vision: issues, problems, principles, and paradigms. chapter A computer, pages 61–62. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1987. ISBN 0-934613-33-8.
- [118] David G. Lowe. Fitting parameterized three-dimensional models to images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(5):441–450, 1991.
- [119] David G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, November 2004. ISSN 0920-5691.
- [120] Bruce D Lucas and Takeo Kanade. An Iterative Image Registration Technique with an Application to Stereo Vision. *Imaging*, 130(x):674–679, 1981. ISSN 17486815. doi: 10.1109/HPDC.2004.1323531.
- [121] Q Luong and O D Faugeras. The Fundamental matrix : theory , algorithms , and stability analysis. *International Journal of Computer Vision*, 17(3):1–46, 2004.
- [122] Y Ma. *An Invitation to 3-D Vision: From Images to Geometric Models*. Interdisciplinary Applied Mathematics. Springer, 2004. ISBN 9780387008936.
- [123] Y Ma. *An Invitation to 3-D Vision: From Images to Geometric Models*. Interdisciplinary Applied Mathematics. Springer, 2004. ISBN 9780387008936.

- [124] J B Antoine Maintz and Max A Viergever. A survey of medical image registration. *Medical Image Analysis*, 2(1):1–36, 1998. ISSN 13618415.
- [125] Agostino Martinelli, Frederic Pont, and Roland Siegwart. Multi-robot localization using relative observations. In *Proceedings - IEEE International Conference on Robotics and Automation*, volume 2005, pages 2797–2802, 2005.
- [126] Petr Matula, M. Kozubek, F. Staier, and M. Hausmann. Precise 3D image alignment in micro-axial tomography. *Journal of Microscopy*, 209(2):126–142, 2003.
- [127] G. P. McCormick. Computability of global solutions to factorable nonconvex programs: part I - convex underestimating problems. *Mathematical Programming*, 10:147–175, 1976.
- [128] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schafalitzky, T. Kadir, and L. Van Gool. A comparison of affine region detectors. *International Journal of Computer Vision*, 65(1-2):43–72, 2005.
- [129] Nikos Georgis Miros law Bober and Josef Kittler. On Accurate and Robust Estimation of Fundamental Matrix. *Comput. Vision Image Understanding*, 72(1):39–53, 1998.
- [130] Kaushik Mitra and Rama Chellappa. A Scalable Projective Bundle Adjustment Algorithm using the L infinity Norm. In *Proceedings of the 2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing, ICVGIP '08*, pages 79–86, Washington, DC, USA, 2008. IEEE Computer Society. ISBN 978-0-7695-3476-3.
- [131] C. M. Shetty Mokhtar S. Bazaraa, Hanif D. Sherali. *Nonlinear Programming, Theory and Algorithms*. Wiley, 1993.
- [132] H.P. Hans Peter Moravec. *Obstacle avoidance and navigation in the real world by a seeing robot rover*. PhD thesis, 1980.
- [133] Francesc Moreno-Noguer, Vincent Lepetit, and Pascal Fua. Accurate non-iterative o (n) solution to the pnp problem. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007.
- [134] Anastasios I. Mourikis and Stergios I. Roumeliotis. Performance analysis of multirobot cooperative localization. *IEEE Transactions on Robotics*, 22(4):666–681, 2006.
- [135] S. Boyd M.S.Lobo, L. Vandenberghe and H. Lebet. Applications of second-order cone programming. *Linear Algebra and its Applications*, 284:193–228, 1998.
- [136] Marius Muja and David G. Lowe. Fast Matching of Binary Features. In *2012 Ninth Conference on Computer and Robot Vision*, pages 404–410, Toronto, ON., May 2012. Ieee. ISBN 978-1-4673-1271-4. doi: 10.1109/CRV.2012.60.
- [137] Abdelkrim Nemra and Nabil Aouf. Robust cooperative UAV Visual SLAM. In *2010 IEEE 9th International Conference on Cyberntic Intelligent Systems*, pages 1–6, Reading UK, September 2010. Ieee. ISBN 978-1-4244-9023-3. doi: 10.1109/UKRICIS.2010.5898125.

- [138] Abdelkrim Nemra and Nabil Aouf. Robust cooperative visual localization with experimental validation for unmanned aerial vehicles. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, 2012.
- [139] Esha D. Nerurkar, Stergios I. Roumeliotis, and Agostino Martinelli. Distributed maximum a posteriori estimation for multi-robot cooperative localization. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 1402–1409, 2009.
- [140] P. Newman, D. Cole, and K. Ho. Outdoor SLAM using visual appearance and laser ranging. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, number May, pages 1180–1187. Ieee, 2006. ISBN 0-7803-9505-0. doi: 10.1109/ROBOT.2006.1641869.
- [141] Paul Newman and Kin Ho. SLAM- Loop closing with visually salient features. In *Proceedings - IEEE International Conference on Robotics and Automation*, volume 2005, pages 635–642, 2005.
- [142] Wolfgang Niehsen. Information fusion based on fast covariance intersection filtering. In *the fifth international conference on information fusion*, pages 901–904, 2002.
- [143] David Nistér. An efficient solution to the five-point relative pose problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):756–770, 2004.
- [144] David Nistér and Henrik Stewénus. Scalable recognition with a vocabulary tree. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 2161–2168, 2006.
- [145] David Nistér, Oleg Naroditsky, and James Bergen. Visual odometry for ground vehicle applications. *Journal of Field Robotics*, 23:2006, 2006.
- [146] J A NOCEDAL and S J Wright. *Numerical optimization*. Springer series in operations research and financial engineering. Springer-Verlag New York, 1999. ISBN 9780387987934.
- [147] Gabriel Nützi, Stephan Weiss, Davide Scaramuzza, and Roland Siegwart. Fusion of IMU and Vision for Absolute Scale Estimation in Monocular SLAM. *J. Intell. Robotics Syst.*, 61(1-4):287–299, January 2011. ISSN 0921-0296. doi: 10.1007/s10846-010-9490-z.
- [148] Aude Oliva, Women Hospital, and Longwood Ave. Modeling the Shape of the Scene : A Holistic Representation of the Spatial Envelope ? *International Journal of Computer Vision*, 42(3):145–175, 2001.
- [149] E. Olson, J. Leonard, and S. Teller. Fast iterative alignment of pose graphs with poor initial estimates. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, number May, pages 2262–2269, Orlando, FL., 2006. Ieee. ISBN 0-7803-9505-0. doi: 10.1109/ROBOT.2006.1642040.

- [150] Amidi Omead, Takeo Kanade, and Keisuke Fujita. Autonomous Systems A visual odometer for autonomous helicopter flight. *Robotics and Autonomous Systems*, 28(02):185–193, 1999.
- [151] Umut Orguner and Fredrik Gustafsson. Statistical characteristics of Harris corner detector. In *IEEE Workshop on Statistical Signal Processing Proceedings*, pages 571–575, 2007.
- [152] T Papadopoulo and MIA Lourakis. Estimating the jacobian of the singular value decomposition: Theory and applications. *Computer Vision-ECCV 2000*, 2000.
- [153] Pedro Piniés and Juan D. Tardós. Large-scale SLAM building conditionally independent local maps: Application to monocular vision. *IEEE Transactions on Robotics*, 24(5):1094–1106, 2008.
- [154] A Pronobis, B Caputo, P Jensfelt, and H I Christensen. A Discriminative Approach to Robust Visual Place Recognition. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3829–3836, 2006.
- [155] Rahul Raguram, Jan-Michael Frahm, and Marc Pollefeys. Exploiting uncertainty in random sample consensus. In *IEEE 12th International Conference on Computer Vision*, pages 2074–2081. Ieee, September 2009. ISBN 978-1-4244-4420-5. doi: 10.1109/ICCV.2009.5459456.
- [156] Douglas A Reynolds. Gaussian Mixture Models. *Encyclopedia of Biometric Recognition*, 31:1047–64, 2008. ISSN 1361-6579. doi: 10.1088/0967-3334/31/7/013.
- [157] Rui Rocha, Jorge Dias, and Adriano Carvalho. Cooperative multi-robot systems: A study of vision-based 3-D mapping using information theory. *Robotics and Autonomous Systems*, 53(3-4):282–311, 2005.
- [158] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 3951 LNCS, pages 430–443, 2006.
- [159] Carsten Rother. *Multi-View Reconstruction and Camera Recovery using a Real or Virtual Reference Plane; PHD THESIS*. PhD thesis, January 2003.
- [160] Hong Seo Ryoo and Nikolaos V. Sahinidis. Analysis of Bounds for Multilinear Functions. *Journal of Global Optimization*, 19(4):403–424, 2001.
- [161] Saad Ali Imran ; Nabil Aouf. Robust $L \infty$ Homography Estimation using Reduced Image Feature Covariances from an RGB Image List of Figures. *Journal of Electronic Imaging*, 21, 2012.
- [162] Juan M. Sáez and Francisco Escolano. 6DOF entropy minimization SLAM for stereo-based wearable devices. *Computer Vision and Image Understanding*, 115(2):270–285, 2011.

- [163] Vincent Samson, Frédéric Champagnat, and Jean-François Giovannelli. Point target detection and subpixel position estimation in optical imagery. *Applied Optics*, 43(2):257–263, 2009.
- [164] Angel Domingo Sappa, Fadi Dornaika, Daniel Ponsa, David Gerónimo, and Antonio López. An efficient approach to onboard stereo vision system pose estimation. *IEEE Transactions on Intelligent Transportation Systems*, 9(3):476–490, 2008. ISSN 15249050.
- [165] Srikanth Saripalli, James F Montgomery, and Gaurav S Sukhatme. Visually-Guided Landing of an Unmanned Aerial Vehicle. *IEEE Trans. Robot. Automat.*, 19(3):371–380, 2003.
- [166] D Scaramuzza and F Fraundorfer. Visual Odometry [Tutorial]. *Robotics Automation Magazine, IEEE*, 18(4):80–92, 2011. ISSN 1070-9932. doi: 10.1109/MRA.2011.943233.
- [167] D Scaramuzza and R Siegwart. Appearance-Guided Monocular Omnidirectional Visual Odometry for Outdoor Ground Vehicles. *Robotics, IEEE Transactions on*, 24(5):1015–1026, 2008. ISSN 1552-3098. doi:10.1109/TRO.2008.2004490.
- [168] Davide Scaramuzza, Friedrich Fraundorfer, and Roland Siegwart. Real-time monocular visual odometry for on-road vehicles with 1-point RANSAC. *2009 IEEE International Conference on Robotics and Automation*, 2009. ISSN 1050-4729.
- [169] Grant Schindler, Matthew Brown, and Richard Szeliski. City-scale location recognition. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2007.
- [170] David Schleicher, Luis M. Bergasa, Manuel Ocaña, Rafael Barea, and María Elena López. Real-time hierarchical outdoor slam based on stereo-vision and GPS fusion. *IEEE Transactions on Intelligent Transportation Systems*, 10(3):440–452, 2009. ISSN 15249050.
- [171] Cordelia Schmid, Roger Mohr, and Christian Bauckhage. Evaluation of interest point detectors. *International Journal of computer vision*, 37(2):151–172, 2000.
- [172] Yongduek Seo and Richard Hartley. Sequential L infinity norm minimization for triangulation. In *Proceedings of the 8th Asian conference on Computer vision - Volume Part II, ACCV'07*, pages 322–331, Berlin, Heidelberg, 2007. Springer-Verlag. ISBN 3-540-76389-9, 978-3-540-76389-5.
- [173] Amnon Shashua. Trilinearity in visual recognition by alignment. *Computer Vision ? ECCV '94*, 800:479–484, 1994.
- [174] Amnon Shashua. Algebraic functions for recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8):779–789, 1995. ISSN 01628828.
- [175] Christian Siagian, Student Member, and Laurent Itti. Rapid Biologically-Inspired Scene Classification Using Features Shared with Visual Attention. *IEEE Transactions on pattern analysis and machine intelligence*, 29(2):300–312, 2007.

- [176] C Silpa-Anan. Visual localization and loop-back detection with a high resolution omnidirectional camera. *Workshop on Omnidirectional Vision*, 2005.
- [177] Chanop Silpa-anan and Richard Hartley. Optimised KD -trees for fast image descriptor matching. In *IEEE Conference on Computer Vision and Pattern Recognition*, Anchorage, AK., 2008. ISBN 9781424422432.
- [178] D Simon. *Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches*. Wiley, 2006. ISBN 9780470045336.
- [179] Gautam Singh and Jana Kosecka. Visual Loop Closing using GIST Descriptors in Manhattan World. In *International Conference on Robotics and Automation*, 2010.
- [180] Josef Sivic and Andrew Zisserman. Video Google: a text retrieval approach to object matching in videos. In *Proceedings Ninth IEEE International Conference on Computer Vision*, number Iccv, pages 1470–1477 vol.2, Nice, UK., 2003. Ieee. ISBN 0-7695-1950-4. doi: 10.1109/ICCV.2003.1238663.
- [181] R. Smith, M. Self, and P. Cheeseman. Estimating uncertain spatial relationships in robotics. *Proceedings. 1987 IEEE International Conference on Robotics and Automation*, 4, 1987.
- [182] SM Smith and JM Brady. SUSAN? a new approach to low level image processing. *International journal of computer vision*, 23(1):45–78, 1997. ISSN 1573-1405. doi: 10.1023/A:1007963824710.
- [183] Sorenson HW. Least Squares estimation. From Gauss to Kalman. *IEEE Spectrum*, 7(7):63–68, 1970.
- [184] M Spetsakis. Structure from motion using line correspondences. *International Journal of Computer Vision*, 1990.
- [185] R.M. Steele and C. Jaynes. Feature Uncertainty Arising from Covariant Image Noise. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 1063–1070, San Diego, CA, USA., 2005. Ieee. ISBN 0-7695-2372-2. doi: 10.1109/CVPR.2005.158.
- [186] Hauke Strasdat and Andrew J Davison. Scale Drift-Aware Large Scale Monocular SLAM. *Robotics: Science and Systems*, 2(3):5–12, 2010.
- [187] Jos F Sturm. Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones, 1998.
- [188] N Sünderhauf and Kurt Konolige. Visual odometry using sparse bundle adjustment on an autonomous outdoor vehicle. *Autonome Mobile Systeme ...*, 2006.
- [189] F Sur, N Noury, and MO Berger. Computing the uncertainty of the 8 point algorithm for fundamental matrix estimation. *19th British Machine Vision Conference- ...*, 2008.
- [190] Richard Szeliski. Computer Vision : Algorithms and Applications. *Computer*, 5:832, 2010. ISSN 10636919. doi: 10.1007/978-1-84882-935-0.

- [191] Sergios Theodoridis and Konstantinos Koutroumbas. *Pattern Recognition*. Elsevier, 2006. ISBN 9780123695314. doi: 10.1016/B978-012369531-4/50007-X.
- [192] D Theodoridis, S., Pikrakis, A., Koutroumbas, K. and Cavouras. *Introduction to pattern recognition: a Matlab approach*. Massachusetts., 2010. ISBN 9780123744869.
- [193] EH Thompson. An exact linear solution of the problem of absolute orientation. *Photogrammetria*, 15:163–179, 1959.
- [194] Sebastian Thrun, Daphne Koller, Zoubin Ghahramani, Hugh Durrant-Whyte, and Andrew Y. Ng. Simultaneous mapping and localization with sparse extended information filters: Theory and initial results. In *Springer Tracts in Advanced Robotics*, volume 7 STAR, pages 363–380, 2004.
- [195] Timothy A. Clarke ; M. A. R. Cooper and John G. Fryer. Estimator for the random error in subpixel target location and its use in the bundle adjustment. In *Proc. SPIE 2252, Optical 3D Measurement Techniques II: Applications in Inspection, Quality Control, and Robotics*, page 161, 1994.
- [196] Carlo Tomasi. Shape and motion from image streams under orthography: a factorization method. *International Journal of Computer Vision*, 9:137—154, 1992.
- [197] N. Trawny and T. Barfoot. Optimized motion strategies for cooperative localization of mobile robots. *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, 1, 2004. ISSN 1050-4729.
- [198] Nikolas Trawny and Stergios I Roumeliotis. Indirect Kalman Filter for 3D Attitude Estimation A Tutorial for Quaternion Algebra Multiple Autonomous Robotic Systems Laboratory Technical Report Number 2005-002 , Rev . 57. (612), 2005.
- [199] Bill Triggs and Long Quan. Camera Pose Revisited - New Linear Algorithms. *Esprit*, pages 6–8, 2000.
- [200] Emanuele Trucco and Alessandro Verri. *Introductory Techniques for 3-D Computer Vision*. Prentice Hall, ISBN: 0132611082, 1998.
- [201] Iwan Ulrich and Illah Nourbakhsh. Appearance-Based Place Recognition for Topological Localization. In *IEEE International Conference on Robotics and Automation*, number April, pages 1023–1029, 2000. ISBN 0780358864.
- [202] Shinji Umeyama. Least-Squares Estimation of Transformation Parameters Between Two Point Patterns. *Intelligence, IEEE Transactions on Pattern Analysis and Machine*, 13(4):376–380, 1991.
- [203] University of Oxford/ Department of Engineering Science. Visual Geometry Group. URL <http://www.robots.ox.ac.uk/~vgg/data/data-mview.html>.
- [204] Matthew Uyttendaele, Antonio Criminisi, Sing Bing Kang, Simon Winder, Richard Hartley, and Richard Szeliski. High-quality Image-based Interactive Exploration. *IEEE Computer Graphics and Applications*, 24(3):52–63, 2004.

- [205] Teresa A. Vidal-Calleja, Cyrille Berger, Joan Solà, and Simon Lacroix. Large scale multiple robot visual mapping with heterogeneous landmarks in semi-structured terrain. *Robotics and Autonomous Systems*, 59(9):654–674, 2011. ISSN 09218890.
- [206] Thierry Vieville, Olivier Faugeras, and Quang-Tuan Luong. Motion of points and lines in the uncalibrated case, 1996. ISSN 0920-5691.
- [207] Joost Van De Weijer and Cordelia Schmid. Coloring Local Feature Extraction. In *ECCV*, pages 334–348, Graz, 2006.
- [208] Juyang Weng, Thomas S. Huang, and Narendra Ahuja. Motion and structure from line correspondences: Closed-form solution, uniqueness, and optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(3):318–336, 1992.
- [209] CF Westin and CJ Westelius. Attention control for robot vision. *Computer Vision and . . .*, pages 726–733, 1996.
- [210] Brian Williams and Ian Reid. On combining visual SLAM and visual odometry. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 3494–3500, 2010.
- [211] Brian Williams, Georg Klein, and Ian Reid. Real-time SLAM relocalisation. In *Proceedings of the IEEE International Conference on Computer Vision*, 2007.
- [212] Changchang Wu, Brian Clipp, Xiaowei Li, Jan Michael Frahm, and Marc Pollefeys. 3D model matching with viewpoint-invariant patches (VIP). In *26th IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2008.
- [213] Ming Wu, Feifei Huang, Long Wang, and Jiyin Sun. Cooperative multi-robot monocular-SLAM using salient landmarks. In *Proceedings - 2009 International Asia Conference on Informatics in Control, Automation, and Robotics, CAR 2009*, pages 151–155, 2009.
- [214] Medha V Wyawahare, Pradeep M Patil, and Hemant K Abhyankar. Image Registration Techniques : An overview. *International Journal of Signal Processing, Image Processing and Pattern Recognition*, 2(3):11–28, 2009.
- [215] Bernhard Zeisl, Pierre Fite Georgel, Florian Schweiger, Eckehard Steinbach, and Nassir Navab. Estimation of Location Uncertainty for Scale Invariant Feature Points. *Proceedings of the British Machine Vision Conference 2009*, pages 57.1–57.12, 2009.
- [216] Hong Zhang. BoRF: Loop-closure detection with scale invariant visual features. In *2011 IEEE International Conference on Robotics and Automation*, pages 3125–3130, Shanghai, China., May 2011. Ieee. ISBN 978-1-61284-386-5. doi: 10.1109/ICRA.2011.5980273.
- [217] Zhengyou Zhang. Determining the Epipolar Geometry and its Uncertainty :. *International Journal of Computer Vision*, 27(2):161–198, 2004.
- [218] Barbara Zitová and Jan Flusser. Image registration methods: A survey, 2003.

-
- [219] Z. Zivkovic, B. Bakker, and B. Krose. Hierarchical map building using visual landmarks and geometric constraints. *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2005. doi: 10.1109/IROS.2005.1544951.
- [220] Danping Zou and Ping Tan. CoSLAM: Collaborative visual SLAM in dynamic environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(2):354–366, 2013.

