CRANFIELD UNIVERSITY

IOANNIS KATRAMADOS

REAL-TIME OBJECT DETECTION USING MONOCULAR VISION FOR LOW-COST AUTOMOTIVE SENSING SYSTEMS

SCHOOL OF ENGINEERING

PhD THESIS

Supervisor: Dr. Toby P. Breckon

February 2013

CRANFIELD UNIVERSITY

SCHOOL OF ENGINEERING

PhD THESIS

IOANNIS KATRAMADOS

Real-time object detection using monocular vision for low-cost automotive sensing systems

Supervisor: Dr. Toby P. Breckon

February 2013

This thesis is submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy

Abstract

This work addresses the problem of real-time object detection in automotive environments using monocular vision. The focus is on real-time feature detection, tracking, depth estimation using monocular vision and finally, object detection by fusing visual saliency and depth information.

Firstly, a novel featuredetection approach is proposed for extracting stable and dense features even in images with very low signal-to-noise ratio. This methodology is based on image gradients, which are redefined to take account of noise as part of their mathematical model. Each gradient is based on a vector connecting a negative to a positive intensity centroid, where both centroids are symmetric about the centre of the area for which the gradient is calculated. Multiple gradient vectors define a feature with its strength being proportional to the underlying gradient-vector magnitude. The evaluation of the Dense Gradient Features (DeGraF) shows superior performance over other contemporary detectors in terms of keypoint density, tracking accuracy, illumination invariance, rotation invariance, noise resistance and detection time.

The DeGraF features form the basis for two new approaches that perform dense 3D reconstruction from a single vehicle-mounted camera. The first approach tracks DeGraF features in real-time while performing image stabilisation with minimal computational cost. This means that despite camera vibration the algorithm can accurately predict the real-world coordinates of each image pixel in real-time by comparing each motion-vector to the ego-motion vector of the vehicle. The performance of this approach has been compared to different 3D reconstruction methods in order to determine their accuracy, depth-map density, noise-resistance and computational complexity. The second approach proposes the use of local frequency analysis of

gradient features for estimating relative depth. This novel method is based on the fact that DeGraF gradients can accurately measure local image variance with sub-pixel accuracy. It is shown that the local frequency by which the centroid oscillates around the gradient window centre is proportional to the depth of each gradient centroid in the real world. The lower computational complexity of this methodology comes at the expense of depth-map accuracy as the camera velocity increases, but it is at least five times faster than the other evaluated approaches.

This work also proposes a novel technique for deriving visual-saliency maps by using Division of Gaussians (DIVoG). In this context, saliency maps express the difference of each image pixel to its surrounding pixels across multiple pyramid levels. This approach is shown to be both fast and accurate when evaluated against other state-of-the-art approaches. Subsequently, the saliency information is combined with depth information to identify salient regions close to the host vehicle. The fused map allows faster detection of high-risk areas where obstacles are likely to exist. As a result, existing object-detection algorithms, such as the Histogram of Oriented Gradients (HOG) can execute at least five times faster.

In conclusion, through a step-wise approach, computationally-expensive algorithms have been optimised or replaced by novel methodologies to produce a fast object-detection system that is aligned to the requirements of the automotive domain.

Acknowledgements

Firstly, I would like to thank my supervisor Dr. Toby Breckon for his continuous support throughout the PhD. He has worked incredibly hard to keep this project going and his help is highly appreciated. With his scientific, engineering and personal advice he also averted many critical errors that I would otherwise have made. Equally important was the support of my future wife Varvara Kokkali, who stood by me in every step of this journey. With her patience, love and care she inspired me to achieve much more than I thought was possible. PhD and work have stolen many hours out of our life, which I promise to return back soon. In addition, thanks go to my family and all the staff of the Applied Mathematics and Computing department for their full support. Especially, I would like to mention my friends Robert Sawko and Greg Flitton for all the wonderful and inspiring coffee break chats that we had together. Finally, I am thankful to my project sponsors, namely EPSRC and TRW Conekt. Without their financial support this PhD would never have started.

Contents

List of Figures

List of Tables

List of Abbreviations

ADAS Advanced Driver Assistance Systems

ADAS Advanced Driver Assistance Systems

ADR Adaptive Dynamic Range

AGAST Adaptive and Generic Accelerated Segment Test

BRIEF Binary Robust Independent Elementary Features

CenSurE Centre Surround Extremas

DeGraF Dense Gradient Features

DIVoG Division of Gaussians

DoG Difference of Gaussians

FAST Features from Accelerated Segment Test

GFTT Good Features To Track

GOA Greedy Optimal Assignment

GraCe Gradient Centroids

HDR High Dynamic Range

HOG Histogram of Oriented Gradients

IGD Inverted Gaussian Dipyramid

KLT Kanade-Lucas-Tomashi (tracking algorithm)

MGE Modified Greedy Exchange

MSER Maximally Stable Extremal Regions

ORB Oriented Fast and Rotated BRIEF

PPS Pedestrian Protection System

SIFT Scale Invariant Feature Transform

SNR Signal-to-noise ratio

SURF Speeded-Up Robust Features

Chapter 1

Introduction

This work addresses the problem of real-time object detection in automotive environments using monocular vision. The motivation is based on the potential of obstacle-detection systems to prevent or mitigate accidents. A key limiting factor is the processing power of low-cost embedded hardware, which requires optimised vision algorithms to make such a system affordable [13]. A wide-range of top-tier vehicles already ship with obstacle-detection systems based on monocular vision, while also performing other tasks such as lane detection and lane departure warning [14–16]. If the cost of these systems could be lowered, while preserving accuracy and reliability, then a wider range of vehicles and autonomous driving systems [17–22] would be able to benefit from this type of safety technology. Achieving this objective requires the optimisation of current computationally-expensive approaches. This thesis presents a set of novel and highly efficient methodologies for object detection in automotive environments. *Figure 1.1* gives an overview of the system architecture, while outlining the specific research areas that the literature review focussed on.

In Chapter 2, the state-of-the-art is reviewed and a number of technological limitations are identified. Each of these limitations is addressed by a chapter of this thesis. The focus is on real-time feature detection and tracking, depth estimation using monocular vision, visual-saliency techniques and object-detection methodologies. Firstly, a broad range of feature detectors are reviewed including Hessian [23], Moravec [24], Förstner [25], Harris [26], Tomasi Kanade [27], Shi Tomasi [28], Haralick [29], Heitger [30], SUSAN [31], SIFT [1], SURF [2], FAST [32], Harris-affine [33],

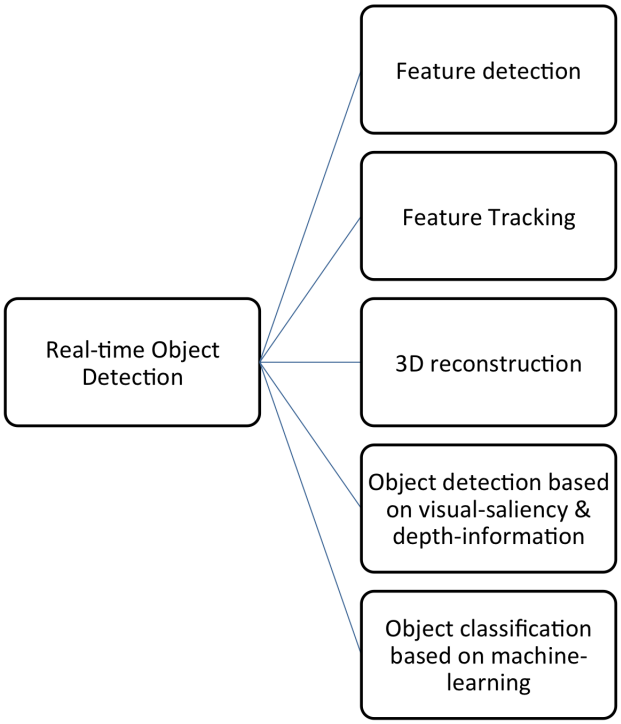


Figure 1.1: An overview of the system architecture.

Hessian-affine [34], MSER [35], Kadir–Brady [36], EBR [37], IBR [37], CenSurE [38], AGAST [39] and ORB [40]. These detectors have been selected based on their area of application. In certain applications, they are used to describe the contents of an image before performing object detection by matching features to an existing database [41]. In other cases, the features are tracked in order to perform 3D reconstruction [5, 42–45], which in turn allows prioritised object detection using the derived depth map. The real-time performance and accuracy of all these detectors is assessed, using a wide range of metrics as proposed by Mikolajczyk *et al.* [34]. This evaluation shows that there is a gap for a feature detector that is computationally efficient, while producing high keypoint density. Most contemporary approaches can only produce sparse keypoints in real-time, which lowers the probability of successful object detection.

A group of 3D reconstruction algorithms has also been reviewed, while mostly focussing on Simultaneous Localisation and Mapping (SLAM) techniques [4, 5, 44, 46–53]. Despite the recent radical improvements in this area, such algorithms are still not suitable for dense 3D reconstruction in automotive environments due to their

high dependance on the aforementioned feature detectors. In scenarios where keypoint density is low, such as motorways, the depth-map density of SLAM approaches is also lower. Finally, the computational complexity of probabilistic localisation makes SLAM hard to implement on embedded hardware [5, 44, 49, 54].

The final section of Chapter 2, examines the literature for detection and classification of targets, since it is an integral part of modern automotive safety applications such as pedestrian protection systems (PPS) [55], traffic sign recognition [56] and collision avoidance [57]. Such systems, also known as Advanced Driver Assistance Systems (ADAS) [58], pose a major machine-vision challenge since both the foreground and the background are highly changeable in terms of colour, shape and texture. For example, in the case of pedestrians there is a very high degree of variability in the type of clothing, articulated pose, skin colour, body size, aspect ratio, viewing angle, lighting direction and more importantly background complexity [8]. Developing accurate models that describe such targets is not feasible, thus machine-learning techniques are used to build an implicit representation based on examples [8]. The majority of these approaches focus on a single object class (e.g. traffic signs, cars, pedestrians) although there are also some multi-class classifiers [59, 60]. Reviewing all of these methodologies is a complex task since each object on its own has been the subject of extensive research over the past decades. As a result, the focus has been shifted to just the pedestrian class, which is one of the most studied and challenging subjects to date. In particular, the state-of-the-art in real-time pedestrian detection is described, while trying to identify limitations in the optimisation process. Different methods for performance evaluation are also investigated, while the most common public datasets are used for reliable benchmarking. The most significant gap is related to the use of prioritised image indexing to reduce the number of unsuccessful iterations. It is proposed that current object-detection approaches [61] could benefit by data fusion of 3D reconstruction algorithms and visual-saliency algorithms [10, 62–67].

In Chapter 3, a novel feature-detection approach is described for extracting stable and dense features even in images with very low signal-to-noise ratio. This methodology is based on image gradients, which are redefined to take account of noise

as part of their mathematical model. Each gradient is based on a vector connecting a negative to a positive intensity centroid, where both centroids are symmetric about the centre of the area for which the gradient is calculated. Multiple gradient vectors define a feature with its strength being proportional to the underlying gradient vector magnitude. The results clearly show superior performance over GFTT (Shi Tomasi) [28], CenSurE [38], AGAST [39], SIFT [1], SURF [2], FAST [32], and ORB [40], MSER [35] in terms of keypoint density, tracking accuracy, illumination invariance, rotation invariance, noise resistance and detection time.

In Chapter 4, the DeGraF features form the basis for two new approaches that perform dense 3D reconstruction from a single vehicle-mounted camera. The first approach tracks DeGraF features in real-time while performing image stabilisation with minimal computational cost. This means that despite camera vibration the algorithm can accurately predict the real-world coordinates of each image pixel in real-time by comparing each motion-vector to the ego-motion vector of the vehicle. The 3D reconstruction performance has been evaluated using different feature-detection methods, including the pyramidal Lucas-Kanade optical flow [68], AGAST [39], FAST [32], GFTT [28], SIFT [1] and ORB [40]. The aim is to determine their accuracy, depth-map density, noise-resistance and computational complexity. The second approach proposes the use of local frequency analysis of gradient features for estimating relative depth. This novel method is based on the fact that DeGraF gradients can accurately measure local image variance with sub-pixel accuracy. It is shown that the local frequency by which the centroid oscillates around the gradient window centre is proportional to the depth of each gradient centroid in the real world. The lower computational complexity of this methodology comes at the expense of depth-map accuracy as the camera velocity increases, but it is at least five times faster than the other evaluated approaches.

In Chapter 5, a novel technique is presented for deriving high-resolution visual-saliency maps in real-time. In this context, saliency maps show how different each image pixel is to its surrounding pixels across multiple pyramid levels. The proposed method replaces the computationally-expensive centre-surround filters [10, 62–64] with a simpler mathematical model named Division of Gaussians ($DIVoG$). The

results are compared to five different approaches [10, 69–72], demonstrating at least six times faster execution than the current state-of-the-art whilst maintaining high detection accuracy. Given the multitude of computer vision applications that make use of visual-saliency algorithms such a reduction in computational complexity is essential for improving their real-time performance. Subsequently, the saliency information is combined with depth information to identify salient regions close to the host vehicle. The fused map allows faster detection of high-risk areas where obstacles are likely to exist. As a result, existing object-detection algorithms, such as the Histogram of Oriented Gradients (HOG) [61] can execute at least five times faster.

Finally, Chapter 6 summarises the four major contributions to knowledge, namely: a) a novel real-time feature-detection algorithm, b) a novel real-time depth-estimation algorithm using monocular vision, c) a fast visual-saliency algorithm and d) a novel image-indexing algorithm for prioritising high-risk areas by fusing visual-saliency and depth information (see Table 1.1). Directions for future research in this domain are also proposed.

	Theme	**Key references**	**Contribution to knowledge**
Chapter 3	Real-time feature detection	[1, 2, 28, 32, 35, 38–40, 68]	- A new method for calculating image gradients that are robust to noise. - A feature detector based on gradients that combines real-time performance with high keypoint density.
Chapter 4	Real-time dense 3D reconstruction	[1, 28, 32, 39, 40, 68]	- A new method based on gradient features for performing dense 3D reconstruction in real-time using monocular vision.
Chapter 5 - Section 2	Real-time visual saliency	[10, 69–72]	- A new (patent-pending) method for deriving high resolution visual-saliency maps in real-time.
Chapter 5 - Section 3	Real-time object detection using prioritised image indexing.	[61]	- A new method for prioritising image indexing and accelerating object detection and classification.

Table 1.1: Existing research and contribution to knowledge

Material from this thesis is presented in the following peer-reviewed publications:

- I. Katramados, S. Crumpler, T.P. Breckon, "Real-Time Traversable Surface Detection by Colour Space Fusion and Temporal Analysis", ICVS '09 Proceedings of the 7th International Conference on Computer Vision Systems:

Computer Vision Systems Pages 265 - 274

- I. Katramados, T.P. Breckon, "Real-time visual saliency by Division of Gaussians", 2011 18th IEEE International Conference on Image Processing (ICIP), pp.1701,1704, 11-14 Sept. 2011

- L. Bordes, T.P. Breckon, I. Katramados, A. Kheyrollahi, "Adaptive object placement for augmented reality use in driver assistance systems", Proceedings of the 8th European Conference for Visual Media Production, London, 16-17 November 2011, paper number sp-1.

The following patent is pending:

- International Application Number PCT/GB2012/000705, MB ref. P8936WOP Cranfield University, "Real-Time Visual Saliency by Division of Gaussians"

Chapter 2

Visual object detection in automotive environments using monocular vision - State of the art

2.1 Overview

This chapter examines a wide range of approaches in order to assess their suitability for object detection in an automotive context. Automotive applications require computationally-efficient algorithms that execute in real-time on embedded hardware. The state-of-the-art review focusses on feature detection and tracking, depth estimation and visual-saliency techniques as a way of accelerating object detection by directly focussing on the high-risk regions of interest, while at the same time using a single monocular camera. Subsequently, a set of object-detection methodologies are analysed with focus on pedestrian detection, which is the most common and widely-studied class of objects in the field of automotive vision. The chapter concludes by identifying a set of technological limitations that are subsequently addressed by each chapter of this thesis.

2.2 Real-time image analysis

2.2.1 Feature Detection

In this section, a broad range of feature detectors are described based on the literature study. The aim is to identify a set of approaches that meet the following criteria:

- Real-time performance
- Suitability for object detection
- Stability under changing conditions in an automotive context

An image feature corresponds to an image pattern that is locally unique within neighbourhood or globally within an image [73]. This means that an image can be characterised by a subset of robust keypoints rather than by the entire set of raw pixel values. Subsequently, the keypoints can be tracked or matched between images. A wide range of feature-detection methodologies exist from basic corner-based detectors to affine detectors. The most common are the following [74]:

- *Hessian detector*

  Hessian features are extracted by calculating the determinant of the Hessian matrix, which is used to detect corners as local maxima. The determinant is rotation invariant since it is derived from the Gaussian curvature of the signal [23].

- *Moravec detector*

  Moravec introduced the idea of "points of interest" by computing the local auto-correlation function of the image in four directions. The lowest results indicate points of interest with large intensity variations in all four directions [24].

- *Förstner detector*

  Förstner features are also based on the auto-correlation function for detecting interest points, edges and regions. However, this approach is more complex with low real-time performance [25].

- *Harris detector*

  Harris features are basically corners derived from the principal curvatures of the auto-correlation function. Two high curvatures indicate an interest point that is invariant to rotation [26].

- *Tomasi-Kanade detector*

  A good feature is detected when two eigenvalues of an image patch are smaller than a pre-specified threshold [27].

- *Haralick operator*

  The Haralick operator extracts interest points as the weighted centre of gravity of all points within a predefined window. The windows of interest are selected based on their gradient and the normal matrix [29].

- *Heitger*

  The Heitger detector uses Gabor filters in different directions to extract key-points. It is computationally very demanding [30].

- *SUSAN corner detector*

  The SUSAN algorithm detects corners within circular regions that have a centroid far from the nucleus [31].

- *Scale Invariant Feature Transform (SIFT)*

  SIFT extracts features by detecting scale-space extrema [1]. It comprises four main steps:

  – Scale-space extrema detection

    In this first step, a pyramid of resolutions is generated from the input image, which is subsequently used to calculate the difference of Gaussians between pyramid levels (*Figure 2.1*). Each pixel is compared to all its neighbours in scale space, which can be visualised as a $3\times3\times3$ cube where its centre is compared to all its neighbours (i.e. 26 neighbours per pixel). An extrema is registered as a keypoint candidate if the central pixel value is greater or less than all its neighbours [1].

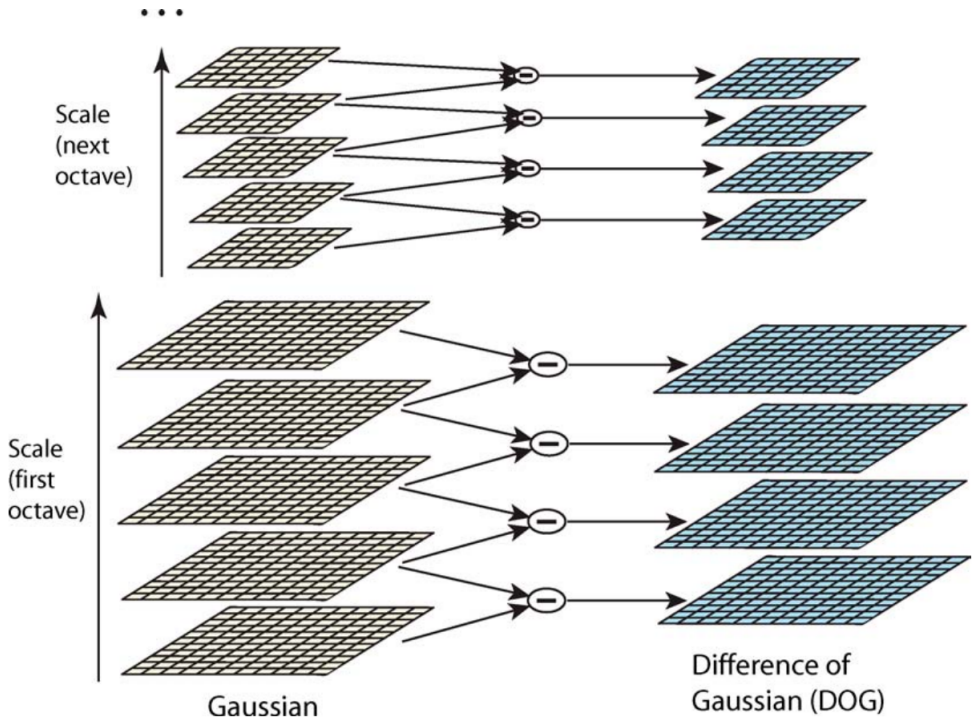


Figure 2.1: SIFT features are detected in scale-space using Difference of Gaussian (image from [1]).

– Accurate keypoint localisation

  The aim of this process is to select the most stable features among all the keypoint candidates that were extracted from scale-space extrema detection. A poor keypoint candidate is normally one with very low contrast compared to its neighbours. Such keypoints are prone to be sensitive to noise, thus they are eliminated. In addition, a threshold on the ratio of principal curvatures (edge responses) is applied for increased stability [1].

– Orientation assignment

  For each image region a histogram of orientations is built from local gradients. The peaks of this histogram denote dominant orientations of each keypoint. This way the keypoint descriptor can be expressed relative to its orientation, which makes it rotation invariant [1]. This is one of the key advantages of SIFT over previous approaches.

– Keypoint descriptor

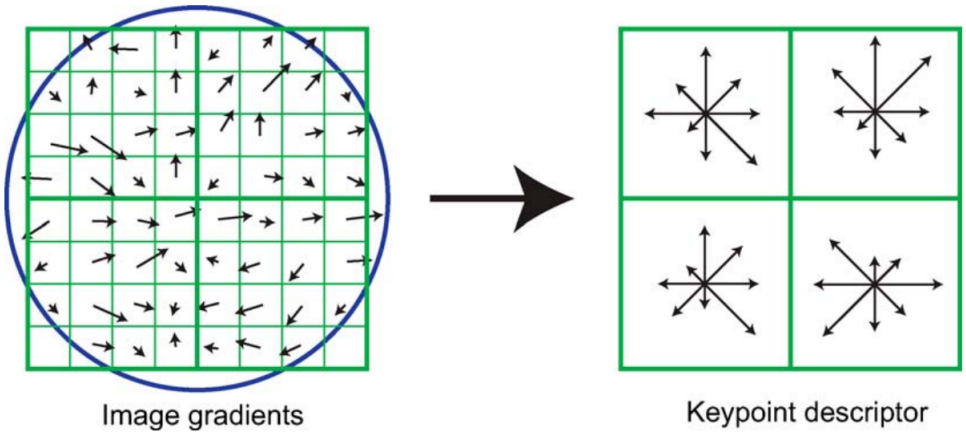


Figure 2.2: SIFT keypoint descriptor derived from image gradients (image from [1]).

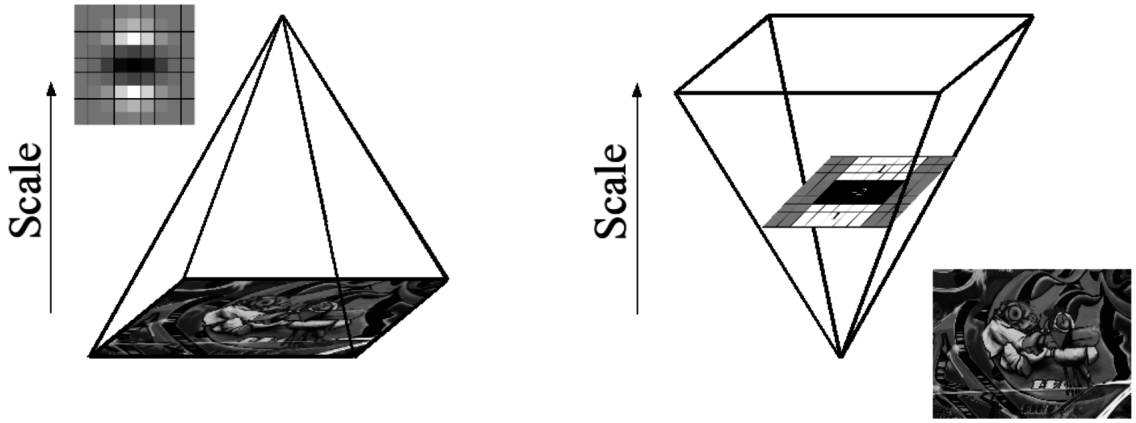


Figure 2.3: SIFT iteratively reduces the image size, whereas SURF up-scales the box filter which is computationally more efficient (image from [2]).

The SIFT descriptor is constructed by combining orientation histograms of neighbouring regions after weighting each gradient's magnitude and orientation using a Gaussian function. Lowe proposed the use of 4×4 descriptors based on 8-bin histograms. This results in a 128 elements feature vector for each keypoint [1] as illustrated in *Figure 2.2*.

- Speeded-Up Robust Features (SURF)

  SURF is a scale and rotation-invariant detector and descriptor with high repeatability, distinctiveness and robustness. It is computationally more efficient than SIFT, which makes it a good candidate for real-time image analysis [2]. It comprises the following steps:

  – Integral image calculation

Figure 2.4: SURF features with each box indicating the size and orientation of the descriptor (image from [2]).

Integral images allow the fast computation of box type convolution filters [2]. A location $a = (x, y)^\intercal$ in an integral image $I_\Sigma(a)$ equals to the sum of all pixels in the input image I within a rectangular region formed by the origin and a (see *Equation 2.1*). This means that the sum of intensities inside any rectangular region of image I can be calculated using only three additions and four memory accesses [2].

$$I_\Sigma(a) = \sum_{i=0}^{i \leq x} \sum_{j=0}^{j \leq y} I(i, j) \tag{2.1}$$

– Derivation of Hessian matrix-based interest points

Blob-like structures can be detected as locations where the determinant of the Hessian matrix is maximum [2]. For a point $\vec{a} = (x, y)$ in an image I, the Hessian matrix $\mathcal{H}(a, \sigma)$ in a at scale σ is:

$$\mathcal{H}(a, \sigma) = \begin{bmatrix} L_{xx}(a, \sigma) & L_{xy}(a, \sigma) \\ L_{xy}(a, \sigma) & L_{yy}(a, \sigma) \end{bmatrix} \tag{2.2}$$

where $L_{xx}(a, \sigma)$ is the convolution of the Gaussian second-order derivative $\frac{\partial^2}{\partial x^2} g(\sigma)$ with the image I at point a and similarly for $L_{xy}(a, \sigma)$ and $L_{yy}(a, \sigma)$ [2].

- Scale-space representation

  Probably the most important contribution of SURF is the the fast approximation of Gaussian derivatives irrespective of the filter size, which is possible due to the use of integral images. Compared to SIFT (*Figure 2.3*), the discretised and cropped Gaussian second-order partial derivatives are replaced by 9×9 box filters [2]. This means that scale-space representation can be achieved without iteratively reducing the image size as ruled by the *Difference of Gaussians (DoG)* approach. Instead the box filters of varying size are applied directly on the integral image.

- Interest point localisation

  The interest points are localised by applying non-maximum suppression in a $3 \times 3 \times 3$ neighbourhood, followed by interpolation in scale and image space. An example of SURF features being localised on a sample image is illustrated in *Figure 2.4*.

- Features from Accelerated Segment Test (FAST)

  The FAST corner detector considers a Bresenham circle of radius $r = 3$ (16 pixels in total) around a point $\vec{p}$ [75]. This point is a corner if a set of 12 contiguous pixels exist which are all brighter (or all darker) than the intensity of the candidate pixel [32]. FAST-9 and FASTer [76] are based on the same principles but are computationally much more efficient.

- *Adaptive and Generic Accelerated Segment Test (AGAST)*

  This technique is based on FAST [32, 39], but has been enhanced to use a pair of a binary decision trees for faster feature extraction, while maintaining the same corner response and repeatability as the (complete) FAST corner detector [39].

- *Centre Surround Extrema (CenSurE)*

CenSurE [38] derives the extrema of the centre-surround filters over multiple scales, using the original image resolution for each scale. They are an approximation to the scale-space Laplacian of Gaussian and can be computed in real-time using integral images [38].

- *Oriented FAST and Rotated BRIEF (ORB)*

  ORB [40] is an efficient alternative to SIFT [1] and SURF [2]. It introduces a fast and accurate orientation component to FAST [32], while proposing a more efficient way of computing the oriented BRIEF features [77].

- *Harris-affine detector*

  The Harris affine detector detects interest points at different scales using the the second-moment matrix of Harris corners. The first step determines localisation and scale using the Harris-Laplace detector. Subsequently, affine shape adaptation is used to normalise candidate regions of interest, which are iteratively estimated by selecting the proper integration and differentiation scale [33, 34, 74].

- *Hessian-affine detector*

  The Hessian-affine detector is highly similar to Harris-affine detector, except that interest points are chosen when the determinant of the Hessian matrix is maximum [33, 34, 74].

- *Maximally Stable Extremal Regions (MSER)*

  The MSER detector extracts regions closed under continuous transformation of the image coordinates and under monotonic transformation of the image intensities [35, 74].

- *Kadir–Brady saliency detector*

  The Kadir–Brady detector identifies salient circle regions at different scales based on the Shannon entropy of local image attributes [36, 74, 78, 79].

- *EBR detector*

The EBR detector extracts regions by combining Harris corners with Canny edges [37, 74].

- *IBR detector*

  The IBR detector extracts affine-invariant regions studying the image intensity function and its local extreme [37, 74].

2.2.1.1 Feature-detector-performance evaluation

A keypoint descriptor has to be highly distinctive allowing a single feature to be identified consistently in an image sequence or database of images [1]. Performance evaluation of feature detectors is a crucial step for identifying the right level of trade-off between real-time performance and accuracy. Mikolajczyk et al. [37] have presented a framework that associates feature-detection performance to:

- **Number of correspondences:** "*The number of corresponding regions detected in images under different geometric and photometric transformations.*" [37]
- **Repeatability of features (%):** "*The repeatability score for a given pair of images is computed as the ratio between the number of region-to-region correspondences and the smaller of the number of regions in the pair of images.*" [37]. Two regions are considered as corresponding only if the overlap error is smaller than a certain threshold, as described below [37]:

$$1 - \frac{R_{\mu_a} \cap R_{(H^T \mu_b H)}}{\left(R_{\mu_a} \cup R_{(H^T \mu_b H)}\right)} < threshold \tag{2.3}$$

where:

R_μ : the elliptic region defined by $x^T \mu x = 1$

H: the homography of two temporally-adjacent frames

A more detailed evaluation and comparison of different feature-detection approaches is performed in Chapter 3, where additional criteria have been added including keypoint density, tracking accuracy, illumination invariance, rotation invariance, noise resistance and detection time.

2.2.2 Feature Tracking

Feature and region tracking is an integral part of most automotive vision applications [80]. The most common approaches can be separated into three groups: a) point tracking, b) kernel tracking and c) silhouette tracking [81].

- Point Tracking

  This approach requires the detection and matching of interest points between image frames. Managing uncertainty is key to creating an accurate point-tracking methodology. In this context there are two main types of point trackers:

  - Deterministic methods

    Deterministic methods use qualitative motion heuristics to constrain the correspondence problem [81,82]. Certain feature correspondences are excluded after applying motion constraints. Such constraints include proximity, maximum velocity, smooth motion assumption, rigidity and proximal uniformity [81]. Common tracking algorithms that belong to this category are *Modified Greedy Exchange (MGE)* [83] and *Greedy Optimal Assignment (GOA)* [82].

  - Statistical methods

    Statistical methods model the object/feature properties such as position, velocity, and acceleration using a state-space approach [81]. Common tracking algorithms that belong to this category are *Kalman* filter [84], *Probabilistic Data Association* (PDA) [85], *Probabilistic Multi-Hypothesis Tracker* (PMHT) [86].

- Kernel Tracking

  Kernel-based approaches track objects by calculating the motion of the kernel in consecutive frames using parametric transformation such as translation, rotation, and affine [81]. There are two main types of kernel trackers:

  - Template and density-based appearance models

Common tracking algorithms that belong to this category are *Meanshift* [87], *Kanade-Lucas-Tomasi* (KLT) [27, 28] and Layering [88].

– Multi-view appearance

Common tracking algorithms that belong to this category are *Eigentracking* [89], *Support Vector Tracking* (SVT) [90].

- Silhouette Tracking

Silhouette-tracking approaches are based on the estimation of image regions in each frame. In the temporal domain they use priors generated from previous frames to estimate the position of each region using appearance density and shape models usually in the form of edge maps [81]. There are two main types of silhouette trackers:

– Contour evolution

Common tracking algorithms that belong to this category are *Active Contours* [91], *Variational methods* [92], *Heuristic methods* [93].

– Matching shapes

Common tracking algorithms that belong to this category are *Hausdorff* [94] and *Hough transform* [95].

Feature tracking is further examined in Chapter 3 and Chapter 4 for performing real-time 3D reconstruction.

2.2.3 Optical Flow

Starting with a point $[u_x, u_y]^\intercal$ in image I_t, optical flow is used to find the corresponding point $[u_x+\delta_x, u_y+\delta_y]^\intercal$ in image I_{t+1}that minimises energy ε. There are two main approaches to solving this problem: a) performing local summation of overlapping regions (known as *patch-based* or *window-based* approach), b) using regularisation or Markov random fields to search for a global minimum [96]. The patch-based approach is usually based on Taylor series expansion of the displaced image function in order to obtain sub-pixel estimates as proposed by Lucas and Kanade [97]. Performance can be improved by using a coarse-to-fine pyramid scheme to estimate

larger motions using a series of discrete search steps [98]. However, uncertainty and errors can easily propagate in a pyramidal approach. Thus several approaches have adopted optimisation methods for uncertainty estimation based on Markov random fields [96]. Overall, there are more than 60 optical flow approaches with excellent evaluation results on the Middlebury dataset [99]. Optical flow is further examined Chapter 4 for performing 3D reconstruction.

2.2.4 Shadow removal

Dynamic environments are often associated with changes in lighting, which may cause shadows and reflections. Detecting and eliminating these features from an image is essential in visual object extraction before identifying the boundaries of each surface. Since this is a major problem in a wide range of computer vision applications, there is a wealth of different approaches available [100–105]. For example, Tao *et al.* [105] propose the use of a fuzzy neural network that has been trained to recognise shadows and water prints. Furthermore, lighting artefacts can be detected by combining the hue and saturation components of the HSV colourspace [100], which in contrast to the RGB colourspace describes colour perception more closely to the human visual system. In a similar approach, Cucchiara *et al.* [101] propose the use of chrominance and luminance to further improve accuracy of object detection. Analysing the results of all these approaches, shows that there is no single colourspace that is shadow invariant under all lighting conditions. Based on this observation, Alvarez *et al.* have created a shadow-invariant feature space combined with a model-based classifier [106]. Finally, detection of moving shadows is addressed by statistical analysis as proposed by Prati *et al.* [103]. Although it is difficult to objectively evaluate the performance of all these methodologies without applying them on a common data-set, their performance is better in environments with light shadows and light reflections [106]. In contrast, dark shadows cannot be handled as efficiently as they tend to alter the visual properties of surfaces and especially their hue. These techniques are further examined in the additional chapters in the Appendix.

2.2.5 Extracting regions of interest using visual saliency

As a concept, visual saliency started as a biologically-inspired process for focusing visual attention to certain parts of an image, thus reducing the complexity of scene analysis [107]. Subsequently, it formed the basis of several computer vision applications, such as in automatic object detection [108–111], medical imaging [112] and robotics [65]. Different saliency definitions exist, however, in this thesis a generalised version of the definition by Achanta *et al.* [10] is used: "*Visual saliency is the perceptual quality that makes a group of pixels stand out relative to its neighbours*". As a research topic, visual saliency theory has evolved rapidly to produce a wide range of approaches. However, their computational cost remains significantly high for real-time applications that require execution at full frame rate ($\geqslant 25$ frames per second (fps)).

Most of the visual saliency models can be categorised into two main groups, as proposed by Achanta *et al.* [62] and Ngau *et al.* [113]: a) biological models and b) computational models. The majority of biological models are using a bottom-up approach for feature extraction mainly based on colour, intensity and orientation [69]. Inspired by the structure of the human eye, this approach detects the contrast difference between an image region and its surroundings, which is also known as centre-surround contrast. Itti *et al.* [69] use the *Difference of Gaussians* (*DoG*) filter for deriving the centre-surround contrast, whereas Walther and Koch [114] take this algorithm further by adopting the concept of salient proto-objects. A common characteristic of these approaches is that they usually produce saliency maps that lack sharpness and detail [111]. Furthermore, the complexity of the biological models means that performance is slow, thus they are more suitable for use in non-real-time applications. One of the few exceptions is found in the approach proposed by Ma and Zhang [70], who calculate the centre-surround contrast by fuzzy growing. The computation takes approximately 60 milliseconds for a 320×240 image on a 2.6 GHz CPU [10], which corresponds to 16.6 fps.

Examples of computational-saliency methods include frequency-tuned salient region detection by Achanta *et al.* [10], graph-based visual saliency by Harel *et al.* [72], affine-invariant salient region detection by Kadir *et al.* [79] and real-time visual-

attention system using integral images by Frintrop *et al.* [115]. The method by Frintrop *et al.* [115], is one of the most successful attempts to produce a real-time visual-saliency algorithm (known as VOCUS) using integral images to reduce execution time. The improvement in performance is impressive with a 400×300 image being processed in approximately 50 milliseconds using a 2.8 GHz CPU, which corresponds to 20 fps. In addition, the approach proposed by Achanta *et al.* [10] comes close to achieving real-time performance by using frequency domain analysis to produce full resolution saliency maps. The execution time for a 400×300 image is 100 milliseconds on a 2.4 GHz notebook. Although, this algorithm is proportionally slower than Frintrop *et al.* [115], it generates maps with significantly higher quality. Ultimately, the target of a new saliency algorithm would be to produce saliency maps of similar quality to those by Achanta *et al.* [10, 64] at full frame rate ($\geqslant$ 25 fps). Visual saliency is further examined in Chapter 5 for performing prioritised image indexing for faster object detection.

2.3 3D reconstruction using monocular vision

Visually localising objects in space requires the estimation of the real-world coordinates of each image pixel. In automotive applications, this transformation between coordinate systems is essential in order to calculate the angle and distance between the host and the target, which can later be used in a wide range of applications such as collision avoidance. One way of addressing this problem is by using stereo-vision to create a depth map based on the disparity between two images of the same scene taken from slightly different angles. On the other hand, in monocular vision there is only one image available at a time, thus reconstructing a three-dimensional scene becomes far more complex The reason is that the temporal properties of multiple sequential images need to be analysed in real-time before estimating the real-world coordinates of each image pixel. Despite that limitation, the majority of existing production-level automotive systems use monocular vision and recover the lost dimension by making assumptions about the visual and geometric properties of the environment, while using pre-defined vehicle-motion models for increased reliabil-

ity [14, 15]. In addition to these approaches the robotics community has developed a wide range of Simultaneous Localisation and Mapping techniques (also known as SLAM) for estimating the position of the vehicle and the objects in space by tracking image features across different frames.

2.3.1 Inverse perspective mapping

A digital image from a monocular camera represents a real-world scene, where the projected object's size is proportional to its distance from the camera due to the perspective effect [16]. Robotic and automotive systems often remap the 2D image onto a grid of real world coordinates by using a methodology known as the inverse perspective transform. This approach is based on the following assumptions:

- The host vehicle and the target object are on the same plane
- The camera height is known
- The focal length is known
- The pixel width and height are known
- The camera model is very close to that of a pin-hole camera

If all the conditions above are satisfied then object localisation can be performed by applying simple trigonometry. For the sake of simplicity, let us assume that the camera xy coordinate system and the real-world xy coordinate system are parallel as illustrated in *Figure 2.5.* Subsequently, the calculation of the distance z_W between the camera lens and a real-world point P_W is given by the following equation:

$$z_W = \frac{f\,h}{y_R} \tag{2.4}$$

where:

f: the focal length (pre-defined)

h: the camera height (pre-defined)

y_R: the projected height of point P_W on the camera plane (equals to image row×pixel height)

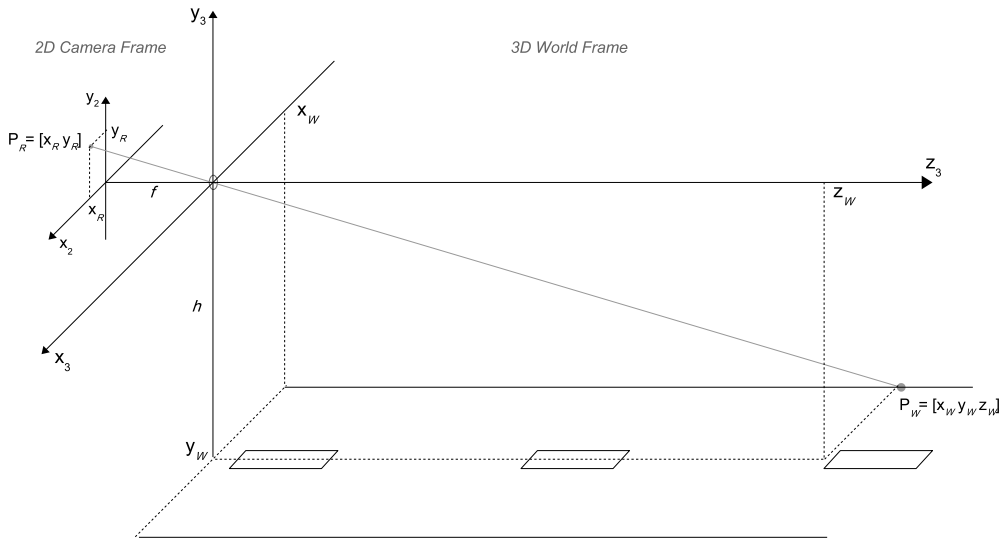


Figure 2.5: Real-world projection in image plane

In the case that the camera xy plane is not parallel to the real-world xy plane then *Equation 2.4* can be adjusted to take into account the roll, pitch and yaw angle, making inverse-perspective transform a very powerful technique for obstacle localisation on flat surfaces. On the other hand, such a technique cannot be used for generic obstacle detection, since there is no guarantee that the host vehicle and the target object will lie on the same plane.

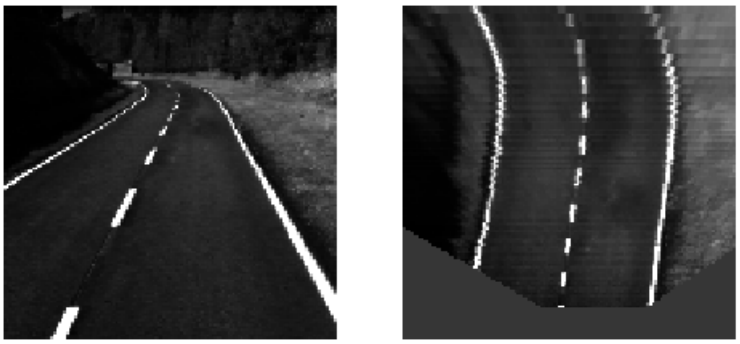


Figure 2.6: Inverse Perspective Transform example - Courtesy of Bertozzi *et al.* [3]

2.3.2 SLAM Approaches

Simultaneous localisation and mapping (SLAM) is a technique commonly used in robotics to incrementally build a map of an unknown environment while estimating the position of the host vehicle. As a methodology it can be found in a wide range of applications using both monocular and stereo vision sensors as well as laser range finders, sonar and data-fusion platforms [116,117]. However, this section only examines how SLAM can be applied in monocular vision applications and introduces the theoretical concepts that form the basis of this technique. *Figure 2.7* illustrates the essence of SLAM which is mathematically described by the following probability [4]:

$$P(\mathrm{x}_k,\mathrm{m}|\mathrm{Z}_{0:k},\mathrm{U}_{0:k},\mathrm{x}_0) \tag{2.5}$$

where:

k : present time

x_k: the vehicle state vector for location and orientation at time k

m: a vector of all landmarks

$\mathrm{U}_{0:k}$: the history of all control vectors applied to the vehicle

$\mathrm{Z}_{0:k}$: the history of all landmark observations from the vehicle's perspective

In other words, at any time k, the vehicle reaches a state x_k following a control input u_k. Thus the probability distribution 2.5, describes a recursive solution where the joint posterior is computed using Bayes theorem, by taking into account the distribution $P(\mathrm{x}_{k-1},\mathrm{m}|\mathrm{Z}_{0:k-1},\mathrm{U}_{0:k-1})$ at time $k-1$. As Durrant-Whyte and Bailey explain in [4], this computation requires a vehicle motion model and an observation model, which are described below:

$$Motion\ Model:\ P(\mathrm{x}_k|\mathrm{x}_{k-1},\mathrm{u}_k) \tag{2.6}$$

$$Observation\ Model:\ P(\mathrm{z}_k|\mathrm{x}_k,\mathrm{m}) \tag{2.7}$$

Since SLAM continuously tries to estimate the vehicle position and the position of the landmarks there are two mechanisms required for prediction (time update) and

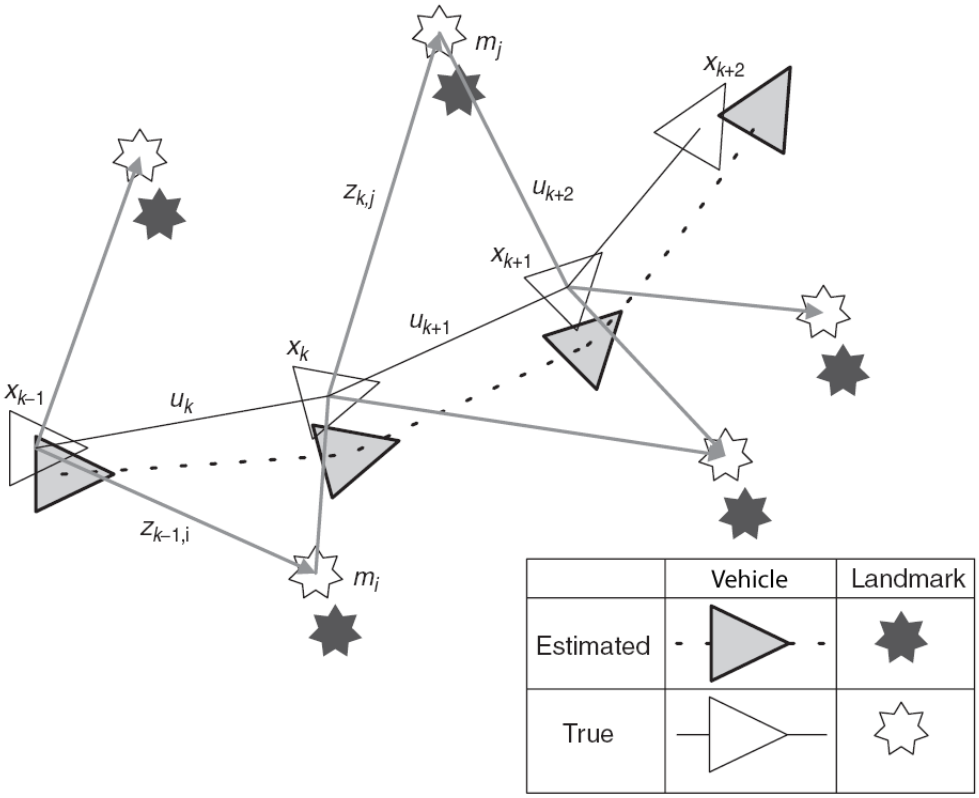


Figure 2.7: SLAM Description: Simultaneous estimation of vehicle's and landmark's position. Note the error between the actual and the estimated locations. Courtesy of Durrant-Whyte and Bailey [4]

correction (measurement update) as specified by the following equations [4]:

Time Update :

$$P(\mathrm{x}_k, \mathrm{m}|\mathrm{Z}_{0:k-1}, \mathrm{U}_{0:k}, \mathrm{x}_0) = \int P(x_k|x_{k-1}, u_k) P(\mathrm{x}_{k-1}, \mathrm{m}|\mathrm{Z}_{0:k-1}, \mathrm{U}_{0:k-1}, \mathrm{x}_0) \mathrm{dx}_{k-1} \tag{2.8}$$

Measurement Update:

$$P(\mathrm{x}_k, \mathrm{m}|\mathrm{Z}_{0:k}, \mathrm{U}_{0:k}, \mathrm{x}_0) = \frac{P(\mathrm{z}_k|\mathrm{x}_k, \mathrm{m}) P(\mathrm{x}_k, \mathrm{m}|\mathrm{Z}_{0:k-1}, \mathrm{U}_{0:k}, \mathrm{x}_0)}{P(\mathrm{z}_k|\mathrm{Z}_{0:k-1}, \mathrm{U}_{0:k})} \tag{2.9}$$

Based on the theory above, two main approaches have been developed known as EKF-SLAM and FastSLAM [48, 51, 53]. EKF-SLAM makes use of the extended Kalman Filter [118] assuming additive Gaussian noise, whereas FastSLAM makes use of the Rao-Blackwellized particle filter by assuming a non Gaussian probability distribution. Both approaches are presented in depth by [4, 116].

Although SLAM is a very powerful technique for solving the localisation problem it also comes with drawbacks such as:

- **Noise:** Noise in SLAM is statistically-dependent which means that errors accumulate over time and thus affect future measurements [119, 120]. Thus accurate noise models or noise filters are required for improved consistency.

- **Performance:** In its basic form SLAM complexity increases quadratically every time a new landmark is detected [116] assuming that all landmarks are updated after every observation. This makes SLAM computationally expensive and often unsuitable for low-cost real-time systems. However, in recent times a multitude of methodologies have been developed for performance optimisation. Examples of such improvements include: state augmentation, sparsification, partitioned updates and submapping. Specifically, state augmentation focuses on reducing computation complexity in the time-update process (described in *Equation 2.8*), whereas, partitioned updates are used to accelerate the measurement update process (described in *Equation 2.9*) by updating only those landmarks that have changed since the last measurement.

- **Data association**: SLAM operation is based on making measurements and associating them with existing landmarks. However, it is possible that data association comes across two indistinguishable landmarks, leading to an incorrect estimate and subsequently to a non-reversible failure [120]. This problem and its solutions are extensively discussed in [116].

- **Environmental changes:** Features in an automotive environment have a different visual signature under varying conditions. Since SLAM depends on accurate feature matching between observations and existing landmarks such environmental changes may lead to false localisation estimates [120]. This drawback also refers to moving objects which may be registered as landmarks leading to incorrect maps. So the question is how can we build an accurate map in dynamic environments? Some papers propose the pre-processing of video data in order to remove dynamic features before applying the SLAM algorithm [116]. Alternatively, it is possible to track stationary and moving

landmarks within a SLAM system however the processing cost is very high. Such an approach is proposed by Fox and Burgard in [121].

Regarding SLAM implementations there is a wealth of solutions within the robotics and autonomous vehicles community, focusing on both indoors [5,44,50,52,122] and outdoors environments [47,51]. However, SLAM systems for automotive applications are limited mainly due to their weakness to deal with changing environmental conditions and moving targets. Current solutions propose the filtering of moving features before applying SLAM [45] or alternatively the introduction of feature tracking for moving objects [51].

As a methodology SLAM has been extensively applied in the robotics community, however, the first successful implementation of a monocular vision-based SLAM system was by Davison *et al.* [5] as outlined in the next paragraph.

2.3.3 Mono-SLAM

MonoSLAM is a technique by Davison *et al.* for recovering "the 3D trajectory of a monocular camera, moving rapidly through a previously unknown scene" [5]. According to its authors this system "is the first successful application of the SLAM methodology from mobile robotics to the 'pure vision' domain of a single uncontrolled camera, achieving real-time but drift-free performance". The following paragraphs outline the main characteristics of this technique mainly based on the core methodology description by Davison *et al.* [5] as well as other contributions by various authors [43,46,49,50,123].

2.3.3.1 3D Scene Reconstruction

Based on a technique by Smith *et al.* [124], Mono-SLAM builds a probabilistic 3D map of the environment by processing sequential images from an uncontrolled monocular camera and continuously tracking features of interest, estimating the motion of the camera and calculating the uncertainty of these estimates [5]. In addition, the map is constantly updated with new estimates, which are derived using an Extended Kalman Filter (EKF). The mathematical representation of the

map is given by the following equation:

$$\hat{\mathrm{x}} = \begin{pmatrix} \hat{x_u} \\ \hat{y_1} \\ \hat{y_2} \\ \vdots \end{pmatrix}, \qquad \mathrm{P} = \begin{bmatrix} P_{xx} & P_{xy_1} & P_{xy_2} & \dots \\ P_{y_1x} & P_{y_1y_1} & P_{y_1y_2} & \dots \\ P_{y_2x} & P_{y_2y_1} & P_{y_2y_2} & \dots \\ \vdots & \vdots & \vdots & \end{bmatrix} \tag{2.10}$$

where:

x: a state vector of all the estimated camera states and features

y_i: the 3D position vectors of the locations of point features as illustrated in *Figure* 2.8

P: a square covariance matrix comprising of covariance sub-matrices

Furthermore,x_u is the camera's state vector as described below:

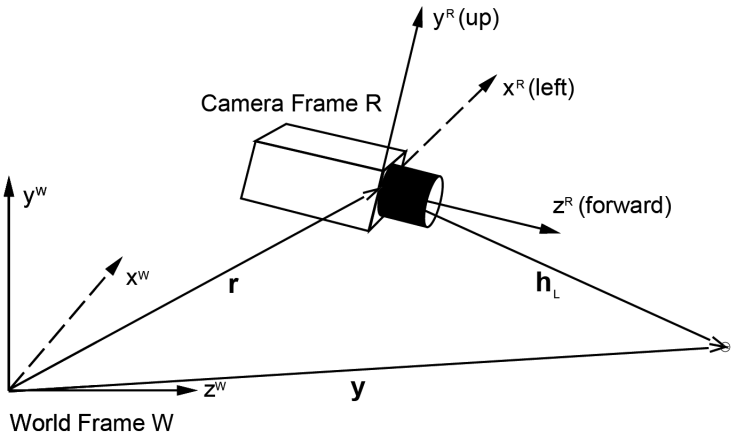


Figure 2.8: "Frames and vectors in camera and feature geometry". Courtesy of Davison *et al.* [5]

$$\mathrm{x}_u = \begin{pmatrix} r^W \\ q^{WR} \\ v^W \\ \omega^R \end{pmatrix} \tag{2.11}$$

where:

r^W: a metric 3D position vector

q^{WR}: the orientation quaternion

v^W: the velocity vector

ω^R: the angular velocity vector

It is important to note that the derived map is not a full 3D representation of the scene, but a map of a sparse set of features that correspond to important landmarks. The remainder of this section describes the process by which these landmarks are chosen, tracked and mapped.

A. Feature Selection

The feature selection process in a SLAM system is directly related to the environment in which the system is required to operate. Davison *et al.* [5] apply a technique by Shi and Tomasi [28] to automatically detect features by using relatively large image patches (11x11 pixels) as visual landmarks. This choice is based on the assumption that larger image segments increase the probability of selecting unique features that can be used for mapping of a small area (e.g. a room) while the camera is freely moved in all directions. On the other hand, in an automotive environment the requirements are different in the sense that the system must detect at least one feature per obstacle, which may vary in size. Thus choosing the right approach requires careful evaluation of different alternatives (e.g. corner detectors).

B. Map Initialisation

Before using MonoSLAM for mapping of an unknown environment, the system needs to go through an initialisation process [5]. This means that the algorithm needs some pre-existing knowledge about the real-world size of of some pre-recorded features in order to:

a) create a precise map scale

b) initialise the estimation process with features that have very low uncertainty, which is certain to improve future performance

This phase is mostly required if the camera is freely moving in an unknown environment. On the other hand, in an automotive environment there are some fixed parameters such as the fixed height and position of the horizon, which make this step redundant.

C. Motion Modelling and Prediction

MonoSLAM requires the use of a camera motion model in order to improve the accuracy of predictions and thus the quality of the estimates [5]. When using a freely moving camera such a model cannot be derived unless some assumptions are made. Specifically, Davison *et al.* [5] consider that the velocity and acceleration of the camera is constant between two sequential frames at 30Hz frame-rate. Although, this assumption has given good results, other techniques make use of more restricted kinematic models [116].

D. Active Feature Measurement and Map Update

One of the major strengths of MonoSLAM is its real-time performance, which partly relies on the use of an active search technique for updating existing features on the map [5]. By predicting the position of the camera and the state of the features, the algorithm expects to find most features within a limited distance from their original position. In other words, this approach firstly defines areas of interest before matching the features, whereas other approaches extract all the image features before matching them to the existing ones [5]. Either of these approaches come with their strengths and weaknesses and the choice is dependent on the application and the expected uncertainty level of the estimates.

E. Depth Estimation

Once a feature is detected in an image its depth is unknown. One way of estimating it would be by tracking the feature and estimating the camera position across a number of sequential frames. Subsequently, the depth of the feature can be derived via triangulation as in stereo vision [5]. However, depending on the amount of features this estimation can become computationally very expensive. In contrast, MonoSLAM has opted out for a different methodology [5], where each new feature is assumed to lie along a 3D line that denotes all the possible depth values. Then a set of hypotheses about the depth of this feature are generated and evaluated across different observations on a frame-by-frame basis using Bayes rule. Once the

uncertainty associated with the depth estimation drops below a certain threshold then the feature is added to the 3D map.

Regarding hardware, it is worth mentioning an experiment performed by Davison *et al.* [44, 50], which proved that using a wide-angle monocular camera (approximately 100 degrees field-of-view) improves the MonoSLAM performance. Practically, this improvement is based on the fact that larger field-of-view allows the object to stay longer in the image thus more observations are made and the 3D map becomes more accurate.

In conclusion, MonoSLAM is powerful technique for real-time object localisation in completely unstructured environments. So far there has been no research on automotive MonoSLAM applications, thus it would be interesting to evaluate its performance when the camera is mounted on a vehicle in a dynamic environment. However, in that case several challenges would have to be addressed including management of dynamic features on the 3D map as well as localisation of objects with low motion parallax. Further analysis of the MonoSLAM approach can be found in [5, 43, 46, 49, 123].

2.3.4 Monocular or Stereo Vision?

Probably the first approach that comes to mind when solving the problem of object detection in 3D space is stereo vision, because the depth information can be derived through basic image processing. The fact is that nowadays stereo vision is a well-established technique that has been thoroughly researched and implemented even in real-time embedded applications. Although it is generally more expensive in terms of required hardware resources and processing overhead, its performance has been proved across a multitude of applications including Mars exploration missions [125]. However, if one of the two cameras fails then the whole system fails since epipolar geometry can no longer be applied in the same way. On the other hand, in nature mammals can still cope very well with one eye [126], which may lead to the conclusion that stereo-vision and monocular-vision should not be considered as two entirely different methodologies. In this context, techniques such as MonoSLAM [5] and Make3D [127] have demonstrated new ways of extracting

3D information from 2D images. Subsequently, stereo-vision systems could benefit from such advances in monocular vision to create more robust and fault-tolerant computer vision approaches.

2.4 Object detection and classification for automotive applications

2.4.1 Overview

Detection and classification of targets is an integral part of modern automotive safety applications such as pedestrian protection systems (PPS) [55], traffic sign recognition [56] and collision avoidance [57]. Such systems, also known as Advanced Driver Assistance Systems (ADAS) [58], pose a major machine vision challenge since both the foreground and the background are highly changeable in terms of colour, shape and texture. For example, in the case of pedestrians there is a very high degree of variability in the type of clothing, articulated pose, skin colour, body size, aspect ratio, viewing angle, lighting direction and more importantly background complexity [8]. Developing accurate models that describe such targets is not feasible, thus machine learning techniques are used to build an implicit representation based on examples [8].

There is a wide range of machine learning techniques developed for detection and classification of objects in automotive environments. The majority of these approaches focus on a single class of objects (e.g. traffic signs, cars, pedestrians) although there are also some multi-class classifiers [59, 60]. Reviewing all of these methodologies is a complex task since each object on its own has been the subject of extensive research over the past decades. In this section, the focus will be on the pedestrian class, which is the most studied and challenging subject to date. In particular, the state-of-the-art in real-time pedestrian detection will be described, while trying to identify gaps in the optimisation process. Different methods for performance evaluation are also investigated, while the most common public datasets are examined for reliable benchmarking.

2.4.2 Image pre-processing

Automotive vision applications need to cope with highly variable weather and lighting conditions in structured, semi-structured and unstructured environments. Achieving high reliability in object detection requires dynamic adjustment of camera parameters in order to reproduce shades and/or colours as accurately as possible. The main techniques used are:

- *Dynamic Range*
  - *High Dynamic Range (HDR)*

    Camera pixels sense light in terms of brightness, which is digitally expressed an as 8-bit or 16-bit value. However, there are often cases where the range of brightness within a scene is so wide that certain image areas appear over-saturated or under-saturated leading to loss of visual information. HDR techniques aim to recover this information by generating an image with non-linear brightness representation [128]. The most common HDR methodologies fuse multiple images of the same scene while varying exposure settings [6]. This way saturation levels are normalised to better reflect the true shades / colours of a scene. For example, this approach is useful when a car is entering or exiting a tunnel, where there is a step change in illumination. This causes underexposure of the scene at the entrance of the tunnel and overexposure at the exit.
  - *Adaptive Dynamic Range (ADR)*

    ADR is able to temporarily adjust the exposure of each individual pixel using a controllable optical attenuator [6]. As *Figure 2.9* illustrates, the exposure of each pixel continuously adapts to the scene's radiance by adjusting the transmittance of the attenuator based on the brightness value of each pixel. According to Nayar *et al.* [6] "this ability to vary the exposures of individual pixels over time gives adaptive imaging a significant edge over conventional HDR techniques".

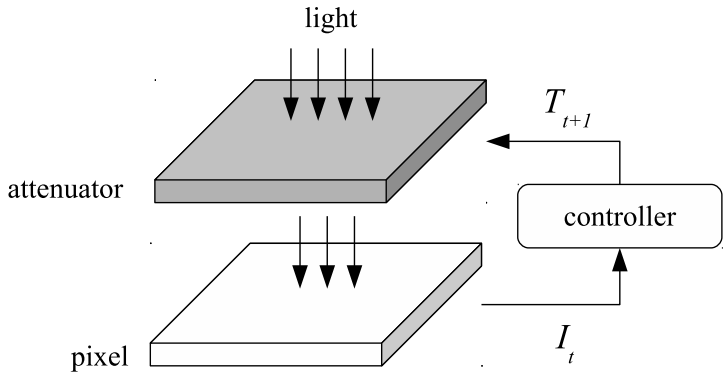


Figure 2.9: Adaptive Dynamic Range (ADR) model. The brightness I of a pixel measured at time t is used to adjust the transmittance T of the attenuator at time $t+1$ [6].

- Multi-resolution image pyramid

  Several object recognition methodologies use a pyramid of resolutions for multi-scale detection. The most popular approaches include the Gaussian and the Laplacian pyramids [129].

- Visual-saliency-map generation

  Image indexing can be accelerated by using via visual-attention algorithms that prioritise candidate locations where an object might exist. This is achieved by measuring the saliency of neighbouring areas throughout the image as described in section 2.2.5.

- Gamma normalisation

  Some approaches [61, 130] use gamma correction to reduce the effect of illumination variance on object detection. This is computed either by calculating the square root or the log of each pixel value [131].

2.4.3 Selecting a region of interest

Localising an object in an image in real-time can be achieved by reducing the search area size. Instead, the majority of object-detection approaches use a sliding window of predefined aspect ratio to scan the entire image [8, 132]. This brute-force approach is computationally very expensive since a moving classifier is searching for

maximal detection responses at all possible positions and scales [133]. Additionally, some automotive approaches restrict the search space by making certain assumptions about the environment (e.g. flat-world, ground-based objects, expected height range and expected aspect ratio) and using fixed camera parameters [8, 55, 61]. Furthermore, the foreground can be separated from the background using stereo depth-map segmentation. In the case, of monocular cameras this can be achieved by motion segmentation between the ego-motion field of the camera and the observed optical flow [8].

2.4.4 Object detection & classification

Object detection and classification comprises numerous approaches based on different cues such as gradients, motion vectors, shape and texture. In this section, a small subset of the most widely used approaches are presented. For a full evaluation of the state of the art researchers can refer to the PASCAL Visual Object Challenge [7], where the best approaches are evaluated every year. *Figure 2.10* illustrates the person detection results for 2011.

2.4.4.1 Histograms of oriented gradients with linear support vector machines

Histogram of Oriented Gradients (HOG) has been at the centre of numerous object detection methodologies and especially pedestrian detection [8, 55]. Originally proposed by Dalal and Triggs [61], this approach models local shape and appearance using dense normalised histograms of oriented gradients. Based on Enzweiler and Gavrila definition [8] "*local gradients are binned according to their orientation, weighted by their magnitude, within a spatial grid of cells with overlapping blockwise contrast normalisation. Within each overlapping block, a feature vector is extracted by sampling the histograms from the contributing spatial cells. The feature vectors for all blocks are concatenated to yield a final feature vector, which is subject to classification using a linear support vector machine (linSVM)*" *[8].* The main drawback of the original approach is that it is based on a sliding window scanning the entire image at different scales, which makes it computationally inefficient for real-time

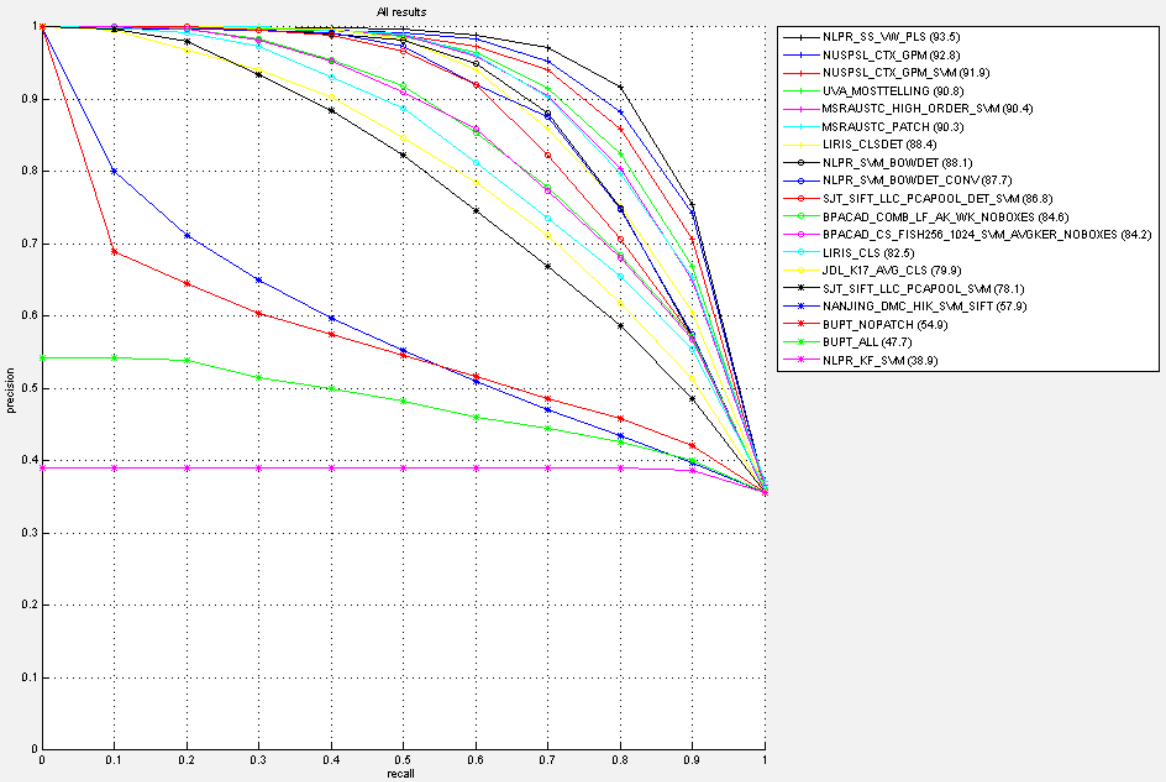


Figure 2.10: Person detection precision-recall graph from the PASCAL Visual Object Challenge 2011 (image from [7])

systems.

2.4.4.2 Neural network using local receptive fields (NN/LRF)

Local receptive fields are limited local regions within an image, connected to specific neurones. In a neural network, each branch is formed of several neurones that share synaptical weights, thus each neural branch forms a spatial feature detector [8]. Object detection can be performed using these features to train a linear support vector machine in combination with a multilayer feed-forward neural network architecture (NN/LRF) [8,134]. Non-linear support vector machines have shown better performance but their training algorithm is computationally very expensive and requires vast amounts of available memory space [8,135].

2.4.4.3 Haar-wavelet-based cascade

Haar-wavelet cascades are computationally more efficient than sliding window approaches due their decision tree architecture that quickly rejects areas of the image

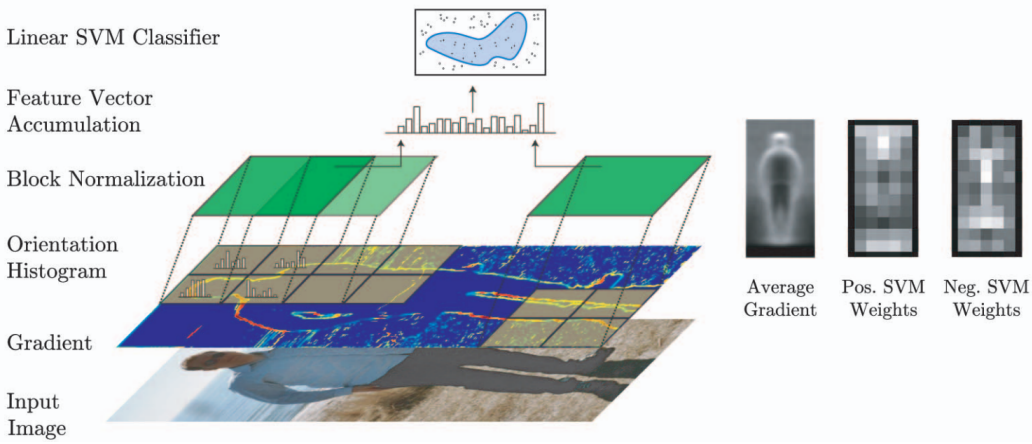


Figure 2.11: Approach overview of Histogram of Oriented Gradients (image from [8])

where an object is unlikely to exist [59, 136]. Haar-wavelet features are extracted at different scales and locations using different types of Haar wavelets (*Figure 2.12*). AdaBoost [137] is used to construct a classifier at each cascade layer using weighted linear combination of selected features of both positive and negative samples [8].

Figure 2.12: Different types of Haar wavelets (image from [8])

2.4.4.4 Combined shape-texture-based detection

This approach is interesting because it combines neural networks (NN/LRF) for classifying image regions based on texture and hierarchical shape-based detection using the Chamfer distance [8, 138]. It is based on the PROTECTOR system [139], which can be implemented in real-time when using a monocular camera. *Figure 2.13* gives a brief overview of this approach for detecting pedestrians.

2.5 Conclusions

This chapter reviewed the state-of-the-art and a number of technological limitations were identified. The focus has been on real-time feature detection and tracking, depth estimation using monocular vision, visual saliency techniques and object detection methodologies. Firstly, a broad range of feature detectors have been reviewed

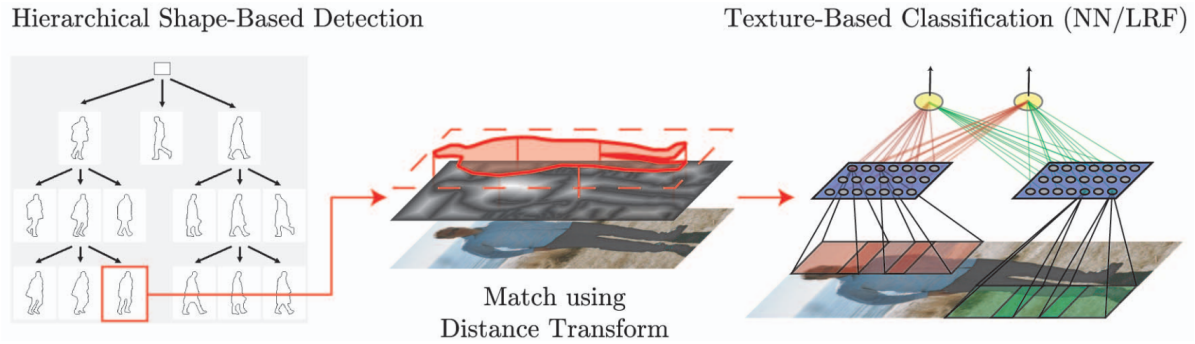


Figure 2.13: Combined shape-based detection and texture-based classification(image from [8])

including Hessian [23], Moravec [24], Förstner [25], Harris [26], Tomasi Kanade [27], Shi Tomasi [28], Haralick [29], Heitger [30], SUSAN [31], SIFT [1], SURF [2], FAST [32], Harris-affine [33], Hessian-affine [34], MSER [35], Kadir–Brady [79], EBR [37], IBR [37], CenSurE [38], AGAST [39] and ORB [40]. In certain applications, these detectors are used to describe the contents of an image before performing object detection by matching features to an existing database. In other cases, the features are tracked in order to perform 3D reconstruction [5, 42–45], which in turn allows prioritised object detection using the derived depth map. The real-time performance and accuracy of all these detectors varies significantly, thus highly complex algorithms have been ruled out, whereas the performance of the remaining methodologies is evaluated in Chapter 3, using a wide range of metrics proposed by Mikolajczyk et al. [34]. This evaluation shows that there is a gap for a feature detector that is computationally efficient, while producing high keypoint density. Most contemporary approaches can only produce sparse keypoints in real-time, which lowers the probability of successful object detection.

A group of 3D reconstruction algorithms has also been reviewed, while mostly focussing on Simultaneous Localisation and Mapping (SLAM) techniques [4, 5, 44, 46–53]. Despite the recent radical improvements in this area, such algorithms are still not suitable for dense 3D reconstruction in automotive environments due to their high dependance on the aforementioned feature detectors. In scenarios where keypoint density is low, such as motorways, the depth map density of SLAM approaches is also lower. Finally, the computational complexity of probabilistic localisation makes

SLAM hard to implement on embedded hardware [4, 116].

The literature for detection and classification of targets has also been examined, since it is an integral part of modern automotive safety applications such as pedestrian protection systems (PPS) [55], traffic sign recognition [56] and collision avoidance [57]. The majority of these approaches focus on single object detection (e.g. traffic signs, cars, pedestrians) although there are also some multi-class classifiers [59,60]. Reviewing all of these methodologies would have been a major complex task since each object on its own has been the subject of extensive research over the past decades. As a result, the focus was shifted to just the pedestrian class, which is one of the most studied and challenging subjects to date. In particular, the state-of-the-art in real-time pedestrian detection was described, while trying to identify limitations in the optimisation process. The most significant gap in this area is related to prioritised image indexing for reducing the number of unsuccessful iterations. It is proposed that the performance of object detection approaches [61] is enhanced by data fusion of 3D reconstruction algorithms and visual saliency algorithms [10, 62–67].

As a conclusion, this chapter has highlighted a set of significant technological gaps that form the basis of each of the following chapters in this thesis. Chapter 3 evaluates the aforementioned feature-detection methodologies and attempts to identify which feature detectors are more suitable for depth-estimation and object detection. The technological gap is addressed by proposing a new type of feature detector that fulfils the criteria of high-keypoint density and real-time performance. In Chapter 4, these features are tracked using an enhanced variant of the Lucas and Kanade tracking approach [97]. This new implementation compensates for the extra computational cost of image stabilisation by tightly integrating it into the tracking algorithm. Finally, in Chapter 5, the literature study has contributed to creating a novel visual-saliency detector that addresses the current limitations by deriving saliency maps at the same resolution as the input image.

Chapter 3

Real-time feature detection in automotive environments

3.1 Overview

The aim of this chapter is to identify the most suitable types of feature detectors for performing dense 3D reconstruction in automotive environments using a monocular camera. Recovering depth information requires the detection and tracking of keypoints across multiple frames. In this context, the real-time requirements of automotive applications are considered in order to propose a computationally-efficient methodology that offers high feature density and tracking accuracy. In section 3.2, a novel feature detector is proposed that produces dense keypoints based on gradient features. Section 3.2.3 compares this new approach to the state-of-the-art using a broad range of evaluation criteria including keypoint density, repeatability and tracking accuracy.

3.2 Dense Gradient-based Features (DeGraF)

A novel feature detector is presented, suitable for dense tracking of features in real-time. This detector, denoted as DeGraF (Dense Gradient Features), is based on the calculation of gradients using intensity-weighted centroids [140].

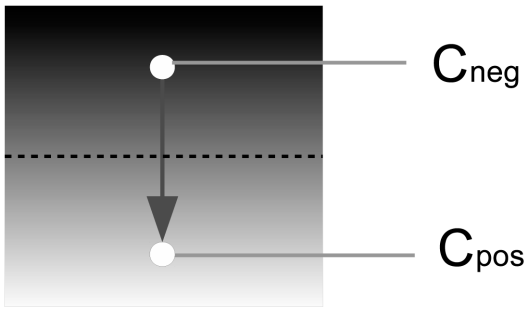


Figure 3.1: The positive centroid C_{pos} is the intensity-weighted centroid of the original image region, whereas the negative centroid C_{neg} is the intensity-weighted centroid of the inverted image region. Both centroids are calculated before choosing the strongest (i.e. the one with the highest weight). The weakest centroid is likely to sensitive to noise, thus it is replaced by a new point which is anti-symmetric to the strongest centroid about the centre of the image region. The vector connecting these two centroids is the gradient of that region. This new way of deriving gradient vectors significantly minimises the effect of image noise.

3.2.1 Gradients from centroids (GraCe)

A method for calculating image gradients using intensity-weighted centroids is proposed as an alternative to the Sobel operator used in other algorithms [61, 136]. In this context, any image area A with dimensions $w \times h$ and central point $C(x_c, y_c)$, has two symmetrical centroids C_{pos} and C_{neg} that define a gradient vector (see *Figure 3.1*). C_{pos} or positive centroid is defined as the weighted average of pixel values as given by *Equation 3.1*.

$$C_{pos}(x_{pos}, y_{pos}) = C_{pos}\left(\frac{\sum_{i=0}^{h-1}\sum_{j=0}^{w-1} iA_{ij}}{S_{pos}}, \frac{\sum_{i=0}^{h-1}\sum_{j=0}^{w-1} jA_{ij}}{S_{pos}}\right) \tag{3.1}$$

where $S_{pos} = \sum_{i=0}^{h-1}\sum_{j=0}^{w-1} A_{ij}$.

The negative centroid is defined as the weighted average of the inverted pixel values as given by *Equation 3.2*.

$$C_{neg}(x_{neg}, y_{neg}) = C_{neg} \left(\frac{\sum_{i=0}^{h-1} \sum_{j=0}^{w-1} i(1 + m - A_{ij})}{S_{neg}}, \frac{\sum_{i=0}^{h-1} \sum_{j=0}^{w-1} j(1 + m - A_{ij})}{S_{neg}} \right) \quad (3.2)$$

where m is the local maximum pixel value of area A and $S_{neg} = \sum_{i=0}^{h-1} \sum_{j=0}^{w-1} (1 + m - A_{ij})$. It should be noted that all pixel values are expressed as floating point numbers greater than 1.0. For example, a greyscale or RGB image with pixel value range 0 - 255 is converted to have a range 1.0 - 256.0. This conversion preserves colourspace linearity while eliminating the unwanted effect of division by zero.

Once C_{pos} and C_{neg} have been derived then the gradient can be expressed as a vector $\overrightarrow{C_{neg}C_{pos}}$. However, such a gradient would be sensitive to noise, since the signal-to-noise ratio (SNR) depends on the value of S_{pos} and S_{neg}. For example, calculating the gradient of a dark noisy image area will lead to a lower S_{pos} value and a higher S_{neg} value, thus C_{neg} will have a better SNR whereas C_{pos} will be unstable. This issue can be addressed by choosing the centroid with the higher SNR (i.e. the centroid with the larger S value) as the positive centroid and then expressing the negative centroid as the symmetric point about the centre $C(x_c, y_c)$ of area A (*see Equation 3.3*). This method dramatically increases the accuracy of the calculated angle and magnitude of each gradient vector in noisy environments. With the centroids having sub-pixel accuracy, stable and dense feature points can be detected using gradients.

$$C_{neg}(x_{neg}, y_{neg}) = (2x_c - x_{pos}, 2y_c - y_{pos}) \quad (3.3)$$

Once the gradient vector has been derived its magnitude r and angle ϕ can be calculated using the following standard equations :

$$r = \sqrt{dx^2 + dy^2} \quad (3.4)$$

$$\phi = atan2(dy, dx) \quad (3.5)$$

where $dx = x_{pos} - x_{neg}$, $dy = y_{pos} - y_{neg}$ and $atan2$ is the quadrant-aware version of $arctan$.

The pseudocode of the gradient calculation is the following:

```
read image with dimensions image_width, image_height;
generate grid with dimensions grid_width, grid_height, cell_width, cell_height;
for each grid cell
    set max_value to the local maximum pixel value;
    set s_pos to the sum of the pixel values;
    set s_neg to the sum of the inverted pixel values, where inverted_pixel_value =
        1 + max_value - pixel_value;
    if s_pos > s_neg
        set c_pos.x to the average of x values weighted by their corresponding
            pixel values;
        set c_pos.y to the average of y values weighted by their corresponding
            pixel values;
    else
        set c_pos.x to the average of x values weighted by their corresponding
            inverted pixel values;
        set c_pos.y to the average of y values weighted by their corresponding
            inverted pixel values;
    endif
    c_neg.x = 2*cell_centre.x - c_pos.x;
    c_neg.y = 2*cell_centre.y - c_pos.y;
    dx = c_pos.x - c_neg.x;
    dy = c_pos.y - c_neg.y;
    gradient_magnitude = sqrt(dx*dx + dy*dy);
    gradient_angle = atan2(dy, dx);
end loop
```

As *Figure 3.2* illustrates, the centroid gradient approach has several advantages over Sobel, Gabor and Lagrange [9] approaches:

- It calculates the gradient vector magnitude and direction for any symmetric or asymmetric image area of any size.
- It offers sub-pixel accuracy, which is essential when processing low-resolution images. In this context, a gradient centroid may exist in the space between four neighbouring pixels.
- It is significantly more resistant to noise.
- It is computationally more efficient than Gabor and Lagrange and only slightly more complex than Sobel, when comparing the number and complexity of the

mathematical operations that are used for each calculation.

3.2.2 From gradients to features

This section describes the process of extracting dense gradient-based features (DeGraF) from a gradient map generated using the GraCe algorithm. The steps are the following:

Difference of Gaussians (DoG) (Optional)

The difference of Gaussians (DoG) image is derived using a novel algorithm called the *Inverted Gaussian Di-pyramid* (IGD). A di-pyramid is a shape comprising two pyramids symmetrically placed base-to-base, whereas an inverted di-pyramid comprises two pyramids symmetrically placed peak-to-peak (see *Figure 3.3*). In image processing terms, the input image is used to perform bottom-up construction of a Gaussian pyramid. Subsequently, the peak of this pyramid is used to perform top-down construction of a second Gaussian pyramid (inverted pyramid). This way the two Gaussian pyramids form an inverted di-pyramid. Calculating the absolute difference between the two Gaussian pyramid bases gives a DoG image with the same resolution as the input image but invariant to illumination. In more detail, the DoG image is derived as follows:

- a Gaussian pyramid U is constructed with n levels
  - the n^{th} level U_n is then used to build an inverted Gaussian pyramid D with base D_0.
  - the DoG image I is given by the following equation:

$$I = |U_0 - D_0| \tag{3.6}$$

Figure 3.4 illustrates an example of a DoG image derived using a 5-level pyramid with a 5×5 Gaussian filter. Without the pyramidal approach Gaussian smoothing would have to be performed on the input image using a 81×81 Gaussian kernel. More generically the equivalent Gaussian

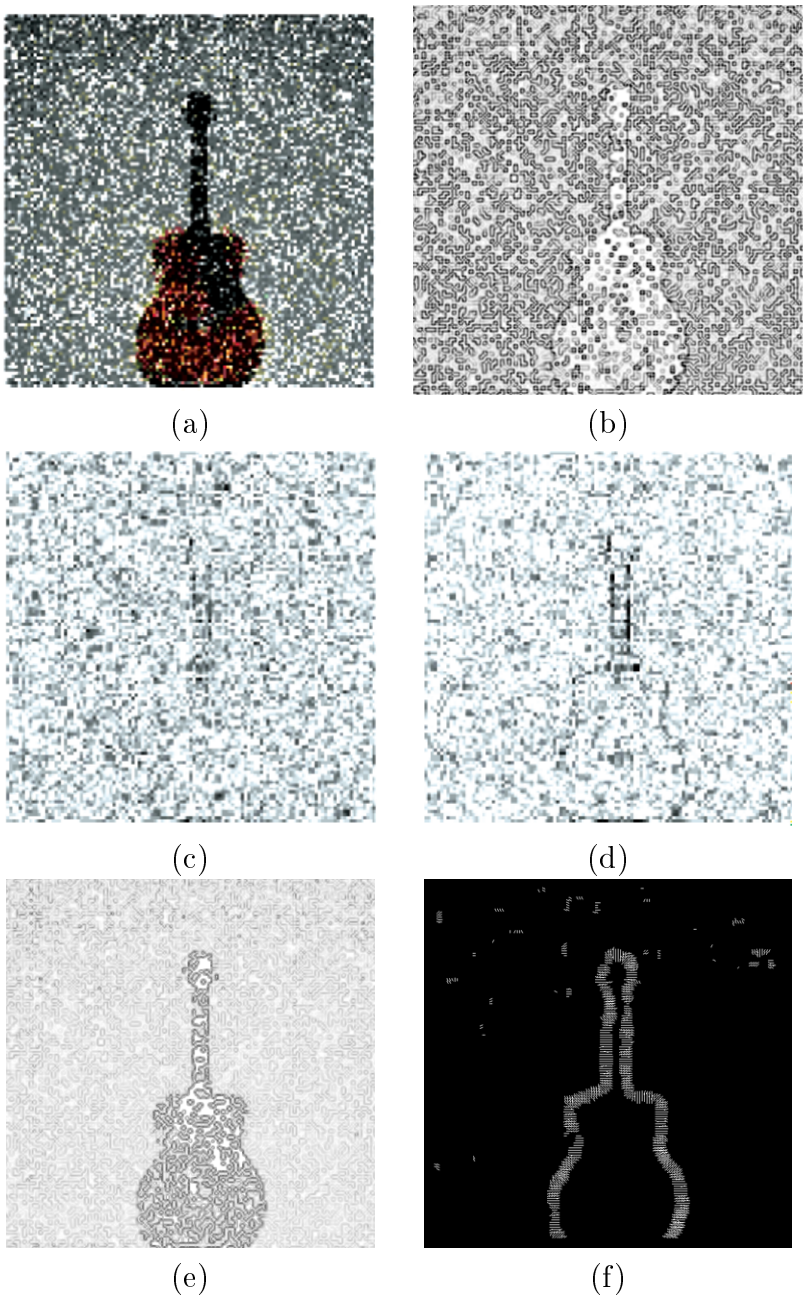


Figure 3.2: Comparison of gradient algorithms: a) dataset image of a guitar, which has been used for evaluation by Ahmad et al. [9], b) Gradient map using the Sobel 3×3 filter, c) gradient map using the Gabor mask [9], d) gradient map using the Lagrange mask [9], e) gradient map using GraCe, f) a subset of the GraCe gradient vectors with the largest magnitude.

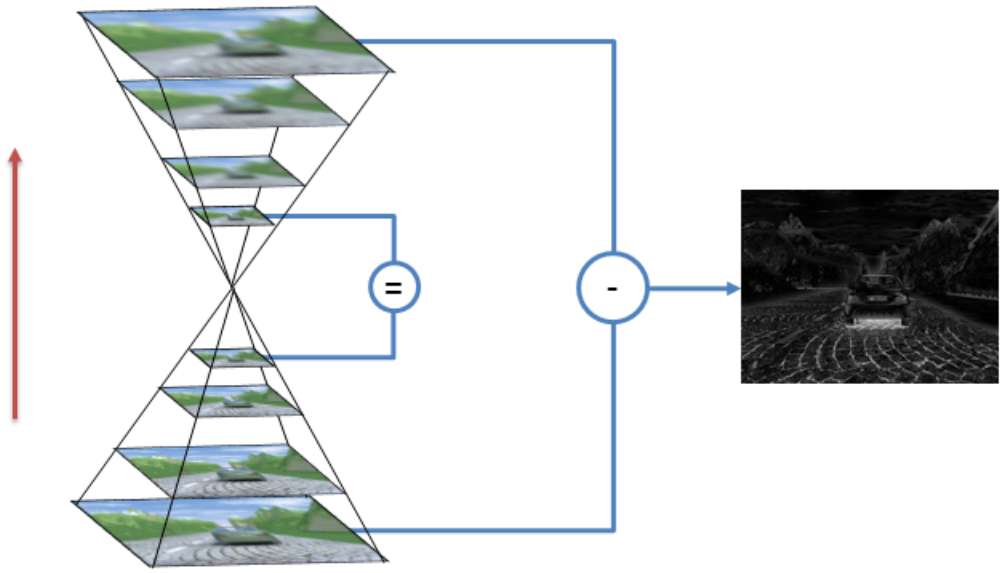


Figure 3.3: Difference of Gaussians calculation using a di-pyramid. The down-sampled image at the top of the first pyramid is used to reconstruct the inverted pyramid. The base images of the two pyramids generate a DoG image when subtracted.

kernel at full resolution has dimensions $((2^{n-1}k_x)+1) \times ((2^{n-1}k_y)+1)$, where n is the number of pyramid levels, which have been derived using a $k_x \times k_y$ Gaussian kernel (usually 5×5). As demonstrated by the SIFT approach [1], the DoG image improves the performance of feature detection. However, the IGD approach produces DoG images far more efficiently when compared to the standard DoG calculation method (using convolution). This is because the IGD algorithm uses a pyramid of lower resolution images as opposed to applying the Gaussian kernel directly on the full-sized image . For example, the difference of Gaussians by convolution would involve the following operations:

* Perform convolution of the input image I_1 with a $k \times k$ Gaussian filter. The mathematical operations involve k^2 multiplications and k^2 additions per pixel (total $2k^2$ operations per pixel). The output image is denoted as I_2.
* Calculate the difference of Gaussians $I_1 - I_2$ (1 operation per pixel).
* As a result, the traditional convolution approach requires $(2k^2+1)$ operations per pixel for all image pixels. For an image with dimen-

sions $w \times h$ the total number of required operations is $(2k^2 + 1)wh$ operations per pixel.

If the difference of Gaussians is calculated using the IGD approach, the required operations would be:

* Build a 3-level Gaussian di-pyramid with a $\frac{k}{2} \times \frac{k}{2}$ Gaussian filter. Note that the Gaussian filter size is smaller than that used by the traditional approach. The reason is that the filter is used across two pyramid levels, thus the combined effect is the same as using a $k \times k$ filter on the full resolution image.
* The first level of the di-pyramid is the input image I_1.
* The second level is derived by resolution reduction of I_1 using a $\frac{k}{2} \times \frac{k}{2}$ Gaussian filter, in order to produce image I_2 with size equal to a quarter of the original image size. The mathematical operations involved are $0.25k^2$ multiplications and $0.25k^2$ additions per pixel (total $0.5k^2$ operations per pixel) but only for a quarter of the total pixels.
* The third level is derived by resolution reduction of I_2 using a $\frac{k}{2} \times \frac{k}{2}$ Gaussian filter, in order to produce image I_3 with size equal to a quarter of the original image size. The mathematical operations involved are $0.25k^2$ multiplications and $0.25k^2$ additions per pixel (total $0.5k^2$ operations per pixel) but only for a quarter of the total pixels of I_2.
* The last two steps are repeated in the inverse order to reconstruct the remaining part of the di-pyramid, where I_4 has the same same as I_2 and I_5 has the same size as I_1.
* The total number of operations for building the di-pyramid is $2\left(\frac{0.5k^2}{4} + \frac{0.5k^2}{16}\right) = 0.3125k^2$ operations per pixel.
* Calculate the difference of Gaussians $I_1 - I_5$ (1 operation per pixel).
* The total number of operations is $(0.3125k^2 + 1)wh$

If for example $k = 10$, $w = 100$ and $h = 100$ the convolution-based

Figure 3.4: Difference of Gaussians using a 5-level pyramid with a 5×5 Gaussian filter.

approach would require 2,010,000 operations, whereas the IGD approach would require only 322,500 operations. As a result, the IGD approach is more than 6 times faster.

This approach differs from SIFT [1] since the difference of Gaussians is not calculated on subsequent pyramid levels of a single pyramid. Instead we calculate the difference of Gaussians between the base a Gaussian pyramid and the base of an inverted Gaussian pyramid.

1. **Gradient matrix calculation**

   A grid G is overlaid over the input image. A gradient vector is calculated for each grid cell C. Let's define the image, grid and cell dimensions as $w \times h$, $w_G \times h_G$ and $w_C \times h_C$ respectively. Cells may overlap by δ_x pixels horizontally and δ_y pixels vertically. Subsequently, the dimensions of the gradient matrix G are given by the following equation:

   $$w_G = \left\lfloor \frac{w - \delta_x}{w_C - \delta_x} \right\rfloor \quad h_G = \left\lfloor \frac{h - \delta_x}{h_C - \delta_y} \right\rfloor \tag{3.7}$$

   where $0 \leq \delta_x \leq w_c - 1$ and $0 \leq \delta_y \leq h_c - 1$. For example, a 640×480 image can be divided into 2×2 cells with no overlap, which generates a 320×240 gradient matrix. However, using 4×4 cells with 2-pixel overlap would give a gradient matrix of similar size (i.e. 319×239) but with more accurate gradients that are less sensitive to noise. This choice of parameters depends

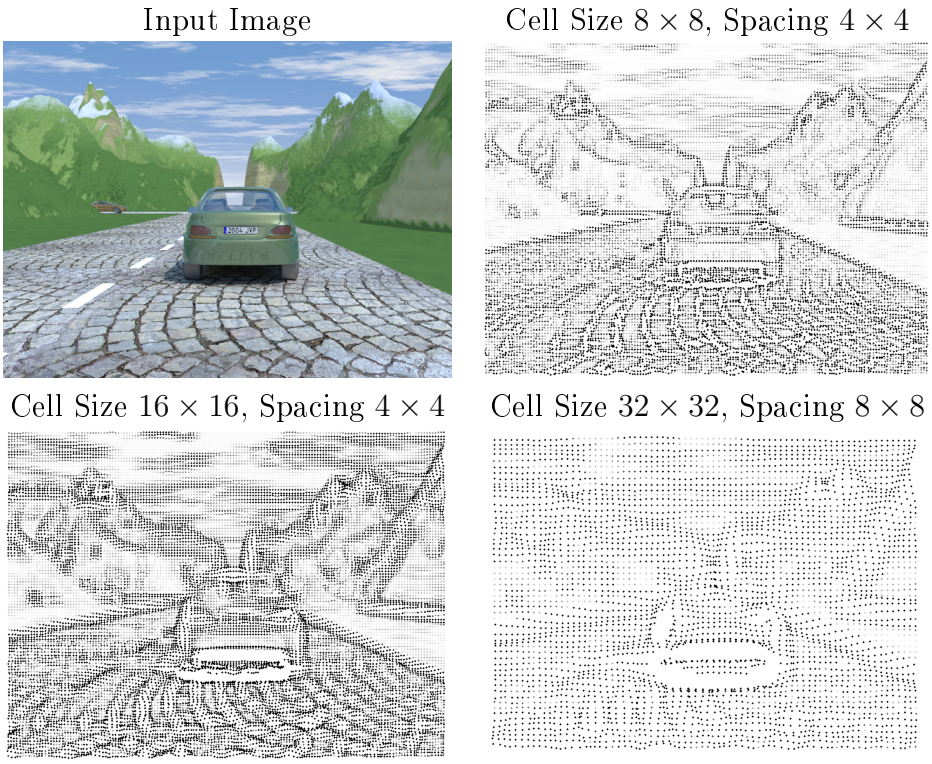


Figure 3.5: Positive centroids generated for different cell size and spacing. The grey shade of the centroid indicates the magnitude of the associated gradient. Stronger gradients have darker shades.

on the trade-off between real-time performance, required density of features and noise resistance. The quality of features also depends on the signal-to-noise ratio of the imaging sensor. Noisy low-cost cameras can still be used to extract stable DeGraF features by increasing the cell size. It should be noted that grid cells may have any size or shape. For example circular cells can be used for deriving rotation invariant features. The size of each cell may also be adjusted dynamically in order to cope with local noise patterns. *Figures 3.5* and *3.6* illustrate some examples where the gradient vectors have been derived with different cell size and overlap. Three graphical representations (vector and centroid point) are used to highlight the distribution of gradients.

2. **Feature extraction**

   In the context of the DeGraF approach, feature extraction is the process of selecting those gradient vectors that are suitable for tracking or matching.

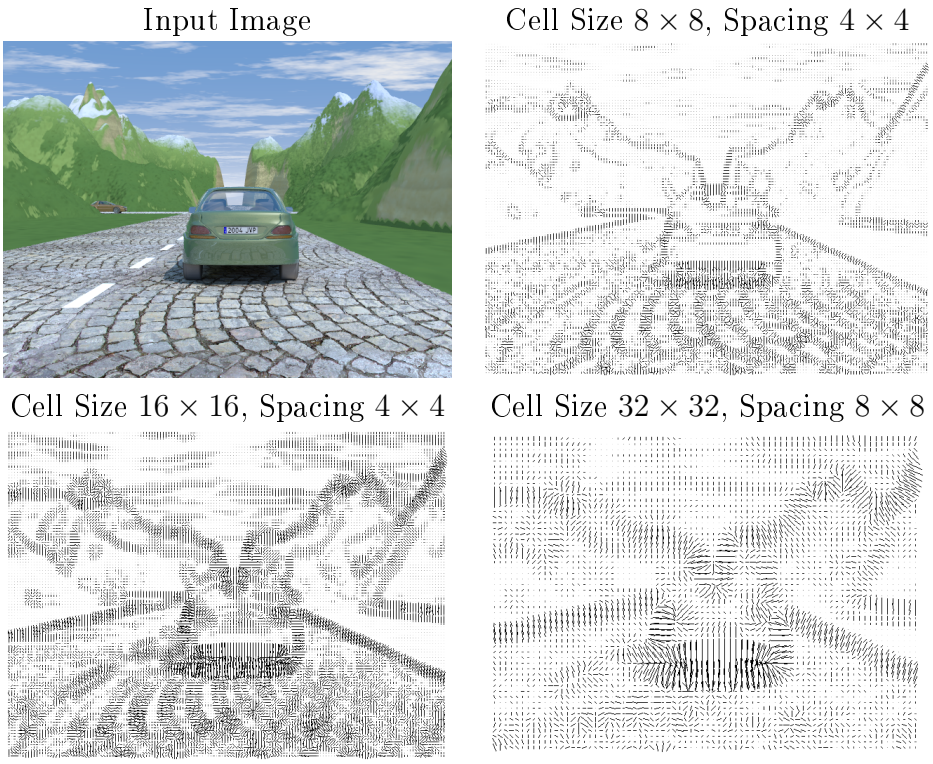


Figure 3.6: Gradient vectors generated for different cell dimensions and spacings.

Feature tracking refers to identifying a feature in subsequent frames, whereas feature matching refers to identifying a feature in two separate frames captured from different viewing angles. Depending on the application, local features can be used for the purpose of tracking keypoints between successive frames, or global features can be detected for the purpose of matching them between two frames where the position or viewing angle of the camera are different.

(a) **Method A: Global feature extraction using α-gradients**

In this method, we use an approach similar to CenSurE [38] and SIFT [1], where local minima and extrema are identified. However, in this case we use the gradient matrix derived by the GraCe approach to detect good gradient features. A good gradient feature is defined as one with the stronger or weaker magnitude amongst all the neighbouring gradient vectors in the gradient matrix. In other words, features are either local gradient peaks or gradient troughs. Such gradient-based features are called an *α-gradients*. The process begins by scanning all the neighbour-

ing gradient vectors for each cell in the gradient matrix. If the stronger or weaker gradient is at the edge of the neighbourhood then it is considered an unstable gradient and is ignored, whereas if it is positioned right at the centre of the neighbourhood then this is an α-*gradient.* An offset can also be used to define the distance from the centre of the neighbourhood that an α-*gradient* is allowed to exist. The size of the neighbourhood is also configurable. Larger neighbourhoods are less likely to have the stronger gradient at the centre, thus the features become sparser. Although this is not desired for most automotive applications, it could have interesting applications in other computer-vision problems, such as detecting the affine transformation of a surface or object in augmented reality applications. In more detail the pseudocode for extracting good gradient features is the following:

```
1  minimum_magnitude = 0;
2  maximum_magnitude = 1000;
3  for y = offset; y < gradient_matrix.width - offset; y++
4      for x = offset; x < gradient_matrix.width - offset; x++
5          // Identify the position of the stronger and weaker gradient
               vector
6          for i = y - offset; i <= y + offset; i++
7              for j = x - offset; j <= x + offset; j++
8                  if gradient(i,j).magnitude > maximum_magnitude
9                      maximum_magnitude = gradient(i,j).magnitude;
10                     feature_point_max.x = gradient(i,j).
                           positive_centroid.x;
11                     feature_point_max.y = gradient(i,j).
                           positive_centroid.y;
12                 endif
13                 if gradient(i,j).magnitude < minimum_magnitude
14                     minimum_magnitude = gradient(i,j).magnitude;
15                     feature_point_min.x = gradient(i,j).
                           positive_centroid.x;
16                     feature_point_min.y = gradient(i,j).
                           positive_centroid.y;
17                 endif
18             end loop
19         end loop
20
21         // Check if the stronger or weaker gradient is positioned at the
               centre (alpha gradient)
22         if(feature_point_max.x == x && feature_point_max.y == y)
```

```
23              return feature_point;
24          else if(feature_point_min.x == x && feature_point_min.y == y)
25              return feature_point;
26          endif
27      end loop
28  end loop
```

(b) **Method B: Local feature extraction using β-gradients**

As an alternative to extracting unique keypoints, the use of every gradient vector as a feature is proposed. Since the stronger gradient vector is no longer detected let's call these β-gradients. The theory is simple and is based on the fact that each gradient is a vector that describes an image area. The positive and negative centroids have been formed using the underlying pixels. As a result, if these pixels change value, the gradient will also change. Since 3D reconstruction is based on tracking points between frames, each positive centroid can be defined as a local keypoint. The distance that each centroid travels between two frames could also be indicative of the underlying change that happened in the scene. Still there may be gradients that describe areas that are too dark, noisy or textureless in which case tracking is impossible. Such gradients can be filtered out by considering some gradient quality metrics such as:

i. Gradient magnitude: gradient vectors with no magnitude are normally describing areas where all the pixels have the same value.

ii. Centroid ratio: Centroid ratio R refers to the signal-to-noise ratio for each of the positive and negative centroids. One of the two centroids is likely to have higher SNR value than the other. This difference is measured using the following equation:

$$R = min\left(\frac{S_{pos}}{S_{neg}}, \frac{S_{neg}}{S_{pos}}\right) \tag{3.8}$$

where S_{pos} and S_{neg} are derived from equations 3.1 and 3.2.

3.2.3 DeGraF characteristics and analysis of α and β types

The DeGraF approach derives stable keypoints from a gradient matrix using intensity-weighted centroids. This algorithm is primarily designed for deriving noise resistant and high density features. The GraCe approach introduces a novel way of handling noise by separately estimating the signal-to-noise ratio for each gradient. In this context, a gradient is defined as the combination of a positive and a negative centroid. Since the weakest of the two centroids is more vulnerable to noise, only the dominant centroid is considered. The weakest centroid is then redefined as the anti-symmetric positive centroid (see *Figure 3.1*). This novel approach of calculating gradients is the key contributing factor to the performance of DeGraF features. In addition extracting DeGraF features from a DoG image makes the detected features become illumination invariant, whereas using different pyramid levels offers higher density and scale-invariance. Most importantly, features can be detected in real-time since the computational complexity is very low. The parallel nature of the algorithm also allows it to be implemented efficiently on multi-processor systems, GPUs or FPGAs, thus making it suitable for automotive applications.Evaluation of feature detectors

The DeGraF feature detector is compared to state-of-the-art methodologies using a wide range of quality metrics that are applicable to automotive vision applications. Firstly, the algorithms evaluated in this section, have been selected based on literature study on real-time dense 3D reconstruction. The evaluated approaches are listed below together with a brief description, the source code used for evaluation and the parameters for running each algorithm in order to achieve maximum feature density.

- Adaptive and Generic Corner Detection Based on the Accelerated Segment Test (**AGAST**) [39]
  - Brief Description: AGAST is based on FAST [32], but uses decision trees to detect corners more reliably and efficiently.
  - Source code: http://www6.in.tum.de/Main/ResearchAgast
  - Evaluation Parameters: b = 1 with non-maximum suppression enabled.

- Centre Surround Extremas for Realtime Feature Detection and Matching (**CenSurE**) [38]
  - Brief Description: CenSurE detects local extrema as features that are scale and rotation invariant. The approach has several similarities to DeGraF, but is is point-based instead of gradient-based.
  - Source code: OpenCV 2.4.2 [11]
  - Evaluation Parameters: maximum size = 5, response threshold = 0, line threshold projected = 10, line threshold binarised = 5, suppress non-max size = 2.
- Dense Gradient Features Alpha (**DeGraF-α**)
  - Brief Description: Local extrema and minima are extracted from a gradient matrix.
  - Source code: Implemented using OpenCV 2.4.2 [11]
  - Evaluation Parameters: gradient window width = 2, gradient window height = 2, gradient window overlap = 1.
- Dense Gradient Features Beta (**DeGraF-β**)
  - Brief Description: The positive centroids in gradient matrix are used as features.
  - Source code: Implemented using OpenCV 2.4.2 [11]
  - Evaluation Parameters: gradient window width = 3, gradient window height = 3, gradient window overlap = 3, minimum magnitude 0.015.
- Features from Accelerated Segment Test (**FAST**) [32]
  - Brief Description: FAST detects corners using an accelerated segment test, which depends on several user defined thresholds. This means that FAST needs to be adapted for each scene.
  - Source code: OpenCV 2.4.2 [11]

  - Evaluation Parameters: threshold = 0 with non-maximum suppression enabled.

- Shi & Tomasi Good features to track (**GFTT**) [28]

  - Brief Description: The GFTT approach computes the local minimum Eigen values as the most suitable features to track.
  - Source code: OpenCV 2.4.2 [11]
  - Evaluation Parameters: maximum corners = 0 (no limit), quality level = 0.001, minimum distance = 1.0, block size = 3, k = 0.04.

- Scale-invariant feature transform (**SIFT**) [1]

  - Brief Description: SIFT detects local scale-space extrema that are scale invariant features. Since each feature is oriented according to the strongest gradient magnitude SIFT features are also rotation-invariant.
  - Source code: OpenCV 2.4.2 [11]
  - Evaluation Parameters: number of features = 0 (no limit), number of octave layers = 3, contrast threshold = 0.015, edge threshold = 10, sigma = 0.7.

- Speeded Up Robust Features (**SURF**) [2]

  - Brief Description: SURF is a faster alternative to SIFT, that uses integral images instead of Gaussian pyramids, whereas features are extracted from the Hessian matrix.
  - Source code: OpenCV 2.4.2 [11]
  - Evaluation Parameters: hessian threshold = 0, number of octaves = 4, number of octave layers = 2.

- Oriented Fast and Rotated BRIEF (**ORB**) [40]

  - Brief Description: The algorithm uses FAST in pyramids to detect stable keypoints, then selects the strongest features using FAST or Harris response. Each keypoint is assigned an orientation using first-order moments.
  - Source code: OpenCV 2.4.2 [11]
  - Evaluation Parameters: number of features = 0 (no limit), scale factor = 1.2, number of levels = 8, edge threshold = 0, first level = 0, number of output points = 2, patch size = 1, score type = 0 (Harris).

- Maximally stable extremal regions (**MSER**) [35]

  - Brief Description: MSER uses blob detection to detect features that are invariant to affine transformation and scale.
  - Source code: OpenCV 2.4.2 [11]
  - Evaluation Parameters: delta = 5, minimum area = 60, maximum area = 1000, maximum variation = 0.25, minimum diversity = 0.2, maximum evolution = 200, area threshold = 1.01, minimum margin = 0.003, edge blur size = 5.

The evaluation is based on the following six criteria:

1. Keypoint Density: Average number of detected keypoints per image frame.
2. Tracking accuracy: Average number of detected keypoints that were tracked successfully between two frames.
3. Repeatability with variable illumination: The error introduced by varying image brightness.
4. Repeatability with variable rotation: The error introduced by varying image rotation.
5. Repeatability with noise: The error introduced by adding noise to the input image.

6. Detection time: Execution time per frame in milliseconds.

The criteria above refer to a slightly different approach than the one proposed by Mikolajczyk et al. [37], since it has been adapted to the requirements of automotive applications. For example, an automotive camera is expected to vibrate or rotate slightly but this will not result in significant affine transformations as is the case with Mikolajczyk's dataset. In the context of this research, an ideal detector would have high density, tracking accuracy and repeatability, while being invariant to illumination changes, resistant to noise and rotation with very low execution time. Although extensive tests have been performed on real-life video sequences, most of the experiments in this section are solely based on 3D modelled scenes from the MiTECH dataset [141, 142] (provided by University of Auckland, DAIMLER AG). The reason for using this dataset is that it allows the algorithms to be evaluated against accurate ground truth data in the absence of noise.

The MiTECH dataset has been used for the experiments in this chapter, which includes more than 3000 image frames that illustrate automotive scenes at variable resolution and with fully-labelled ground truth. All of the images are used in every experiment. The results correspond to the average performance across all frame sequences. The optimised parameters for each algorithm where initially set using each author's recommendations and verified using the OpenCV documentation [11]. Manual parameter adjustments were also performed, followed by visual inspection of the output, in order to ensure that the maximum keypoint density has been achieved using the recommended parameters.

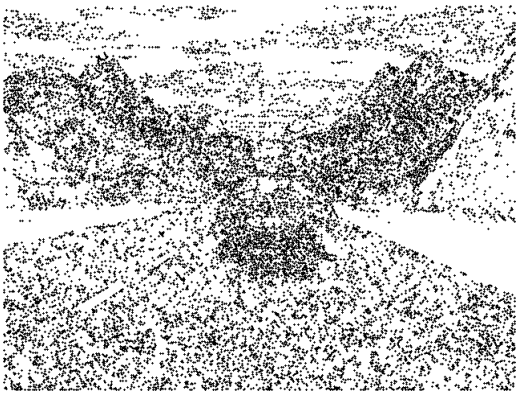
3.2.4 Keypoint Density

One of the key requirements for accurate 3D reconstruction is the density of keypoints. This is tested by analysing a set of automotive scenes, extracting keypoints and calculating the average number of keypoints per frame normalised by the image size. The configuration parameters of each feature detector are adjusted in order to achieve the highest possible feature density. Table 3.1 illustrates some characteristic examples.

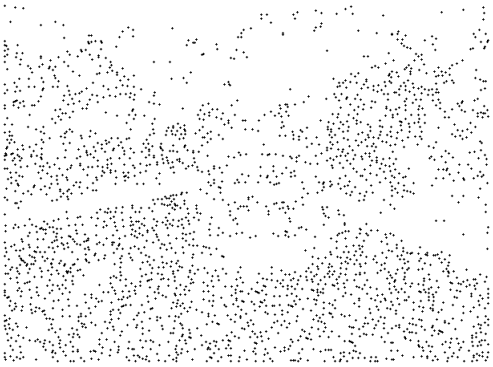
Table 3.1: Features detection using different algorithms

Input Image

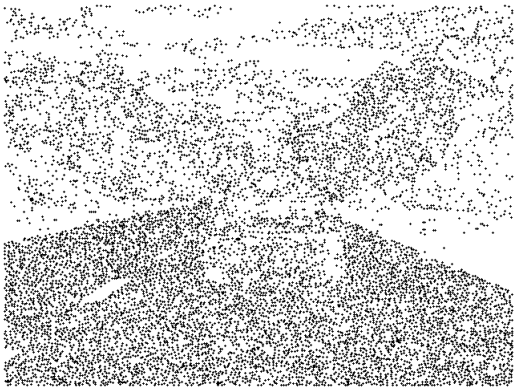
AGAST



CenSurE (STAR)

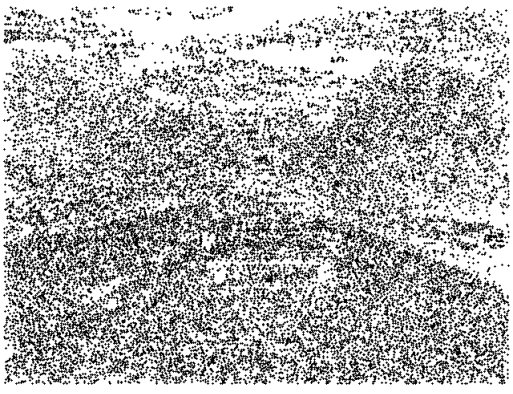


FAST



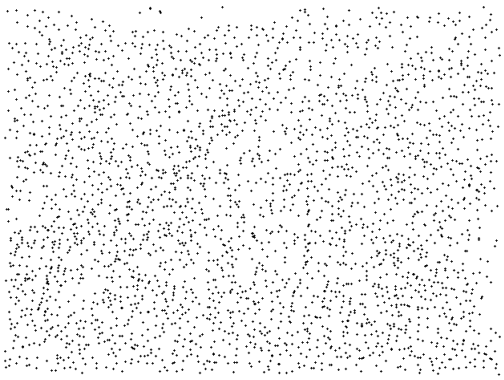
GFTT

SIFT



SURF

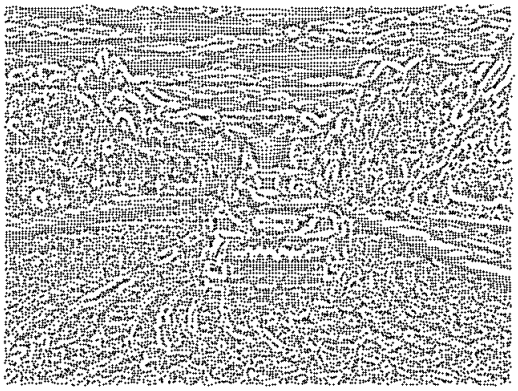
ORB

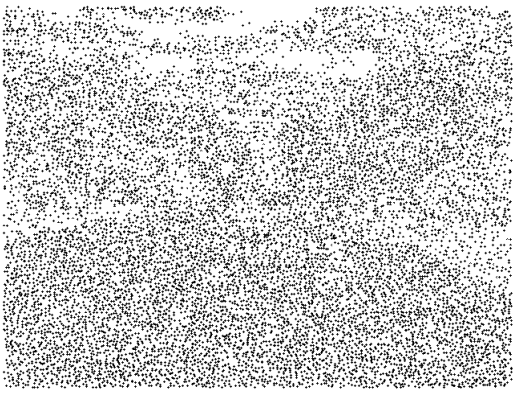


MSER

DeGraF-α

DeGraF-β





The derived keypoint density results can be viewed on *Table 3.2* and *Figure 3.7*. The values correspond to the average keypoint density across all images of the processed dataset. In this case, an error measurement is not included since the keypoint density is different for each image frame. Analysing those results shows that DeGraF-β produces the highest density, although this is to be expected since the entire gradient-matrix is used to produce one feature per positive centroid. Disregarding DeGraF-β, high keypoint density is also demonstrated by DeGraF-α, AGAST, SIFT, FAST GFTT and ORB. Interestingly, during the experimental phase both

Feature Detector	Keypoint Density (%)
AGAST	4.35
CenSurE	0.44
DeGraF-α	4.47
DeGraF-β	6.07
FAST	3.41
GFTT	3.18
SIFT	3.84
SURF	0.97
ORB	2.96
MSER	0.19

Table 3.2: Keypoint density results for different feature detectors (higher is better in the context of this thesis)

SIFT and DeGraF-α reached a density peak around 11% of the image resolution, however since the execution speed was significantly lower, these measurements have been excluded. Other feature detectors, such as CenSurE, SURF and MSER did not produce very dense keypoints. In the case of MSER, such low performance was expected since each keypoint is describing a region. As a conclusion, keypoint density by itself is not a meaningful measure without taking other parameters into consideration. There is a wide range of feature detectors that can produce high density keypoints, but these are only useful if they can be tracked accurately and within the real-time constraints.

3.2.5 Tracking Accuracy

Tracking accuracy is tested in an automotive context for each of the evaluated feature detectors. Achieving high accuracy for 3D reconstruction is essential but also challenging using dense keypoints. In this experiment, tracking accuracy is evaluated for each feature detector using image sequences with artificial vibration of amplitude equal to 1, 2, 4, 8, 16 and 32 pixels in different directions. This is achieved by displacing the image off-centre by the corresponding amount of pixels. This is representative of the vibration experienced in automotive environments from motorways to rural roads. Each frame is separated by a distance of d pixels from

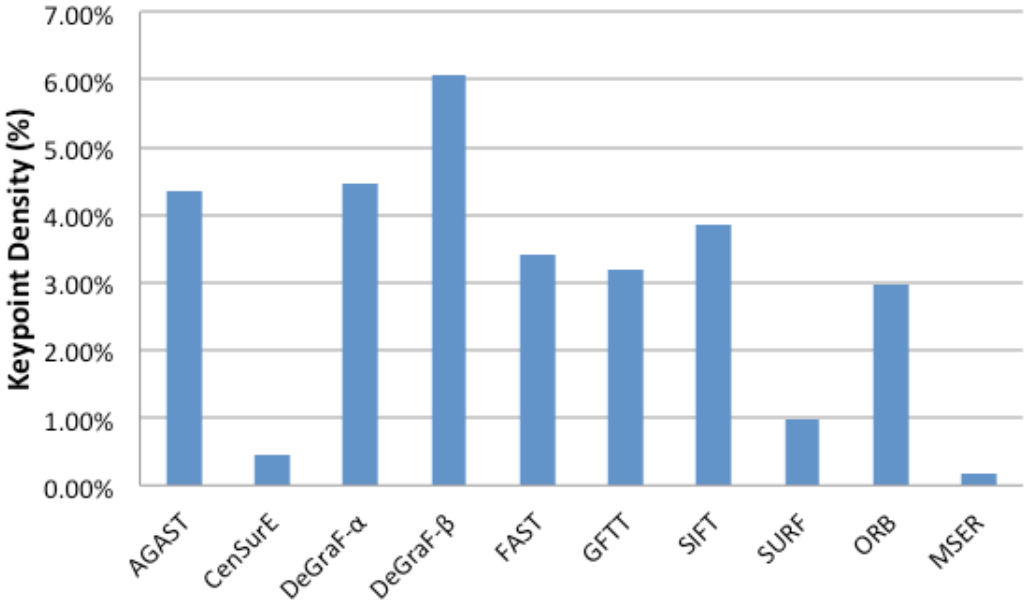
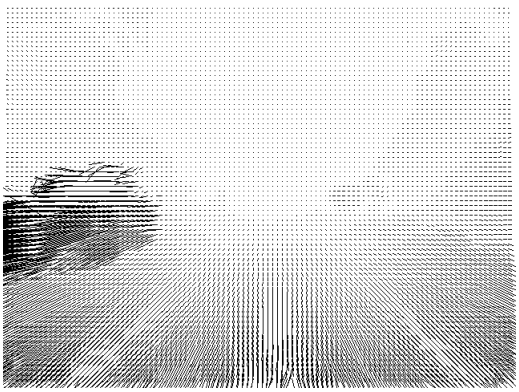
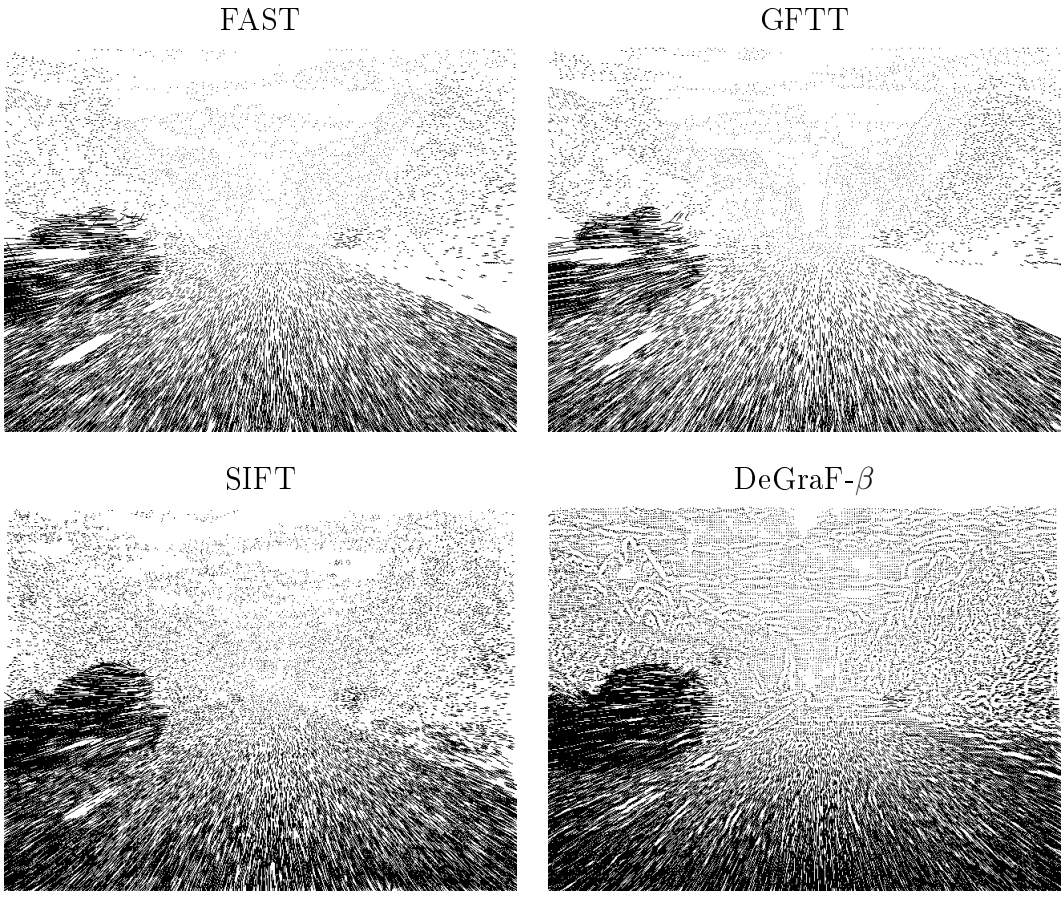


Figure 3.7: Evaluation of keypoint density for a wide range of features detectors.

the next, thus the ground truth measurement is known a priori. This choice of evaluation approach is based on the condition that the tracking performance should be measured for all detected keypoints under the same conditions. Using a standard sequence of a moving camera would mean that only part of the detected keypoints are actually moving (as illustrated in Table 3.3). In addition, establishing the ground truth in such scenarios would be a challenging task with variable accuracy.

Table 3.3: Using pyramidal KLT tracker to highlight the difference between optical flow and feature tracking of FAST, GFTT, SIFT and DegraF-β features.

Input Image	Optical Flow
	



A pyramidal implementation of the iterative Lucas-Kanade tracking algorithm [68] is used to track keypoints between frames. *Figure 3.8* shows the average tracking error for each feature detector, which is defined as:

$$error = \frac{\frac{\sum_{i=1}^{k} |d-s|}{k}}{n} \tag{3.9}$$

where d is the predefined vibration amplitude, s is the measured displacement of each keypoint, k denotes the number of detected keypoints and n denotes the total number of frames in the image sequence. The chart clearly shows that most feature detectors demonstrate similar performance under vibration with DeGraF-α and DeGraF-β being the most reliable. As expected, higher vibration leads to lower tracking performance. The majority of the evaluated feature detectors extract most

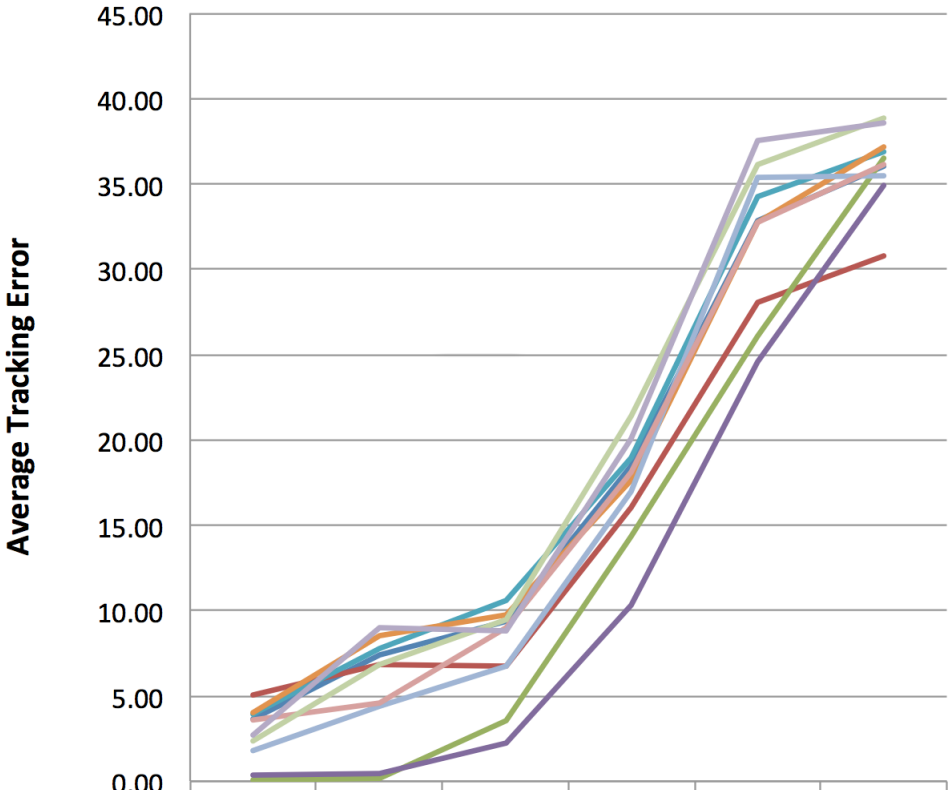
of the keypoints in high-saliency areas, leading to over-concentration of features in a small part of the image. Such keypoints are hard to track and match between different frames. DeGraF features address this issue by ensuring the detected features are evenly distributed throughout the image, which increases the probability of detecting the same feature in subsequent frames. The effect of this algorithmic design is reflected in the results, where the DeGraF features have significantly lower tracking error, when the vibration is up to eight-pixels wide. For 16 and 32-pixel vibration the tracking error is still competitive relative to the other approaches, but DeGraF is not the most accurate method. The reason is that DeGraF features describe an image area (in this case 4x4 pixels), which is significantly smaller than the vibration amplitude.

3.2.6 Repeatability with variable illumination

Automotive cameras often need to dynamically adjust to lighting changes (e.g. driving out of tunnel), which introduces a requirement for illumination-invariant algorithms. In this context, feature detectors are evaluated using image sequences with variable brightness settings. Given a set of keypoints in the input image, the detection error is derived by measuring the repeatability of features in adjusted images with 25%, 50%, 75% and 100% higher brightness level. For each of the processed images a binary image is generated where the keypoints are represented as circles with radius equal to one pixel. The conjunction of the original binary image with each of the adjusted images expresses the repeatability of each feature detector. Practically, this means that the overlap of such binary images will be pixel-perfect if identical keypoints are detected despite changes in brightness. As a result, the repeatability error can be expressed as:

$$error = \frac{\sum_{y=1}^{h}\sum_{x=1}^{w} \mathrm{A} \vee \mathrm{B} - \sum_{y=1}^{h}\sum_{x=1}^{w} \mathrm{A} \wedge \mathrm{B}}{\sum_{y=1}^{h}\sum_{x=1}^{w} \mathrm{A} \vee \mathrm{B}} \tag{3.10}$$

where binary images A, B have dimensions $w \times h$.



	1	2	4	8	16	32
AGAST	3.64	7.44	9.39	18.44	32.83	36.03
CenSurE	5.02	6.82	6.74	16.08	28.08	30.80
DeGraF-α	0.07	0.17	3.56	14.39	26.13	36.48
DeGraF-β	0.36	0.47	2.25	10.34	24.62	34.91
FAST	3.90	7.80	10.59	18.96	34.27	36.88
GFTT	4.00	8.54	9.71	17.63	32.73	37.17
SIFT	1.80	4.39	6.76	16.95	35.34	35.44
SURF	3.59	4.54	9.03	18.14	32.70	36.11
ORB	2.36	6.84	9.49	21.36	36.17	38.83
MSER	2.67	8.97	8.77	20.07	37.58	38.58

Figure 3.8: Chart of keypoint tracking accuracy for a wide range of features detectors. The horizontal axis describes the vibration amplitude in pixels. The measurement unit is pixels, denoting the the offset between the detected and the actual feature position.

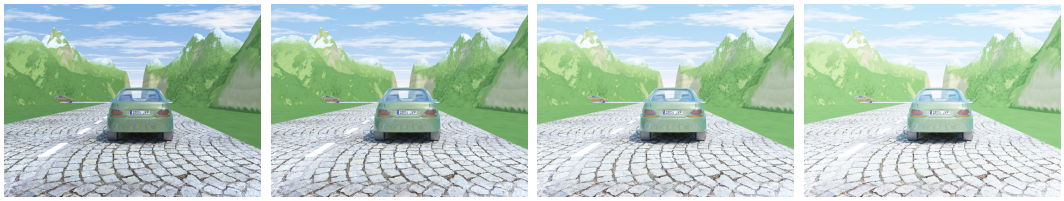
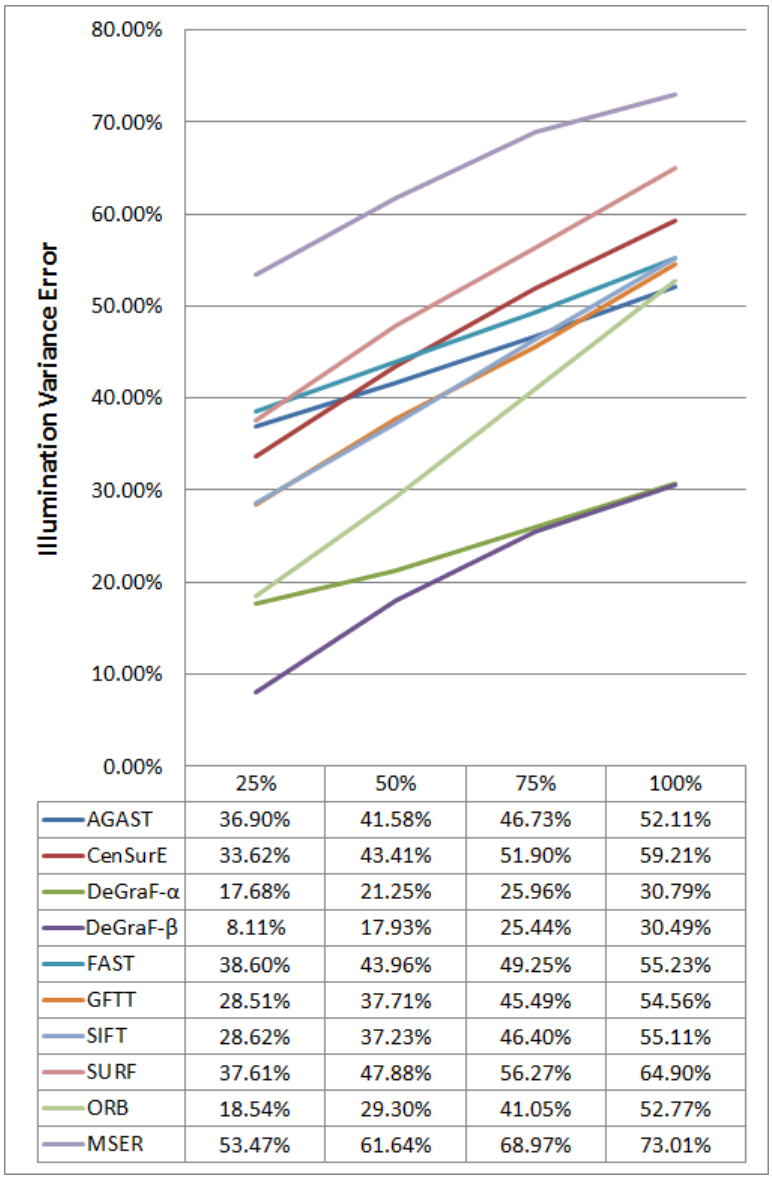


Figure 3.9: Sample images used in the illumination test. From left to right the brightness is increased by 25%, 50%, 75%, 100%.

Figure 3.10 shows the average error caused by illumination variance. DeGraF-β is outperforming all other methodologies with the lowest error, while DeGraF-α shows comparable performance at higher brightness levels. ORB has the third lowest error rate, which is important since it is the only other approach apart from DeGraF that is based on intensity centroids, albeit low density. This is a clear indication that using gradient centroids to extract features leads to illumination-invariant features. MSER demonstrates the poorest performance, which can be justified since each keypoint corresponds to a larger region. Finally, AGAST, FAST, GFTT, SIFT and SURF are demonstrating an average error rate of 37% at the lower brightness level. Such features detectors are designed to detect fewer good features rather than dense poor features. Adjusting their parameters in order to achieve high density has a counter effect on their illumination invariance.

3.2.7 Repeatability with variable rotation

In this test the error associated with variable image rotation is measured. The simulation represents the camera rolling effect between -3 and 3 degrees (see *Figure 3.11*). The rotation range corresponds to the maximum expected rolling angle of most road vehicles. Features are detected on the input image as well as on each rotated image. The rotated features are then back-projected on the original image since the initial rotation angle is known. This back-projection is achieved by aligning the feature matrix of each image to the feature matrix of the non-rotated image. For example, the feature matrix of the with 2-degree image is rotated by -2 degrees. The evaluation process is then exactly the same as in the illumination test above. Having a set of aligned binary images, allows the overlap of corresponding features to be measured and the error to be calculated as illustrated in *Figure 3.12*. This



	25%	50%	75%	100%
AGAST	36.90%	41.58%	46.73%	52.11%
CenSurE	33.62%	43.41%	51.90%	59.21%
DeGraF-α	17.68%	21.25%	25.96%	30.79%
DeGraF-β	8.11%	17.93%	25.44%	30.49%
FAST	38.60%	43.96%	49.25%	55.23%
GFTT	28.51%	37.71%	45.49%	54.56%
SIFT	28.62%	37.23%	46.40%	55.11%
SURF	37.61%	47.88%	56.27%	64.90%
ORB	18.54%	29.30%	41.05%	52.77%
MSER	53.47%	61.64%	68.97%	73.01%

Figure 3.10: Chart of error introduced by varying image brightness for a wide range of features detectors. The horizontal axis describes the increase in illumination as a percentage of the original brightness level.

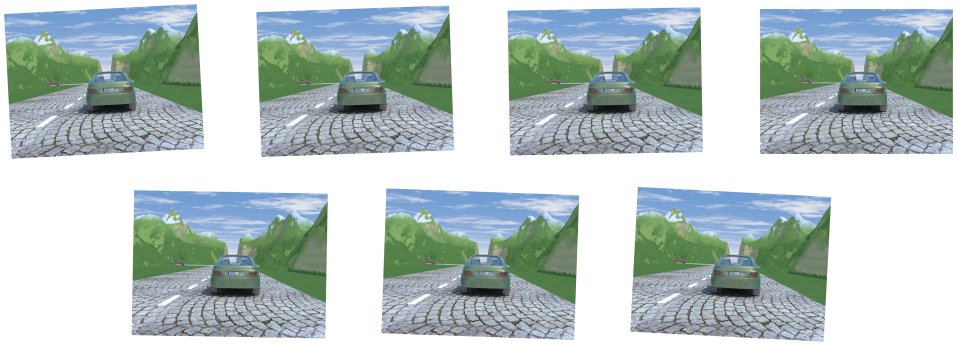
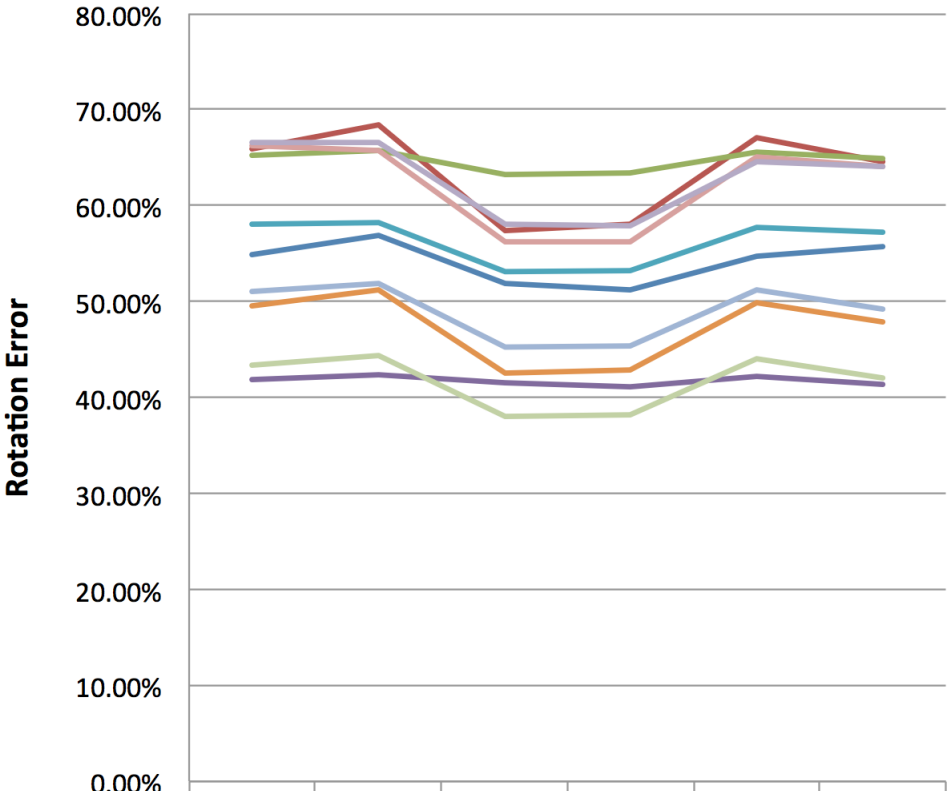


Figure 3.11: Sample images used in the rotation test. From left to right the angle of rotation is -3 ,-2, -1, 0, 1, 2, 3 degrees.

chart clearly shows that rotation-angle variance directly affects the performance of most feature detectors. In particular, ORB and DeGraF-β have the highest rotation invariance, albeit with significant error of around 41%. This means that a significant number of detected keypoints are displaced by at least 2 pixels when the image is rotated. GFTT and SIFT are followed by AGAST and FAST with medium performance. Finally, SURF, MSER, CenSurE and DeGraF-α have the highest error rate, which means that the majority of keypoints are displaced by more than 2 pixels following rotation. SURF and DeGraF-αuse rectangular image areas to derive the properties of each feature, thus their rotation invariance could be improved by using circular patches instead.

3.2.8 Repeatability in noisy images

In this test the error associated with variable image noise is measured. The simulation represents the camera noise as a Gaussian distribution with each image having 5% more added noise than the previous one (see *Figure 3.13*). Features are detected on the input image as well as on each noisy image. Subsequently, the same evaluation process is applied as in the illumination test above. The error is calculated from the aligned binary images and the results are shown in *Figure* 3.14. This chart clearly illustrates that DeGraF-β outperforms all other approaches by a significant margin. For example, comparing DeGraF-β to the second best approach (ORB) shows a performance gap between 28% and 62% when image noise increases by 5% and 20% respectively. The fact that ORB and DeGraF-α are second and third in



	-3	-2	-1	1	2	3
AGAST	54.87%	56.81%	51.87%	51.19%	54.66%	55.78%
CenSurE	65.91%	68.40%	57.35%	58.11%	67.05%	64.53%
DeGraF-α	65.24%	65.68%	63.21%	63.39%	65.60%	64.84%
DeGraF-β	41.81%	42.30%	41.49%	41.11%	42.21%	41.42%
FAST	58.13%	58.26%	53.11%	53.27%	57.80%	57.28%
GFTT	49.51%	51.17%	42.54%	42.82%	49.82%	47.92%
SIFT	50.96%	51.93%	45.25%	45.35%	51.19%	49.21%
SURF	66.19%	65.69%	56.25%	56.16%	65.14%	64.08%
ORB	43.42%	44.45%	38.00%	38.24%	43.99%	41.98%
MSER	66.51%	66.60%	58.05%	57.84%	64.57%	64.07%

Figure 3.12: Chart indicating the error introduced by image rotation for a wide range of features detectors. The horizontal axis describes the image rotation in degrees.

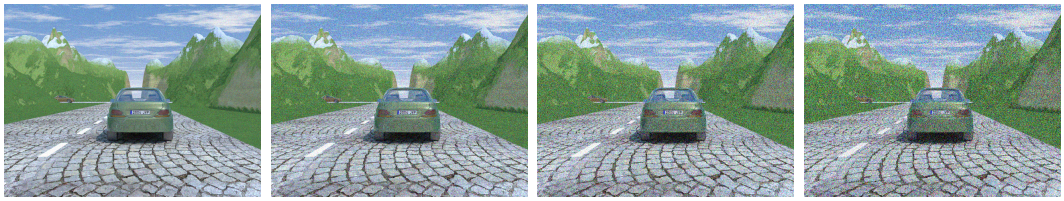


Figure 3.13: Sample images used in the noise test. From left to right the Gaussian noise is increasing by 5%, 10%, 15%, 20%.

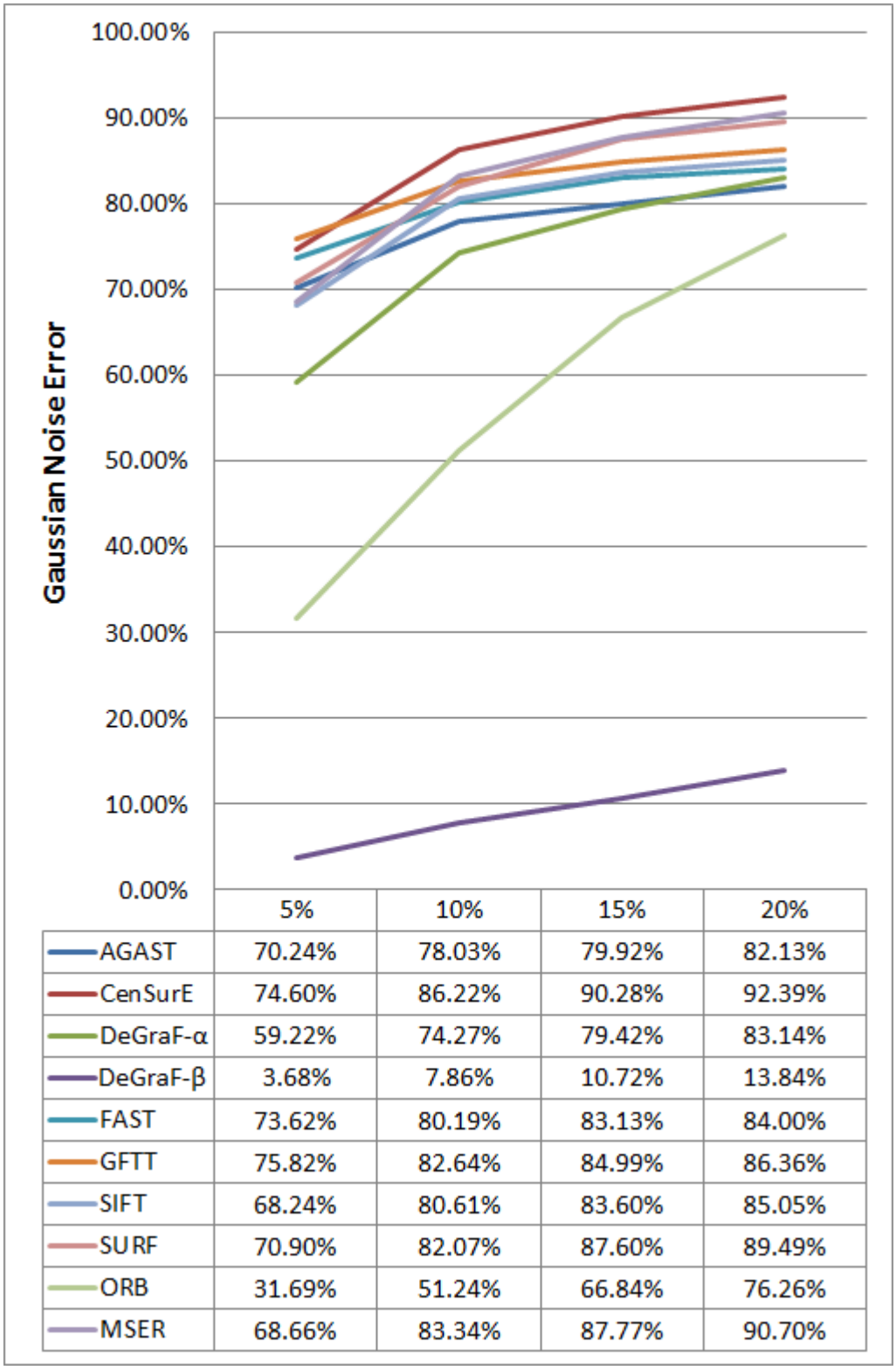
the performance order is important, since they are the only other approaches that use intensity-weighted centroids as part of the feature-extraction process. This is a remarkable result since it allows the development of real-time feature detectors that extract stable features in low-quality images. The remaining approaches have an estimated error rate of at least 68-76% in images with 5% added noise.

3.2.9 Detection time

The execution time of a feature detector is one of the key factors in performing real-time 3D reconstruction. *Table 3.4* and *Figure 3.15* show the average detection time in milliseconds across all the dataset images. The number of mathematical operations for each detected feature is fixed thus an error measurement is negligible in all cases, since the difference in CPU execution time is measured in nanoseconds. FAST, DeGraF-β and CenSurE are the fastest with similar performance. AGAST, DeGraF-a, GFTT, SURF and ORB are slightly slower but still suitable for real-time applications. SIFT and MSER are the most computationally-expensive. In the case of SIFT, higher performance can be achieved by adjusting the σ parameter, however, this happens at the expense of keypoint density. Generally, the presented results should be considered only as a rough guideline since execution time of each detector depends on hardware-specific optimisation and the nature of the evaluated dataset.

3.3 Discussion & Conclusions

This chapter presented a detailed evaluation of a wide range of feature-detection approaches in order to measure their suitability for developing automotive vision applications and more specifically for performing real-time 3D reconstruction. In



	5%	10%	15%	20%
AGAST	70.24%	78.03%	79.92%	82.13%
CenSurE	74.60%	86.22%	90.28%	92.39%
DeGraF-α	59.22%	74.27%	79.42%	83.14%
DeGraF-β	3.68%	7.86%	10.72%	13.84%
FAST	73.62%	80.19%	83.13%	84.00%
GFTT	75.82%	82.64%	84.99%	86.36%
SIFT	68.24%	80.61%	83.60%	85.05%
SURF	70.90%	82.07%	87.60%	89.49%
ORB	31.69%	51.24%	66.84%	76.26%
MSER	68.66%	83.34%	87.77%	90.70%

Figure 3.14: Chart of error introduced by the presence of noise for a wide range of features detectors. The horizontal axis describes the added Gaussian noise as a percentage of the affected image pixels.

Feature Detector	Time (msec)
AGAST	0.033
CenSurE	0.027
DeGraF-α	0.035
DeGraF-β	0.024
FAST	0.02
GFTT	0.043
SIFT	0.109
SURF	0.039
ORB	0.037
MSER	0.293

Table 3.4: Feature detector execution time

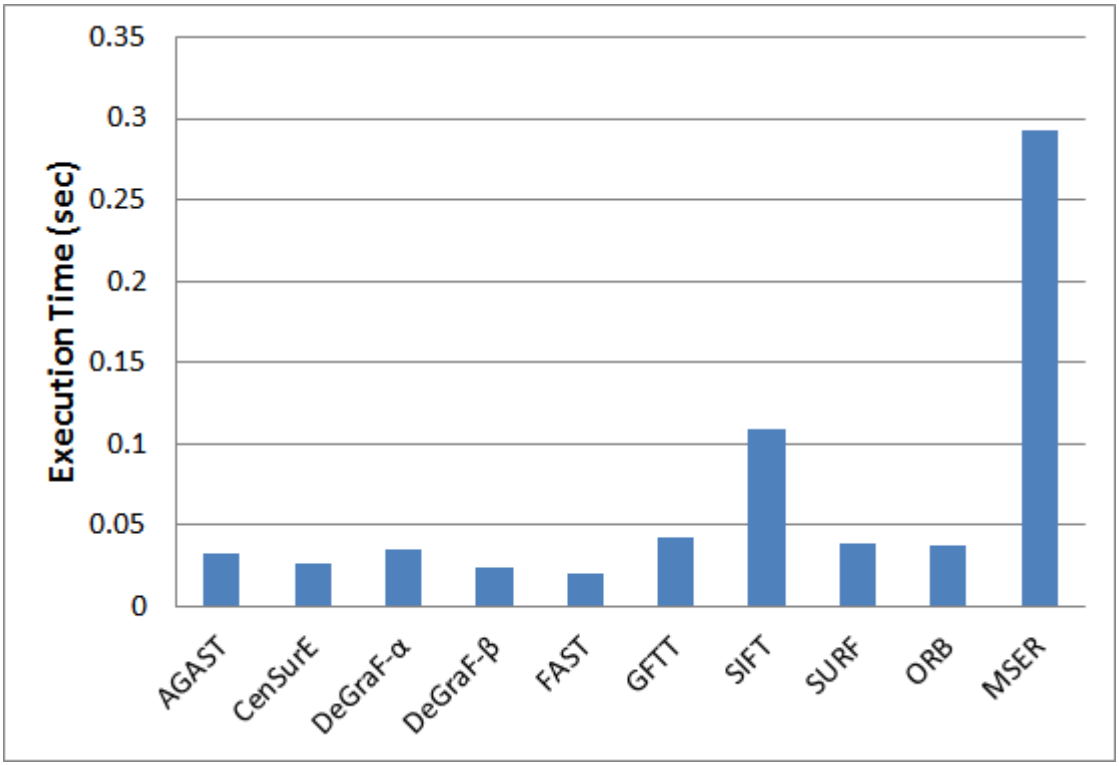


Figure 3.15: Feature detector execution time chart.

addition, a novel approach for real-time dense feature extraction has been proposed based on gradient maps. Like ORB, DeGraF uses intensity weighted centroids but using a new noise-resistant mathematical model. A gradient is redefined as a vector connecting a negative to a positive centroid, where both centroids are symmetric about the centre of the area for which the gradient is calculated. As *Figure 3.2* illustrates, the proposed gradient calculation approach outperforms other approaches when applied to noisy images. In this context, each gradient centroid is a stable local feature that describes the underlying area. This class of features is called DeGraF-β. Combining neighbouring DeGraF-β features allows the detection of more distinctive features. DeGraF-α features are defined as centre-surround features (similar to CenSurE) where the central gradient vector is either a local maxima or minima. Both DeGraF-α and DeGraF-β features are compared to AGAST, CenSurE, FAST, GFTT, SIFT, SURF, ORB and MSER by analysing a diverse range of quality criteria such as keypoint density, tracking accuracy, illumination invariance, rotation invariance, resistance to noise and execution time.

The first evaluation methodology focussed on keypoint density, which is essential for dense 3D reconstruction. In this case, all existing methodologies are optimised for maximum keypoint density, which is achieved by lowering the value of various feature quality-related thresholds. DeGraF-β demonstrated the highest performance followed by DeGraF-α, AGAST, SIFT, FAST GFTT and ORB. On the other hand CenSurE, SURF and MSER had the lowest density values. However, keypoint density by itself is not a useful measure unless it is combined with high tracking accuracy and repeatability of features.

For measuring tracking accuracy, a car vibration simulation model was chosen. This way, the tracking error of each detected keypoint is measured for 1, 2, 4, 16 and 32-pixel vibration amplitude. As expected, most feature detectors start with low error rates at low amplitudes that gradually get worse. DeGraF-α and DeGraF-β performed slightly better than the other methodologies, although the difference is small compared to the potential measurement error that might be caused by non-optimum algorithm configuration. Generally, finding the perfect evaluation parameters for each approach is a time-consuming task that should be considered

in future work.

Repeatability of features under variable illumination conditions was evaluated by gradually increasing the brightness of images. In this case, DeGraF-α and DeGraF-β outperformed all other approaches by a significant margin, with ORB being the only close competitor. MSER was by far the approach with the largest error margin, whereas the remaining approaches demonstrated similar behaviour. The main conclusion from this test is that methodologies based on intensity-weighted centroids (DeGraF-α, DeGraF-β and ORB) perform reliably under variant illumination conditions.

Repeatability of features in rotated images is another important aspect of the evaluation process since it guarantees the stability of feature-detection algorithms when the camera rotates around the axis of motion (rolling effect). In a car, this effect can be described as uneven vibration of the front suspension causing the image to rotate by ± 3 degrees. This behaviour was simulated by artificially rotating the images around their centre. Subsequently features were extracted from each rotated image before being back-projected on the original image. The error caused by image rotation was then measured, showing that each feature detector exhibits different behaviour. The total error margin was in the range of 38% to 68% with DeGraF-β and ORB giving better results albeit with significant error. GFTT and SIFT also demonstrated reasonable performance, whereas all the other approaches failed to achieve high repeatability in this test. Specifically, for DeGraF-α and SURF this can be explained by the fact that the features are extracted from rectangular areas that are dependent on rotation by definition.

The resistance of feature detectors to noise was also evaluated by incrementally adding Gaussian noise to the dataset. The results in this case were remarkable. DeGraF-β outperformed all other approaches by a large margin of 28% for low-noise images and 62% for high-noise images. ORB had the second lowest error rating with also a significant margin over DeGraF-a that came third. All other approaches demonstrated high sensitivity to noise. The conclusion from this test is similar to the illumination test in that intensity-weighted centroids lead to more reliable features being detected in poor quality images.

Finally, the real-time performance of all approaches was evaluated with most approaches demonstrating low execution times with the exception of SIFT and MSER that were significantly slower. The fastest detectors were FAST, DeGraF-β and CenSurE followed by AGAST, DeGraF-a, GFTT, SURF and ORB.

Overall, some interesting conclusions can be drawn from comparing a wide range of feature detectors. Firstly, assessing feature-detection methodologies in an automotive context highlights some different challenges which would not be obvious using generic non-automotive datasets. For the first time, the effect of increasing keypoint density is evaluated based on a wide range of quality criteria such as tracking accuracy, illumination, rotation and noise variance. The novel DeGraF-β approach proves competitive in most tests with exceptional performance in the noise and illumination tests. On the other hand, DeGraF-a demonstrates similar performance to well established feature detectors, but still needs further work on optimising certain aspects of the proposed methodology (e.g. rotation variance). However, since 3D reconstruction is likely to be based on local features rather than global features, DeGraF-β is a good starting point for producing dense 3D point clouds since it demonstrated the highest keypoint density of any detector, highest tracking accuracy, second highest repeatability at the rotation test, highest repeatability score at the illumination and noise tests by a significant margin and finally the second fastest execution time.

Chapter 4

Real-time depth estimation using monocular vision

4.1 Overview

The aim of this chapter is to produce a 3D map of the environment around a moving vehicle in order to facilitate faster obstacle detection by prioritising high-likelihood areas. Specifically, the developed methodology is required to work with a single monocular camera and low-cost embedded hardware. Performing real-time 3D reconstruction with such constraints requires a trade-off between low computational complexity, high accuracy and reliability. As outlined in Chapter 2, the first step towards monocular 3D reconstruction is the detection of reliable feature points. A detailed comparison of different feature detectors was performed and it was demonstrated that DeGraF features are the most suitable in terms of high density, repeatability, tracking accuracy, illumination invariance, rotation invariance and low execution time. Most approaches detect such features in multiple frames in order to perform triangulation between different viewpoints. Triangulation data is then combined with the ego-motion parameters of the moving vehicle in order to accurately measure depth. Such approaches also rely on visual odometry for ego-motion estimation, however, in this case the ego-motion parameters are available through the vehicle's Controller-Area-Network (CAN) bus, thus visual odometry is not required.

This chapter describes two novel approaches for performing real-time 3D recon-

struction by dense feature tracking. The first approach tracks DeGraF features with sub-pixel accuracy in order to produce relative depth information. Apart from the novelty of the DeGraF feature detector, this approach also introduces a different method for storing tracking information within the gradient matrix, which significantly increases real-time performance. Furthermore, this methodology eliminates the need for image stabilisation, since noise patterns such as vibration are automatically filtered out by the feature-tracking approach.

A second approach is also proposed for estimating depth by local frequency analysis of DeGraF features. In this case, each image region is described by its gradient. Depth is estimated by measuring the accumulative displacement of the gradient centroid over time. Although the results are not as accurate as with the former approach, the computational overhead is significantly lower.

Finally, a detailed evaluation is performed where the DeGraF-based approaches are compared to other state-of-the-art methodologies. It is shown that DeGraF feature tracking produces the most dense and accurate depth maps in real-time even under challenging conditions.

4.2 Depth estimation by dense feature tracking

4.2.1 Background

This section describes a novel way of using Bougeut's variant of the Lucas Kanade (LK) algorithm [68] to accurately track dense gradient features and estimate depth. The input to the LK algorithm is a set of feature points that can be tracked reliably. Previous approaches have used known feature detectors such as AGAST [39], FAST [32], CenSurE [38], Good Features To Track (GFTT) [28], SIFT [1], SURF [2], ORB [40] and others. The primary aim of all these feature detectors is the extraction of locally or globally unique features, so they are usually sparse. In order to achieve dense 3D reconstruction, dense features are needed. The keypoint density of feature detectors such as SIFT, FAST or GFTT can be increased by lowering certain quality-related thresholds, however in this case the detected features are not uniformly distributed across the entire image area, leading to high concentration of tracked

features only in certain parts of the image. In this case the computational overhead is significant. Alternatively, feature matching can be performed by using the feature descriptors to associate corresponding features between two images, thus eliminating the need for tracking. However, in a high frame rate video sequence feature matching does not have any advantages over the aforementioned LK approach since the dense features tend to have similar descriptors making their detection problematic. Finally, another way of solving this problem is by using dense optical flow algorithms [98, 99, 143], which are generally featureless and can track each image pixel separately [144]. However, their accuracy is dependent on image texture, whereas high real-time performance is achieved by image subsampling, which causes partial loss of information.

The proposed approach is a hybrid between traditional feature tracking and dense optical flow that demonstrates high real-time performance without image subsampling. Instead of tracking each pixel in an image, the image is divided into a grid of overlapping regions, with each region described by its gradient. Although the gradient-matrix resolution can be lower than the image resolution, the gradient vectors have been formed by all the underlying pixels, thus accuracy remains high. In this case, the LK-tracking algorithm uses evenly distributed gradient features so the maximum number of tracked points is constant regardless of texture. Textureless surfaces have zero gradient so they can be excluded *a priori*. It is shown that this approach outperforms other real-time feature-tracking and optical-flow approaches while producing very dense motion fields that are subsequently converted to depth maps. Finally, by using a unique way of storing tracking information within the gradient matrix, the need for image stabilisation is eliminated, which further reduces the overall computational complexity of the algorithm.

The proposed methodology for estimating depth from gradient features can be outlined as follows:

1. Generate gradient matrix for image I_{t-1}

2. Extract DeGraF features (DeGraF-α or DeGraF-β) for image I_{t-1}

3. Track features between images I_{t-1} and I_t using the LK algorithm (without

Kalman filter)

4. Store the horizontal and vertical displacement (dx, dy) within the gradient matrix

5. Calculate the accumulative euclidean displacement over n frames

6. Calculate the accumulative ego-motion vector over n frames

7. Estimate depth of each gradient feature by comparing its motion vector to the ego-motion vector

The following section describes this methodology in depth.

4.2.2 Methodology

The first step is to generate a gradient matrix G for image I_{t-1} with each cell corresponding to an image area with dimensions $w_C \times h_C$ pixels (see Section 3.2.2). Neighbouring image regions overlap by δ_x horizontally and δ_y vertically. In this chapter the results have been produced using $w_C = h_C = 15$ and $\delta_x = \delta_y = 5$, unless otherwise mentioned. The choice of these parameters is based on the size of the processed images and the real-time requirements of the 3D reconstruction algorithm. Lower w_C and h_C values can be used for faster processing, whereas lower δ_x and δ_y lead to a denser gradient matrix. A denser gradient matrix would in turn increase the resolution of the reconstructed 3D map. Subsequently, DeGraF-α or DeGraF-β features are extracted before they are passed to a pyramidal LK tracker. Since DeGraF-β features are denser this section will focus only on those.

Bougeut's implementation of the pyramidal Lucas Kanade (LK) tracking algorithm[1] [68] tracks DeGraF-β keypoints between two subsequent frames I_{t-1} and I_t. Practically these keypoints correspond to the weighted centroids of the underlying image region. As a result, they are optimal for tracking even when the visible texture is minimal. They are also very robust to noise, thus guaranteeing repeatability and tracking accuracy even in poorly illuminated areas of the image. Once

[1] The OpenCV implementation of the pyramidal LK tracking algorithm is used with the following parameters: $\varepsilon = 0.03$, $max\_count = 20$, $window\_width = 31$, $window\_height = 31$.

tracking information has been extracted the horizontal and vertical displacement of each feature are derived as:

$$d_x = x_t - x_{t-1} \tag{4.1}$$

$$d_y = y_t - y_{t-1} \tag{4.2}$$

where (x_t, y_t) and (x_{t-1}, y_{t-1}) are the coordinates of the tracked point $P_{x,y}$ at time t and $t-1$ respectively. Tracking information is stored in the gradient matrix by moving the gradient from position (i_{t-1}, j_{t-1}) to gradient-matrix position (i_t, j_t), where:

$$i_t = i_{t-1} + \frac{d_x}{\delta_x} \tag{4.3}$$

$$j_t = i_{t-1} + \frac{d_y}{\delta_y} \tag{4.4}$$

Historical information of the position of each feature in the past n frames is also stored in the gradient matrix structure. As a result, the magnitude d and angle φ of the motion vector of each feature are defined as:

$$d = \sqrt{\left(\frac{\sum_{t-n}^{t} d_x}{n}\right)^2 + \left(\frac{\sum_{t-n}^{t} d_y}{n}\right)^2} \tag{4.5}$$

$$\varphi = atan2(d_x, d_y) \tag{4.6}$$

where $atan2$ is the quadrant-aware version of $arctan$.

The advantage of this approach is that unwanted noise such as vibration is eliminated since the accumulative displacement in the case of vibration is close to 0. This behaviour is clearly illustrated on *Table 4.5* in the results section below.

Relative depth can be estimated by comparing the motion vector of each feature with the ego-motion vector $\vec{\varepsilon}$ of the vehicle. Firstly, the vehicle-velocity vector is projected on the image plane so that it represents the pixels per frame travelled

by a point on the bottom row of the image. In *Figure 4.1*, the ego-motion vector ε_1 is projected on the ground plane as vector ε_2, which is equal in magnitude to ε_1, but points in the opposite direction since it indicates how fast a given point is approaching the moving vehicle. The projection of ε_2 on the image as vector ε_3 indicates how fast an image feature moves in pixels per frame. Since the real-world location of this point is known as well as its actual velocity vector, then each other motion vector on the image can be calculated by comparing its angle and magnitude to the projected ego-motion vector. The relative depth D is derived using the following equation:

$$D_{(x,y)} = \sqrt{\left(\frac{\sum_{t-n}^{t} \frac{d_x sin\vartheta}{1+\varepsilon_x}}{n} \right)^2 + \left(\frac{\sum_{t-n}^{t} \frac{d_y cos\vartheta}{1+\varepsilon_y}}{n} \right)^2} \tag{4.7}$$

where ε_x and ε_y is the projection of the ego-motion vector on the x and y axis respectively (measured in pixels) and ϑ is the angle between the z-axis and the ego-motion vector, as illustrated in *Figure 4.1*. In practice, when a vehicle is moving in a straight line then $\vartheta = 0$ and $\varepsilon_x = 0$, so the equation is simplified as:

$$D_{(x,y)} = \frac{\sum_{t-n}^{t} \frac{d_y}{1+\varepsilon_y}}{n} \tag{4.8}$$

The main contribution of this approach is that it produces dense depth maps from monocular video sequences with minimal computational overhead. This is down to the novel DeGraF features as well as the integration of tracking information with the gradient matrix. Only dense optical flow approaches can demonstrate similar performance in terms of density but at significantly lower execution speed [54]. Furthermore, the proposed approach performs stabilisation of the tracked features thus eliminating the need for full-image stabilisation as proposed by alternative methodologies [120, 145]. Finally, most approaches work best in well-textured images with the camera moving laterally. In contrast, the above methodology, works equally well for both longitudinal and lateral camera motion. This is an important advantage, considering that a vehicle is moving forward and in a straight line for most of the

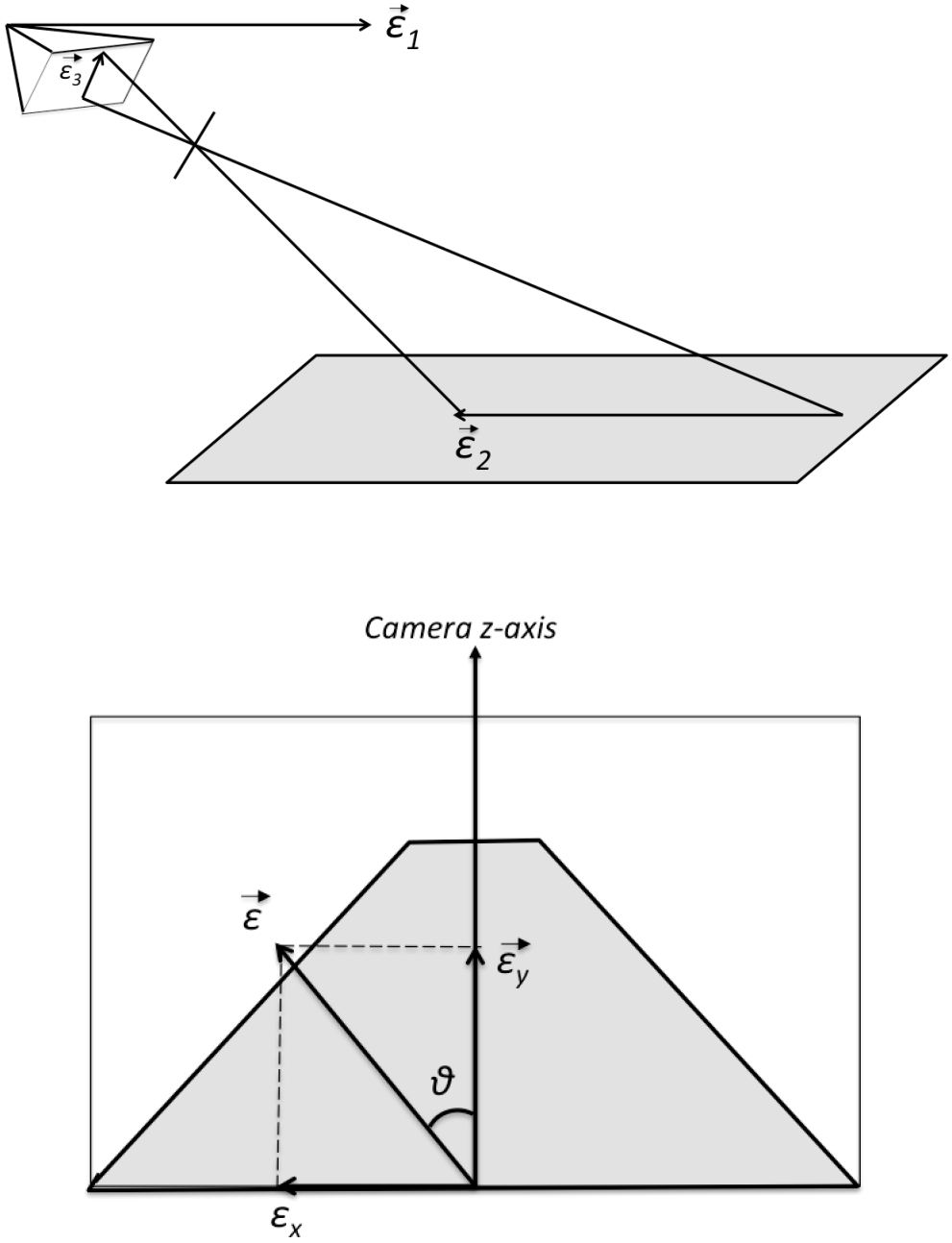


Figure 4.1: Top: The projection of the ego-motion vector ε_1 to the ground plane as vector ε_2 and subsequently to the image plane as vector ε_3. Bottom: The projected ego-motion vector is projected onto the image x and y axis as ε_x and ε_y.

time.

4.3 Depth estimation by local frequency analysis

In this section an alternative methodology is presented for estimating depth by measuring the oscillation frequency of local gradient features. This novel method is based on the fact that DeGraF gradients can accurately measure local image variance with sub-pixel accuracy. It is shown that the local frequency by which the centroid oscillates around the gradient window centre is proportional to the depth of each gradient centroid in the real world. Additionally, by eliminating the need for conventional feature tracking, significant gains in real-time performance can be made. Of course the lower computational complexity of this methodology comes at the expense of depth-map accuracy as the camera velocity increases. However, it is still mentioned as an alternative solution for low-cost obstacle-detection applications that only require a rough depth map in order to prioritise higher-risk areas during image indexing. The steps described in the following paragraphs are:

1. Perform image stabilisation (Optional)
2. Calculate difference of Gaussians image
3. Generate gradient matrix
4. Calculate local oscillation frequency
5. Generate depth map

Firstly, the input image is stabilised using the method by Grundmann et al. [145] in order to reduce the effect of vibration that may be present during capture. This step is recommended if no optical stabilisation is present.

In the second step, the input image is converted to the equivalent DoG image as outlined in Section 3.2.2. This conversion makes the gradients illumination invariant.

In the third step, the gradient matrix is derived with each gradient being calculated for a predefined window with dimensions $w_g \times h_g$. The choice of window size depends on:

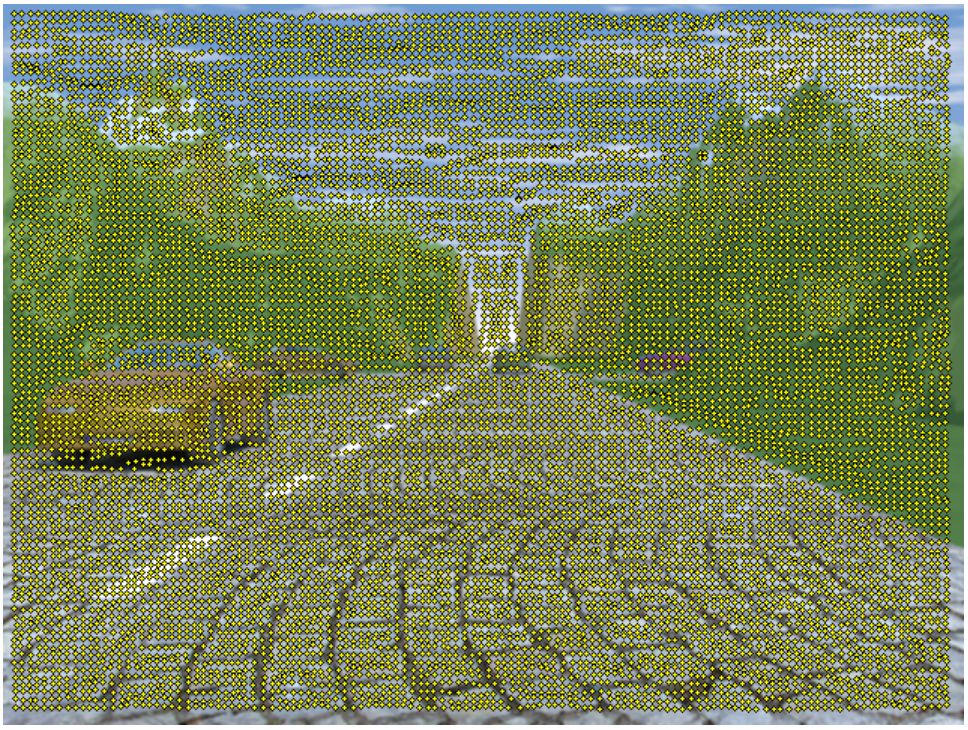


Figure 4.2: The input image with overlaid centroids (yellow) that have been calculated from the DoG image. Here the centroids are sparser than in practice for improved visualisation.

- the image noise level (larger window size leads to more robust gradient measurement in noisy images)

- the expected maximum displacement of features between subsequent frames. For example, if a feature moves by 4 pixels or more in a single frame then a 3×3 window is too small for measuring this displacement. In this case, the solution is to either use a larger gradient window or perform local frequency analysis at multiple pyramid levels.

Figure 4.2 illustrates an example where the gradient centroids have been derived for each gradient window. In this case, the centroids appear sparse for better visualisation since the actual density is too high.

In the final step, the horizontal and vertical displacement of each centroid is calculated between two subsequent frames I_{t-1} and I_t. The centroid displacement is accumulated over n frames in order to derive the local oscillation frequency. This frequency is linked to the depth of each centroid in the real world.

The main difference between this approach and general structure from motion (SfM) approaches is that DeGraF features are not being explicitly tracked. Instead the accumulative displacement of the centroids is measured over n frames (i.e. oscillation frequency). Each time one or more pixels change value then the gradient centroid moves accordingly. Since the pixel values represent the difference of Gaussians, the centroid motion is directly related to the actual displacement of each image feature. Using this novel method, the need for a tracking algorithm is eliminated, albeit at the expense of depth-map accuracy. The algorithm pseudocode is as follows:

```
1  read image with dimensions image_width, image_height;
2  generate grid with dimensions grid_width, grid_height, cell_width, cell_height,
       cell_spacing_x, cell_spacing_y;
3  for each grid cell
4      set max_value to the local maximum pixel value;
5      set s_pos to the sum of the pixel values;
6      set s_neg to the sum of the inverted pixel values, where inverted_pixel_value =
            1 + max_value - pixel_value;
7      if s_pos > s_neg
8          set c_pos.x to the average of x values weighted by their corresponding
               pixel values;
9          set c_pos.y to the average of y values weighted by their corresponding
               pixel values;
10     else
11         set c_pos.x to the average of x values weighted by their corresponding
               inverted pixel values;
12         set c_pos.y to the average of y values weighted by their corresponding
               inverted pixel values;
13     endif
14     c_neg.x = 2*cell_centre.x - c_pos.x;
15     c_neg.y = 2*cell_centre.y - c_pos.y;
16     dx = c_pos.x - c_neg.x;
17     dy = c_pos.y - c_neg.y;
18     gradient_magnitude = sqrt(dx*dx + dy*dy);
19
20     add gradient_magnitude to accumulative_displacement;
21     average_displacement = accumulative_displacement / accumulator_size;
22 end loop
```

The size of the accumulator is dependent on the application and the framerate of the camera. For example, when using a camera capturing images at 30 frames per second and the required reaction time is 1 second then the accumulator uses the

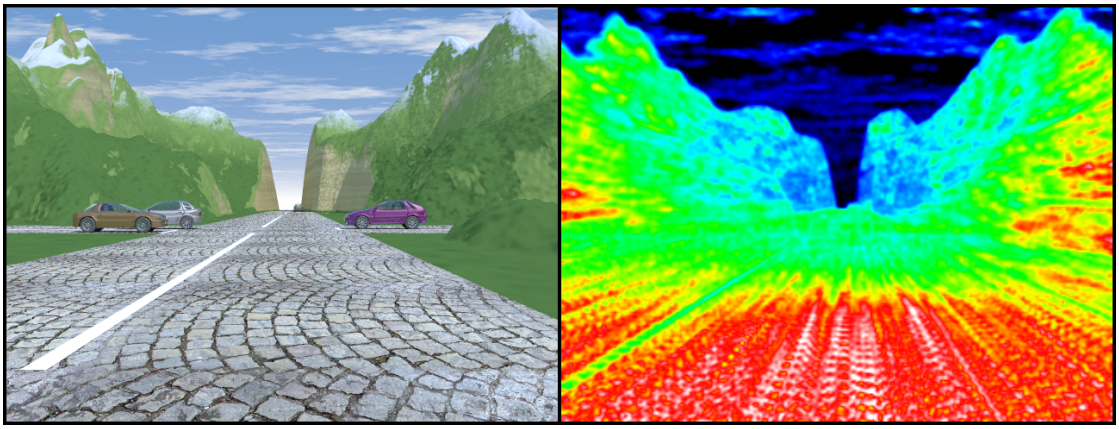


Figure 4.3: A relative depth map derived by measuring the oscillation frequency of DeGraF features.

past 30 frames.

Assuming a scene with all objects stationary and a vehicle-mounted camera moving forward in a straight line then the displacement of the gradient-feature centroids over time is proportional to the depth of each feature in the real-world. Any moving object is detected as an anomaly on the depth map since it has significantly higher frequency. In the case, of lateral camera motion then the ego-motion parameters shall be used to correct the depth map. *Figure 4.3* illustrates a relative depth map derived using this approach. It should be noted that it is called relative because it only calculates the position of each feature relative to the camera, but not the absolute position in the real-world coordinate system. Still such an output is useful for prioritising image indexing, if depth accuracy is not an issue. This method is mentioned as a lower-performance and faster alternative to the feature-tracking method since it is at least five times faster. The speed increase is caused by the elimination of the Lucas-Kanade tracking algorithm [68]. Evaluating such a method presents a problem. There are no datasets with ground-truth on relative-depth maps and on the other hand, there is no way to convert an absolute depth-map into a relative one. A new dataset would have to be developed but such a task is outside the scope of this thesis. As a result, this methodology is presented as an alternative type of 3D reconstruction but it is not evaluated in the following sections.

4.4 Experimental Methodology

The DeGraF-based approach for 3D-reconstruction is compared to other state-of-the-art approaches. The aim is to identify the methodology that offers the most accurate and dense depth maps by just using monocular vision while keeping computational complexity low. The selected approaches derive structure from motion either by feature tracking or by dense optical flow. Based on the results from Chapter 3, the following approaches were chosen: 1) AGAST [39], 2) SIFT [1], 3) FAST [32], 4) GFTT [28], 5) ORB [40].

4.4.1 Data Capture Hardware

Ego-motion measurement is normally derived by automotive sensors including yaw-rate sensor, speedometer and accelerometer. Such sensor data is available on the Controller Area Network (CAN bus), which exists in most modern vehicles. However, practical limitations such as availability and high cost of commissioning a real vehicle with accessible CAN information, meant that alternative hardware had to be used to simulate normal vehicle behaviour. In this case, a phidget sensor was used that combines a digital compass, 3-Axis Gyroscope and 3-Axis Accelerometer. Since none of these sensors measures velocity, this data was extracted from a bluetooth GPS. Finally, a wide range of cameras were tested from low-cost webcams to high-quality PointGrey FireflyMV cameras with a wide angle lens attached.

4.4.2 Data Capture Software

Capturing data from multiple sensors requires accurate synchronisation especially in the case where different hardware components capture data at different frequencies. For example, in this case camera frames are acquired at a frequency of 30 Hz, the motion sensor is operating at 200 Hz and the GPS at 1Hz. Logging and synchronisation of data from these sensors can be achieved using one of the following approaches:

1. The most straightforward solution to sensor synchronisation is to use the camera as the main sensor and store motion and GPS information within each

frame header. This approach is commonly used and it is also the easiest to implement. However, such an implementation would mean that the motion sensor data is less accurate since its actual refresh rate is significantly faster than the camera framerate. Attempting to lower the motion sensor frequency to match the camera framerate produces very noisy results. Also in the scenario where data is captured from multiple cameras operating at different framerates the cameras cannot remain in sync using this approach.

2. The second option is to use commercially-available or open-source software. Unfortunately, none of the available options was suitable either due to high cost or hardware incompatibility.

3. The third option is to develop an interrupt driven approach for capturing data from each sensor separately as soon as information becomes available. In the case of using multiple sensors the information bandwidth can significantly increase, thus all sensor data needs to be stored in memory before storing it on the hard-disk. In addition, a global time-stamping mechanism needs to be used so each sensor reading can be synchronised with the rest. Although, this approach offers several advantages, the main disadvantage is that it is hard to implement and requires separate data-capture and data-playback mechanisms for online and offline processing.

After careful consideration and with the aim of producing a long-term solution for different computer-vision projects, the third option was implemented producing a software solution known as Visioner. Its main features include:

1. Interrupt driven data capture and synchronisation with global time-stamping.

   Each time data becomes available on a sensor an interrupt service routine is executed on a separate thread. This routine stores the acquired information in memory. Subsequently, a background low-priority thread stores the information on the hard-disk. This is done to avoid the usual bottleneck of low hard-drive bandwidth, especially when writing large video files from multiple cameras.

2. Synchronisation using global timestamp

   Each sensor output is timestamped as soon as an interrupt is raised. Using a global timestamp means that each sensor output can be logged in a separate file and played back using a timer.

3. Multiple camera support.

   Although, this specific project uses monocular-vision, video may need to be captured using multiple cameras in order to assess different camera configurations. Recording multiple cameras concurrently poses an interesting challenge due to the large amount of information that needs to be managed in real-time. Having a global timestamp allows the capture of synchronised video even if the framerate is different.

4. Memory buffering

   All information is firstly stored in RAM to ensure no data loss. Out of memory exceptions pause recording in order to allow the memory contents to be saved.

5. Asynchronous data logging to hard-disk.

   A low priority background thread saves memory contents to the hard-disk. This functionality ensures that data quality is the same regardless of the underlying hardware. Systems with slow hard-drive bandwidth will run out memory faster when the system is overloaded. However, most systems should be able to read and write information in real-time when using a single camera, motion sensor and GPS.

6. Synchronised data playback.

   In offline mode a global timer is used to playback the captured information. As the timer value increases the system checks for expired timestamps and thus all sensors remain fully synchronised.

7. Online and offline algorithm evaluation.

   Algorithms can run both in online and offline mode, making Visioner suitable for both algorithm development and system deployment.

8. Software development library for hardware independent computer-vision-algorithm development.

   A software library has been developed to allow hardware-independent vision algorithm development, that works with any type of camera, motion sensor and GPS.

9. Plugin support for GUI independent development.

   Each algorithm can be developed in the form of a plugin using the standard Visioner API. This means that the GUI implementation is separated from the core-algorithm development. Likewise, wherever, GUI is required there are API functions for fast deployment.

10. Cross-platform support (Windows, Linux, MacOSX).

    Modern computer-vision systems run on a variety of platforms. Visioner has been built using cross-platform development tools such Qt and OpenCV, thus it is compatible with the most commonly used operating systems. It can also run in embedded linux systems making it suitable for a wide-range of embedded applications.

4.4.3 Datasets

Evaluating a 3D-reconstruction algorithm requires accurate ground-truth data including the real-world coordinates of each voxel. Such datasets have recently emerged where real-world coordinates are derived by a multitude of sensors including laser-scanners (see KITTI dataset [17]). Still, the easiest and most accurate way of testing is using a 3D simulator with rendered automotive environments. The 3D-simulator data guarantees noise-free input for initial algorithm evaluation and allows stepwise noise addition in order to measure robustness. In this case, the 3D coordinates are known with sub-pixel accuracy regardless of range. The results in this chapter have been derived using the MITEC 3D-rendered dataset [141] to compare algorithm

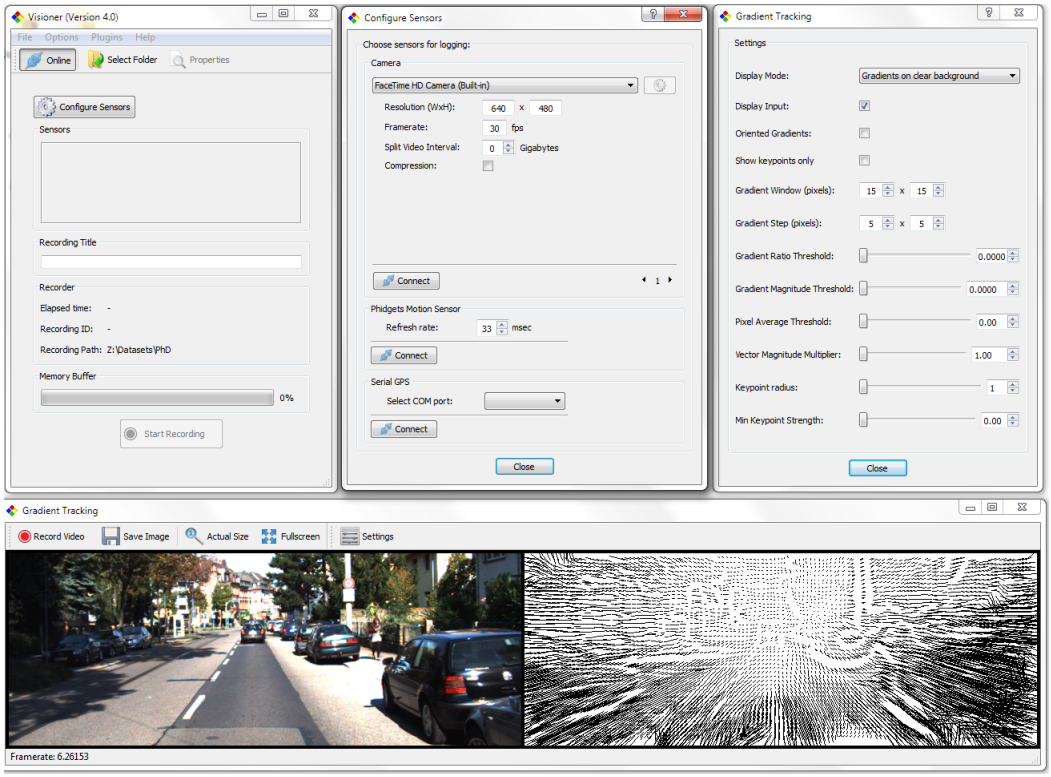


Figure 4.4: Visioner: A data-logging and algorithm-evaluation platform. Here the gradient-tracking algorithm is evaluated on the KITTI dataset.

performance. Subsequently, it is shown that the algorithm performs equally well in real conditions with a noticeable resistance to camera noise and vibration. For this purpose, a dataset has been created using the capture hardware and software mentioned above. Video sequences from the KITTI dataset [17] are also used.

4.5 Results

In this section, 3D reconstruction is performed using a wide range of approaches. These approaches are based on different feature detectors that produce depth maps, which are then evaluated using simulated data. The simulated data includes ground-truth information about the inter-frame motion of each pixel as well as its depth. Real-world examples are also shown in order to illustrate the relative strengths and weaknesses of each approach in the presence of noise and camera vibration.

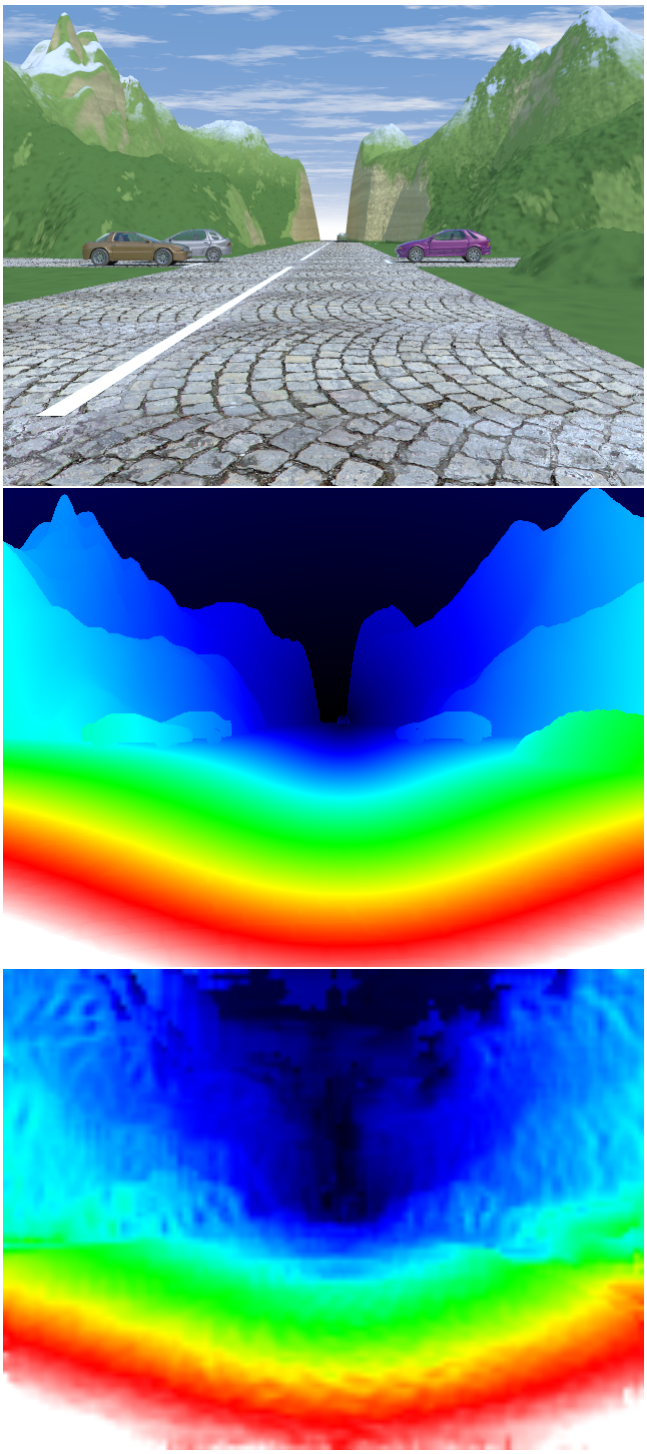


Figure 4.5: Top: The input image from the MITEC dataset. Middle: The motion vector ground truth. Bottom: The motion vector output produced by DeGraF feature tracking.

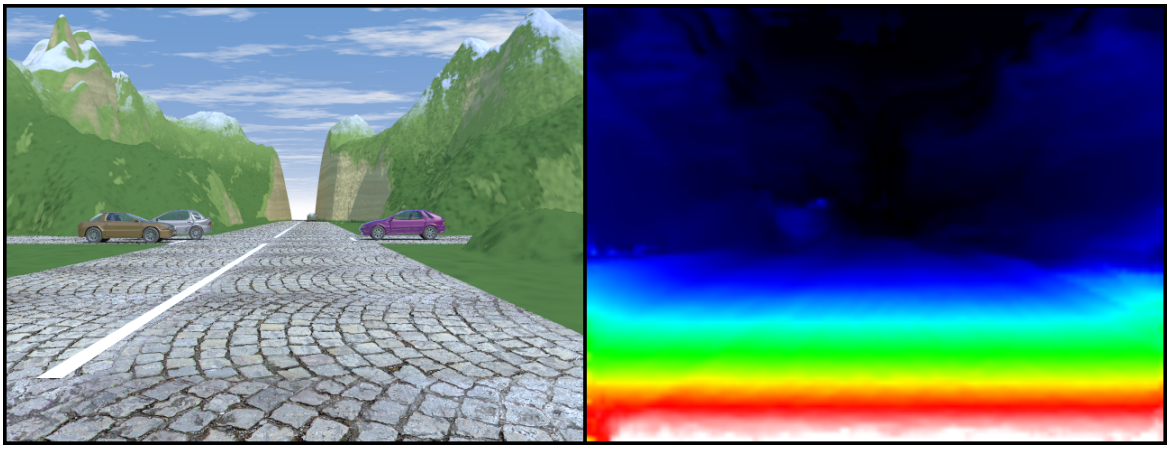


Figure 4.6: The motion-vector map derived by feature-tracking is converted to depth map.

4.5.1 Optical-flow Accuracy

The accuracy of the optical-flow algorithms is tested in an automotive context for each of the evaluated methodologies. As part of the evaluation process, all the algorithms produce velocity vectors for each tracked feature or pixel. These vectors represent the displacement of each point over time on the image plane. All the following experiments have been performed using the 3D-rendered MITEC dataset [141], where the ground truth is a matrix with dimensions equal to the input image. Each matrix element corresponds to the displacement of each image pixel between two consecutive frames with sub-pixel accuracy. Such ground-truth images are illustrated in *Figure 4.5* and are usually used for evaluating dense optical-flow algorithms. Although, most of the evaluated methodologies do not fall in this category, the key component is the accuracy of the displacement of each feature between temporally-adjacent frames. Unfortunately, real data, as provided by the KITTI dataset [17], cannot be used since the resolution of the ground-truth data is not high enough for accurate evaluation. Finally, it is important to note, that the optical-flow accuracy is used as a direct measure of 3D reconstruction accuracy based on *Equation 4.7*. This equation shows that the accuracy of the motion vector is the only contributing factor that affects the actual accuracy of the produced depth map. The remaining parameters are related to the ego-motion vector of the vehicle that is derived using an accelerometer. Since the contribution to knowledge

is based on the image-processing aspect of 3D reconstruction, the ego-motion measurement has been excluded from this evaluation. Otherwise, possible inaccuracies in the ego-motion estimation would result in errors that cannot be attributed to a single source.

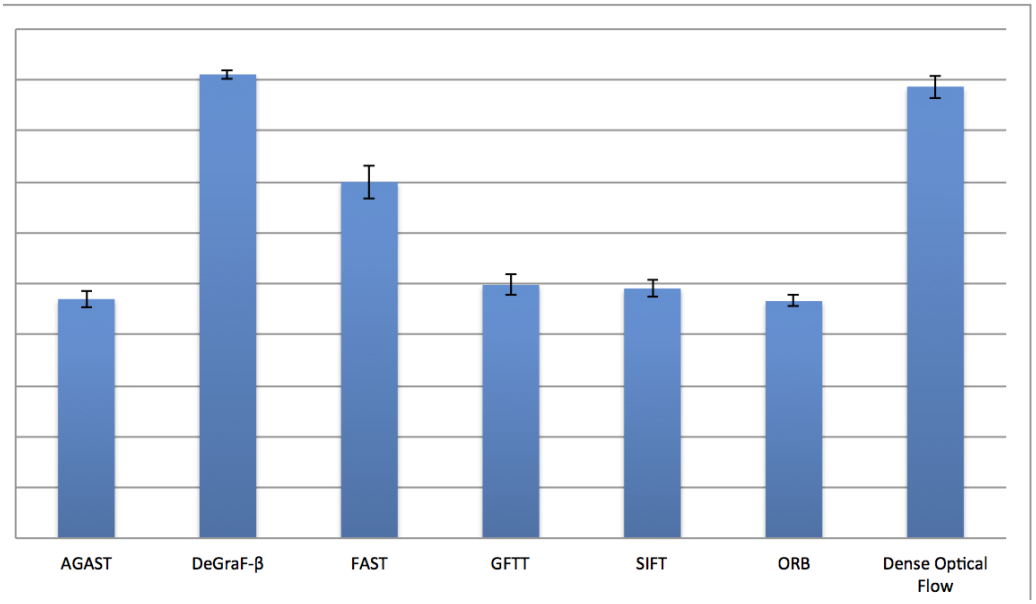


Figure 4.7: Optical-flow-accuracy results using different approaches.

A pyramidal implementation of the iterative Lucas-Kanade tracking algorithm [68] is used to track keypoints between frames. The output is a motion-vector map M as outlined in *Table 4.2*. The displacement of each keypoint is compared to the

ground-truth map G and the average error is defined as:

$$error = \frac{1 - \sum_{i=1}^{h}\sum_{j=1}^{w} min\left(\frac{1+M_{(i,j)}}{1+G(i,j)}, \frac{1+G_{(i,j)}}{1+M_{(i,j)}}\right)}{n} \tag{4.9}$$

where $M_{(i,j)}$ and $G_{(i,j)}$ correspond to the measured and actual displacement of an image point between time $t-1$ and t. Also n denotes the total number of frames in the image sequence with each frame having a resolution $w \times h$. This technique is useful for error measurement when comparing image matrices since it guarantees that the error will always be in the range between 0.0 and 1.0. As a result, the algorithm accuracy can then be expressed as a percentage. Without this method, the error would be expressed in pixels, but such a measurement would not take the size of the image into account. Alternative ratio-metric equations for error measurement are also suitable, but then the error would not be in the 0.0-1.0 range, while division by zero could cause discontinuities in the results. The chart on *Figure 4.7* clearly shows that the DeGraF-β approach produced the most accurate motion-vector map, closely followed by dense optical flow. The FAST-based approach also performed well whereas GFTT, AGAST, SIFT and ORB approaches were less accurate at motion estimation. It is worth noting that these results are derived from the raw output of the tracking algorithm without any interpolation. Interpolation could have significantly increased the accuracy of some techniques, however, it would not give a good indication of the underlying performance. The main conclusions that can be drawn from these results is that using gradient-based features for tracking proves more robust than any other technique. Of course, different feature detectors have specific strengths that favour certain applications. For example, it could be argued that SIFT is more suited to global feature matching for object detection than local feature tracking [1], however, no previous work had measured its performance in this field.

Approach	Optical flow accuracy (%)
PLK Dense Optical Flow	88.72
AGAST	46.97
DeGraF-β	91.11
FAST	69.98
GFTT	49.76
SIFT	49.06
ORB	46.58

Table 4.1: Optical-flow accuracy results based on different feature detectors

4.5.2 Depth-map density

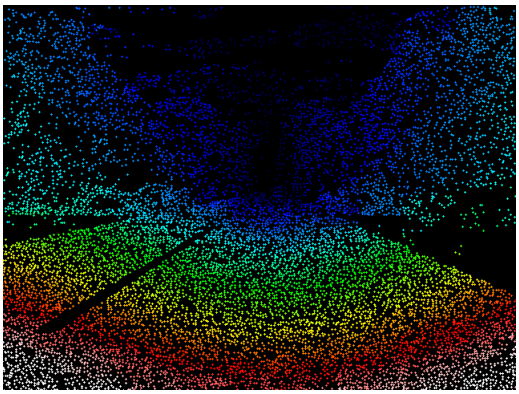
One of the key qualities of a depth map is its density. This measurement indicates what percentage of the image pixels has an allocated depth value. If an approach is based on feature tracking then the result is directly proportional to the keypoint density as presented in Section 3.2.4. In the case of dense optical flow the density is always 100% since every pixel is tracked separately, although this is not necessarily linked to its accuracy. In this section, a wide range of 3D reconstruction approaches are tested by analysing a set of automotive scenes and calculating the average pixel density per motion-vector map over 1000 frames. The relation between motion-vector map density and depth-map density is linear based on *Equation 4.7*. For feature-based approaches the configuration parameters of each detector are adjusted in order to achieve the highest possible feature density. *Table 4.2* illustrates some characteristic examples.

-insert

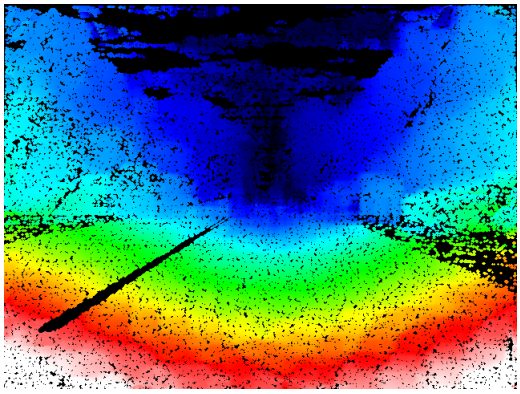
Table 4.2: Motion-vector maps using different approaches.

Input Image

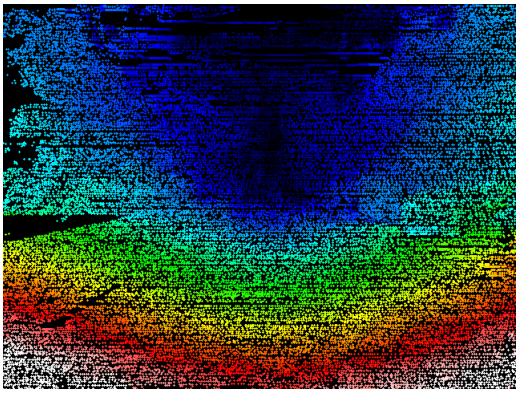
AGAST



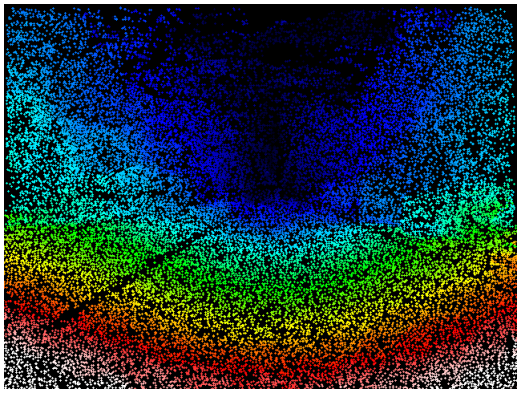
FAST



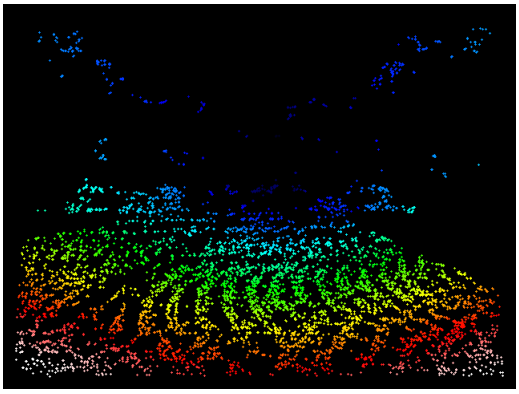
GFTT



SIFT



ORB



Dense Optical Flow

DeGraF-β

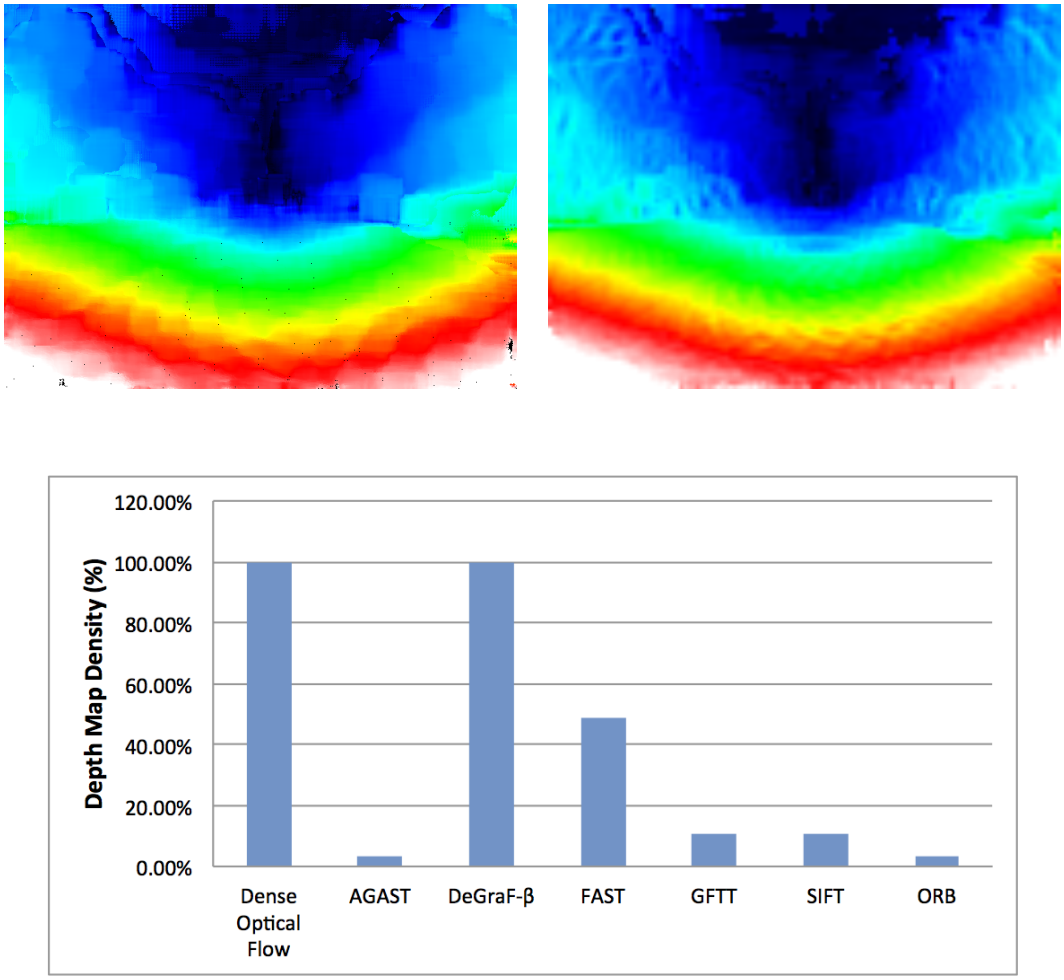


Figure 4.8: Evaluation of depth-map density for a wide range of 3D reconstruction approaches based either on feature-detection or dense optical flow.

The derived motion-vector map density results can be viewed on *Table 4.3* and *Figure 4.8*. Analysing those results shows that performing 3D reconstruction using dense optical flow based on the pyramidal Lucas-Kanade algorithm produces the densest output with 100% coverage. This means that every pixel from the input image has an allocated depth-value. The DeGraF-β-based approach closely matches this performance. In addition, the DeGraF-β output produces more surface detail than the optical-flow approach, which appears blurred. The reason for this behaviour is that the DeGraF approach guarantees that each feature will be located at a centroid, which in turns increases the maximum likelihood of accurate tracking. On the other hand, the optical-flow approach pre-selects fixed keypoints without measuring their quality. As a result, the tracking algorithm may track unstable features, leading to lower accuracy. The FAST-based approach also performs relatively well

Feature Detector	Motion-vector Map Density (%)
PLK Dense Optical Flow	100
AGAST	3.62
DeGraF-β	99.64
FAST	48.71
GFTT	10.54
SIFT	10.47
ORB	3.40

Table 4.3: Motion-vector map density results for different feature detectors and dense optical flow.

by estimating depth for almost half of the image pixels, while the rest can easily be interpolated. Approaches based on AGAST, GFTT and SIFT produce evenly distributed sparse motion vectors that again could provide high area coverage via interpolation. Finally, ORB shows low performance since motion vectors are produced only for highly textured areas. On their own, these measurements are not enough for deriving any useful conclusions without considering motion-vector accuracy and other quality criteria. However, the DeGraF-β-based approach has the highest performance amongst feature-based methodologies, whereas on the other end ORB proves unsuitable for 3D reconstruction.

4.5.3 Noise Sensitivity

The source of noise in a moving-vehicle scenario is dependent on:

- Camera vibration and rolling
- Illumination changes
- Camera sensor noise

In the last chapter, the sensitivity of each feature detector to vibration noise was measured by artificially vibrating an image using a predefined amplitude (1, 2, 4, 8, 16 or 32 pixels). The sensitivity to image rotation was also tested in order to evaluate the rolling-camera effect caused by uneven vehicle-suspension vibration. The sensitivity to illumination was then evaluated by gradually increasing the brightness

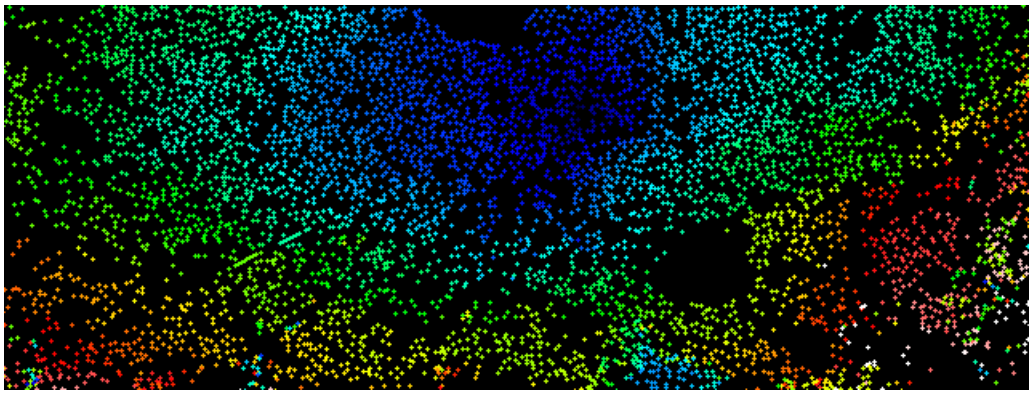
of the image. These results directly affect the sensitivity of the 3D-reconstruction techniques to noise, because if the features are not repeatable across a range of frames, the tracking algorithm will certainly fail to measure the motion-vector magnitude. As a result, the noise sensitivity of a 3D reconstruction algorithm is directly related to the results that have already been presented in *Figures 3.10, 3.12* and *4.4*. This evaluation refers to simulated data. However, it is still interesting to evaluate the noise sensitivity of the aforementioned techniques in real conditions. The problem is that for real scenarios there is no ground-truth data on the amount nor the source of noise, since the environment is changing dynamically in an unpredictable manner. As a result, it is impossible to measure the accuracy of the produced motion-vector maps. Some real-world examples are illustrated in *Tables 4.4* and *4.5*. These can be assessed only by visual inspection, which shows that DeGraF-β produces the least noisy output. The effect of noise can be observed either as error on the motion-vector magnitude or as error in the feature-tracking leading to missed to areas with zero motion-vector magnitude (black shade). These errors can be viewed by examining the colour-coded images in *Table 4.4*. By examining the bottom-right part of the image, it can be seen that the motion-vector magnitude of the vehicle-surface features does not always correspond to the expected values. For example, the corner of the vehicle should appear in white or red shade since it is closest point to the camera. This is not the case on all of the images. The error can be attributed to tracking inaccuracy, since there is a significant inter-frame displacement of features at this part of the image. On the other hand, features that are further away from the camera appear more accurate since the inter-frame displacement is minimal. In the presence of vibration it is also clear that DeGraF-β returns by far the most accurate measurements by making use of the built-in feature stabilisation mechanism. This behaviour is illustrated in *Table 4.5* that shows that DeGraF is the only approach that estimates the distance of the building at the top-centre part of the image correctly. The other approaches are affected by camera rolling and vibration and attempt to track the features on the building, which in turn leads to incorrect depth estimation. As already discussed, these results cannot be mathematically verified, however, this is a common evaluation technique that

other authors have adopted, like for example on the YouTube image stabilisation algorithm [145].

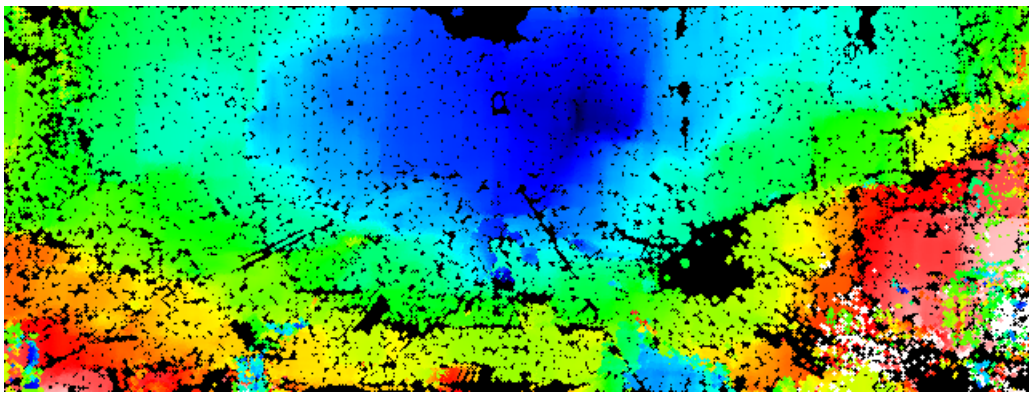
Table 4.4: Motion-vector maps using different approaches on real-world data in order to assess sensitivity to noise.

Input Image

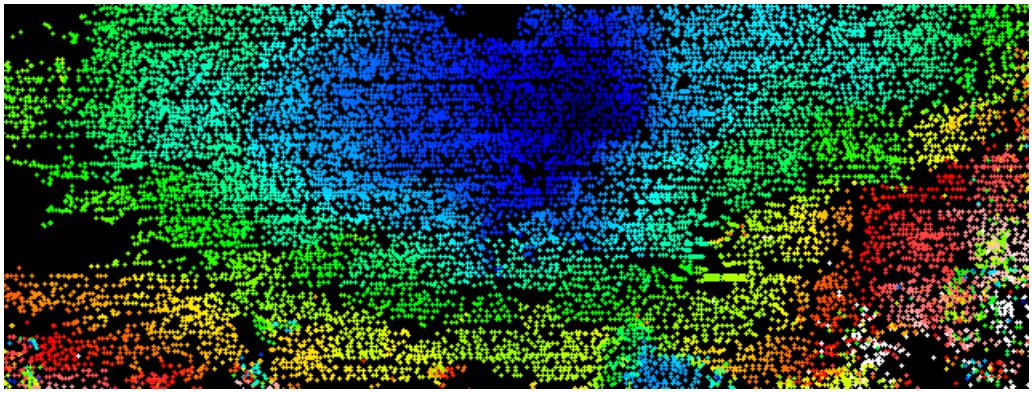
AGAST



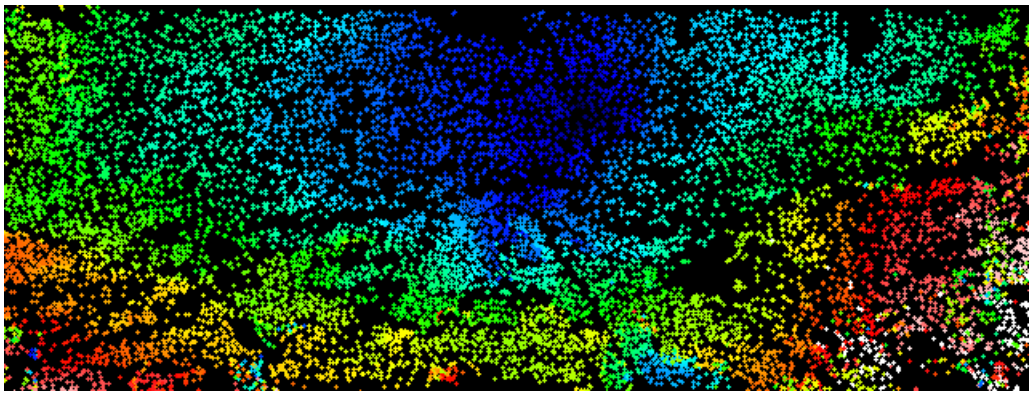
FAST



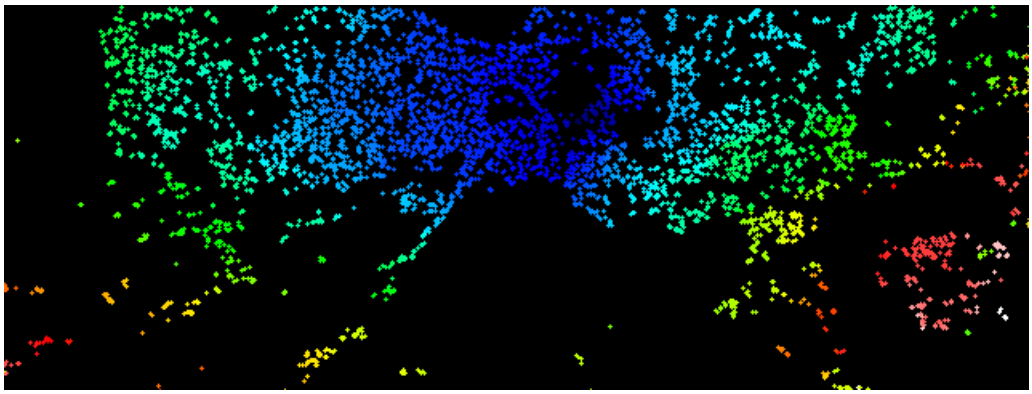
GFTT



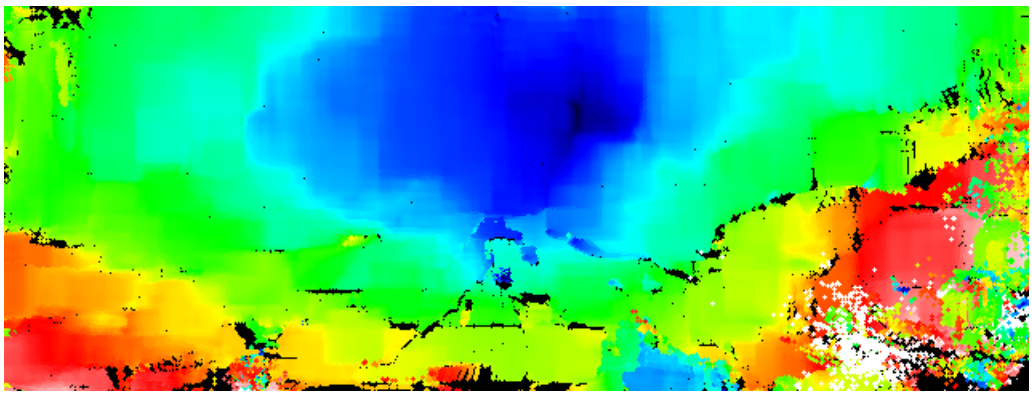
SIFT



ORB



Dense Optical Flow



DeGraF-β

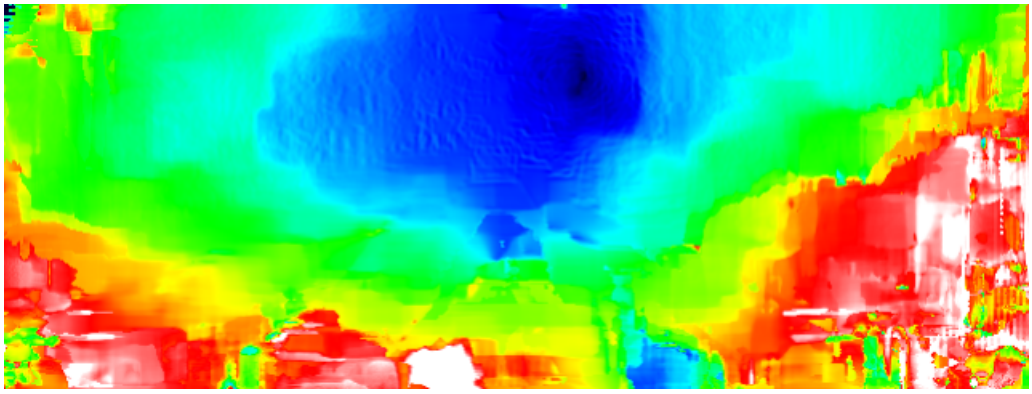
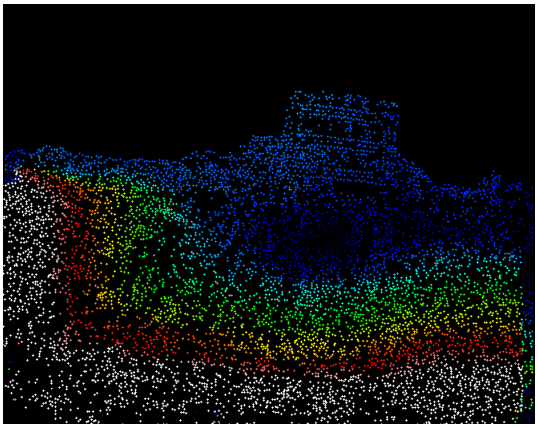


Table 4.5: Motion-vector maps using different approaches on real-world data in order to assess sensitivity to vibration. Note the difference in performance of the DeGraF approach which uses its built-in feature stabilisation function to filter camera vibration.

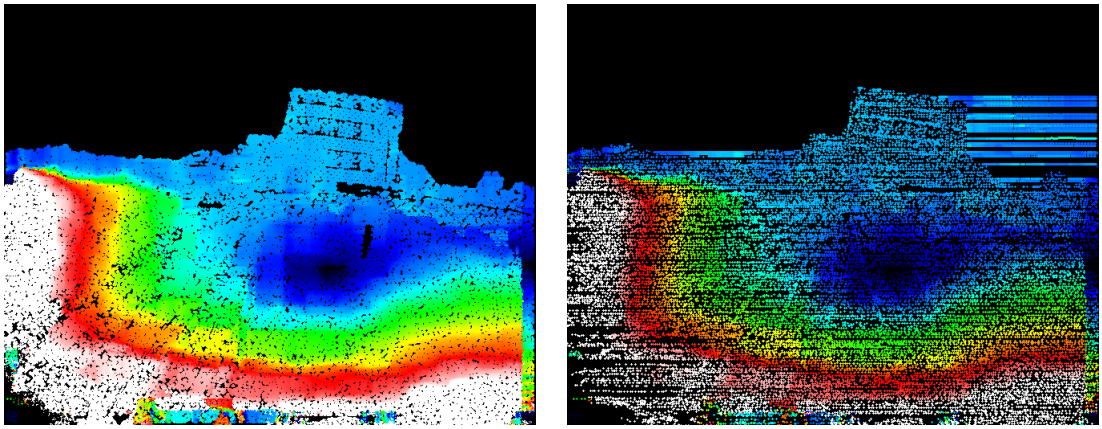
Input Image

AGAST

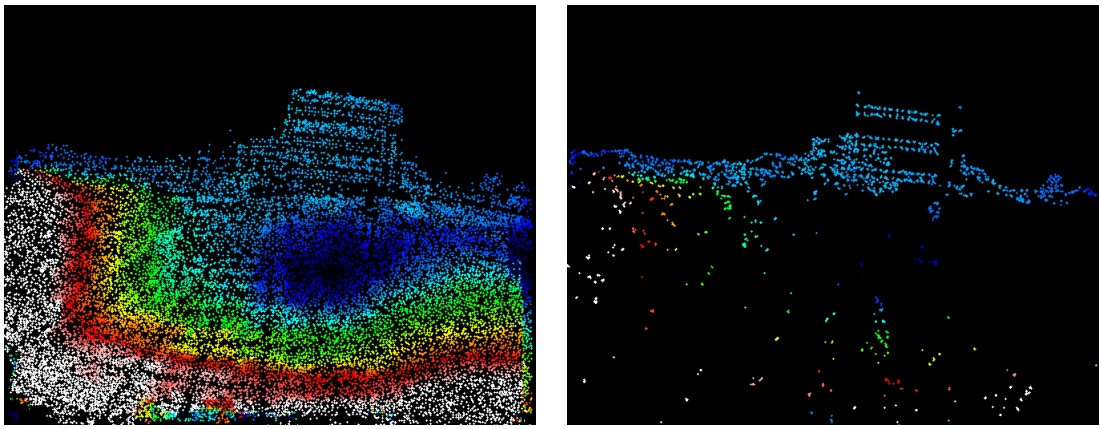


FAST

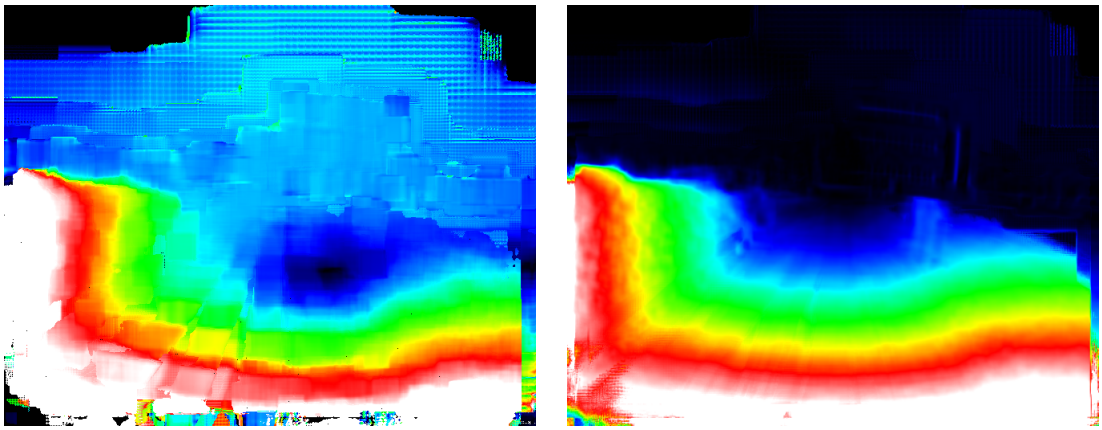
GFTT



SIFT ORB



Dense Optical Flow DeGraF-β



4.5.4 Computational complexity

Low computational complexity is essential for embedded 3D reconstruction algorithms. Currently, most real-time approaches are based on parallel execution of complex algorithms on GPUs [54]. In this section, the real-time performance of the aforementioned approaches is analysed. *Table 4.6* and *Figure 4.9* show the average 3D

Table 4.6: Feature detector execution time

Feature Detector	Time (msec)
PLK Dense Optical Flow	1379.99
AGAST	60.27
DeGraF-β	56.05
FAST	703.08
GFTT	160.34
SIFT	167.07
ORB	41.65

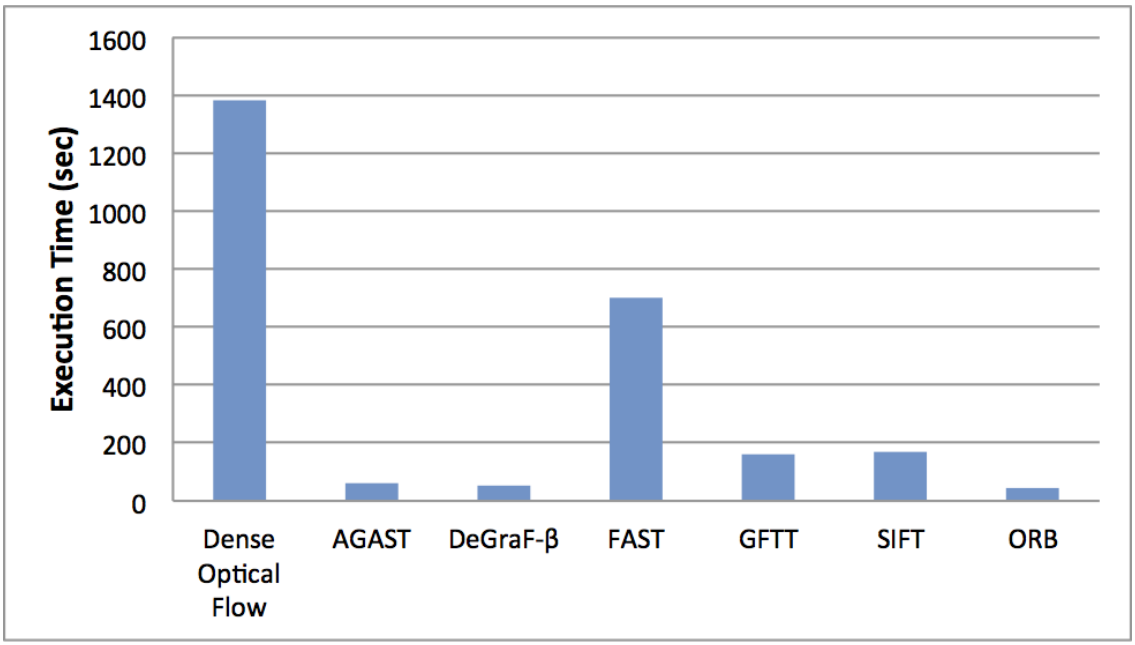


Figure 4.9: Feature detector execution time chart.

reconstruction time in milliseconds across all the datasets. The error is negligible compared to the execution time of each algorithm and has not been included in these results. The fastest algorithms are based on tracking AGAST, DeGraF-β and ORB features. Producing depth maps by tracking GFTT and SIFT is a slightly slower process. Finally, approaches based on dense optical flow and FAST features are by far the most computationally expensive. Of course real-time performance is directly related to the number of features being tracked, which explains why dense optical flow is the slowest since it attempts to track every pixel. Generally, the presented results should be considered only as a rough guideline since execution time of methodology depends on hardware-specific optimisation and the nature of the evaluated dataset.

4.6 Discussion & Conclusions

In this chapter, two novel approaches were presented for performing real-time 3D reconstruction using monocular vision. The first approach tracks DeGraF-β features using Bougeut's variant of the Lucas-Kanade (LK) algorithm [68] and produces a dense motion-vector map. Subsequently, this map is converted to a depth-map by comparing individual motion vectors to the ego-motion vector of the camera. The performance of this approach was compared to different 3D-reconstruction methods in order to determine their accuracy, depth-map density, noise-resistance and computational complexity. The evaluated approaches were based either on dense optical flow or dense feature tracking using a set of feature detectors including AGAST, FAST GFTT, SIFT and ORB. The motion-vector field of each methodology was evaluated using the MITEC-rendered dataset and the output of each algorithm was compared to the ground-truth data. Real-world examples were also used to demonstrate the performance of each algorithm in the presence of noise and camera vibration.

The second approach proposed the use of local frequency analysis of gradient features for estimating relative depth. This novel method is based on the fact that DeGraF gradients can accurately measure local image variance with sub-pixel accuracy. It was shown that the local frequency by which the centroid oscillates around the gradient-window centre is proportional to the depth of each gradient centroid in the real world. Of course the lower computational complexity of this methodology comes at the expense of depth-map accuracy as the camera velocity increases, however, it is at least five times faster than any other approach. Another disadvantage is that the produced depth map shows relative depth only, meaning that the real-world depth of each image point cannot be measured. Certain applications that perform distance-based prioritisation of image indexing may still find this output useful, but in this case it has been excluded from evaluation.

Other state-of-the-art 3D reconstruction approaches, such as the one proposed by Newcombe, Lovegrove and Davison [54] perform accurate 2D to 3D estimation by stereo-type triangulation, which works best in the presence of lateral camera motion. However, a vehicle moves forward in a straight line for the majority of the

time. Therefore, triangulation-based monocular approaches were not considered. Likewise, SLAM-based approaches [4, 5, 44, 46–53] were considered but proved to be more effective for ego-motion estimation while measuring the depth of specific landmarks without producing dense 3D maps.

Overall, 3D reconstruction based on DeGraF feature-tracking emerged as the most accurate feature-based approach, producing dense depth maps in real-time, while being resistant to noise and vibration. Starting with optical-flow accuracy, the DeGraF exceeded 90% accuracy followed by the dense-optical flow approach. The remaining approaches did not perform equally well, which can be attributed to the repeatability of the features on which optical flow was measured. This means that certain features were not stable between temporally-adjacent frames and as a result tracking failed. The next experiment assessed the motion-vector map density, where the proposed approach achieved higher than 99% coverage, which is only matched by the dense optical flow approach that by definition has 100% coverage. In terms of computational complexity, DeGraF produced the highest score while being 24 times faster than the runner up and the second-fastest overall behind ORB. ORB produced the least dense depth-maps with the lowest accuracy, thus its low-computational complexity could not be exploited further. Finally, the built-in feature stabilisation of the DeGraF approach meant that it performs equally well in both simulated and real environments in the presence of noise and vibration. All the other approaches would require an image-stabilisation algorithm before performing 3D reconstruction, which would further increase their computational complexity.

The DeGraF approach makes several contributions to the current state-of-the-art, however it could also be questioned as just being another dense optical flow variant. Contrary to this argument a number of reasons stand against this opinion:

- Dense optical flow [96] tries to track every pixel, regardless if that is possible or not. With the DeGraF approach, the gradient of an image region is calculated before choosing whether to track it or not. The results may look similar in the end if the scene is well textured everywhere. However, in scenarios where textureless surfaces are part of the image, then the optical flow error is significant, compared to the DeGraF approach.

- The DeGraF approach produces uniformly arranged asymmetrical features. This is a unique advantage. Most feature detectors detect numerous keypoints in highly textured areas and no keypoints in areas with little or no texture. On the other hand, dense optical flow approaches track either every pixel, or equally spaced pixels, which means that accuracy cannot be predicted *a priori*. The DeGraF approach addresses these weaknesses, by tracking image segments that are always centred around gradient features. The results clearly show that such regions are more suitable for LK tracking. At the same time by distributing the keypoints evenly across the entire image, the real-time performance is dramatically increased.

- The DeGraF approach performs feature stabilisation using a unique way of storing tracking information within the gradient matrix. By eliminating the need for a separate image-stabilisation step the computational complexity remains low.

Chapter 5

Real-time object detection by fusing visual saliency and depth information

5.1 Overview

This chapter addresses the problem of real-time object detection in automotive environments using a monocular camera. Such systems can already be found in modern vehicles for detecting pedestrians, obstacles and traffic signs. The majority of the existing approaches scan the entire image while looking for specific patterns either by using a sliding-search window or a cascade. These methodologies work well for low-resolution images, however, as the pixel count increases there is a demand for higher efficiency. Previous work has focussed on using visual cues such as optical flow for image-search optimisation, but in this case an image-stabilisation module (hardware or software) is also required for reliable operation. This chapter makes two important contributions in this direction.

A novel method is proposed for deriving highly accurate visual-saliency maps by division of Gaussians. Such maps highlight areas of interest, where potential targets such as pedestrians and vehicles, are likely to be located. Subsequently, the saliency map is fused with the DeGraF depth map leading to significantly faster object localisation. The approach is validated using the Histogram of Oriented Gradients (HOG) approach for detecting pedestrians. The results show that the visual-saliency algorithm outperforms all other approaches by a significant margin,

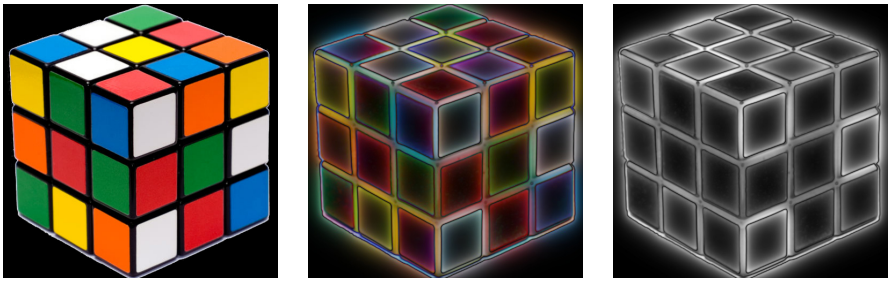


Figure 5.1: Colour and greyscale saliency maps of Rubik's cube using the *DIVoG* approach. Darker colours/shades indicate areas of low-saliency and *vice-versa*.

whereas data fusion of saliency and depth information leads to accelerated HOG performance.

5.2 Real-time visual saliency by division of Gaussians

Different saliency definitions exist, however, in this thesis a generalised version of the definition by Achanta *et al.* [10] is used: "*Visual saliency is the perceptual quality that makes a group of pixels stand out relative to its neighbours*". This section introduces a novel method for deriving visual saliency maps in real-time without compromising the quality of the output. This is achieved by replacing the computationally-expensive centre-surround filters with a simpler mathematical model named Division of Gaussians (*DIVoG*). The results are compared to five other approaches, demonstrating at least six times faster execution than the current state-of-the-art whilst maintaining high detection accuracy. Given the multitude of computer-vision applications that make use of visual-saliency algorithms such a reduction in computational complexity is essential for improving their real-time performance.

5.2.1 Methodology

The Division of Gaussians approach comprises of three distinct steps: 1) Bottom-up construction of Gaussian pyramid, 2) Top-down construction of Gaussian pyramid based on the output of *Step 1*, 3) Element-by element division of the input image with the output of *Step 2*.

Figure 5.2: Saliency map of a pedestrian using *DIVoG*.

Step 1: The Gaussian pyramid U comprises of n levels, starting with an image U_1 as the base with resolution $w \times h$. Higher pyramid levels are derived via down-sampling using a 5×5 Gaussian filter. The top pyramid level has a resolution of $(w/2^{n-1}) \times (h/2^{n-1})$. Let us call this image U_n.

Step 2: U_n is used as the top level D_n of a second Gaussian pyramid D in order to derive its base D_1. In this case, lower pyramid levels are derived via up-sampling using a 5×5 Gaussian filter.

Step 3: Element-by-element division of $U_1$and D_1 is performed in order to derive the minimum ratio matrix M (also called MiR matrix) of their corresponding values as described by the following equation:

$$M_{i,j} = min\left(\frac{D_{1_{i,j}}}{U_{1_{i,j}}}, \frac{U_{1_{i,j}}}{D_{1_{i,j}}}\right) \tag{5.1}$$

The saliency map S is then given by *Equation 5.2*, which means that saliency is expressed as a floating-point number in the range $0 - 1$.

$$S_{i,j} = 1 - M_{i,j} \tag{5.2}$$

The described approach can be further expanded to include element-by-element division of all corresponding levels of pyramids U and D. In this case, the MiR matrix is initialised as a unit matrix (i.e. for each matrix element $M_{0_{i,j}} = 1$). Subsequently, each pair of pyramid levels U_n and D_n is scaled up to the input's resolution. Next, the MiR matrix M_n is multiplied by M_{n-1} as described by the

DIVoG equation below, which is a generalised form of *Equation 5.1.*

$$M_{n_{i,j}} = min\left(\frac{D_{n_{i,j}}}{U_{1_{i,j}}}, \frac{U_{1_{i,j}}}{D_{n_{i,j}}}\right) M_{n-1_{i,j}} \tag{5.3}$$

for $n \geqslant 1$. The saliency map is then derived using *Equation 5.2.* Deriving the MiR matrix through processing of all pyramid levels produces more accurate saliency maps than *Equation 5.1*, but also increases the computational complexity of the algorithm. In practice, the difference between the two approaches is visually minimal, thus in this thesis all MiR matrices have been calculated using *Equation 5.1.* Practically, the choice of n value depends on the size of the salient objects that need to be detected. For example, $n = 2$ may be adequate calculating the saliency of an area smaller than 5×5 pixels. Alternatively, calculating the saliency of larger areas will require incrementally higher n value. On the other hand, calculating the saliency of large areas with a small n value will result in detecting the salient edges of this area, however, the centre of the area will appear as non-salient. The reason is that there are not enough pyramid levels to separate the foreground from the background. Finally, a major advantage of this approach is that it is colourspace-independent, thus it can derive saliency maps even from greyscale images, which significantly reduces computational cost.

Implementation notes:

a) All operations are performed using 32-bit floating point matrices.

b) To avoid division by zero, or division with floating point numbers in the range 0 to 1, we define the minimum pixel value equal to k^n, where k is the size of the Gaussian kernel. This ensures that pyramidal downsampling will always result in a value greater than 1.

c) For colour images, the algorithm can be used with any colourspace. Each channel is processed separately to produce a salience map.

d) All the saliency maps in this thesis have been produced using 24-bit colour images in the RGB colourspace. The Gaussian pyramid is constructed with $n = 5$.

e) All saliency maps in *Figure 5.1, 5.2, 5.3*, have been normalised to fit the $0 - 255$ pixel range for effective visualisation.

5.3 Real-time object detection using saliency & depth information

This section proposes a basic yet powerful methodology for fusing saliency and depth information into a single matrix which represents the likelihood of each image pixel belonging to an object. Subsequently, the HOG algorithm [61] is evaluated on the high-likelihood areas in order to confirm the existence of an object. As a result, the process of object detection in significantly accelerated.

5.3.1 Methodology

Firstly, the DIVoG saliency map is derived using the aforementioned technique. This map assigns a value to each image pixel ranging from 0.0 to 1.0, with 1.0 representing the highest saliency value. Likewise, the DeGraF depth map is extracted by feature tracking as described in the last chapter. This map assigns a value to each image pixel which represents the depth (distance) between the camera and the real-world coordinate of the point. In an automotive scenario, the areas of interest refer to objects within a certain range from the vehicle that could potentially cause an accident. In practice, this translates to relatively low depth value and high saliency value. As a result the most basic fusion could be performed by element-wise multiplication of the saliency and depth matrices. *Figures 5.13, 5.14* and *5.15* illustrate the saliency, depth and fused output, which uses false colour to highlight areas of interest. The data fusion equation is:

$$F_{i,j} = S_{i,j} * D_{i,j} \tag{5.4}$$

where element-wise multiplication of saliency matrix S is performed with depth map D to derive the fused map F. This map forms the basis for extracting a list of regions of interest within the image. Currently, this is achieved by running the sliding window HOG algorithm [61] only in windows with high accumulative likelihood. However, the fused map can also be used to prioritise the candidate

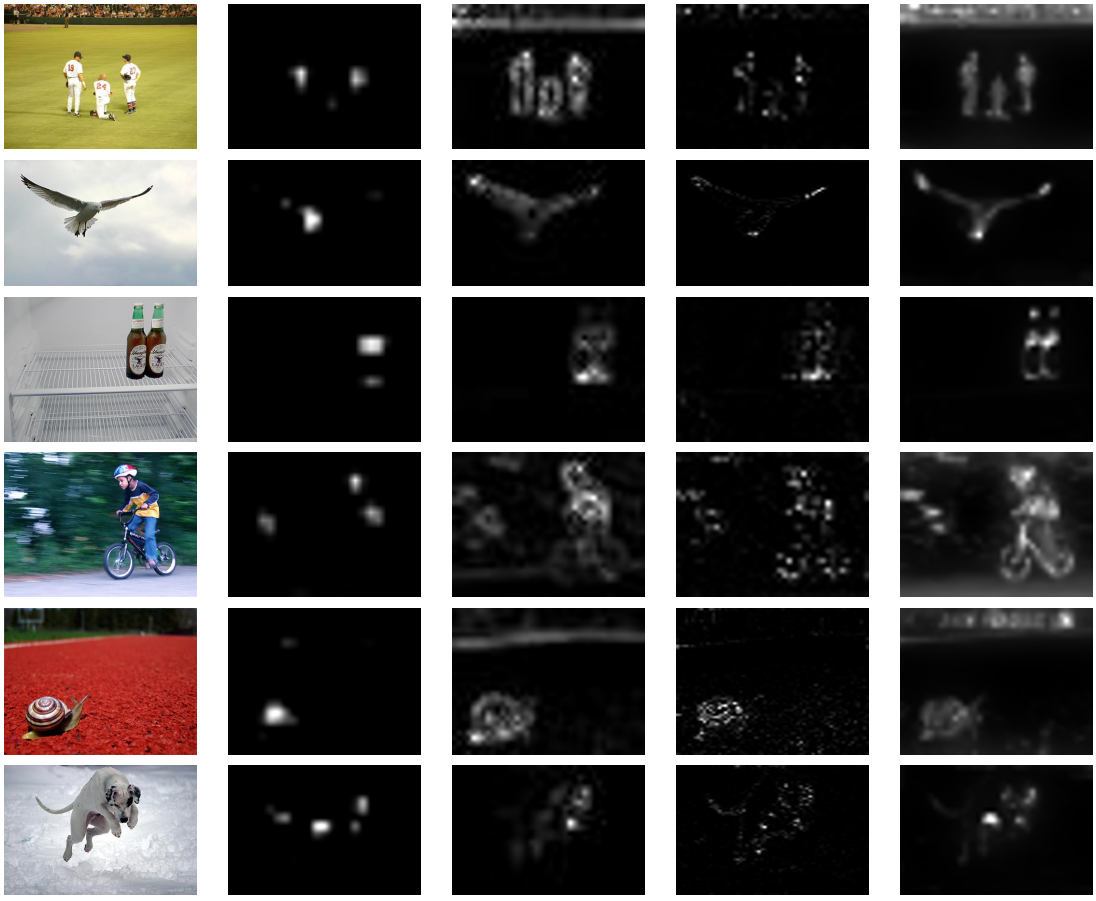


Figure 5.3: A set of saliency maps generated using different approaches (based on work by Achanta *et al.* [10]).

windows by likelihood. This is useful in the case of real-time detection algorithms that must return a response within a given time limit before advancing to the next frame.

5.4 Results

5.4.1 Visual saliency results

The *DIVoG* approach is compared with five other saliency algorithms using an evaluation framework created by Achanta *et al [10, 64].* As part of this procedure, saliency maps are extracted for 1000 images using five different approaches [10, 69–72], as illustrated in *Figure 5.3, 5.4.* Bright shades indicate high saliency values. For example, in images with a single object and simple background the object surface

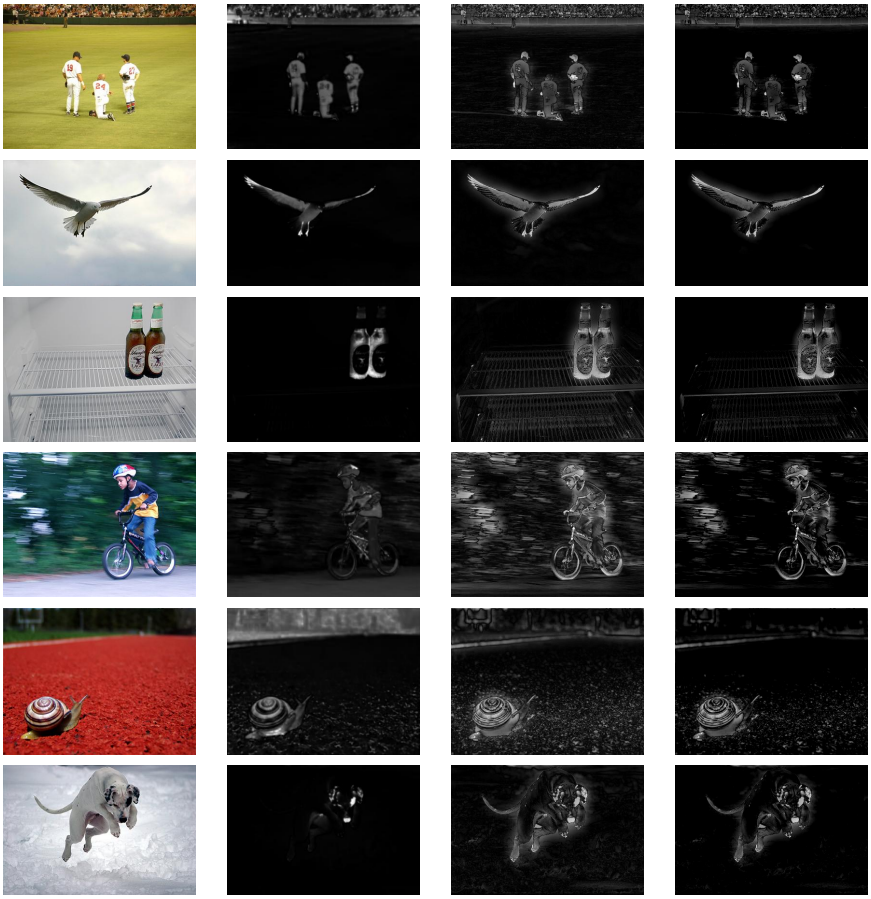


Figure 5.4: DIVoG-F enhances these results of the standard DIVoG algorithm by adding a low-pass filter to reduce background noise.

appears as the most salient part of the image. However, this is also dependent on the contrast-difference between the foreground and the background. For example, in the first image there are three players in a green field. The players appear as salient objects whereas the green field as a non-salient area. Most approaches separate the players relatively clearly. On the other hand, the last image shows a white dog against a white background. In this case, most approaches only identify the black nose and ears of the dog as salient but completely miss the body. Of course, there are also examples with complex foreground and background, where saliency measurement is more problematic (e.g. kid cycling in front of trees). By visual inspection of the processed dataset, the DIVoG approach gives the clearer separation between foreground and background. Previous approaches [10,62–64] have used the saliency maps to segment the images and compare the extracted segments to the

ground-truth in order to derive the algorithm's accuracy. This is a reasonable approach for simple scenes with a small number of distinct objects. However, for more complex images the specification of ground-truth is more subjective. In addition, the performance of the chosen segmentation approach directly affects the performance of the saliency detector, which makes the evaluation of saliency algorithms problematic. Since the main contribution of this section is related to the real-time performance of the algorithm, we compare the execution time of our approach with Achanta *et al. [10]*, which is one of the most efficient saliency methodologies for producing high-resolution maps.

For performance evaluation a mobile 2.4GHz Intel Core 2 Duo processor was used with 4GB RAM. *Figure 5.5* and *Table 5.1* show a comparison in execution time between *DIVoG* and *[10]* at different resolutions using colour and greyscale images. Furthermore, *Figures 5.3* and *5.4* show some examples of saliency maps generated using DIVoG and five other approaches.

The original implementation by Achanta *et al [10]* (*AC09*), produces much sharper saliency maps than *IT98 [69], MA03 [70], HO07 [71]* and *HA07 [72].* In terms of computational performance *AC09 [10]* is at least comparable to the aforementioned approaches as presented in *[10].* On the other hand, the *DIVoG* approach demonstrates similar or higher quality saliency maps to *AC09 [10]*, but at a fraction of the time. *DIVoG* is faster than *AC09 [10]* by a factor of 6 when processing 24-bit colour images and by a factor of 16 when processing greyscale images. This massive gap could not be justified by the theoretical difference in computational complexity, thus the *AC09 [10]* was re-implemented using the *OpenCV* library [11] (*AC-OPENCV*). This way the execution time reduced by a factor of 3. Even so, *AC-OPENCV* remained 56% slower than *DIVoG*. An indication of performance can also be given by quoting the achieved framerate. At the lowest resolution of 320×240, *DIVoG* executed at 333 fps on greyscale images and 111 fps on colour images, showing a linear relationship between data size and execution time. Overall, the *DIVoG* approach has demonstrated an ability to calculate full resolution saliency maps with the minimum computational cost.

	***AC09** [10]*		***AC-OPENCV***	
Resolution	*Time (s)*	*fps*	*Time (s)*	*fps*
320×240	0.078	12.8	0.015	66.6
512×512	0.187	5.3	0.052	19.2
640×480	0.218	4.6	0.057	17.5
1024×1024	0.718	1.4	0.200	5.0
2048×2048	2.699	0.4	0.803	1.6
	DIVoG-3CH		***DIVoG- 1CH***	
320×240	0.009	**111**	0.003	**333**
512×512	0.032	**31.2**	0.009	**111**
640×480	0.036	**27.7**	0.012	**83.3**
1024×1024	0.115	**8.7**	0.041	**24.3**
2048×2048	0.456	**2.2**	0.161	**6.2**

Table 5.1: Performance evaluation data showing execution time and framerate. *AC09* is the original implementation by Achanta *et al.* [10].

5.4.2 Object detection results

The proposed object detection approach is evaluated on the ETH pedestrian detection dataset [12]. The performance of HOG-based pedestrian detection [61] is initially measured on the raw images using the standard sliding window approach. Subsequently, the saliency and depth maps are used to select a subset of regions of interest. The HOG algorithm is then evaluated only on these regions. Ideally, the output of both approaches should be exactly the same with the proposed approach being significantly faster.

The evaluation methodology is on deriving the number of true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN) for both the original HOG sliding window approach and the proposed approach. Subsequently, ROC curves are used to measure the overall performance. These curves highlight the relationship between the True Positive Rate (TPR) and the False Positive Rate (FPR) as given below:

$$TPR = \frac{TP}{TP + FN} \tag{5.5}$$

$$FPR = \frac{FP}{FP + TN} \tag{5.6}$$

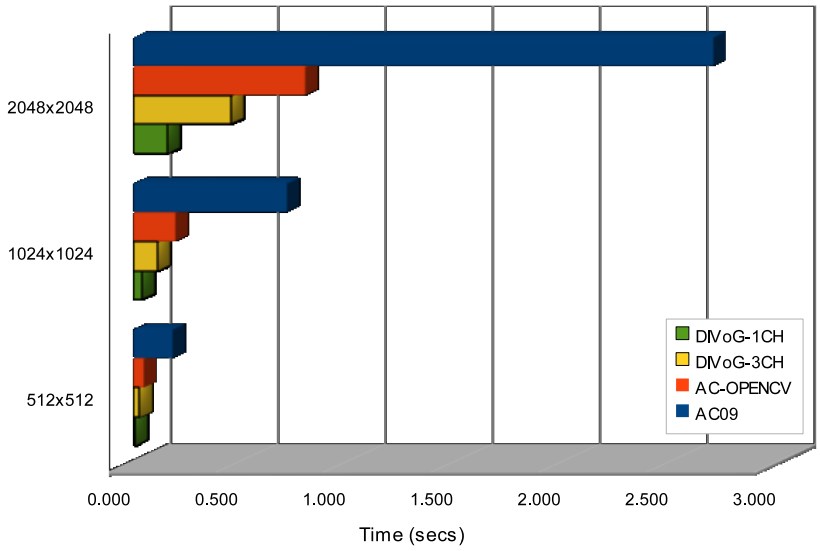


Figure 5.5: Performance evaluation of *DIVoG* and "Frequency-tuned Salient Region Detection" by Achanta *et al.* [10] (*AC09*). *AC-OPENCV* is our *AC09* real-time implementation using the *OpenCV* library [11]. *DIVoG-3CH* denotes the *DIVoG* algorithm running on 3 channel input (i.e. RGB image), whereas *DIVoG-1CH* denotes the *DIVoG* algorithm running on a single channel input (i.e. greyscale 8-bit image).

Figure 5.6, shows the ROC curves for both the sliding window and the salient-region approaches. The sliding window indexing is slightly more accurate mainly due to detecting pedestrians at a longer range than our approach. However, it is important to note that these results are derived from the ETH Bahnhof dataset [12], which is captured at 15 frames per second rate. Such a low rate reduces the accuracy of the feature tracker, which in turn leads to lower than expected performance. Nevertheless, the two ROC curves are only separated by a small margin. Finally, is worth noting that the proposed approach executed five times faster than the sliding window algorithm. Practically, this means that there were five times fewer windows that the HOG algorithm had to process. The conclusion is that by selecting salient regions of interest the HOG algorithm can be accelerated without a significant impact on accuracy.

Below there is a comprehensive set of examples illustrating:

- The output of the DIVoG saliency algorithm (*Figures 5.7, 5.8, 5.9*)

  In these images, saliency is denoted by a grayscale shade with black denoting zero saliency and white denoting the maximum saliency. For example, most

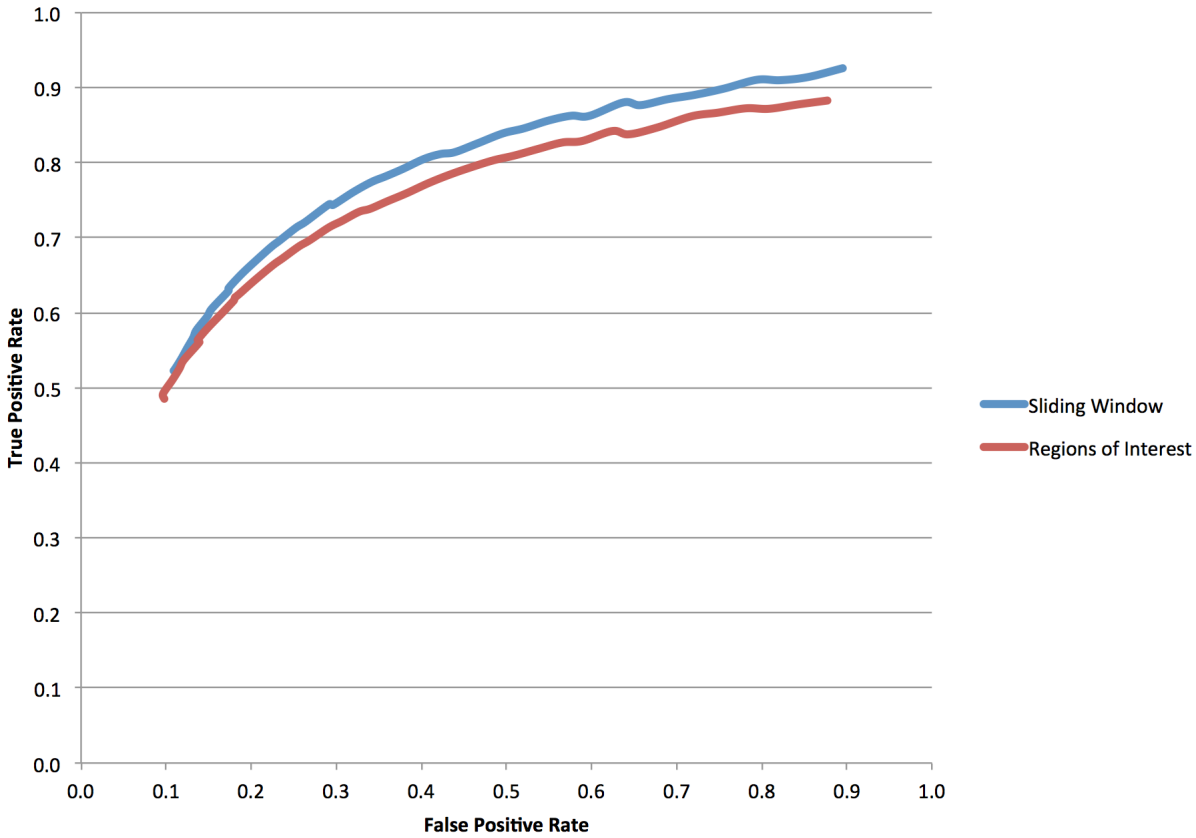


Figure 5.6: ROC curve denoting the relation between true positive rate and false positive rate.

pedestrians in the distance appear much more salient than the pavement, where saliency is zero. On the other hand, objects that are very close to the camera often have large non-salient regions. This is expected since the DIVoG algorithm performs multi-scale element-wise saliency calculation, thus it is limited by the number of pyramid levels. In this case six pyramid levels have been used to produce each saliency map. In addition, noise filtering and normalisation have been applied on the raw output in order to better highlight the salient objects.

- The output of the DeGraF-depth-estimation algorithm (*Figures 5.10, 5.11, 5.12*)

  In these images, false colour has been used to denote depth. The sequence from white to red, yellow, green, blue and black denotes the distance from the camera (close to far). Visually, the output does not look as sharp as in the previous chapter, since the Bahnhof sequence from the ETH dataset [12] is only captured at 15 frames per second. This means that the tracking algorithm needs a larger temporal buffer for performing accurate stabilisation. However, this optimisation leads to blurred depth maps. Still given the poor quality, low framerate and extreme camera vibration the results are at the expected level.

- The fused output from depth and saliency information (*Figures 5.13, 5.14, 5.15*)

  In these images, false colour has been used to denote the likelihood of the corresponding pixel belonging to an object. The sequence from white to red, yellow, green, blue and black denotes the order in which regions should be scanned for objects. Again the output looks slightly noisy, which is due to the poor quality, low framerate and extreme camera vibration of the ETH Bahnhof dataset [12]. This means that although the saliency map is very clean when it is fused with a noisy depth map it generates a noisy fused map. Still it is clearly evident that the algorithm is choosing the right areas as regions of interest. For example, all of the pedestrians appear with brighter shades than

the background, thus highlighting them as areas of interest.

- The HOG algorithm output based only on salient regions of interest (*Figures 5.16, 5.17, 5.18*)

  In these images the output of the HOG algorithm is denoted by drawing rectangles around the detected pedestrians. The HOG algorithm uses the default parameters as proposed by Dalal *et al.* [61]. The ROC curve in *Figure 5.6* confirms that the pedestrian detection accuracy is comparable to the original HOG approach, albeit significantly faster.

5.5 Discussion & Conclusions

In this chapter, a novel visual saliency algorithm was presented for calculating full resolution saliency maps in real-time by using Division of Gaussians. Compared to recent work by Achanta *et al.* [10], *DIVoG* showed a significant increase in performance by a factor of 6 when using colour images. A real-time implementation of Achanta's work was also performed using the OpenCV library [11], which is more than three times faster than the original implementation, but still 56% slower than the *DIVoG* approach. Given that for VGA resolution the achieved framerate exceeds 80 fps on greyscale images, this algorithm could significantly improve the performance of a wide range of applications including salient feature detection, object extraction and classification.

In addition, the DeGraF depth estimation approach was used to improve the real-time performance of object detection methodologies and in particular the HOG pedestrian detection algorithms. Fusing information from a DIVoG saliency map and a DeGraF depth map, gives a clear indication of the regions of interests where objects are likely to exist. The results show comparable accuracy between the original HOG implementation and our accelerated variant that improves real-time performance by at least five times.

Overall, this chapter offers major contributions to knowledge by proposing a faster- visual saliency algorithm and a novel methodology for fusing *DIVoG* saliency and depth information for more efficient object detection.

Figure 5.7: DIVoG saliency output based on the ETH pedestrian detection dataset [12].

Figure 5.8: DIVoG saliency output based on the ETH pedestrian detection dataset [12].

Figure 5.9: DIVoG saliency output based on the ETH pedestrian detection dataset [12].

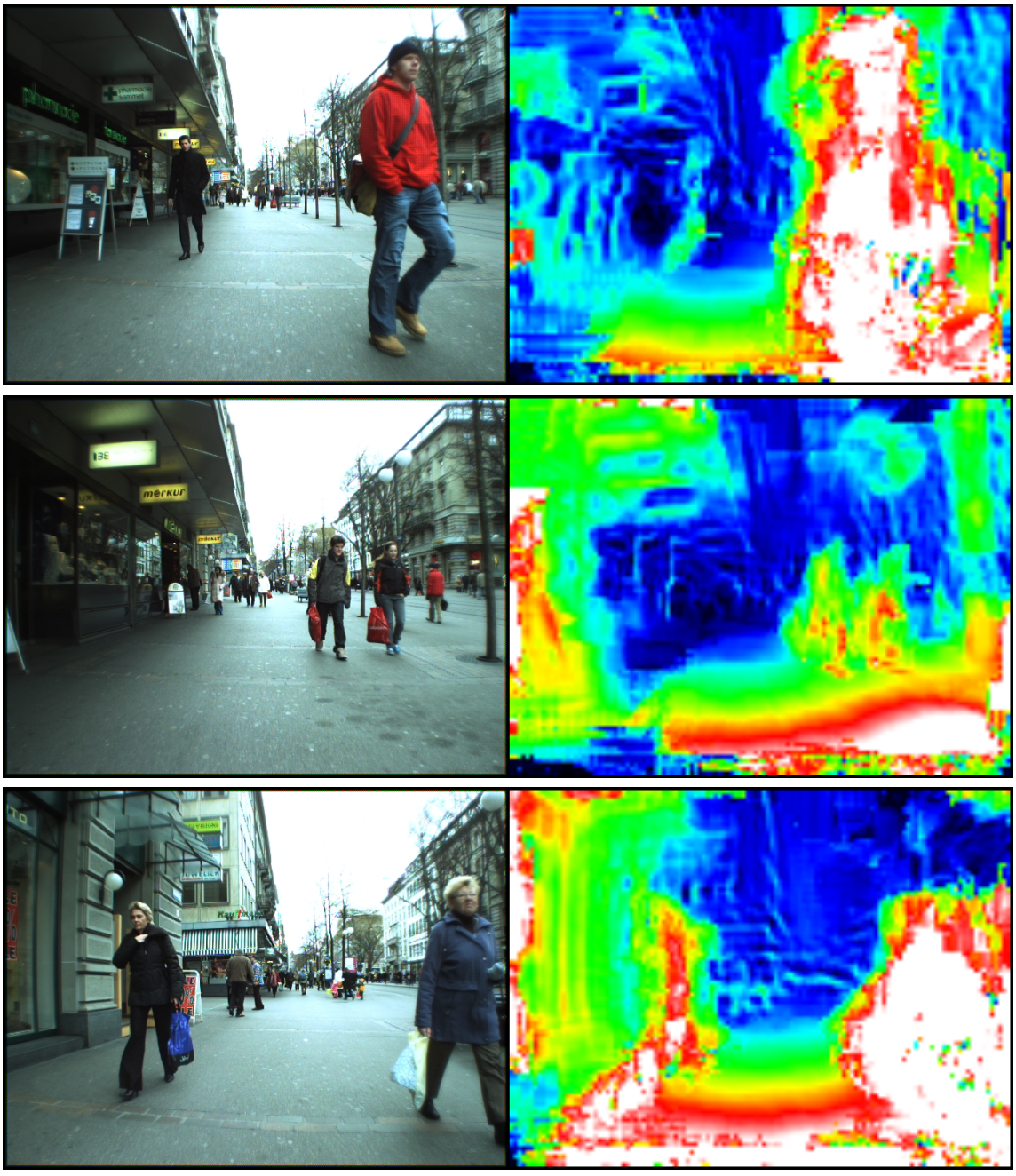


Figure 5.10: DeGraF depth map based on the ETH pedestrian detection dataset [12].

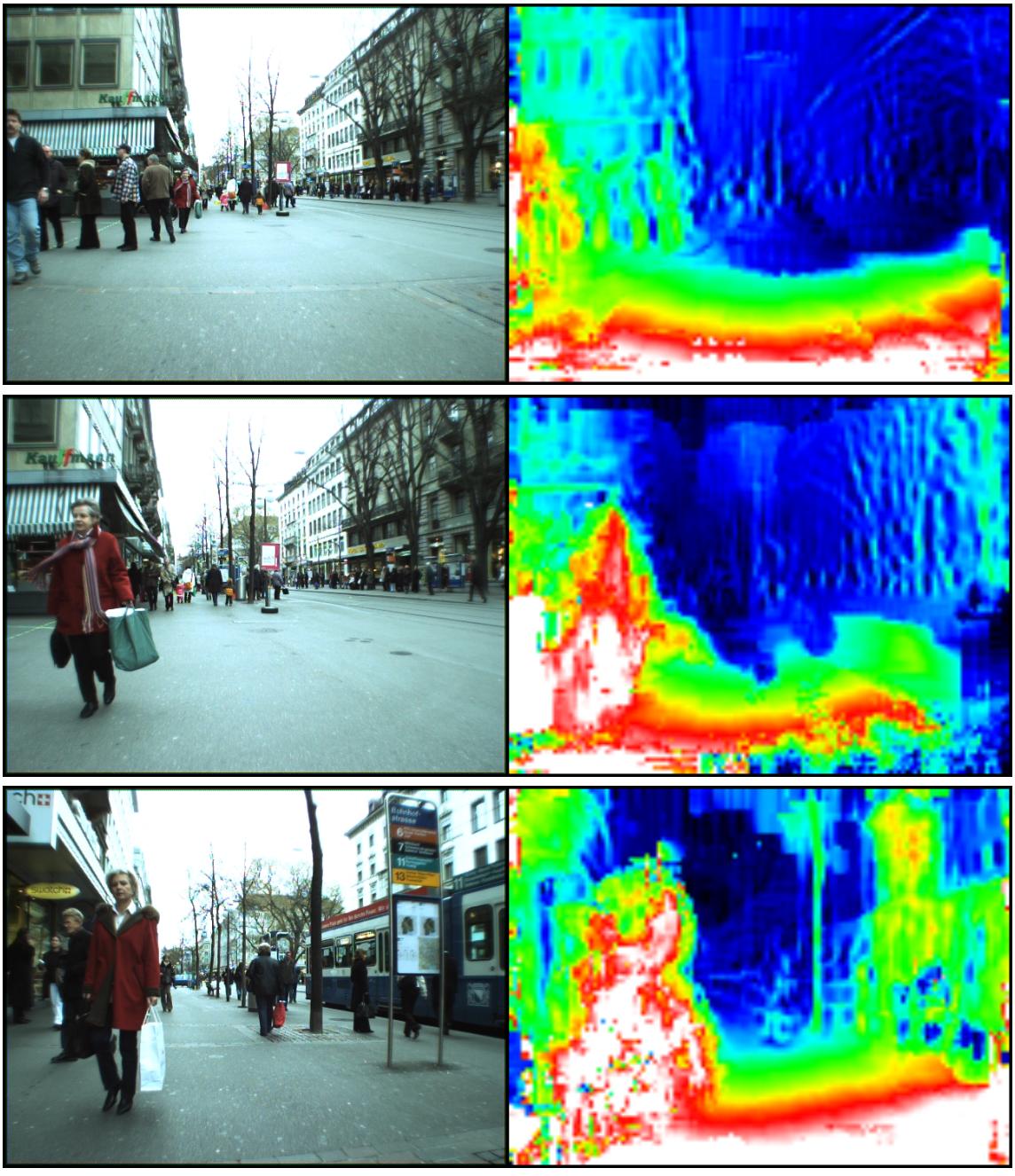


Figure 5.11: DeGraF depth map based on the ETH pedestrian detection dataset [12].

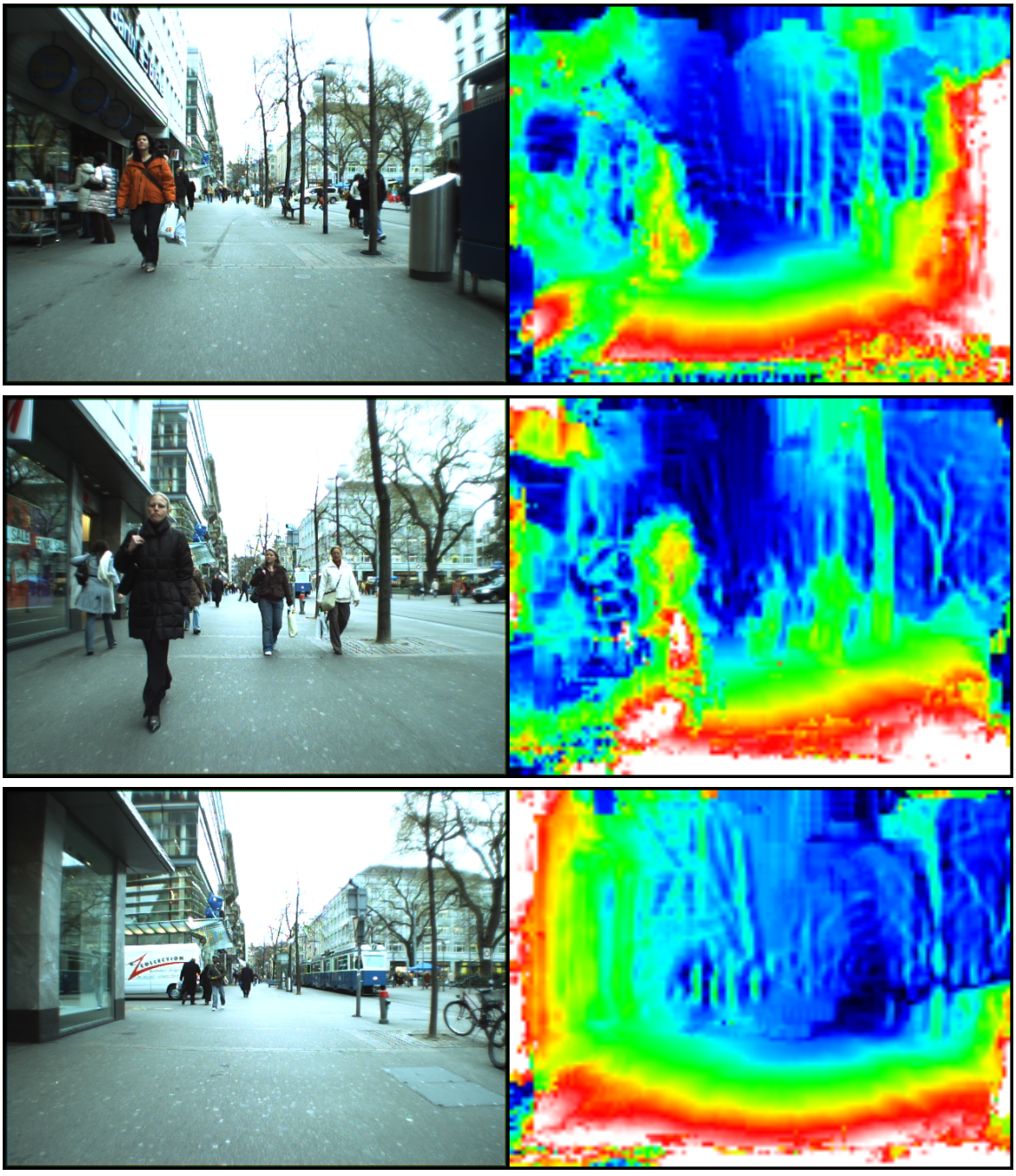


Figure 5.12: DeGraF depth map based on the ETH pedestrian detection dataset [12].

Figure 5.13: Fusion of DIVoG saliency and DeGraF depth information in order to accelerate pedestrian detection. (ETH Dataset [12])

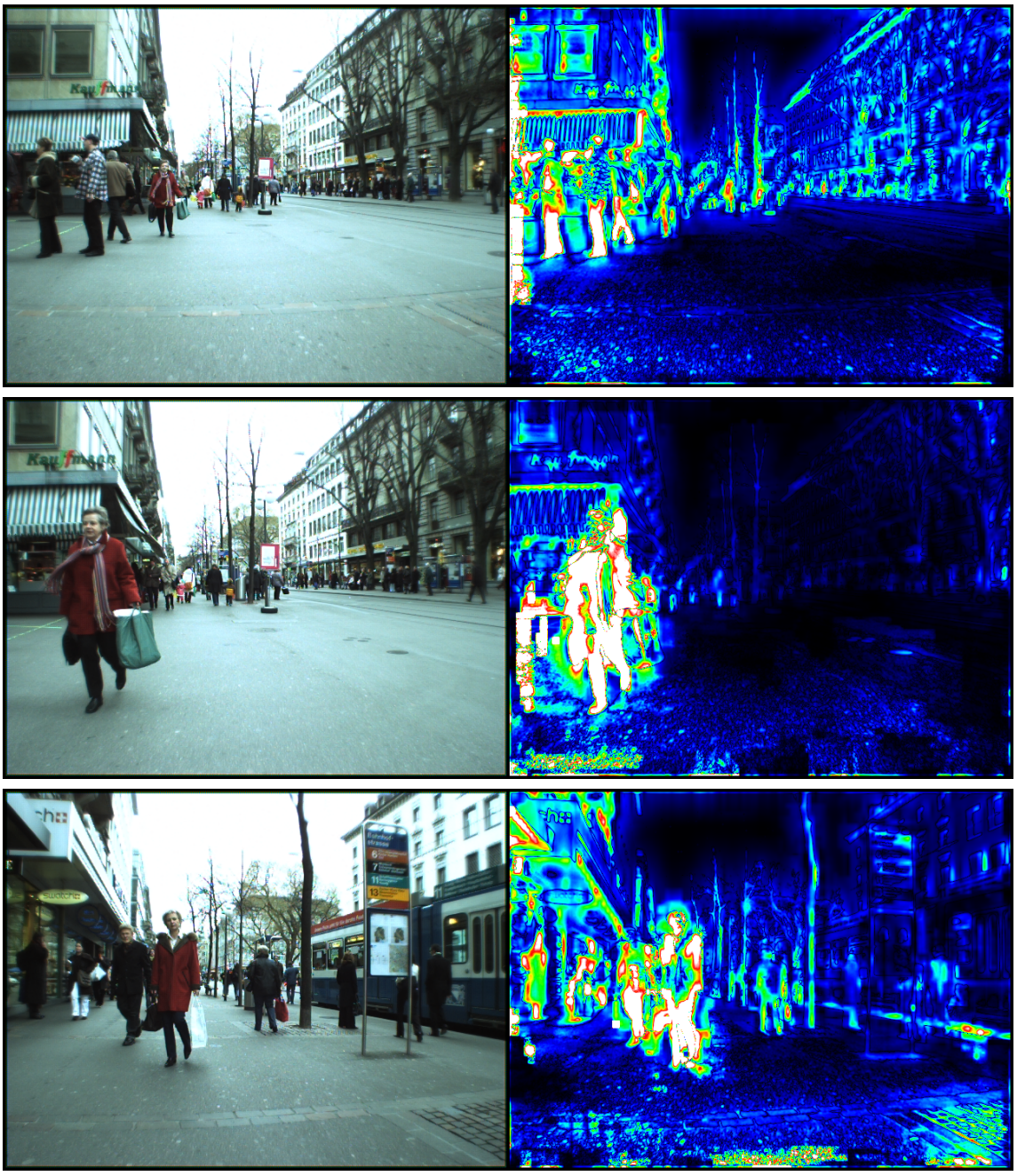


Figure 5.14: Fusion of DIVoG saliency and DeGraF depth information in order to accelerate pedestrian detection. (ETH Dataset [12])

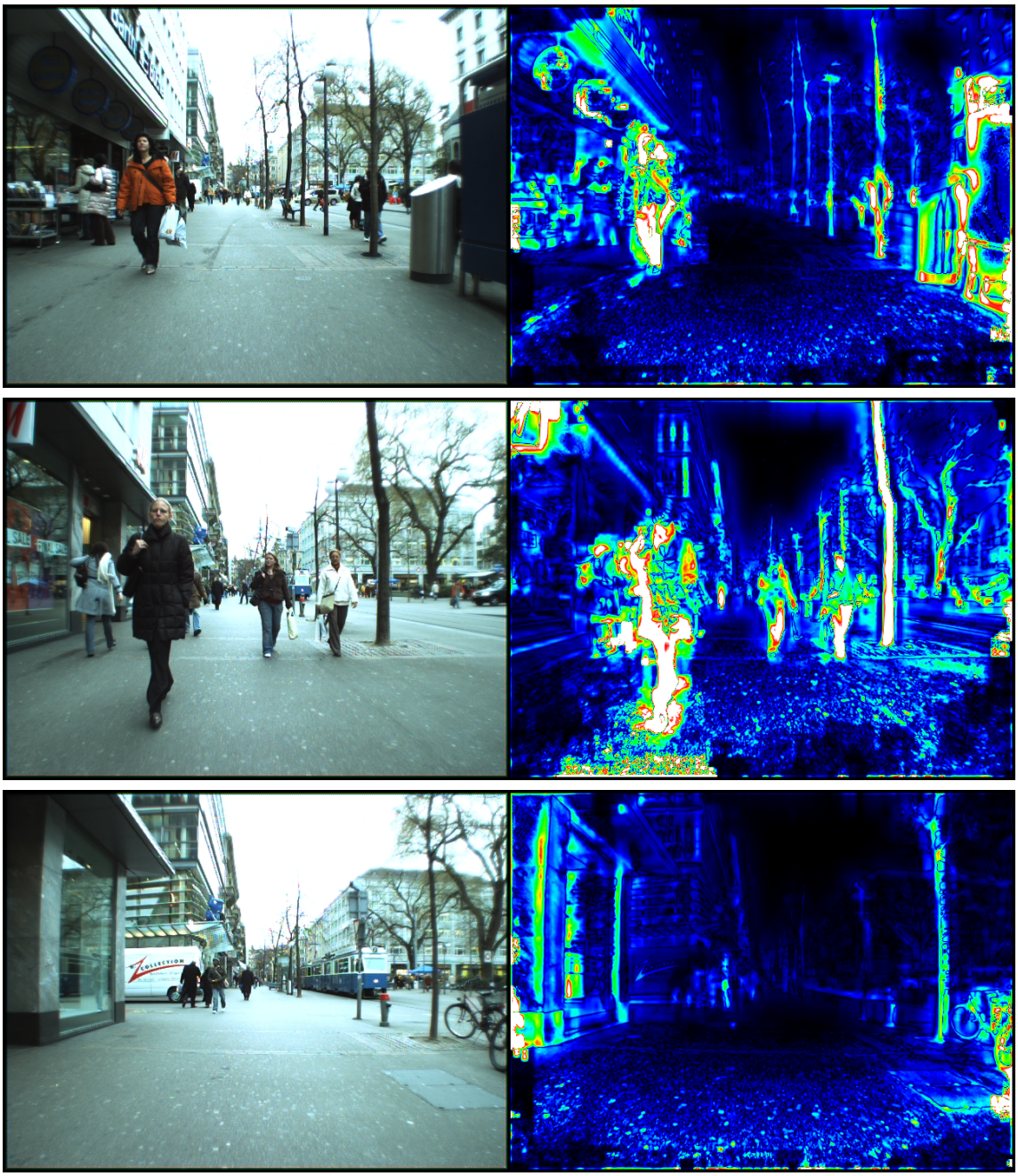


Figure 5.15: Fusion of DIVoG saliency and DeGraF depth information in order to accelerate pedestrian detection. (ETH Dataset [12])

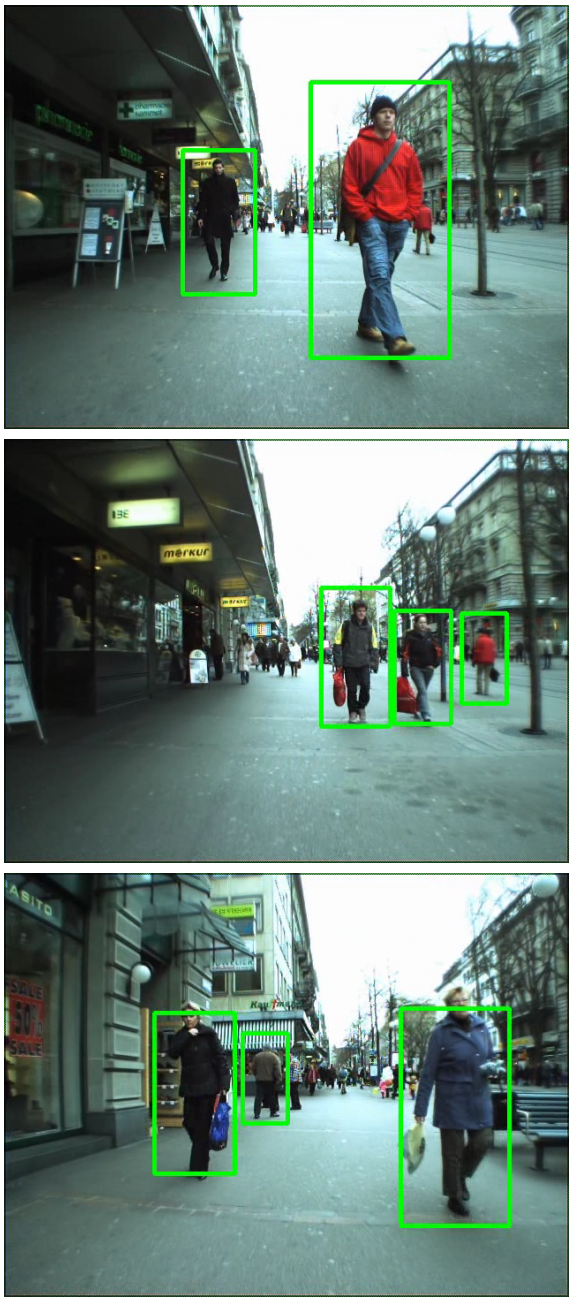


Figure 5.16: Pedestrian detection using visual saliency and monocular depth estimation. (ETH Dataset [12])

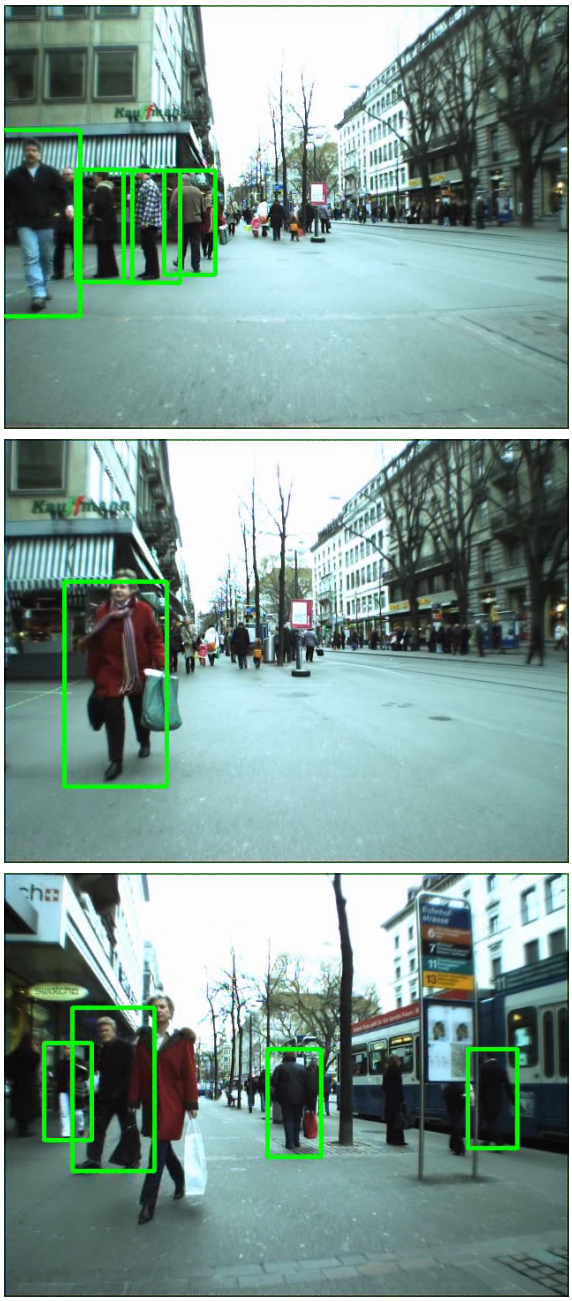


Figure 5.17: Pedestrian detection using visual saliency and monocular depth estimation. (ETH Dataset [12])

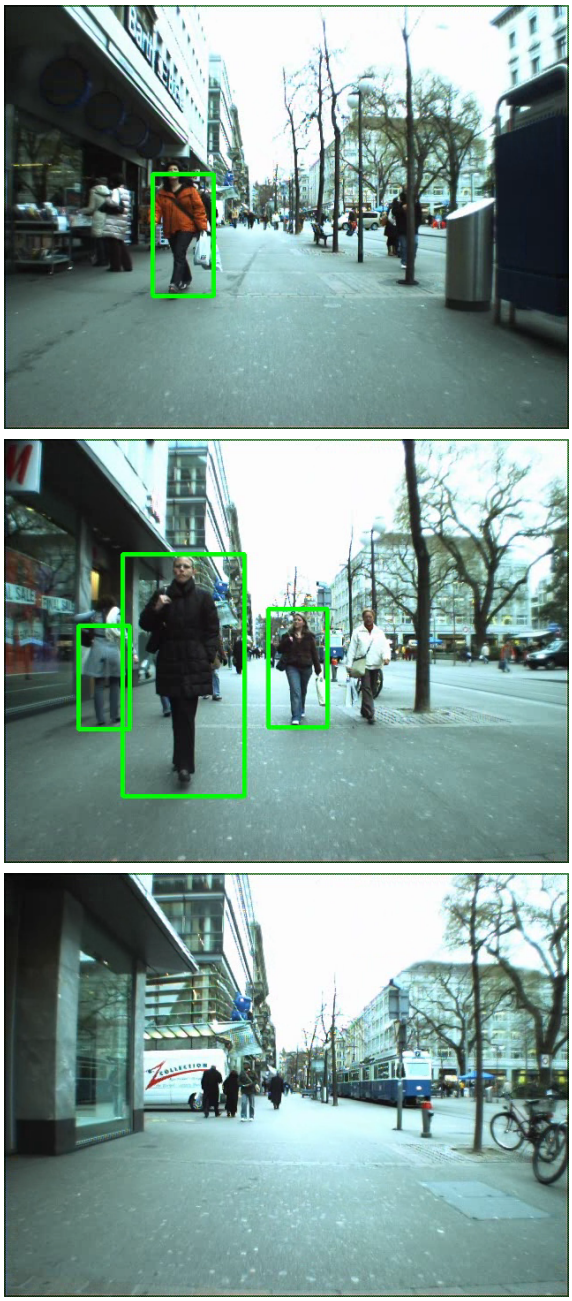


Figure 5.18: Pedestrian detection using visual saliency and monocular depth estimation. (ETH Dataset [12])

Chapter 6

Conclusions & Future Work

6.1 Concluding remarks

This thesis addresses the problem of real-time object detection in automotive environments using monocular vision. Through a step-wise approach computationally-expensive algorithms have been optimised or replaced by novel methodologies to produce a fast object detection system that is aligned to the requirements of the automotive domain.

In Chapter 2, the state-of-the-art was reviewed and the main technological limitations were identified. Each of these limitations was addressed in a separate chapter focussing on real-time feature detection and tracking, depth estimation using monocular vision and finally, object detection by fusing visual saliency and depth information. The common theme across all of these chapters is the design and implementation of real-time algorithms that facilitate object detection.

Chapter 3 proposed a novel approach for real-time dense feature extraction based on gradient maps. This method, DeGraF, uses intensity weighted centroids as proposed by the ORB detector [40] but implemented using a new mathematical model with significantly lower noise sensitivity. In this context, the gradient vector connects a negative to a positive centroid, where both centroids are symmetric about the centre of the area for which the gradient is calculated. Each pair of positive and negative centroids forms a stable local feature that describes the underlying area. These local features are called DeGraF-β. Combining neighbouring DeGraF-β features al-

lows the detection of more distinctive features, called DeGraF-α. These are defined as centre-surround features, similar to CenSurE [38], where the central gradient vector is either a local maxima or minima. Gradient-derived features (DeGraF-α and DeGraF-β) are compared to GFTT (Shi Tomasi) [28], CenSurE [38], AGAST [39], SIFT [1], SURF [2], FAST [32], and ORB [40] and MSER [35] by analysing a diverse range of quality criteria [37] such as keypoint density, tracking accuracy, illumination invariance, rotation invariance, resistance to noise and execution time. DeGraF performance proved superior in the majority of the tests, making it suitable for 3D reconstruction algorithms based on feature tracking.

More specifically, the first evaluation methodology focussed on keypoint density, which is essential for dense 3D reconstruction. DeGraF-β demonstrated the highest performance followed by DeGraF-α, AGAST [39], SIFT [1], FAST [32], GFTT [28] and ORB [40]. On the other hand CenSurE [38], SURF [2] and MSER [35] had the lowest density values.

For measuring tracking accuracy, a car vibration simulation model was chosen. This way, the tracking error of each detected keypoint was measured for 1, 2, 4, 16 and 32-pixel vibration amplitude. As expected, most feature detectors start with low error rates at low amplitudes that gradually increase. In this case, DeGraF-α and DeGraF-β performed only slightly better than the other methodologies.

Repeatability of features under variable illumination conditions was evaluated by gradually increasing the brightness of images. In this case, DeGraF-α and DeGraF-β outperformed all other approaches by a significant margin, with ORB [40] being the only close competitor. MSER [35] was by far the approach with the largest error margin, whereas the remaining approaches demonstrated similar behaviour. Subsequently, the main conclusion from this test is that methodologies based on intensity-weighted centroids (DeGraF-α, DeGraF-β and ORB [40]) perform reliably under variant illumination conditions.

Repeatability of features in rotated images is another important aspect of the evaluation process since it guarantees the stability of feature detection algorithms when the camera rotates around the axis of motion (rolling effect). In a car, this effect can be described as uneven vibration of the front suspension causing the

image to rotate by ±3 degrees. This behaviour was simulated by artificially rotating the images around their centre. Subsequently features were extracted from each rotated image before being back-projected onto the original image. The error caused by image rotation was then measured, showing that each feature detector exhibits different behaviour. The total error margin was in the range of 38% to 68% with DeGraF-β and ORB [40] giving better results albeit with significant error. GFTT [28] and SIFT [1] also demonstrated reasonable performance, whereas all the other approaches failed to achieve high repeatability in this test. Specifically, for DeGraF-α and SURF [2] this can be explained by the fact that the features are extracted from rectangular areas that are dependent on rotation by definition.

The resistance of feature detectors to noise was also evaluated by incrementally adding Gaussian noise to the dataset. The results in this case were remarkable. DeGraF-β outperformed all other approaches by a large margin of 28% for low-noise images and 62% for high-noise images. ORB had the second lowest error rating with also a significant margin over DeGraF-a that came third. All other approaches demonstrated high sensitivity to noise. The conclusion from this test is similar to the illumination test in that intensity-weighted centroids lead to more reliable features being detected in poor quality images.

Finally, the real-time performance of all approaches was evaluated with most approaches demonstrating low execution times with the exception of SIFT [1] and MSER [35] that were significantly slower. The fastest detectors were FAST [32], DeGraF-β and CenSurE followed by AGAST [39], DeGraF-a, GFTT [28], SURF [2] and ORB [40].

Overall, some interesting conclusions can be drawn by comparing a wide range of feature detectors. Firstly, assessing feature detection methodologies in an automotive context highlights some different challenges which would not be obvious using generic non-automotive datasets. For the first time, the effect of increasing keypoint density is evaluated based on a wide range of quality criteria such as tracking accuracy, illumination, rotation and noise variance. The novel DeGraF-β approach proves competitive in most tests with exceptional performance in the noise and illumination tests. On the other hand, DeGraF-a demonstrates similar performance to

well established feature detectors, but still needs further work on optimising certain aspects of the proposed methodology (e.g. rotation variance). However, since 3D reconstruction is likely to be based on local features rather than global features, DeGraF-β is a good starting point for producing dense 3D point clouds since it demonstrated the highest keypoint density of any detector, highest tracking accuracy, second highest repeatability at the rotation test, highest repeatability score at the illumination and noise tests by a significant margin and finally the second fastest execution time.

In chapter 4, two novel approaches have been presented for performing real-time 3D reconstruction using monocular vision. The first method tracks DeGraF-β features using Bougeut's variant of the Lucas Kanade (LK) algorithm [68] and produces a dense motion-vector map. Subsequently, this map is converted to a depth-map by comparing individual motion vectors to the ego-motion vector of the camera. The performance of this approach has been compared to different 3D reconstruction methods in order to determine their accuracy, depth-map density, noise-resistance and computational complexity. The evaluated approaches were based either on dense optical flow or dense feature tracking using a set of feature detectors including AGAST [39], FAST [32], GFTT [28], SIFT [1] and ORB [40]. A motion-vector field was produced using each of these methodologies, which was then evaluated using a 3D rendered dataset. In this case, the output of each algorithm was compared to the ground truth data. Real-world examples were also used to demonstrate the performance of each algorithm in the presence of noise and camera vibration. Overall, 3D reconstruction based on DeGraF feature-tracking emerged as the most accurate feature-based approach, producing dense depth maps in real-time, while being resistant to noise and vibration. Starting with depth-map density, the proposed approach achieved higher than 99% coverage, which is only matched by the dense optical flow approach, which by definition has 100% coverage. In terms of accuracy, DeGraF produced the highest score while being 24 times faster than the runner up and the second-fastest overall behind ORB [40] . ORB [40] produced the least dense depth-maps with the lowest accuracy, thus its low-computational complexity could not be exploited further. Finally, the built-in stabilisation feature of the DeGraF

approach meant that it performs equally well in both simulated and real environments in the presence of noise and vibration. All the other approaches would require a separate image stabilisation algorithm before performing 3D reconstruction, which would further increase their computational complexity.

The second approach proposed in Chapter 4, performs local frequency analysis of gradient features for estimating relative depth. This novel method is based on the fact that DeGraF gradients can accurately measure local image variance with sub-pixel accuracy. It was shown that the local frequency by which the centroid oscillates around the gradient window centre is proportional to the depth of each gradient centroid in the real world. Of course the lower computational complexity of this methodology comes at the expense of depth map accuracy as the camera velocity increases, however, it is at least five times faster than any other approach.

In chapter 5, a novel visual saliency algorithm was presented for calculating full resolution saliency maps in real-time by using division of Gaussians. The method comprises three distinct steps: 1) Bottom-up construction of Gaussian pyramid, 2) Top-down construction of Gaussian pyramid based on the output of *Step 1*, 3) Element-by element division of the input image with the output of *Step 2*. Compared to recent work by Achanta *et al.* [10], *DIVoG* showed a significant increase in performance by a factor of 6 when using colour images. A real-time implementation of Achanta's work has also been carried out, leading to three times faster execution time than the original implementation, which is still 56% slower than the *DIVoG* approach. Given that for VGA (640×480) resolution the achieved framerate exceeds 80 fps on greyscale images using standard computer hardware, this algorithm could significantly improve the performance of a wide range of applications based on salient feature detection. In addition, the DeGraF depth estimation approach was used to improve the real-time performance of object detection methodologies and in particular the HOG pedestrian detection algorithm [61]. Fusing information from a DIVoG saliency map and a DeGraF depth map, gives a clear indication of the regions of interest where objects are likely to exist. The results show comparable accuracy between the original HOG implementation and our accelerated variant while execution time has improved by at least five times.

Overall, this thesis offers some major contributions to knowledge by proposing significantly faster algorithms for performing efficient object detection. The chosen approach is based on detecting and tracking features in order to perform 3D reconstruction before fusing the derived depth map with a saliency map. An object detector then scans a limited subset of regions of interest in order to confirm whether the detected object belongs in a predefined class. Each algorithm contributes not only to more efficient object detection but also to a wide range of computer vision applications that could benefit from faster feature detection, 3D reconstruction or visual saliency measurement.

6.2 Future work

This thesis has developed a set of real-time algorithms for automotive applications, however, there is more research still required in several directions. The first area is related to feature detection. The approach presented in Chapter 3 proposes a new noise-resistant model for calculating gradients, which fundamentally increases the accuracy and performance of gradient-based approaches. DeGraF-α features are derived by grouping DeGraF-β features and and then marking the local maxima and minima as potential features. Instead a probabilistic framework could be used for making a weighted decision on which is the most dominant feature. In addition, when this approach is combined with existing feature descriptors, such as those proposed by SIFT [1] and SURF [2], then DeGraF feature matching could be performed for more efficient object detection.

The algorithms in Chapter 4 that perform 3D reconstruction by tracking DeGraF features should also be revisited. In this case, the focus should be on evaluating different types of feature tracking techniques, especially those that perform faster than Bougeut's variant of the Lucas Kanade (LK) algorithm [68]. The fact that gradient centroids offer subpixel accuracy with high noise resistance should lead to a wide range of applications that could be developed using low-cost cameras. In addition, the 3D reconstruction algorithm shall be enhanced to support complex vehicle motion. Currently, the majority of the tests are performed with the vehicle

moving in a straight line. More testing is required when the vehicle is moving with different patterns in both motorways and urban environments.

The novel visual saliency algorithm proposed in Chapter 5 should firstly be tested with more object detection approaches in order to assess the benefits of saliency information. A particular area of further research would be the accuracy of deriving gradients from the saliency map rather than the raw image. Additionally, the proposed approach could also be expanded into new areas outside the automotive domain. For example, the proposed visual saliency approach could be used in photography for faster autofocus or in security applications for more efficient identification of suspects.

Finally, different fusion strategies need to be adopted for combining visual saliency and depth information. Currently, this is performed by pixel-wise multiplication of the two input maps, which performs at a satisfactory level, however, a more probabilistic approach would ensure that the two maps do not always have equal weight. For example, in an environment with very low saliency information, the depth map should be the main source of extracting information and vice versa. With the current approach both depth and saliency information need to be present for the algorithm to operate reliably.

Bibliography

[1] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, Nov. 2004.

[2] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (SURF)," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[3] M. Bertozzi, A. Broggi, D. Colla, and R. Fascioli, "Sensing of automotive environments using stereo vision," in *In 30th International Symposium on Automotive Technology and Automation ISATA Special Session on Machine Vision and Intelligent Vehicles and Autonomous Robots*, 1997, pp. 187–193.

[4] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part I," *IEEE Robotics Automation Magazine*, vol. 13, no. 2, pp. 99–110, 2006.

[5] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "MonoSLAM: real-time single camera SLAM." *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, no. 6, pp. 1052–67, Jun. 2007.

[6] S. K. Nayar and V. Branzoi, "Adaptive dynamic range imaging: optical control of pixel exposures over space and time," *Proceedings Ninth IEEE International Conference on Computer Vision*, pp. 1168–1175 vol.2, 2003.

[7] M. Everingham and J. Winn, "The PASCAL Visual Object Classes Challenge 2011 (VOC2011) Development Kit," pp. 1–29, 2010.

[8] M. Enzweiler and D. M. Gavrila, "Monocular pedestrian detection: survey and experiments." *IEEE transactions on pattern analysis and machine intelligence*, vol. 31, no. 12, pp. 2179–95, Dec. 2009.

[9] F. Ahmad, M. Gilbert, S. Myers, J. Pacheco, R. Castellane, and E. Miller, "Lagrange Gradient Mask for Optical Image Processing," *The Open Optics Journal*, vol. 1, no. 1, pp. 4–7, 2007.

[10] R. Achanta, S. Hemami, F. Estrada, and S. Süsstrunk, "Frequency-tuned salient region detection," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009, pp. 1597–1604.

[11] G. R. Bradski and A. Kaehler, *Learning OpenCV: Computer Vision in C++ with the OpenCV Library*, ser. Oreilly and Associate Series. O'Reilly Vlg. GmbH & Company, 2013. [Online]. Available: http://books.google.co.uk/books?id=qXH_uAAACAAJ

[12] A. Ess, B. Leibe, K. Schindler, and L. Van Gool, "A mobile vision system for robust multi-person tracking," in *2008 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, Jun. 2008, pp. 1–8. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4587581' escapeXml='false'/>

[13] W. Jones, "Building safer cars," *IEEE Spectrum*, vol. 39, no. 1, pp. 82–85, 2002.

[14] Mobileye, "Driver Assistance Vision Applications, Vehicle, Pedestrian, Lane Detection." [Online]. Available: http://www.mobileye-vision.com/default.asp?PageID=202

[15] C. Connolly, "Driver assistance systems aim to halve traffic accidents," *Sensor Review*, vol. 29, no. 1, pp. 13–19, Jan. 2009. [Online]. Available: http://www.emeraldinsight.com/journals.htm?issn=0260-2288&volume=29&issue=1&articleid=1768982&show=html

[16] V. Kastrinaki, M. Zervakis, and K. Kalaitzakis, "A survey of video processing techniques for traffic applications," *Image and Vision Computing*, vol. 21, no. 4, pp. 359–381, 2003.

[17] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3354–3361, Jun. 2012. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6248074

[18] M. L. Eichner and T. P. Breckon, "Real-Time Video Analysis for Vehicle Lights Detection using Temporal Information," in *Proc. 4th European Conference on Visual Media Production*. IET, Nov. 2007, pp. I–9. [Online]. Available: http://www.cranfield.ac.uk/ toby.breckon/publications/papers/eichner07headlights.pdf

[19] I. Katramados, S. Crumpler, and T. Breckon, "Real-Time Traversable Surface Detection by Colour Space Fusion and Temporal Analysis," in *Lecture Notes in Computer Science, Volume 5815/2009*, ser. Lecture Notes in Computer Science, M. Fritz, B. Schiele, and J. Piater, Eds. Springer Berlin / Heidelberg, 2009, vol. 5815, pp. 265–274.

[20] A. Kheyrollahi and T. P. Breckon, "Automatic Real-time Road Marking Recognition Using a Feature Driven Approach," *Machine Vision and Applications*, vol. 23, no. 1, pp. 123–133, 2012. [Online]. Available: http://www.cranfield.ac.uk/ toby.breckon/publications/papers/kheyrollahi12marking.pdf

[21] F. Mroz and T. P. Breckon, "An Empirical Comparison of Real-time Dense Stereo Approaches for use in the Automotive Environment," *EURASIP Journal on Image and Video Processing*, vol. 2012, no. 13, pp. 1–19, 2012. [Online]. Available: http://www.cranfield.ac.uk/ toby.breckon/publications/papers/mroz12stereo.pdf

[22] I. Tang and T. P. Breckon, "Automatic Road Environment Classification," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 2, pp. 476–484, Jun. 2011. [Online]. Available: http://www.cranfield.ac.uk/ toby.breckon/publications/papers/tang11classification.pdf

[23] P. R. Beaudet, "Rotationally invariant image operators," in *International Joint Conference on Pattern Recognition*, vol. 579, no. 2. Proceedings of the International Joint Conference on Pattern Recognition, 1978, pp. 579–583.

[24] H. P. Moravec, "Visual mapping by a robot rover," in *Proc of the 6th International Joint Conference on Artificial Intelligence*. Morgan Kaufmann Publishers Inc., 1979, pp. 598–600.

[25] W. Förstner and E. Gülch, "A fast operator for detection and precise location of distinct points, corners and centres of circular features," in *ISPRS Intercommission Workshop Interlaken*, 1987, pp. 281–305.

[26] C. Harris and M. Stephens, "A combined edge and corner detector," *Proc 4th Alvey Vision Conference*, 1988.

[27] C. Tomasi and T. Kanade, "Detection and Tracking of Point Features," *Order A Journal On The Theory Of Ordered Sets And Its Applications*, vol. 7597, no. 7597, p. 22, 1991.

[28] J. Shi and C. Tomasi, "Good features to track," *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition CVPR94*, vol. 94, no. June, pp. 593–600, 1994.

[29] R. M. Haralick, "Digital step edges from zero crossing of second directional derivatives." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 6, no. 1, pp. 58–68, 1984.

[30] F. Heitger, L. Rosenthaler, R. Von Der Heydt, E. Peterhans, and O. Kubler, "Simulation of neural contour mechanisms: from simple to end-stopped cells," *Vision Research*, vol. 32, no. 5, pp. 963–981, 1992.

[31] S. M. Smith and J. M. Brady, "SUSAN - A New Approach to Low Level Image Processing," *International Journal of Computer Vision*, vol. 23, no. 1, pp. 45–78, 1997.

[32] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," *European Conference on Computer Vision (ECCV)*, pp. 430–443, 2006.

[33] K. Mikolajczyk and C. Schmid, "An affine invariant interest point detector," *Image Rochester NY*, vol. 1, no. 1, pp. 128–142, 2002.

[34] K. Mikolajczyk, "Scale & Affine Invariant Interest Point Detectors," *International Journal of Computer Vision*, vol. 60, no. 1, pp. 63–86, 2004.

[35] J. Matas, "Robust wide-baseline stereo from maximally stable extremal regions," *Image and Vision Computing*, vol. 22, no. 10, pp. 761–767, 2004.

[36] T. Kadir and M. Brady, "Saliency, Scale and Image Description," *International Journal of Computer Vision*, vol. 45, no. 2, pp. 83–105, 2001.

[37] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. V. Gool, "A Comparison of Affine Region Detectors," *International Journal of Computer Vision*, vol. 65, no. 1-2, pp. 43–72, 2005.

[38] M. Agrawal, K. Konolige, and M. R. Blas, "CenSurE: Center Surround Extremas for Realtime Feature Detection and Matching," *Lecture Notes in Computer Science*, vol. 5305/2008, pp. 102–115, 2008.

[39] E. Mair, G. D. Hager, D. Burschka, M. Suppa, and G. Hirzinger, "Adaptive and Generic Corner Detection Based on the Accelerated Segment Test," *Computer Vision ECCV 2010*, vol. 6312, pp. 183–196, 2010. [Online]. Available: http://www6.in.tum.de/Main/ResearchAgast

[40] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," *2011 International Conference on Computer Vision*, pp. 2564–2571, Nov. 2011. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6126544

[41] B. Zitova, "Image registration methods: a survey," *Image and Vision Computing*, vol. 21, no. 11, pp. 977–1000, Oct. 2003. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0262885603001379

[42] M. Andriluka, S. Roth, and B. Schiele, "Monocular 3D pose estimation and tracking by detection," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, no. 2, pp. 623–630, Jun. 2010.

[43] J. Civera, A. J. Davison, and J. M. M. Montiel, "Inverse Depth to Depth Conversion for Monocular SLAM," *Proceedings 2007 IEEE International Conference on Robotics and Automation*, vol. 10, no. April, pp. 2778–2783, 2007.

[44] A. J. Davison, Y. G. Cid, and N. Kita, "Real-Time 3D SLAM with Wide-Angle Vision," in *Proc IFAC Symposium on Intelligent Autonomous Vehicles Lisbon*. Citeseer, 2004.

[45] D. Hahnel, R. Triebel, W. Burgard, and S. Thrun, "Map building with mobile robots in dynamic environments," *2003 IEEE International Conference on Robotics and Automation Cat No03CH37422*, vol. 2, pp. 1557–1563, 2003.

[46] J. Civera, A. J. Davison, and J. Montiel, "Inverse Depth Parametrization for Monocular SLAM," *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 932–945, 2008.

[47] J. E. Guivant and E. M. Nebot, "Optimization of the simultaneous localization and map-building algorithm for real-time implementation," pp. 242–257, 2001.

[48] T. Lemaire, C. Berger, I.-K. Jung, and S. Lacroix, "Vision-Based SLAM: Stereo and Monocular Approaches," *International Journal of Computer Vision*, vol. 74, no. 3, pp. 343–364, 2007.

[49] J. M. M. Montiel, J. Civera, and A. J. Davison, "Unified Inverse Depth Parametrization for Monocular SLAM," *Proc Robotics Science and Systems*, vol. 9, p. 1, 2006.

[50] O. Stasse, A. Davison, R. Sellaouti, and K. Yokoi, "Real-time 3D SLAM for Humanoid Robot considering Pattern Generator Information," *Camera*, pp. 348–355, 2006.

[51] C.-c. Wang, C. Thorpe, and S. Thrun, "Online simultaneous localization and mapping with detection and tracking of moving objects: theory and results from a ground vehicle in crowded urban areas," *2003 IEEE International Conference on Robotics and Automation Cat No03CH37422*, vol. 1, pp. 842–849, 2003.

[52] B. Williams, G. Klein, and I. Reid, "Real-Time SLAM Relocalisation," *IEEE 11th International Conference on Computer Vision (2007)*, vol. 07, pp. 1–8, 2007.

[53] D. C. K. Yuen and B. A. MacDonald, "An evaluation of the sequential Monte Carlo technique for simultaneous localisation and map-building," *Robotics and Automation 2003 Proceedings ICRA 03 IEEE International Conference on*, vol. 2, pp. 1564–1569, 2003.

[54] R. Newcombe, S. J. Lovegrove, and A. J. Davison, "DTAM: Dense tracking and mapping in real-time," *2011 International Conference on Computer Vision*, pp. 2320–2327, Nov. 2011. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6126513

[55] D. Gerónimo, A. M. López, A. D. Sappa, and T. Graf, "Survey of pedestrian detection for advanced driver assistance systems." *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 7, pp. 1239–58, Jul. 2010.

[56] M. L. Eichner and T. P. Breckon, "Integrated speed limit detection and recognition from real-time video," in *IEEE Intelligent Vehicles Symposium*. IEEE, Jun. 2008, pp. 626–631.

[57] T. Gandhi and M. Trivedi, "Pedestrian Protection Systems: Issues, Survey, and Challenges," *IEEE Transactions on Intelligent Transportation Systems*, vol. 8, no. 3, pp. 413–430, Sep. 2007.

[58] R. Bishop, *Intelligent Vehicle Technology and Trends*. Artech House, 2005.

[59] I. Kallenbach, R. Schweiger, G. Palm, and O. Lohlein, "Multi-class Object Detection in Vision Systems Using a Hierarchy of Cascaded Classifiers," in *IEEE Intelligent Vehicles Symposium*. IEEE, 2006, pp. 383–387.

[60] F. Xiaodong, "Efficient Multiclass Object Detection by a Hierarchy of Classifiers," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1. IEEE, 2005, pp. 716–723.

[61] N. Dalal and W. Triggs, "Histograms of Oriented Gradients for Human Detection," *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition CVPR05*, vol. 1, no. 3, pp. 886–893, 2004.

[62] R. Achanta, F. Estrada, P. Wils, and S. Süsstrunk, "Salient Region Detection and Segmentation," in *Computer Vision Systems*, ser. LNCS. Springer Berlin / Heidelberg, 2008, vol. 5008, pp. 66–75.

[63] R. Achanta and S. Süsstrunk, "Saliency detection for content-aware image resizing," in *IEEE ICIP*, 2009, pp. 1005–1008.

[64] R. Achanta and S. Susstrunk, "Saliency detection using maximum symmetric surround," in *IEEE ICIP*, 2010, pp. 2653–2656.

[65] N. J. Butko, L. Z., G. W. Cottrell, and J. R. Movellan, "Visual saliency model for robot cameras," in *IEEE ICRA*, May 2008, pp. 2398–2403.

[66] D. G. and N. Vasconcelos, "Bottom-up saliency is a discriminant process," in *IEEE ICCV*, 2007, pp. 1–6.

[67] D. Gao, V. Mahadevan, and N. Vasconcelos, "On the plausibility of the discriminant center-surround hypothesis for visual saliency," *Journal of Vision*, vol. 8, no. 7, 2008.

[68] J. Bouguet, "Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm," *Intel Corporation*, vol. 1, no. 2, pp. 1–9, 2001. [Online]. Available: http://robots.stanford.edu/cs223b04/algo_affine_tracking.pdf

[69] L. Itti, C. Koch, and E. Niebur, "A model of saliency-based visual attention for rapid scene analysis," *IEEE PAMI*, vol. 20, no. 11, pp. 1254–1259, Nov. 1998.

[70] Y.-F. Ma and H.-J. Zhang, "Contrast-based image attention analysis by using fuzzy growing," in *Proceedings of the 11th ACM international conference on Multimedia*. ACM, 2003, pp. 374–381.

[71] X. Hou and L. Zhang, "Saliency Detection: A Spectral Residual Approach," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007, pp. 1–8.

[72] J. Harel, C. Koch, and P. Perona, "Graph-based visual saliency," *Advances in Neural Information Processing Systems*, vol. 19, pp. 545–552, 2007.

[73] T. Tuytelaars and K. Mikolajczyk, "Local Invariant Feature Detectors: A Survey," *Foundations and Trends in Computer Graphics and Vision*, vol. 3, no. 3, pp. 177–280, 2007. [Online]. Available: http://www.nowpublishers.com/product.aspx?product=CGV&doi=0600000017

[74] F. Remondino, "Detectors and descriptors for photogrammetric applications," *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 36, no. 3, pp. 49–54, 2006.

[75] E. Rosten and T. Drummond, "Fusing points and lines for high performance tracking," *10th IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, pp. 1508–1515 Vol. 2, 2005.

[76] E. Rosten, R. Porter, and T. Drummond, "Faster and better: a machine learning approach to corner detection." *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 1, pp. 105–19, Jan. 2010.

[77] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "BRIEF: binary robust independent elementary features," pp. 778–792, Sep. 2010. [Online]. Available: http://dl.acm.org/citation.cfm?id=1888089.1888148

[78] T. Kadir and M. Brady, "Scale Saliency: a novel approach to salient feature and scale selection," in *International Conference on Visual Information Engineering VIE 2003 Ideas Applications Experience*. IET, 2003, pp. 25–28.

[79] T. Kadir, A. Zisserman, and M. Brady, "An Affine Invariant Salient Region Detector," in *ECCV*, ser. LNCS. Springer Berlin / Heidelberg, 2004, vol. 3021, pp. 228–241.

[80] T. Morris, *Computer Vision and Image Processing*, ser. Cornerstones of Computing Series. Palgrave Macmillan Limited, 2003.

[81] A. Yilmaz, O. Javed, and M. Shah, "Object tracking," *ACM Computing Surveys*, vol. 38, no. 4, p. 13, 2006.

[82] C. J. Veenman, M. J. T. Reinders, and E. Backer, "Resolving motion correspondence for densely moving points," pp. 54–72, 2001.

[83] V. Salari and I. K. Sethi, "Feature point correspondence in the presence of occlusion," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 1, pp. 87–91, 1990.

[84] T. J. Broida and R. Chellappa, "Estimation of Object Motion Parameters from Noisy Images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 1, pp. 90–99, 1986.

[85] T. Kirubarajan and Y. Bar-Shalom, "Probabilistic data association techniques for target tracking in clutter," pp. 536–557, 2004.

[86] R. L. Streit and T. E. Luginbuhl, "Maximum likelihood method for probabilistic multihypothesis tracking," in *Proceedings of SPIE*, vol. 2235, 1994, p. 394.

[87] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 5, pp. 564–577, 2003.

[88] H. T. H. Tao, H. S. Sawhney, and R. Kumar, "Object tracking with Bayesian estimation of dynamic layer representations," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 1, pp. 75–89, 2002.

[89] M. J. Black and A. D. Jepson, "EigenTracking : Robust Matching and Tracking of Articulated Objects Using a View-Based Representation," *International Journal of Computer Vision*, vol. 26, no. 1, pp. 63–84, 1998.

[90] S. Avidan, "Support vector tracking." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 8, pp. 1064–1072, 2004.

[91] A. Blake and M. Isard, "Active Contours," *Signal Processing Image Communication*, vol. 17, no. 6, 1998.

[92] M. Bertalmio, L. Cheng, S. Osher, and G. Sapiro, "Variational Problems and Partial Differential Equations on Implicit Surfaces," *Journal of Computational Physics*, vol. 174, no. 2, pp. 759–780, 2001.

[93] R. Ronfard, "Region-based strategies for active contour models," *International Journal of Computer Vision*, vol. 13, no. 2, pp. 229–251, 1994.

[94] D. P. Huttenlocher, J. J. Noh, and W. J. Rucklidge, "Tracking non-rigid objects in complex scenes," in *Computer Vision 1993 Proceedings Fourth International Conference on*, no. TR92-1320, Cornell University. IEEE Computer Society Press, 1993, pp. 93–101.

[95] K. Sato and J. Aggarwal, "Temporal spatio-velocity transform and its application to tracking and interaction," *Computer Vision and Image Understanding*, vol. 96, no. 2, pp. 100–128, 2004.

[96] R. Szeliski, *Computer Vision: Algorithms and Applications*, 1st ed., Nov. 2010.

[97] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," *International Joint Conference on Artificial Intelligence*, vol. 130, no. x, pp. 674–679, 1981. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.49.2019&rep=rep1&type=pdf

[98] P. Anandan, "A computational framework and an algorithm for the measurement of visual motion," *International Journal of Computer Vision*, vol. 2, no. 3, pp. 283–310, 1989. [Online]. Available: http://www.springerlink.com/index/10.1007/BF00158167

[99] S. Baker, D. Scharstein, and J. Lewis, "A database and evaluation methodology for optical flow," *Vision, 2007. ICCV*, vol. 92, no. 1, pp. 1–31, Nov. 2007.

[100] V. Marion, "Method of color image processing to eliminate shadows and reflections," *Word Intellectual Property Organisation*, vol. WO 2004/02, 2002.

[101] R. Cucchiara, C. Grana, M. Piccardi, A. Prati, and S. Sirotti, "Improving shadow suppression in moving object detection with HSV color information," *ITSC 2001 2001 IEEE Intelligent Transportation Systems Proceedings Cat No01TH8585*, pp. 334–339, 2001.

[102] C. Fredembach and G. Finlayson, "Simple Shadow Removal," *Pattern Recognition*, pp. 18–21, 2006.

[103] A. Prati, I. Mikic, M. M. Trivedi, and R. Cucchiara, "Detecting moving shadows: algorithms and evaluation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 7, pp. 918–923, 2003.

[104] L. Xu, F. Qi, and R. Jiang, "Shadow Removal from a Single Image," *Sixth International Conference on Intelligent Systems Design and Applications*, vol. 2, pp. 1049–1054, 2006.

[105] X. Tao, M. Guo, and B. Zhang, "A neural network approach to the elimination of road shadow for outdoor mobile robot," *1997 IEEE International Conference on Intelligent Processing Systems Cat No97TH8335*, vol. 2, pp. 1302–1306, 1997.

[106] J. Alvarez and A. Lopez, "Road Detection Based on Illuminant Invariance," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 1, pp. 184–193, 2011.

[107] J. K. Tsotsos, S. M. Culhane, W. Y. K. Wai, Y. Lai, N. Davis, and F. Nuflo, "Modeling visual attention via selective tuning," *Artificial Intelligence*, vol. 78, no. 1-2, pp. 507–545, 1995.

[108] N. Ouerhani and H. Hugli, "MAPS: multiscale attention-based presegmentation of color images," in *Proceedings of the 4th International conference on scale space methods in computer vision*. Berlin, Heidelberg: Springer-Verlag, 2003, pp. 537–549.

[109] S. L. M. Won W.J. Jeong, "Road Traffic Sign Saliency Map Model," in *Proceedings of Image and Vision Computing New Zealand*, 2007, pp. 91–96.

[110] J. Sokalski, T. P. Breckon, and I. Cowling, "Automatic Salient Object Detection in UAV Imagery," in *Proc. 25th International Unmanned Air Vehicle Systems*, Apr. 2010, pp. 11.1–11.12.

[111] H. Yu, J. Li, Y. Tian, and T. Huang, "Automatic interesting object extraction from images using complementary saliency maps," in *Proceedings of the international conference on Multimedia*. ACM, 2010, pp. 891–894.

[112] Z. Gu and B. Qin, "Nonrigid Registration of Brain Tumor Resection MR Images Based on Joint Saliency Map and Keypoint Clustering," *Sensors*, vol. 9, no. 12, pp. 10270–10290, 2009.

[113] C. W. H. Ngau, L. M. Ang, and K. P. Seng, "Bottom-up visual saliency map using wavelet transform domain," in *IEEE ICCSIT*, vol. 1, 2010, pp. 692–695.

[114] D. Walther and D. Koch, "Modeling attention to salient proto-objects," *Neural Networks*, vol. 19, no. 9, pp. 1395–1407, 2006.

[115] S. Frintrop, M. Klodt, and E. Rome, "A real-time visual attention system using integral images," *ICVS*, 2007.

[116] T. Bailey and H. F. Durrant-Whyte, "Simultaneous Localization and Mapping (SLAM): Part II State of the Art," *IEEE Robotics and Automation Magazine*, vol. 13, no. 3, pp. 108–117, 2006.

[117] C. Brenneke, O. Wulf, and B. Wagner, "Using 3d laser range data for slam in outdoor environments," *Proceedings 2003 IEEERSJ International Conference on Intelligent Robots and Systems IROS 2003 Cat No03CH37453*, vol. 1, no. section 2, pp. 188–193, 2003.

[118] P. Newman, J. Leonard, J. D. Tardos, and J. Neira, "Explore and return: experimental validation of real-time concurrent mapping and localization," in *Proceedings 2002 IEEE International Conference on Robotics and Automation Cat No02CH37292*, vol. 2, no. May, IEEE Internation Conference on Robotics and Automation. IEEE, 2002, pp. 1802–1809.

[119] S. Thrun, "Robotic Mapping : A Survey," *Science*, vol. 298, no. February, pp. 1–35, 2002.

[120] T. A. V. Calleja, "Visual Navigation and Environment Modeling for Wheeled Mobile Robots," Ph.D. dissertation, Institut de Robotica i Informatica Industrial, Universitat Politecnica de Catalunya, 2007.

[121] D. Fox, W. Burgard, and S. Thrun, "Markov localization for mobile robots in dynamic environments," *Journal of Artificial Intelligence Research*, vol. 11, no. 3, pp. 391–427, 1999.

[122] 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 No03CH37422*, vol. 2, no. September, pp. 1899–1906, 2003.

[123] J. Civera, A. J. Davison, J. A. Magallón, and J. M. M. Montiel, "Drift-Free Real-Time Sequential Mosaicing," *International Journal of Computer Vision*, vol. 81, no. 2, pp. 128–137, 2008.

[124] R. Smith, M. Self, and P. Cheeseman, "A stochastic map for uncertain spatial relationships," in *Proceedings of the 4th international symposium on Robotics Research*, O. Faugeras and G. Giralt, Eds., no. 0262022729, MIT Press. MIT Press, 1988, pp. 467–474.

[125] S. B. Goldberg, M. W. Maimone, and L. Matthies, "Stereo vision and rover navigation software for planetary exploration," *Proceedings IEEE Aerospace Conference*, vol. 5, no. March, pp. 5–2025–5–2036, 2002.

[126] J. Adkisson, *Lost Eye: Coping with Monocular Vision after Enucleation or Eye Loss from Cancer, Accident, Disease.* iUniverse, 2006.

[127] A. Saxena, M. Sun, and A. Y. Ng, "Make3D: learning 3D scene structure from a single still image." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 5, pp. 824–40, 2009.

[128] P. E. Debevec and J. Malik, "Recovering high dynamic range radiance maps from photographs," in *ACM SIGGRAPH 2008 classes on - SIGGRAPH '08.* New York, New York, USA: ACM Press, Aug. 2008, p. 1.

[129] U. Rutishauser, D. Walther, C. Koch, and P. Perona, "Is Bottom-Up Attention Useful for Object Recognition?" *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, pp. 37–44, 2004.

[130] V. A. Prisacariu and I. Reid, "fastHOG - a real-time GPU implementation of HOG," *Science*, vol. 2310, no. 2310, pp. 1–13, 2009.

[131] B. Verma and M. Blumenstein, *Pattern recognition technologies and applications: recent advances.* Idea Group Inc (IGI), 2008.

[132] M. Verma and P. W. McOwan, "A semi-automated approach to balancing of bottom-up salience for predicting change detection performance," *Journal of Vision*, vol. 10, no. 6, 2010.

[133] M. Pedersoli, J. Gonz, A. D. Bagdanov, and J. J. Villanueva, "Recursive Coarse-to-Fine Localization for fast Object Detection," *European Conference on Computer Vision (ECCV)*, vol. 6316, pp. 280–293, 2010.

[134] C. Wohler and J. K. Anlauf, "An adaptable time-delay neural-network algorithm for image sequence analysis." pp. 1531–1536, 1999. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/18252656

[135] S. Munder and D. M. Gavrila, "An experimental study on pedestrian classification." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 11, pp. 1863–1868, 2006. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/17063690

[136] S. Avidan, "Fast Human Detection Using a Cascade of Histograms of Oriented Gradients," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, no. c, pp. 1491–1498, 2006.

[137] Y. Freund, "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting," *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119–139, 1997. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S002200009791504X

[138] G. Borgefors, "Distance transformations in digital images," *Computer Vision Graphics and Image Processing*, vol. 34, no. 3, pp. 344–371, 1986. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0734189X86800470

[139] D. M. Gavrila and S. Munder, "Multi-cue Pedestrian Detection and Tracking from a Moving Vehicle," *International Journal of Computer Vision*, vol. 73, no. 1, pp. 41–59, 2006. [Online]. Available: http://www.springerlink.com/index/10.1007/s11263-006-9038-7

[140] P. L. Rosin, "Measuring corner properties," *Computer Vision and Image Understanding*, vol. 73, no. 2, pp. 291–307, 1999. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1077314298907196

[141] T. Vaudrey, C. Rabe, R. Klette, and J. Milburn, "Differences between stereo and motion behaviour on synthetic and real-world stereo sequences," *23rd International Conference Image and Vision Computing New Zealand*, pp. 1–6, Nov. 2008. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4762133

[142] A. Wedel, C. Rabe, T. Vaudrey, and T. Brox, "Efficient dense scene flow from sparse or dense stereo data," pp. 1–12, 2008. [Online]. Available: http://researchspace.auckland.ac.nz/handle/2292/3257

[143] S. Baker and I. Matthews, "Lucas-Kanade 20 Years On : A Unifying Framework," *International Journal of Computer Vision*, vol. 56, no. 3, pp. 221–255, 2004.

[144] M. Tao, J. Bai, P. Kohli, and S. Paris, "SimpleFlow: A Non-iterative, Sublinear Optical Flow Algorithm," *Computer Graphics Forum*, vol. 31, no. 2pt1, pp. 345–353, May 2012. [Online]. Available: http://doi.wiley.com/10.1111/j.1467-8659.2012.03013.x

[145] M. Grundmann, V. Kwatra, and I. Essa, "Auto-Directed Video Stabilization with Robust L1 Optimal Camera Paths," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2011)*, 2011.

Appendix A

Publications

A.1 Real-Time Traversable Surface Detection by Colour Space Fusion and Temporal Analysis

The following paper has been published in the "*Lecture Notes in Computer Science*", Volume 5815/2009, p265-274. This paper was presented in the "*7th International Conference on Computer Vision Systems* (ICVS'09)" in Liège, Belgium on 14th October 2009. The details are:

- **Paper title**

  Real-Time Traversable Surface Detection by Colour Space Fusion and Temporal Analysis

- **Authors**

  Ioannis Katramados, Steve Crumpler, Toby P. Breckon

A copy of this paper is attached.

Real-time traversable surface detection by colour space fusion and temporal analysis

Ioannis Katramados[1], Steve Crumpler[2] and Toby P. Breckon[1]

[1]Cranfield University, School of Engineering, Cranfield, MK43 0AL, UK
{i.katramados, toby.breckon}@cranfield.ac.uk
[2]TRW Conekt, Stratford Road, Solihull, B90 4GW, UK
steve.crumpler@trw.com

Abstract. We present a real-time approach for traversable surface detection using a low-cost monocular camera mounted on an autonomous vehicle. The proposed methodology extracts colour and texture information from various channels of the HSL, YCbCr and LAB colourspaces by temporal analysis in order to create a "traversability map". On this map lighting and water artifacts are eliminated including shadows, reflections and water prints. Additionally, camera vibration is compensated by temporal filtering leading to robust path edge detection in blurry images. The performance of this approach is extensively evaluated over varying terrain and environmental conditions and the effect of colourspace fusion on the system's precision is analysed. The results show a mean accuracy of 97% over this comprehensive test set.

1 Introduction

This work addresses the problem of autonomous vehicle navigation in semi-structured or unstructured environments where geometrical road models are not applicable. Specifically, a real-time approach is presented which facilitates the detection of traversable surfaces via temporal analysis of multiple image properties. These properties are specifically selected to provide maximally descriptive image information with a minimal computational overhead per image frame. Initially, a multi-stage approach is proposed for feature extraction based on colour and texture analysis. This information is then stored in a temporal memory structure to improve algorithm robustness by means of noise filtering. The proposed methodology has been implemented on the SATURN unmanned ground vehicle as part of the MoD Grand Challenge competition (2008).

Engineering road and obstacle detection systems has long been at the centre of academic and industrial research, leading to a number of successful implementations, ranging from the early road-following systems [4, 16] to the most recent fully automated vehicles in the DARPA Urban Challenge competition (2007) [1, 5, 14]. Additionally, significant research has been motivated by various vehicle platforms for Mars exploration missions [7, 8]. However, the sensing and processing complexity of these systems has often led to costly solutions which whilst useful for exploiting the current limits of technology, do not address the

demand for low-cost autonomous platforms utilising widely available low-cost sensors. Creating such vision systems is not a new concept [3, 11]. Embedded lane-departure warning systems [9, 10], are increasingly becoming commonplace in commercial vehicles, motivated by the demand for improved driver safety. However, not every driving environment is as structured as a conventional roadway and an autonomous vehicle may also be required to traverse unstructured environments under varying conditions.

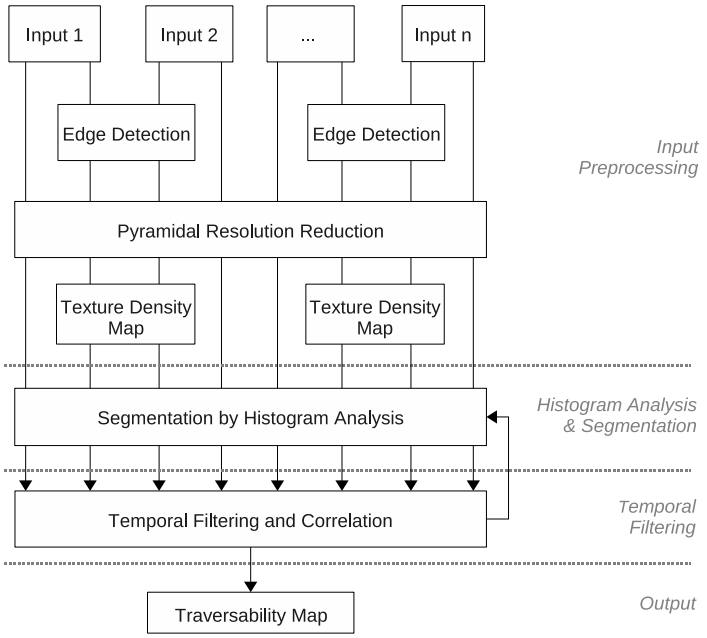


Fig. 1. Traversable area detection methodology

Several prior approaches focus on obstacle detection and avoidance by analysing basic image properties such as texture, colour, hue and saturation of the monocular image. Such approaches are often built on the assumption that the area directly in front of the vehicle is always traversable (initial state assumption) and use a "safe" window to derive the properties of that surface [13]. Obstacles and non-traversable areas are normally identified through a probabilistic model which is based on the similarity of each image pixel to the "safe window" [2, 13, 7]. This becomes the initial *a priori* model from which the system is driven as demonstrated by the Pebbles III robot [13]. The advantages of this approach include flexibility to changing conditions/terrains, limited training requirements and real-time performance. On the other hand, a major disadvantage is its inability to distinguish between surfaces with similar properties due to noise, il-

lumination and environmental effects. To solve this problem Kröse *et al.* [12] proposed the use of optical flow-based techniques, however this is often sensitive to camera vibration and incurs additional computational cost. The work of [12] does however introduce the important aspect of temporal analysis (via frame-to-frame optical flow) as a driver to overcome the earlier limitations of [16, 13]. By contrast, this paper proposes a real-time solution as inspired by the Navlab "Road Following" module [16] and Pebbles III robot [13], with some fundamental changes in the image feature selection from multiple colourspaces and the addition of a novel temporal memory model.

2 Feature extraction for traversable area detection

The following methodology aims to extract information from the video stream output of a vehicle-mounted camera in order to create a map of the traversable and non-traversable areas in real-time. The main challenge is the creation of an algorithm that is adaptable to variable environmental conditions while utilising the least possible computational resource that would facilitate execution on a low-cost processing unit. *Figure 2* provides some examples of such challenging conditions that were experienced during the MoD Grand Challenge competition. The proposed approach is divided into four incremental stages: a) *camera image pre-processing*, b) *multi-dimensional segmentation by histogram analysis*, c) *temporal information processing*, d) *traversable area mapping*. As illustrated in the overview diagram of *Figure 1*, the first stage deals with colour and texture extraction by using intensity-invariant channels of differing colourspaces. The resolution of each input channel is then pyramidically reduced in order to improve system performance and reduce noise (*Figure 1 centre*). Finally, the lower-resolution images are segmented and filtered using a temporal memory model that produces the "traversability" map (*Figure 1 lower*).

Fig. 2. Examples of challenging environmental conditions with shadows, reflections from standing water and wet prints

2.1 Camera Image Pre-processing

First we describe the noise-filtering approach that is applied prior to segmentation in order to eliminate shadows, reflections and water prints. This is achieved

by combining individual channels from differing colourspaces to extract colour and texture information that is insensitive to illumination changes. Prior research [17, 6, 15, 13] has shown that choosing the right colourspace is crucial for extracting accurate path and obstacle features. In fact this methodology combines the *HSL*, *YCbCr* and *LAB* colourspaces [15] to derive four illumination invariant features as listed below:

- **Saturation** (based on the S channel of the *HSL* colourspace)
  By converting the *RGB* colourspace to *HSL*, the saturation channel is extracted (as illustrated in *Figure 3)* and further resized to a coarse 64×48 saturation intensity map by Gaussian pyramid decomposition of the 320×240 input image.
- **Saturation-based texture**
  This can be derived by applying an edge detector on the S channel of the *HSL* colourspace (*Figure 3*). Then the texture is defined as the density of edges in different parts of the image. Practically, this is achieved by Gaussian pyramid decomposition of the output of the *Sobel* edge detector in order to generate a low-resolution 64×48 grid.
- **Mean Chroma** (based on combining the *Cb* and *Cr* components of the *YCbCr* colourspace with the A component of the *LAB* colourspace)
  Chroma provides luminance-independent colour information in the *YCbCr* colourspace. As with the S channel of the *HSL* colourspace, dark shadows and reflections alter the chroma level making their detection difficult. To solve this problem Wu et al. [17] propose the combination of the two chroma components (*Cb* and *Cr*) in order to detect features that are entirely light intensity invariant. However, the *Cb* and *Cr* components have a relatively small variation range when compared to the Y component. Based on this observation, the *Cb* and *Cr* values are scaled to fit the $0 - 255$ (8-bit) range and subsequently their mean value is derived. The A channel of the *LAB* colourspace also provides intensity invariant information, thus by combining it with the mean value of *Cb* and *Cr*, a map of colour distribution (*Figure 4a*) is created as described by *equation 1.*

$$chroma\ map = \frac{sCb + sCr + 2sA}{4} \tag{1}$$

  where sCb is the *Cb* channel of the *YCbCr* colourspace, sCr is the *Cr* channel of the *YCbCr* colourspace and is the A channel of the *LAB* colourspace. These three parameters have been scaled to 8-bit ($0 - 255$ range).
  An example of a *mean chroma* map is illustrated in *Figure* 4*a*, where most reflections have successfully been eliminated. This map is also pyramidically reduced to a coarse 64×48 grid.
- **Chroma-based texture** (based on the *Cb* and *Cr* components of the *YCbCr* colourspace)
  This is derived by calculating the mean value of the *Cb* and *Cr* components to generate a new chroma map. The *Sobel* edge detector is subsequently applied to this map in order to calculate a chroma-based texture density

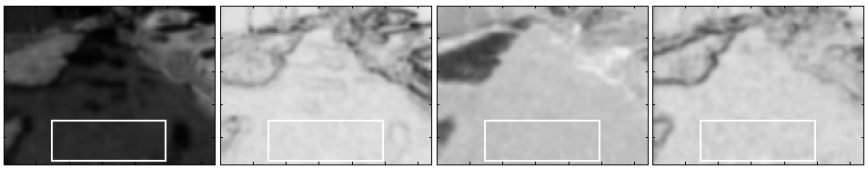


Fig. 3. Image analysis into four input channels: saturation, saturation-based texture, mean chroma and chroma-based texture

using the process described in the saturation-based texture above (*Figure 3*).

At this point we have four 64×48 arrays (8-bit) representing a set of characteristic image properties. These arrays form the input of the segmentation algorithm as described in the following section.

2.2 Segmentation by Histogram Analysis

Several prior path-following techniques have been developed around the assumption that the area immediately in front of the vehicle is initially traversable and thus they identify the pathway by comparison to "safe" window near the bottom of the image [13, 2]. The current approach also adopts this idea since the "safe" window can always be validated by low-cost active short-range sensors such as ultrasonic or infrared. A histogram is calculated for each of the four input image arrays (from the pre-processing stage) within the safe area. The histogram resolution is then reduced by a factor of 8 in order to simplify its processing and improve performance. Thus four different histograms are derived, from which the dominant features of the traversable area are extracted by detecting the histogram peaks based on the assumption that each surface is characterised by a certain combination of saturation, chrominance and texture density levels. Each histogram peak is considered as a feature with five attached properties:

- **Left histogram peak edge**: The point where the left side of the peak meets the "mean level"[1] line
- **Right histogram peak edge**: The point where the right side of the peak meets the "mean level" line
- **Histogram peak value**: The peak value of the low-resolution histogram
- **Mean segment value**: The mean value of the left and right edges of the histogram peak
- **Age**: The time that the peak has remained consistent (in terms of persistence over multiple image frames). A peak is considered as a valid feature only if its age is above a certain threshold. In our tests, the age threshold was set to 10 frames (0.4 sec) with a maximum possible age of 30 frames (1.2 sec).

[1] Defined as the mean of all the histogram values

The left and right histogram peak edges form a histogram segment. Each image pixel is marked as traversable only if its value falls within one of the histogram segments. The remaining pixels are marked as non-traversable. In most cases, the histogram will have only one main segment thus the image will essentially be thresholded. However, more complex surfaces may result in two or more histogram peaks and thus two or more segments. This feature makes the current approach suitable for identifying simple as well as composite traversable surfaces. At this stage, we have four segmented image arrays for each of the four inputs. These arrays are then stored in a temporal memory structure as described in the next section.

2.3 Temporal memory model and correlation

Creating high-level representations of complex raw data can be improved by introducing a temporal memory structure as a way of reducing noise and increasing system accuracy and reliability. This approach proposes the use of temporal behaviour analysis on the output of the segmentation as a top-level filter before correlation. Specifically, the segments identified by histogram analysis are tracked over a series of video frames in order to check their consistency. This is done by assigning a confidence level to each type of surface, which adjusts depending on whether a similar surface appears repeatedly or not. In this way, the system compensates for noise and image blur on a frame-by-frame basis. Similarly, each grid cell of the segmented images is also assigned a confidence level, which increases if its status as "traversable" or "non-traversable" remains unchanged over time. The final output consists of four new "*traversability*" maps based on the saturation, saturation-based texture, mean chroma and chroma-based texture analysis over time. The final traversability map is then derived by majority voting. Although, more sophisticated techniques could have been implemented, this specific one was preferred as the best compromise between overall robustness and real-time performance. Four different levels of traversability are possible for each pixel as illustrated in *Figure* 4*b*, where the darker shades of grey indicate non-traversable areas.

3 Results

The presented approach has been evaluated using a video dataset comprising of sequences captured under a wide range of environmental conditions and different terrain types (*Table 1, Figure* 5). In each video, path and obstacle boundaries (ground truth) were manually marked at *1 sec* intervals. The algorithm output was compared to the ground truth and its accuracy was derived as follows:

$$Accuracy\,(\%) = \left(1 - \frac{\sum_{i=0}^{M-1} \sum_{j=0}^{N-1} (|g_{ij} - o_{ij}|)}{M \times N} \right) \times 100 \qquad (2)$$

where g_{ij} is the ground truth array of size $M \times N$ and o_{ij} is the output array of size $M \times N$. In each of the g_{ij} and o_{ij} arrays the traversable pixels are denoted by '1' and the non-traversable pixels by '0'. Error measurement is then performed by calculating the absolute difference of the two arrays. Note that throughout testing no horizon level was used although this would normally increase the system performance and accuracy further. The results for each scenario are listed in *Table 2*, where the algorithm accuracy was derived using different number of input channels as follows: ***a) 1-channel test*:** Using saturation only, ***b) 2-channel test*:** Using saturation and saturation-based texture, ***c) 3-channel test*:** Using saturation, saturation-based texture and mean chroma and ***d) 4-channel test*:** Using saturation, saturation-based texture, mean chroma and chroma-based texture.

a) *b)*

Fig. 4. *a)* Chroma-based analysis: Areas of low chrominance are eliminated including the foreground water reflections, *b)* Segmentation result after temporal analysis

The algorithm has generally been robust in predicting the traversability of an area regardless of the image quality, noise and camera vibration. *Figure 5* provides some characteristic examples of the system output. As we can see from *Table* 2, a performance of between 95.2% - 97.8% against the ground truth is achieved over a range of conditions (cloudy, wet, sunny, shadow, snow) and a range of terrains (concrete, grass, soil, tarmac, snow) with varying levels of vibration (empirically) recorded on the vehicle platform (*Figures 4b,* 5). The error is measured for each test by calculating the standard deviation of the samples. The overall accuracy and error are then derived by calculating the weighted mean. It should also be noted that using more input channels does not always increase the system accuracy and as a matter of fact the system can sometimes perform better with fewer inputs. This is logical since the colour properties of a surface change with weather and lighting conditions. As a matter of fact, if the system had chosen the right number of input channels for each test, the mean accuracy would have been 97.9% $\pm$ 2.5% (based on the maximum accuracy per test as highlighted by italic characters in *Table 3*). Given the subjective nature of ground truth labelling such a result is also subject to a $\approx$2% error, which is highly acceptable within an autonomous driving scenario.

The evaluation was done using the architecture described in *Figure 1*, which performed in real-time (25 frames per second) when implemented in C++ and ex-

ecuted on a 2GHz Intel Core2Duo CPU using up to four input channels. The camera was mounted on a vehicle that was moving at approximately walking pace. While testing, most obstacles were accurately detected as non-traversable areas except in situations where they were indistinguishable from the underlying surface. Regarding changing environmental conditions (*Table 1*), the performance was good, although reflections were sometimes detected as non-traversable areas. *The video dataset, ground truth data and results can be accessed via the following URL: http://tiny.cc/yannis.*

ID	**Conditions**	**Terrain Type**	**Vibrations**	**Samples**
1	Cloudy Dry	concrete	Light	81
2	Cloudy Wet	concrete	Light	103
3	Cloudy Muddy	soil, grass, gravel	Medium	10
4	Sunny Wet	concrete	Light	20
5	Complex Shadows	tarmac	Very Intense	100
6	Sunny Dry	poor quality tarmac	Very Intense	18
7	Strong shadows	concrete	Light	56
8	Snow	snow-covered tarmac	Medium	104
Total				**492**

Table 1. Environmental and terrain conditions during testing

		1-channel		**2-channel**		**3-channel**		**4-channel**	
ID	*Weight*	*Accuracy*	*Error*	*Accuracy*	*Error*	*Accuracy*	*Error*	*Accuracy*	*Error*
1	0.16	94.87%	3.38%	***97.63%***	1.73%	96.18%	2.20%	97.04%	1.47%
2	0.21	93.68%	4.04%	98.33%	1.09%	98.52%	1.32%	***98.72%***	0.86%
3	0.02	94.35%	2.42%	95.47%	2.96%	97.79%	1.90%	***98.10%***	2.37%
4	0.04	97.12%	4.04%	99.12%	0.74%	***99.42%***	0.24%	99.24%	0.41%
5	0.20	92.47%	8.75%	***96.07%***	5.45%	95.44%	5.50%	95.41%	6.03%
6	0.04	96.46%	3.22%	***98.24%***	1.32%	96.70%	2.81%	96.47%	2.30%
7	0.11	98.95%	1.07%	99.14%	0.73%	***99.20%***	0.66%	99.05%	0.86%
8	0.21	97.01%	4.43%	***98.18%***	3.63%	98.06%	3.04%	98.09%	3.81%
Weighted mean		**95.19%**	**4.57%**	**97.79%**	**2.61%**	**97.44%**	**2.63%**	**97.60%**	**2.70%**

Table 2. Algorithm accuracy results in changing conditions using varying number of input channels (the numbers in bold-italic font denote the test with the highest accuracy in each row)

4 Conclusions

In this paper an effective real-time methodology was presented for detecting traversable surfaces by fusing colour and texture information from *HSL*, *YCbCr* and *LAB* colourspaces to perform image segmentation using a temporal memory model. By initially assuming that the area in front of the vehicle is traversable, the algorithm compares the characteristics of the "safe window" to the rest of the image and creates a "traversability" map. Furthermore, the temporal information is used to filter noise and thus improve system robustness. Testing has

proved that this approach is well-suited for autonomous navigation in unstructured or semi-structured environments (up to 97.8% ±2.6% accuracy) and can perform in real-time on platforms with limited processing power. Future work will concentrate on developing an algorithm that can be trained to classify the environmental and terrain conditions in order to optimise colour space fusion.

Acknowledgements

This research has been supported by the Engineering and Physical Sciences Research Council (EPSRC, CASE/CNA/07/85) and TRW Conekt.

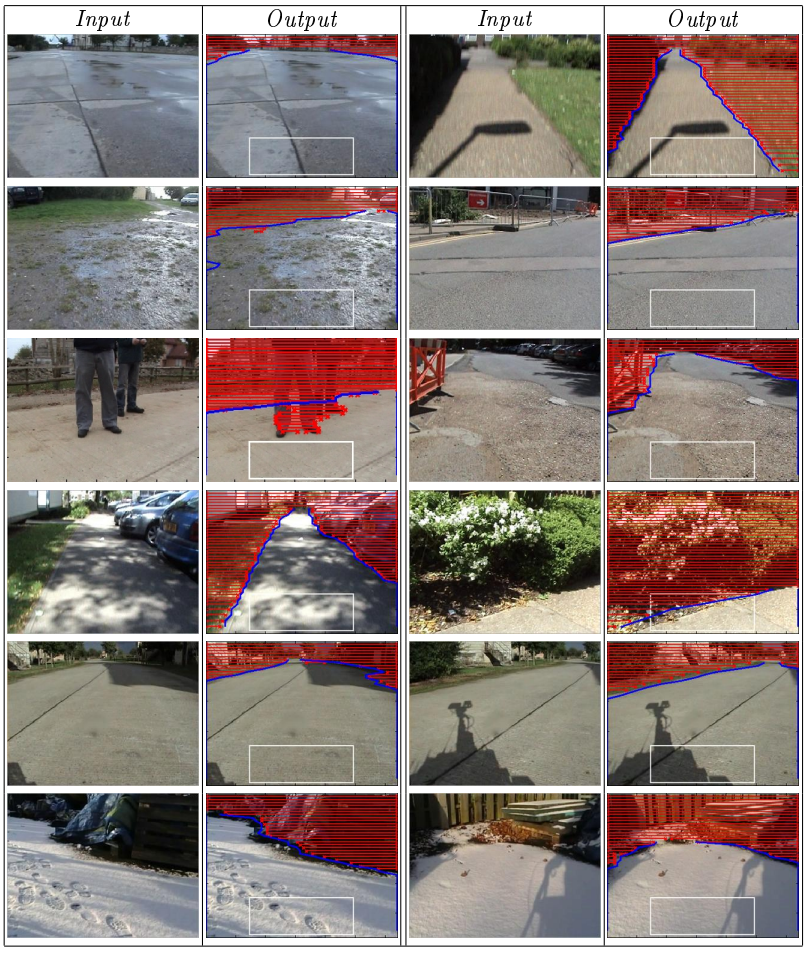


Fig. 5. System input and output examples. Areas covered with red lines are non-traversable. The blue lines indicate the path boundaries and the white box indicates the "*safe window*".

References

1. Y. Alon, A. Ferencz, and A. Shashua. Off-road path following using region classification and geometric projection constraints. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1:689–696, 2006.
2. E. M. Amendt and F. W. DePiero. Multi-dimensional k-means image segmentation for off-road autonomous navigation. *10th IASTED International Conference on Signal and Image Processing (SIP)*, 623-065, 2008.
3. P.H. Batavia and S. Singh. Obstacle detection in smooth high curvature terrain. *IEEE International Conference on Robotics and Automation*, 3:3062–3067, 2002.
4. M. Bertozzi, A. Broggi, and A. Fascioli. Vision-based intelligent vehicles: State of the art and perspectives. *Robotics and Autonomous Systems*, 32(1):1–16, July 2000.
5. A. Broggi, C. Caraffi, P.P. Porta, and P. Zani. The single frame stereo vision system for reliable obstacle detection used during the 2005 darpa grand challenge on terramax. *IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 745–752, 2006.
6. R. Cucchiara, C. Grana, M. Piccardi, A. Prati, and S. Sirotti. Improving shadow suppression in moving object detection with hsv color information. *IEEE Intelligent Transportation Systems*, pages 334–339, 2001.
7. G.N. Desouza and A.C. Kak. Vision for mobile robot navigation: a survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(2):237–267, 2002.
8. S.B. Goldberg, M.W. Maimone, and L. Matthies. Stereo vision and rover navigation software for planetary exploration. *IEEE Aerospace Conference Proceedings*, 5:5–2025–5–2036 vol.5, 2002.
9. P. Hsiao and C. Yeh. A portable real-time lane departure warning system based on embedded calculating technique. *IEEE 63rd Vehicular Technology Conference*, 6:2982–2986, 2006.
10. J. Kaszubiak, M. Tornow, R.W. Kuhn, B. Michaelis, and C. Knoeppel. Real-time vehicle and lane detection with embedded hardware. *IEEE Intelligent Vehicles Symposium*, pages 619–624, 2005.
11. K. M. Agrawal Konolige and J. Sola. Large-scale visual odometry for rough terrain. In *International Symposium on Research in Robotics, Hiroshima , Japan*, 2007.
12. B. J. A Krï¿œse, A. Dev, and F. C. A. Groen. Heading direction of a mobile robot from the optical flow. *Image and Vision Computing*, 18(5):415–424, 2000.
13. L.M. Lorigo, R.A. Brooks, and W.E.L. Grimsou. Visually-guided obstacle avoidance in unstructured environments. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1:373–379 vol.1, 1997.
14. A.V. Nefian and G.R. Bradski. Detection of drivable corridors for off-road autonomous navigation. *IEEE International Conference on Image Processing*, pages 3025–3028, 2006.
15. Y. Shan, F. Yang, and R. Wang. Color space selection for moving shadow elimination. *Fourth International Conference on Image and Graphics (ICIG)*, pages 496–501, 2007.
16. C. Thorpe, M.H. Hebert, T. Kanade, and S.A. Shafer. Vision and navigation for the carnegie-mellon navlab. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(3):362–373, 1988.
17. J. Wu, Z. Yang, J. Wu, and A. Liu. Virtual line group based video vehicle detection algorithm utilizing both luminance and chrominance. *2nd IEEE Conference on Industrial Electronics and Applications (ICIEA)*, pages 2854–2858, May 2007.

A.2 Real-time visual saliency by division of Gaussians

The following paper has been accepted for publication in the proceedings of the IEEE International Conference on Image Processing (ICIP) in Brussels, Belgium on 11th September 2009. The details are:

- **Paper title**

  Real-time visual saliency by division of Gaussians

- **Authors**

  Ioannis Katramados, Toby P. Breckon

A copy of this paper is attached.

REAL-TIME VISUAL SALIENCY BY DIVISION OF GAUSSIANS

Ioannis Katramados[1,2] *and Toby P. Breckon*[1]

[1]School of Engineering, Cranfield University, UK
[2]TRW Conekt, UK
{i.katramados, toby.breckon}@cranfield.ac.uk

ABSTRACT

This paper introduces a novel method for deriving visual saliency maps in real-time without compromising the quality of the output. This is achieved by replacing the computationally expensive centre-surround filters with a simpler mathematical model named Division of Gaussians (*DIVoG*). The results are compared to five other approaches, demonstrating at least six times faster execution than the current state-of-the-art whilst maintaining high detection accuracy. Given the multitude of computer vision applications that make use of visual saliency algorithms such a reduction in computational complexity is essential for improving their real-time performance.

Index Terms— division of gaussians, DIVoG, salient features, center-surround, ratiometric saliency

1. INTRODUCTION

As a concept, visual saliency started as a biologically inspired process for focusing visual attention to certain parts of an image, thus reducing the complexity of scene analysis [1]. Subsequently, it formed the basis of several computer vision applications, such as in automatic object detection [2, 3, 4, 5], medical imaging [6] and robotics [7]. Different saliency definitions exist, however, in this paper a generalised version of the definition by Achanta *et al.* [8] is used: "*Visual saliency is the perceptual quality that makes a group of pixels stand out relative to its neighbours*". As a research topic, visual saliency theory has evolved rapidly to produce a wide range of approaches. However, their computational cost remains significantly high for real-time applications that require execution at full frame rate ($\geqslant$ 25 frames per second (fps)). This paper proposes a fast alternative to calculating visual saliency maps by using Division of Gaussians (*DIVoG*), which delivers a multifold increase in performance when compared to the current state-of-the-art.

This research has been supported by the Engineering and Physical Sciences Research Council (EPSRC, CASE/CNA/07/85) and TRW Conekt.

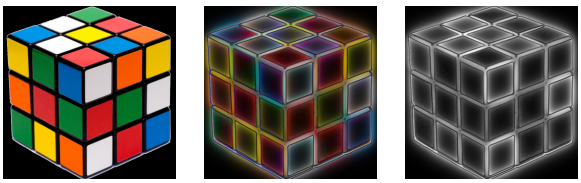


Fig. 1. Colour and greyscale saliency maps of Rubik's cube using the *DIVoG* approach. Darker colours/shades indicate areas of low-saliency and *vice-versa*.

2. EXISTING APPROACHES

Most of the visual saliency models can be categorised into two main groups, as proposed by Achanta *et al.* [9] and Ngau *et al.*[10]: a) biological models and b) computational models. The majority of biological models are using a bottom-up approach for feature extraction mainly based on colour, intensity and orientation [11]. Inspired by the structure of the human eye, this approach detects the contrast difference between an image region and its surroundings, which is also known as centre-surround contrast. Itti *et al.* [11] use the Difference-of-Gaussians (*DoG*) filter for deriving the centre-surround contrast, whereas Walther and Koch [12] take this algorithm further by adopting the concept of salient proto-objects. A common characteristic of these approaches is that they usually produce saliency maps that lack sharpness and detail [5]. Furthermore, the complexity of the biological models means that performance is slow, thus they are more suitable for use in non-real-time applications. One of the few exceptions is found in the approach proposed by Ma and Zhang [13], who calculate the centre-surround contrast by fuzzy growing. The computation takes approximately 60 milliseconds for a 320×240 image on a 2.6 Ghz CPU [8], which corresponds to 16.6 fps.

Examples of computational saliency methods include frequency-tuned salient region detection by Achanta *et al.* [8], graph-based visual saliency by Harel *et al.* [14], affine invariant salient region detection by Kadir *et al.* [15] and real-time visual attention system using integral images by Frintrop *et al.* [16]. The method by Frintrop *et al.* [16], is one

Fig. 2. Saliency map of a pedestrian using *DIVoG*.

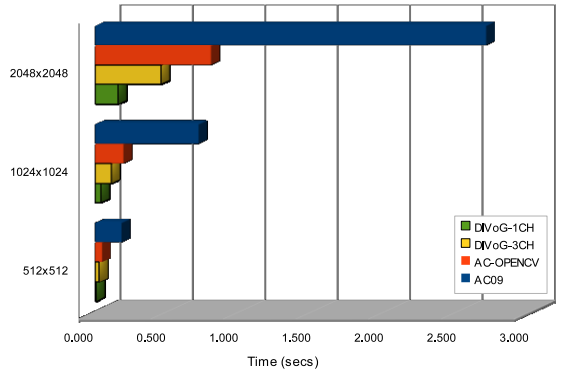


Fig. 3. Performance evaluation of *DIVoG* and "Frequency-tuned Salient Region Detection" by Achanta *et al*. [8] (*AC09*). *AC-OPENCV* is our *AC09* real-time implementation using the *OpenCV* library [18]. *DIVoG-3CH* denotes the *DIVoG* algorithm running on 3 channel input (i.e. RGB image), whereas *DIVoG-1CH* denotes the *DIVoG* algorithm running on a single channel input (i.e. greyscale 8-bit image).

of the most successful attempts to produce a real-time visual saliency algorithm (known as VOCUS) using integral images to reduce execution time. The improvement in performance is impressive with a 400×300 image being processed in approximately 50 milliseconds using a 2.8 Ghz CPU, which corresponds to 20 fps. In addition, the approach proposed by Achanta *et al*. [8] comes close to achieving real-time performance by using frequency domain analysis to produce full resolution saliency maps. The execution time for a 400×300 image is 100 milliseconds on a 2.4 Ghz notebook. Although, this algorithm is proportionally slower than Frintrop *et al*. [16], it generates maps with significantly higher quality.

Ultimately, the target of our algorithm was to produce saliency maps of similar quality to those by Achanta *et al*. [8, 17] at full frame rate ($\geqslant$ 25 fps). In fact, we will show that for a 400×300 image the *DIVoG* approach generates high-detail saliency maps at 50 fps (20 milliseconds per frame) using a 2.4 Ghz CPU.

3. ALGORITHM DESCRIPTION

The Division of Gaussians approach comprises of three distinct steps: 1) Bottom-up construction of Gaussian pyramid, 2) Top-down construction of Gaussian pyramid based on the output of *Step 1*, 3) Element-by element division of the input image with the output of *Step 2*.

Step 1: The Gaussian pyramid U comprises of n levels, starting with an image U_1 as the base with resolution $w \times h$. Higher pyramid levels are derived via downsampling using a 5×5 Gaussian filter. The top pyramid level has a resolution of $(w/2^{n-1}) \times (h/2^{n-1})$. Let us call this image U_n.

Step 2: U_n is used as the top level D_n of a second Gaussian pyramid D in order to derive its base D_1. In this case, lower pyramid levels are derived via upsampling using a 5×5 Gaussian filter.

Step 3: Element-by-element division of $U_1$and D_1 is performed in order to derive the minimum ratio matrix M (also called MiR matrix) of their corresponding values as described by the following equation:

$$M_{i,j} = min\left(\frac{D_{1_{i,j}}}{U_{1_{i,j}}}, \frac{U_{1_{i,j}}}{D_{1_{i,j}}}\right) \quad (1)$$

The saliency map S is then given by *equation 2*, which means that saliency is expressed as a floating-point number in the range $0 - 1$.

$$S_{i,j} = 1 - M_{i,j} \quad (2)$$

The described approach can be further expanded to include element-by-element division of all corresponding levels of pyramids U and D. In this case, the MiR matrix is initialised as a unit matrix (i.e. for each matrix element $M_{0_{i,j}} = 1$). Then each pair of pyramid levels U_n and D_n is scaled up to the input's resolution. Then the MiR matrix M_n is multiplied by M_{n-1} as described by the *DIVoG* equation below, which is a generalised form of *equation 1*.

$$M_{n_{i,j}} = min\left(\frac{D_{n_{i,j}}}{U_{1_{i,j}}}, \frac{U_{1_{i,j}}}{D_{n_{i,j}}}\right) M_{n-1_{i,j}} \quad (3)$$

for $n \geqslant 1$. The saliency map is then derived using *equation 2*. Deriving the MiR matrix through processing of all pyramid levels produces more accurate saliency maps than *equation 1*, but also increases the computational complexity of the algorithm. In practice, the difference between the two approaches is visually minimal, thus in this paper all MiR matrices have been calculated using *equation 1*. Finally, a major advantage of this approach is that it is colourspace-independent, thus it can derive saliency maps even from greyscale images, which significantly reduces computational cost.

Implementation notes: a) All operations are performed using 32-bit floating point matrices. b) To avoid division by zero, or division with floating point numbers in the range 0 to 1, we define the minimum pixel value equal to k^n, where k is the size of the Gaussian kernel. This ensures that pyramidal

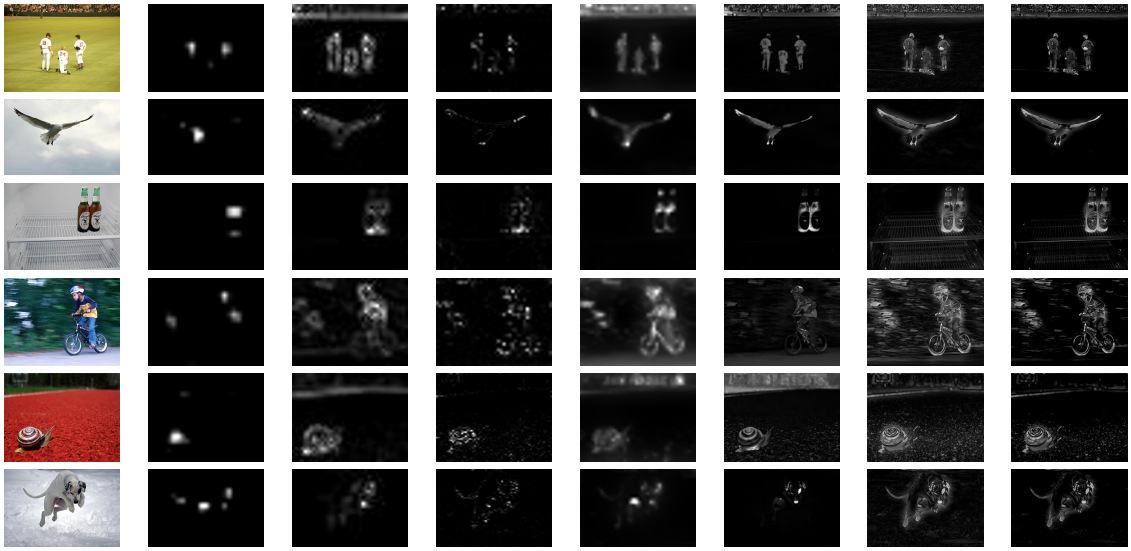


Fig. 4. A set of saliency maps generated using different approaches (based on work by Achanta *et al*. [8]). DIVoG-F enhances these results of the standard DIVoG algorithm by adding a low-pass filter to reduce background noise.

downsampling will always result into a value greater than 1. c) For colour images, the algorithm can be used with any colourspace. Each channel is processed separately to produce a salience map. d) All the saliency maps in this paper have been produced using 24-bit colour images in the RGB colourspace. The Gaussian pyramid is constructed with $n = 5$. e) All saliency maps in Fig. 1, 2, 4, have been normalised to fit the $0 - 255$ range.

4. RESULTS

The *DIVoG* approach is compared with five other saliency algorithms using an evaluation framework created by Achanta *et al [8, 17].* As part of this procedure, saliency maps are extracted for 1000 images using five different approaches [11, 13, 19, 14, 8], as illustrated in Fig. 4. These maps are then used to segment the images. Finally, the extracted segments are compared to the ground-truth in order to derive the algorithm's accuracy. This is a reasonable approach for simple scenes with a small number of distinct objects. However, for more complex images the specification of ground-truth is becoming subjective. Since the main contribution of this paper is related to the real-time performance of the algorithm, we compare the execution time of our approach with Achanta *et al. [8]*, which is one of the most efficient saliency methodologies for producing high-resolution maps.

For performance evaluation a mobile 2.4GHz Intel Core 2 Duo processor was used with 4GB RAM. Fig. 3 and Table 1 show a comparison in execution time between *DIVoG* and *[8]* at different resolutions using colour and greyscale images. Furthermore, Fig. 4 shows some examples of saliency maps generated using DIVoG and five other approaches.

The original implementation by Achanta *et al [8]* (*AC09*), produces much sharper saliency maps than *IT98, MA03, HO07* and *HA07*. In terms of computational performance *AC09* is at least comparable to the aforementioned approaches as presented in *[8].* On the other hand, the *DIVoG* approach demonstrates similar or higher quality saliency maps to *AC09*, but at a fraction of the time. *DIVoG* is faster than *AC09* by a factor of 6 when processing 24-bit colour images and by a factor of 16 when processing greyscale images. This massive gap could not be justified by the theoretical difference in computational complexity, thus the *AC09* was re-implemented using the *OpenCV* library [18] (*AC-OPENCV*). This way the execution time reduced by a factor of 3. Even so, *AC-OPENCV* remained 56% slower than *DIVoG*. An indication of performance can also be given by quoting the achieved framerate. At the lowest resolution of 320×240, *DIVoG* executed at 333 fps on greyscale images and 111 fps on colour images, showing a linear relationship between data size and execution time. Overall, the *DIVoG* approach has demonstrated an ability to calculate full resolution saliency maps with the minimum computational cost.

5. CONCLUSIONS

We presented a novel visual saliency algorithm for calculating full resolution saliency maps in real-time by using Divi-

	AC09 [8]		*AC-OPENCV*	
Resolution	*Time (s)*	*fps*	*Time (s)*	*fps*
320×240	0.078	12.8	0.015	66.6
512×512	0.187	5.3	0.052	19.2
640×480	0.218	4.6	0.057	17.5
1024×1024	0.718	1.4	0.200	5.0
2048×2048	2.699	0.4	0.803	1.6
	DIVoG-3CH		***DIVoG- 1CH***	
320×240	0.009	**111**	0.003	**333**
512×512	0.032	**31.2**	0.009	**111**
640×480	0.036	**27.7**	0.012	**83.3**
1024×1024	0.115	**8.7**	0.041	**24.3**
2048×2048	0.456	**2.2**	0.161	**6.2**

Table 1. Performance evaluation data showing execution time and framerate. *AC09* is the original implementation by Achanta *et al*. [8].

sion of Gaussians. Compared to recent work by Achanta *et al*. [8], *DIVoG* showed a significant increase in performance by a factor of 6 when using colour images. This paper also introduced a real-time implementation of Achanta's work using the OpenCV library [18], which is more than three times faster than the original implementation, but still 56% slower than the *DIVoG* approach. Given that for VGA resolution the achieved framerate exceeds 80 fps on greyscale images, this algorithm could significantly improve the performance of a wide range of applications including salient feature detection, object extraction and classification.

6. REFERENCES

[1] J.K. Tsotsos, S.M. Culhane, Winky Yan Kei Wai, Yuzhong Lai, N. Davis, and F. Nuflo, "Modeling visual attention via selective tuning," *Artificial Intelligence*, vol. 78, no. 1-2, pp. 507 – 545, 1995.

[2] N. Ouerhani and H. Hugli, "Maps: multiscale attention-based presegmentation of color images," in *Proceedings of the 4th International conference on scale space methods in computer vision*, Berlin, Heidelberg, 2003, pp. 537–549, Springer-Verlag.

[3] S. Lee M. Won, W.J. Jeong, "Road traffic sign saliency map model," in *Proceedings of Image and Vision Computing New Zealand*, 2007, pp. 91–96.

[4] J. Sokalski, T.P. Breckon, and I. Cowling, "Automatic salient object detection in uav imagery," in *Proc. 25th International Unmanned Air Vehicle Systems*, April 2010, pp. 11.1–11.12.

[5] Haonan Yu, Jia Li, Yonghong Tian, and Tiejun Huang, "Automatic interesting object extraction from images using complementary saliency maps," in *Proceedings of the international conference on Multimedia*. 2010, pp. 891–894, ACM.

[6] Zhijun Gu and Binjie Qin, "Nonrigid registration of brain tumor resection mr images based on joint saliency map and keypoint clustering," *Sensors*, vol. 9, no. 12, pp. 10270–10290, 2009.

[7] N.J. Butko, Lingyun Z., G.W. Cottrell, and J.R. Movellan, "Visual saliency model for robot cameras," in *IEEE ICRA*, May 2008, pp. 2398 –2403.

[8] R. Achanta, S. Hemami, F. Estrada, and S. Süsstrunk, "Frequency-tuned salient region detection," in *IEEE CVPR*, 2009, pp. 1597 –1604.

[9] R. Achanta, F. Estrada, P. Wils, and Sabine Süsstrunk, "Salient region detection and segmentation," in *Computer Vision Systems*, vol. 5008 of *LNCS*, pp. 66–75. Springer Berlin / Heidelberg, 2008.

[10] C.W.H. Ngau, L.M. Ang, and K.P. Seng, "Bottom-up visual saliency map using wavelet transform domain," in *IEEE ICCSIT*, 2010, vol. 1, pp. 692 –695.

[11] L. Itti, C. Koch, and E. Niebur, "A model of saliency-based visual attention for rapid scene analysis," *IEEE PAMI*, vol. 20, no. 11, pp. 1254 –1259, Nov. 1998.

[12] D. Walther and D. Koch, "Modeling attention to salient proto-objects," *Neural Networks*, vol. 19, no. 9, pp. 1395 – 1407, 2006.

[13] Yu-Fei Ma and Hong-Jiang Zhang, "Contrast-based image attention analysis by using fuzzy growing," in *Proceedings of the 11th ACM international conference on Multimedia*. 2003, pp. 374–381, ACM.

[14] J. Harel, C. Koch, and P. Perona, "Graph-based visual saliency," *Advances in Neural Information Processing Systems*, vol. 19, pp. 545–552, 2007.

[15] T. Kadir, A. Zisserman, and M. Brady, "An affine invariant salient region detector," in *ECCV*, vol. 3021 of *LNCS*, pp. 228–241. Springer Berlin / Heidelberg, 2004.

[16] S. Frintrop, M. Klodt, and E. Rome, "A real-time visual attention system using integral images," *ICVS*, 2007.

[17] R. Achanta and S. Süsstrunk, "Saliency detection using maximum symmetric surround," in *IEEE ICIP*, 2010, pp. 2653 –2656.

[18] OpenCV, "Open computer vision library," *http://opencv.willowgarage.com/wiki/*, Last visited on 10/01/2011.

[19] Xiaodi Hou and Liqing Zhang, "Saliency detection: A spectral residual approach," in *IEEE CVPR*, 2007, pp. 1 –8.

A.3 Real-time visual saliency by division of Gaussians

The following paper has been accepted for publication in the proceedings of 8th European Conference for Visual Media Production, London, 16-17 November 2011. The details are:

- **Paper title**

  Adaptive object placement for augmented reality use in driver assistance systems

- **Authors**

  L. Bordes, T.P. Breckon, I. Katramados, A. Kheyrollahi

A copy of this paper is attached.

ADAPTIVE OBJECT PLACEMENT FOR AUGMENTED REALITY USE IN DRIVER ASSISTANCE SYSTEMS

Lucie Bordes, Toby P. Breckon, Ioannis Katramados, Alireza Kheyrollahi
School of Engineering, Cranfield University, UK.

Abstract
We present an approach for adaptive object placement for Augmented Reality (AR) use in driver assistance systems. Combined vanishing point and road surface detection enable the real-time adaptive emplacement of AR objects within a drivers' natural field of view for on-road information display. This work combines both automotive vision and multimedia production aspects of real-time visual engineering.

Keywords: automotive vision, augmented reality.

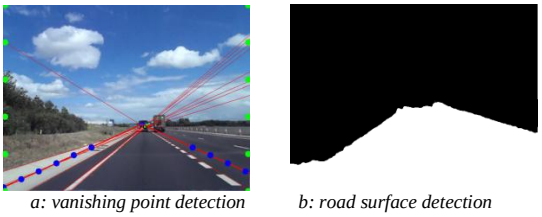
1 Introduction

Recent advances in vehicle technology embed a range of different sensors for road environment monitoring and additionally employ computer vision techniques to extract road information from onboard cameras [1, 2, 3]. The accumulation of this data presented to the driver, in addition to existing satellite navigation, can distract the drivers from their natural road view. This work aims at using combined vanishing point and road surface detection to enable the adaptive emplacement of such information, on the road surface, within the driver's natural field of view. Augmented Reality (AR) could be used in this context for the presentation of both navigation, vehicle status and environment sensed information. Specifically, this work investigates the use of intelligent placement techniques to avoid other environment objects present on the road surface.

2 Road surface detection

In an initial calibration stage, we determine the vanishing point of the road scene using the approach of [1]. This approach uses temporal averaging of an initial calibration sequence to calculate the RANSAC-based intersection of straight lines within the scene. This one time calibration facilitates the recovery of the scene vanishing point from which the road plane homography can be recovered (see Figure 1a).

Subsequently, we have to identify the available space within the defined road area. This is performed using a histogram analysis technique on the saturation channel of the HSL colour transform of the original road image using the technique of [4]. Histogram back-projection is then used to create a probability map of road surface colour occurrence. This is averaged over 10 frames from which a segmented available road surface area (free of other vehicles and environment clutter) is identified. An example is shown in Figure 1b of the identified available road surface area from the original Figure 1a road image.

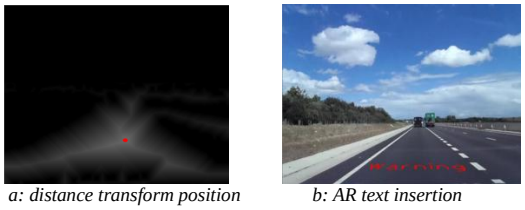


a: vanishing point detection *b: road surface detection*

Figure 1: Road area estimation using [1, 4]

3 Adaptive object placement

The AR information should be presented without interfering with existing objects present in the road environment. A distance transform is used to identify the central point of the segmented road area (Figure 1b) which is maximally distant from the usable road area edges for object emplacement. An inverse perspective mapping using the recovered homography transforms the road surface to a bird eye view upon which an AR marker can be placed. An AR marker and text are rendered onto this view before inverse perspective mapping. The output of the distance transform upon a segmented road image represents the distance to the nearest boundary of the usable (free) road area (Figure 2a). The maximum which is the most central space within the usable road surface is suitable for AR object placement (Figure 2b).



a: distance transform position *b: AR text insertion*

Figure 2: Position identification and emplacement of AR text in road scene

The approach of [5] is used to place AR text and 3D objects upon the road surface at the identified position (Figure 2b).

4 Results

Overall the usable road area detection approach is successful but can be somewhat dependent on the video image quality. A range of different road examples with AR display is shown in Figure 3 where we see adaptation to a number of road occupancy conditions.

Figure 3: Adaptive AR object placement on road scenes

5 Conclusions

The combination of vanishing point detection [1] and road surface detection [4] enables the adaptive emplacement of AR objects within a dynamic road environment suitable for a driver assistance system. Future work could investigate improved road area detection using feature-based or adaptive machine learning techniques.

[1] Kheyrollahi, A. and Breckon, T. P. (2010), "Automatic real-time road marking recognition using a feature driven approach", Machine Vision and Applications, pp. 1-11.

[2] Eichner, M. L. and Breckon, T. P. (2008), "Integrated speed limit detection and recognition from real-time video", IEEE Intel. Vehicles Symp., pp. 626-631.

[3] Tang, I. and Breckon, T. P. (2011), "Automatic road environment classification", IEEE T. on Intel. Trans. Sys., vol. 12, no. 2, pp. 476-484.

[4] Katramados, I., Crumpler, S. and Breckon, T. P. (2009), "Real-time traversable surface detection by colour space fusion and temporal analysis", Proc. Int. Conf. on Computer Vision Systems, pp. 265-274.

[5] Kato, H., Billinghurst, M., Poupyrev, I., Imamoto, K. and Tachibana, K. (2000), "Virtual object manipulation on a table-top AR environment", Proc. Int. Symp. on Augmented Reality, pp. 111-119.

Appendix B

Additional Chapters

This appendix contains additional chapters that were authored during the earlier part of this degree, but are no longer linked to the main theme of the thesis. However, certain aspects of this work are novel and they have also been included in the aforementioned peer-reviewed publications. As a result, these chapters are included for future reference since they could form the basis of further work in the field of autonomous navigation, terrain recognition and real-time feature detection.

Chapter 4

Implementation and Results

This chapter presents the results of the work to date based on the methodology of *Chapter 3*. Specifically, the results of implementing a terrain classification algorithm (*Methodology E)* are described, which was developed out of sequence in order to meet the deadlines of the MoD Grand Challenge competition [9]. The following sections describe a novel implementation of this methodology and evaluate its performance.

4.1 Methodology A - Evaluation of monocular SLAM in an automotive environment with no moving obstacles

One of the main objectives of this project is to reconstruct the vehicle environment in 3D using a monocular camera in order to perform obstacle detection. Practically, this problem is similar to simultaneous localisation and mapping (SLAM) except that the generated 3D map is constructed and analysed in real-time before being discarded. In addition, the height of the camera is fixed thus its motion is limited to the same degrees of freedom as the vehicle. Based on the state-of-the-art review [1], a number of monocular SLAM approaches were identified and evaluated for their performance in automotive environments. Such an evaluation has been challenging since most of the monocular approaches reconstruct the 3D maps of structured environments, while assuming smooth camera movement [3]. However, this assumption

is not valid in most urban and rural road datasets, where camera vibration and image blur are significantly reducing the accuracy of feature detection and tracking. These are two essential elements of any SLAM system, thus their performance needs to be enhanced in order to get a reliable output. Based on this fact, a comparative evaluation of feature detection approaches was performed in order to identify these group of features that best describe automotive environments, as presented in the section below.

4.1.1 Feature detection - Comparative evaluation of affine covariant region detectors

Building a 3D map from monocular camera images requires reliable tracking of image features over several frames in order to derive their motion vectors and subsequently calculate their depth in the real-scene. This means that features need to be **dense**, **repeatable** over time and **resistant** to image noise including rotation caused by vibration. In addition, each feature detector needs to be able to execute in **real-time**. Based on these criteria a set of affine covariant region detectors were evaluated using a method introduced by Mikolajczyk *et al.* [10]. This methodology compares the performance of the following state-of-the-art approaches:

- Harris-Affine [11]
- Hessian-Affine [11]
- Maximally stable extremal regions (MSER) [12, 13]
- Intensity extrema based detector (IBR) [14]
- Edge based detector (EBR) [14]
- Salient Regions [15]

From these approaches, the last two have a long execution time ($>$ 10 seconds / frame), thus they were excluded from the evaluation as they do not meet the real-time requirements of this project. The remaining algorithms were evaluated on an automotive-specific dataset, whereas Mikolajczyk *et al.* [10] had performed the same

Figure 4.1: Sample shots from a custom-made dataset. This dataset is used to evaluate a set of feature detector algorithms in automotive environments. Note that the image has been cropped above a certain height in order to prevent the algorithm detecting features that are far off the road surface.

evaluation on a more generic dataset including a wide range of image scenes. *Figure 4.1* illustrates a set of characteristics input samples.

The performance of each of the feature detectors above is quantified in terms of:

- **Number of correspondences:** "*The number of corresponding regions detected in images under different geometric and photometric transformations.*" [10]
- **Repeatability of features (%):** "*The repeatability score for a given pair of images is computed as the ratio between the number of region-to-region correspondences and the smaller of the number of regions in the pair of images.*" [10]. Two regions are considered as corresponding only if the overlap error is smaller than a certain threshold, as described below [10]:

$$1 - \frac{R_{\mu_a} \cap R_{(H^T \mu_b H)}}{\left(R_{\mu_a} \cup R_{(H^T \mu_b H)}\right)} < threshold$$

where:

R_μ : the elliptic region defined by $x^T \mu x = 1$

H: the homography of two subsequent frames

The results are illustrated in the two graphs of Figure 4.2, where the average number of correspondences and repeatability percentage are measured against time. In this case, time is measured as frame difference between the key-frame and the query frame. In other words, the feature detectors are evaluated at different frame rates in order to derive their consistency. These results are illustrated in *Figure 4.3* and show that:

- The "Hessian Affine" and "Harris Affine" approaches have the highest number of correspondences with a relatively high rate of repeatability across different frames.
- MSER has the highest rate of repeatability although the average number of correspondences is significantly lower.
- The "IBR" approach has the lowest rate of repeatability and the lowest average number of correspondences, thus it is clearly not suitable for automotive applications.
- As expected all approaches demonstrate a decreasing performance at lower frame-rate.
- In terms of execution time, the only algorithm that could be executed in less than 1 second was MSER. The rest of the algorithms executed in 1-10 seconds depending on the complexity of the image. *(Please note that these times do not refer to real-time implementations and could possibly be improved).*

Based on these observations, the "MSER" affine region detector would be satisfying most of the original criteria although the low density of the detected features is not ideal. The "Hessian Affine" and "Harris Affine" approaches offer higher feature density but at a lower repeatability rate and have a higher execution time. As a conclusion, none of the evaluated algorithms combines high repeatability with sufficiently high number of correspondences and as a result a new approach may need to be designed around the automotive requirements. *Figure 4.3* illustrates the output of each of the evaluated feature detectors.

Before focusing on alternative feature detection techniques, an attempt was also made to evaluate a slightly different implementation of the MSER algorithm as described by Forssen [16]. This technique, known as MSCR (Maximally Stable Colour Regions for Recognition and Matching) adapts the original MSER approach to use colour and as a result produces more accurate results. However, after evaluating Forssen's implementation it became apparent that the performance of this technique does not fulfill the real-time requirements of this project. An example of the MSCR algorithm being applied to an automotive scene is illustrated in *Figure 4.4*.

4.1.2 Feature detection - Alternative approaches

The field of large-scale visual SLAM and visual odometry [17–20] has made a significant contribution to the area of feature detection for both structured and unstructured environments. Specifically, the CenSurE (Center Surround Extremas for Real-time Feature Detection and Matching) approach solves the problem of low feature repeatability between frames and compares its performance to standard detectors such as SIFT [2], FAST [21] and Harris [22]. Such a feature detector has been used for 3D scene reconstruction using stereo-vision, however there is no information about whether the feature density is sufficient for monocular SLAM in automotive environments. This approach has not yet been evaluated due to lack of available source-code or executable, whereas the risk of making a full implementation was high for the current project plan.

4.1.3 A novel real-time feature detection approach

The evaluation of state-of-the-art feature-detectors proved that current approaches cannot combine high repeatability with high correspondence of features in automotive environments. Subsequently, the development of a new methodology is justified in order to allow accurate measurement of motion vectors before performing 3D scene reconstruction. This section describes a novel approach to feature detection and tracking as inspired by the ability of swarms of various living organisms to move in groups without crashing into each other [23]. The proposed technique is presented

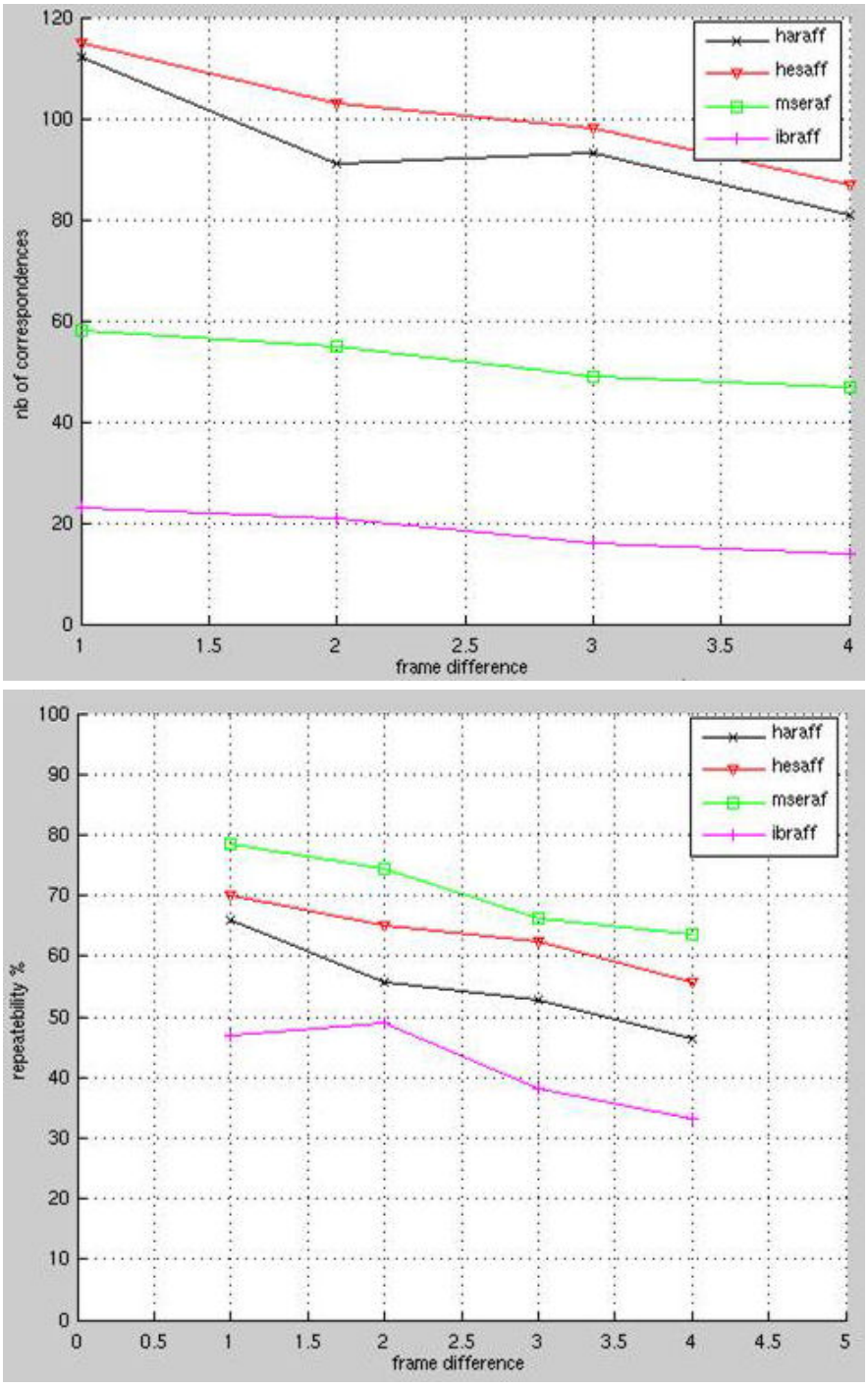


Figure 4.2: Measuring the performance of four different feature detectors by means of correspondence and repeatability.

Figure 4.3: The feature detection output for: 1) Harris affine, 2) Hessian Affine, 3) MSER, 4) IBR.

Figure 4.4: The MSCR feature detection algorithm input and output

in three stages.

4.1.3.1 Stage 1: Pyramidal Processing for identifying areas of interest

Describing an image feature requires the clustering of prime image elements into groups that share similar properties. Thus the first step includes identifying which image elements are suitable for clustering. For example, running a standard Sobel edge detection algorithm on a series of raw images shows that edges are significantly affected by noise and image blur and as a result any edge-based features are also going to be unstable. Thorough experiments proved that most approaches are not effective when using the unfiltered raw input (also verified by Agrawal and Konolige [18,19]), but show improved performance when using higher levels of a Gaussian pyramid. This is a well established noise-filtering technique which has a blurring effect on the image. In the SIFT feature detection algorithm [2], the Difference of Gaussian (DoG) has successfully been used to describe image features by combining successive pyramid levels as illustrated in *Figure 4.5*. Our feature detection approach, is based on Lowe's methodology but introduces some effective changes that further improve its repeatability and number of feature correspondences as analysed below:

1. **Excluding level-0 of the Gaussian pyramid**: Level-0 of the Gaussian pyramid is essentially the raw input image. In this technique, this level is only used to derive the level-1 of the pyramid and is subsequently discarded. Depending on the noise level of the input more Gaussian pyramid levels may need to be excluded. The results show significant stabilisation in feature extraction.

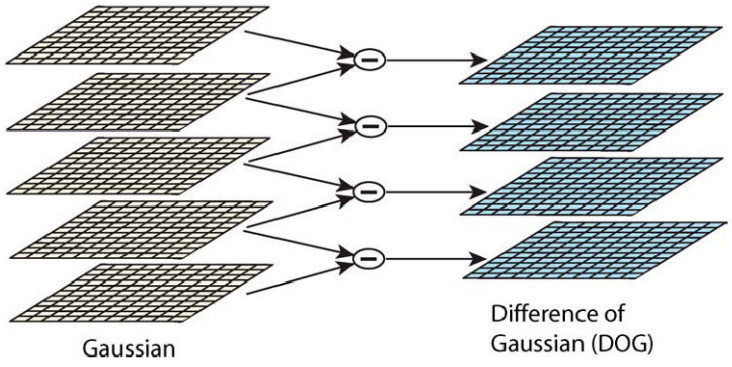


Figure 4.5: Difference of Gaussian as used in the SIFT feature detection algorithm. Courtesy of David G. Lowe [2].

2. **Difference of Gaussian Output (DoG) with high-pass filter:** As *Figure 4.7* illustrates all the levels of the Gaussian pyramid are scaled to create a set of matrices with the same dimensions. Then the absolute DoG between two successive pyramid levels is calculated in order to identify feature-rich areas. In the output matrix, non-zero values correspond to potential feature points, whereas zero-values indicate smooth image areas that are not significantly altered by the blurring effect. Starting with n pyramid levels produces $n-1$ DoG matrices, each of which is thresholded. In the thresholded images white pixels indicate feature-rich areas. Subsequently, the $n-1$ thresholded DoG matrices are logically combined into a single matrix using an OR operator as shown in *Figure 4.7*. Furthermore, the second image of *Figure 4.6* shows an example of the output matrix.

3. **Allocating colour properties to feature points:** Each white pixel in the second image of *Figure 4.6* could potentially group with other pixels to form a feature. However, for features to be formed they need to share some characteristic properties. In this case, the colour footprint of each feature point in the first-level of the Gaussian pyramid is used for feature-point clustering as illustrated in the third image of *Figure 4.6*.

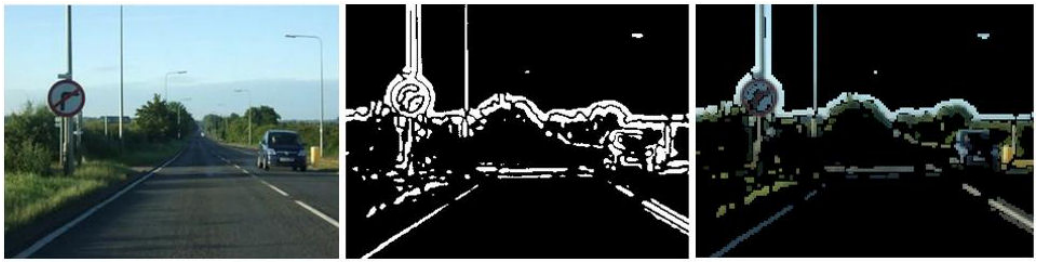


Figure 4.6: Identification of feature-rich areas by pyramidal processing.

4.1.3.2 Stage 2: Pyramidal Processing for feature extraction

Once the areas of interest in an image have been identified, the next step is to extract the actual features from the image. The approach of Stage 1 is repeated with minor changes as illustrated in *Figure 4.7*. These changes are:

1. The input to the Gaussian pyramid is the output of Stage 1 and not the raw input image.

2. The DoG is calculated without a threshold.

3. The output of the DoG between pyramid levels is summed up to create an accumulative matrix, which forms the output of Stage 2. In this matrix, each value represents **the footprint of each feature point** on the Gaussian pyramid. Practically, this pyramidal footprint characterises a feature point more accurately than its actual colour value in the raw input image.

Note that in *Figure 4.7* only one information channel is shown, however in practice we repeat the same procedure for each colour channel separately. For the sake of simplicity we are using RGB channels, however, these can be substituted by any other number or type of channels, such as HSL, YCbCr or even texture.

4.1.3.3 Stage 3: Feature matching & tracking

In Stage 2, the pyramidal footprint of each feature point was calculated. The next step is to cluster similar feature points into groups in order to register them into the features space, as shown in *Figure 4.7*. The sequence of events is the following:

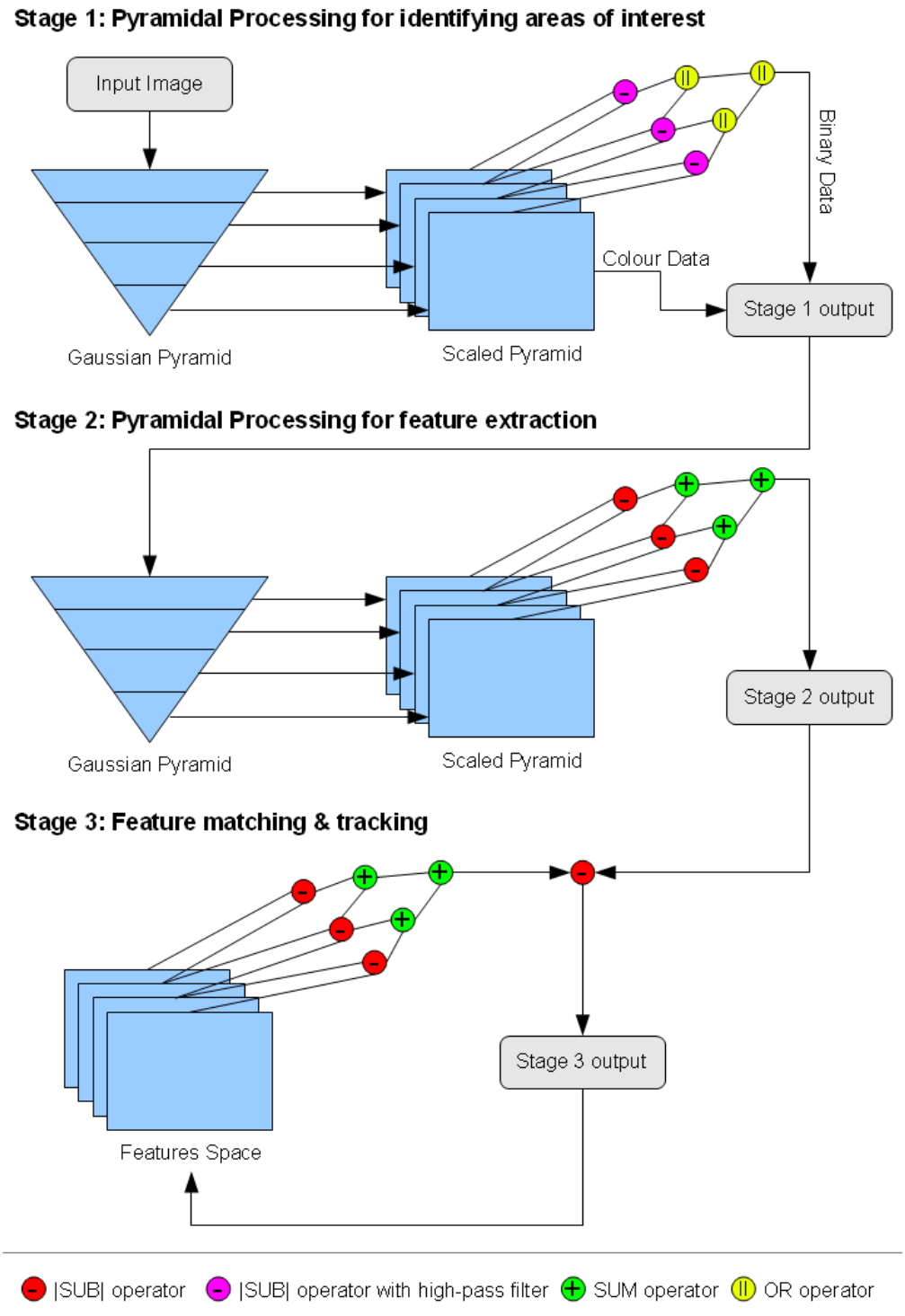


Figure 4.7: Overview of feature detection and tracking methodology using pyramidal processing

- The pyramidal footprint of each feature point is compared to the database of features. (Please note that this is a localised search within a window of predefined size, e.g. 5x5 pixels).
  - If the similarity score exceeds a certain threshold then the feature point joins the feature.
    - The parameters of the feature are updated to include the new feature point. These parameters are:
      - Feature dimensions
      - Pyramidal footprint of the feature
      - Age (i.e. number of frames for which the feature has existed)
  - Else if a sufficiently similar feature cannot be found in the features space, then a new feature is initialised.
- This process repeats every new frame that gets processed, while a history of the location and characteristics of all features is kept in the memory buffer.
- A feature is considered initialised if it has been detected consistently for a certain number of frames (in this case 3).
  - If a feature is not detected in a certain frame, then its age gets decremented.
    - If the age of a feature is zero, then it gets deleted from the features space.
- The motion vector of a feature is the vector connecting the current (x, y) coordinate of the feature to the (x, y) coordinate of the feature in the last frame.

The output of this stage consists of a list of features with allocated motion vectors as illustrated in *Figures 4.8 and 4.9*. Although a full evaluation of this technique against other state-of-the-art methodologies has not yet been completed, our initial results show a high degree of accuracy even using a standard webcam. The

importance of these results is further enhanced if we consider that no predictions are made about the future position of the features. Using techniques such as a Kalman filter [24] could further increase the reliability of this feature detection and tracking approach. Finally, this algorithm can also be used as a real-time optical flow technique [25–29].

4.1.3.4 Stage 4: 3D reconstruction

Once the features and their motion vectors have been derived, the next step is to reconstruct the 3D scene. This work is still in progress at the time of writing, thus only some preliminary results are presented. In this methodology, we start with the assumption that there are no moving obstacles in the scene and that the vehicle is moving in a straight line at constant speed. Based on this assumption, it can be stated that for every forward movement of the vehicle, each of the detected features is getting closer by an amount which is proportional to the vehicle's movement. Due to the perspective effect, features closer to the vehicle appear to be approaching faster than features further away. Although, this is not how the final algorithm is going to estimate depth, it is sufficient for demonstrating the accuracy of the motion vectors derived in Stage 3. In *Figures 4.8 and 4.9* a range of examples are illustrated in the form of depth maps. In each depth map the brighter pixels represent features that are closer to the vehicle and vice-versa. Overall, the feature density, the motion vectors and the quality of the depth maps are reasonably good for detecting different types of objects such as cars, lorries, lamp posts, low bridges and post signs. Of course, there is still a lot of work to be done on compensating vehicle vibration as well as predicting depth when the vehicle is not moving in a straight line. As mentioned above, this is still work in progress, but overall the preliminary results appear to be promising.

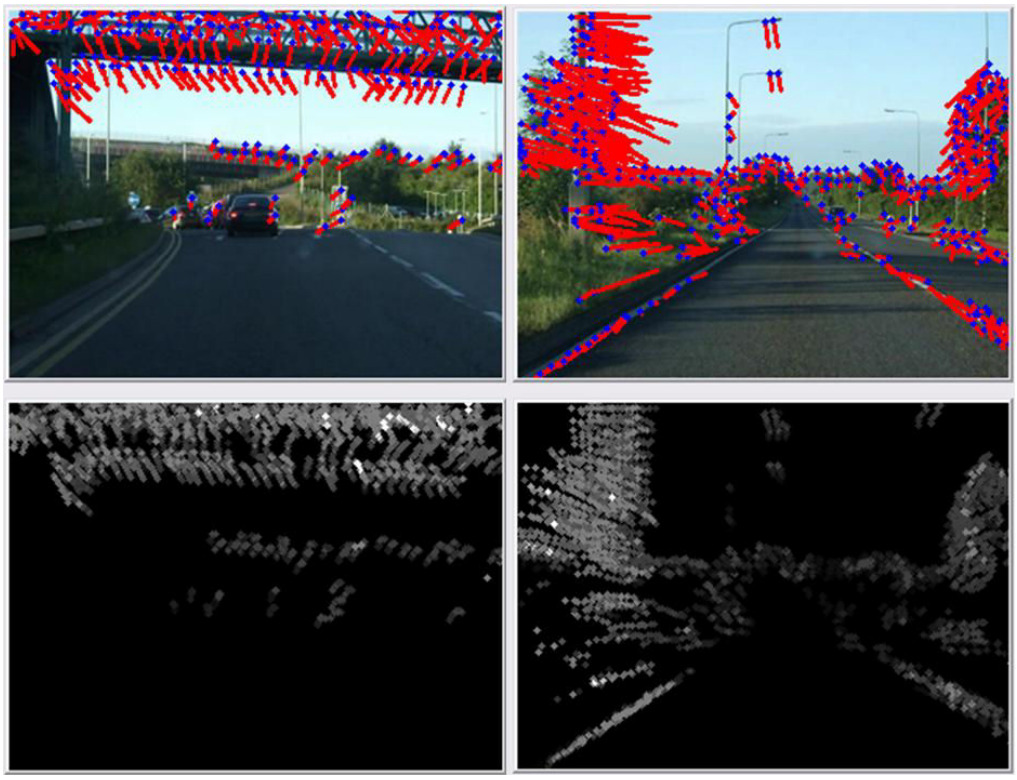


Figure 4.8: Results of feature detection and tracking as well as 3D reconstruction of the scene. In the depth map brighter pixels are closer to the vehicle.

4.2 Methodology B - Evaluation of monocular SLAM in an automotive environment with moving obstacles

Initially, the aim of this methodology was to evaluate the MonoSLAM approach in dynamic environments. However, since Methodology A could not get satisfactory results when evaluating MonoSLAM in static environments, this part of the methodology has been withdrawn. Three-dimensional scene reconstruction in dynamic environments is still going to be implemented as part of Methodology D.

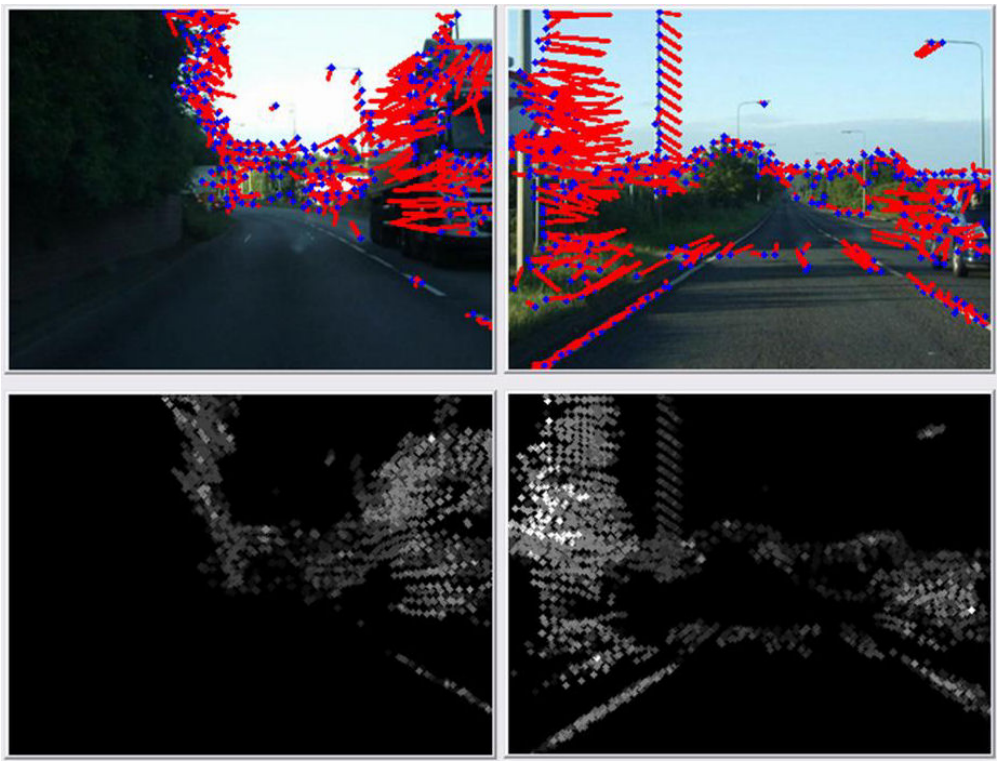


Figure 4.9: More results of feature detection, tracking and derived depth map.

4.3 Methodology C - Removing lighting artifacts

A methodology for eliminating lighting artifacts, such as shadows and light reflections, has been published in the ICVS '09 conference [30]. The paper title is: "*Real-time traversable surface detection by colour space fusion and temporal analysis*" (see *Chapter* B).

4.4 Methodology D - Managing moving obstacles

This methodology, has not yet been implemented. See the project plan in Chapter 5 for more details.

4.5 Methodology E - Terrain Classification Algorithm

A terrain classification algorithm has already been implemented as described in [1] and published in ICVS '09 conference (see *Chapter* B).

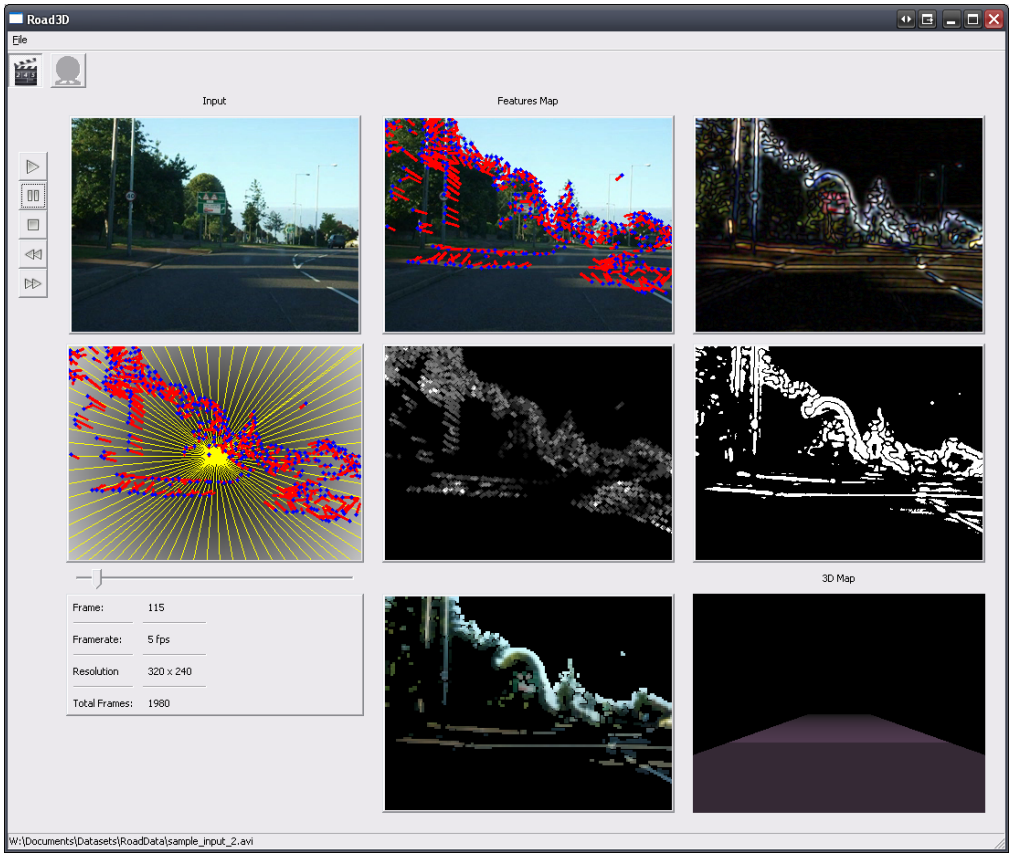


Figure 4.10: Graphical User Interface

4.6 GUI Implementation

A good graphical user interface facilitates software evaluation and is particularly useful in the field of image processing, where visualisation is an essential part of the development lifecycle. As part of this project a cross-platform GUI has been implemented as shown in *Figure 4.10*. The major challenge in designing this GUI was the integration of diverse tools such as OpenCV, OpenGL and wxWidgets into a single

development environment that is operating system independent. This challenge has successfully been completed.

4.7 Conclusions

This chapter described a novel approach to feature detection and tracking in automotive environments and presented some preliminary results on performing 3D scene reconstruction using a monocular camera. Firstly, the MonoSLAM methodology was evaluated in automotive scenarios. The conclusion of this evaluation was that feature detection and tracking is the bottleneck of most SLAM techniques, when they are applied to noisy and dynamic environments. Based on this observation a range of state-of-the-art feature detection techniques was evaluated for their performance in automotive environments, before designing a new real-time technique that seems to outperform the existing state-of-the-art in both performance and accuracy. A quantitative evaluation between our technique and other relevant methodologies is currently underway.

Bibliography

[1] I. Katramados, "9-month progress report: Real-time obstacle detection for low-cost automotive systems using monocular vision," tech. rep., Cranfield University, December 2008.

[2] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, pp. 91–110, 2004.

[3] A. Davison, I. Reid, N. Molton, and O. Stasse, "Monoslam: Real-time single camera slam," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, pp. 1052–1067, June 2007.

[4] A. J. Davison, "Reserach website, "http://www.doc.ic.ac.uk/ ajd/", [last visited on: 20/11/2008]."

[5] R. Cucchiara, C. Grana, M. Piccardi, A. Prati, and S. Sirotti, "Improving shadow suppression in moving object detection with hsv color information," *IEEE Intelligent Transportation Systems*, pp. 334–339, 2001.

[6] J. Wu, Z. Yang, J. Wu, and A. Liu, "Virtual line group based video vehicle detection algorithm utilizing both luminance and chrominance," *2nd IEEE Conference on Industrial Electronics and Applications (ICIEA)*, pp. 2854–2858, May 2007.

[7] D. Hahnel, R. Triebel, W. Burgard, and S. Thrun, "Map building with mobile robots in dynamic environments," *IEEE International Conference on Robotics and Automation, 2003. Proceedings. ICRA '03*, vol. 2, pp. 1557–1563 vol.2, Sept. 2003.

[8] C.-C. Wang, C. Thorpe, and S. Thrun, "Online simultaneous localization and mapping with detection and tracking of moving objects: theory and results from a ground vehicle in crowded urban areas," *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, vol. 1, pp. 842–849 vol.1, Sept. 2003.

[9] M. of Defence, "Mod grand challenge (2008)," tech. rep., http://www.challenge.mod.uk/, [Last visited: 15/06/2009].

[10] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool, "A comparison of affine region detectors," *Int. J. Comput. Vision*, vol. 65, no. 1-2, pp. 43–72, 2005.

[11] K. Mikolajczyk and C. Schmid, "Scale & affine invariant interest point detectors," *Int. J. Comput. Vision*, vol. 60, pp. 63–86, October 2004.

[12] J. Matas, O. Chum, M. Urban, and T. Pajdla, "Robust wide baseline stereo from maximally stable extremal regions," vol. 1, pp. 384–393, 2002.

[13] M. Donoser and H. Bischof, "Efficient maximally stable extremal region (mser) tracking," in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, vol. 1, pp. 553–560, June 2006.

[14] T. Tuytelaars and L. Van Gool, "Matching widely separated views based on affine invariant regions," *Int. J. Comput. Vision*, vol. 59, pp. 61–85, August 2004.

[15] T. Kadir, A. Zisserman, and M. Brady, "An affine invariant salient region detector," vol. 3021, pp. 228–241, 2004.

[16] P.-E. Forssen, "Maximally stable colour regions for recognition and matching," in *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pp. 1–8, June 2007.

[17] B. Morisset, R. Rusu, A. Sundaresan, K. Hauser, M. Agrawal, J. Latombe, and M. Beetz, "Leaving flatland. toward real-time 3d navigation," *IEEE International Conference on Robotics and Automation*, 2009.

[18] M. Agrawal, K. Konolige, and M. R. Blas, "Censure: Center surround extremas for realtime feature detection and matching," in *ECCV (4)*, pp. 102–115, 2008.

[19] M. A. Konolige, K. and J. Sola., "Large-scale visual odometry for rough terrain," in *International Symposium on Research in Robotics, Hiroshima , Japan*, 2007.

[20] K. Konolige and M. Agrawal, "Frameslam: From bundle adjustment to real-time visual mapping," *Robotics, IEEE Transactions on*, vol. 24, pp. 1066–1077, Oct. 2008.

[21] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," vol. 3951, pp. 430–443, 2006.

[22] C. Harris and M. Stephens, "A combined corner and edge detector," in *Alvey Vision Conference*, pp. 147–151, 1988.

[23] M. Adioui, J. Treuil, and O. Arino, "Alignment in a fish school: a mixed lagrangian-eulerian approach," *Ecological Modelling*, vol. 167, no. 1-2, pp. 19 – 32, 2003.

[24] Y. Yoon, A. Kosaka, and A. Kak, "A new kalman-filter-based framework for fast and accurate visual tracking of rigid objects," *Robotics, IEEE Transactions on*, vol. 24, pp. 1238–1251, Oct. 2008.

[25] T. Low and G. Wyeth, "Obstacle detection using optical flow," *Proceedings of the 2005 Australasian Conference on Robotics & Automation*, 2005.

[26] C. Braillon, C. Pradalier, J. Crowley, and C. Laugier, "Real-time moving obstacle detection using optical flow models," *Intelligent Vehicles Symposium, 2006 IEEE*, pp. 466–471, June 2006.

[27] G. Desouza and A. Kak, "Vision for mobile robot navigation: a survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 2, pp. 237–267, 2002.

[28] M. Yeasin, "Optical flow in log-mapped image plane - a new approach," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, pp. 125–131, Jan 2002.

[29] G.-S. Young, T.-H. Hong, M. Herman, and J. Yang, "Obstacle detection for a vehicle using optical flow," *Intelligent Vehicles '92 Symposium., Proceedings of the*, pp. 185–190, Jun-1 Jul 1992.

[30] I. Katramados, S. Crumpler, and T. P. Breckon, "Real-time traversable surface detection by colour space fusion and temporal analysis," *Lecture Notes in Computer Science (LNCS)*, 2009.

Chapter 5

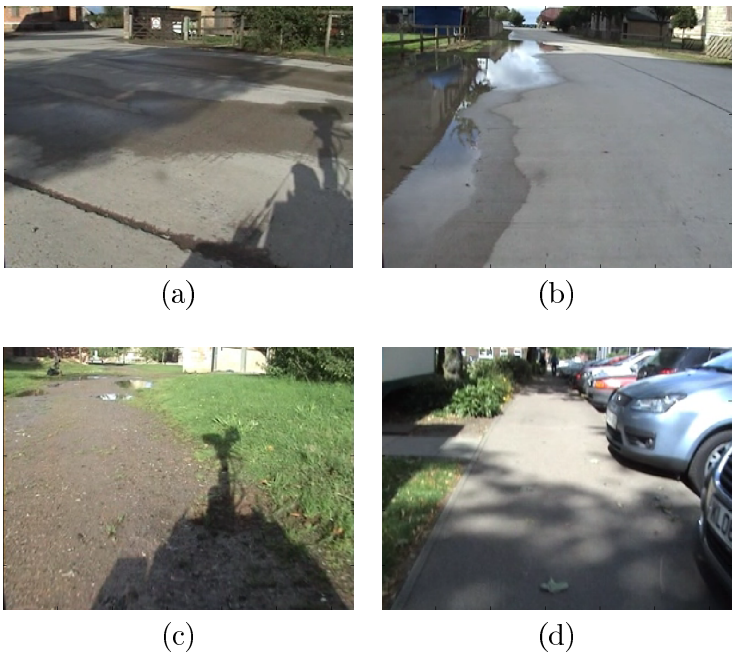
Implementation and Results

This chapter presents the results of the work to date based on the methodology of *Chapter 4*. Specifically, the results of implementing a terrain classification algorithm (*Methodology E)* are described, which was developed out of sequence in order to meet the deadlines of the MoD Grand Challenge competition [89]. The following sections describe a novel implementation of this methodology and evaluate its performance.

5.1 Methodology E - Terrain Classification Algorithm

Designing a terrain classification algorithm requires the definition of a set of feature vectors that can be used to uniquely characterise different surface types. In other words, this can be considered as a segmentation problem, where areas with similar characteristics are grouped together to form the terrain segment. Furthermore, such an algorithm should be designed to cope with dynamic environments including shadows, reflections and a mixture of wet and dry surfaces. *Figure 5.1* gives some examples of such challenging surface conditions. In addition, *Figure 5.2* describes the core algorithm functionality, which is divided into the following four modules:

- Image Pre-processing
- Segmentation by Histogram Analysis



(a) (b)

(c) (d)

Figure 5.1: Examples of challenging terrain surfaces

- Temporal Filtering and Correlation
- Traversability Map Generation and Terrain Classification

Each of these modules is analysed in the following paragraphs.

5.1.1 Image Pre-processing

Terrain classification and obstacle detection are relevant techniques since non-terrain surfaces are very likely to be obstacles. Thus both approaches use similar computer vision methodologies such as edge detection and segmentation for feature extraction. However, before applying them, the input needs to be pre-processed in order to eliminate noise. In this case, most of the noise originates from lighting artifacts (i.e. shadows/reflections), water artifacts (i.e. wet/dry surfaces), road markings and road clutter. Based on this observation, it became apparent that a methodology had to be adopted for eliminating all those artifacts that are not part of the road surface. Current research [10, 11, 31, 87] has shown that choosing the right colourspace is crucial to extracting accurate light-invariant information. In fact this methodology

combines the *HSL* and *YCbCr* colourspaces to derive four light invariant features namely:

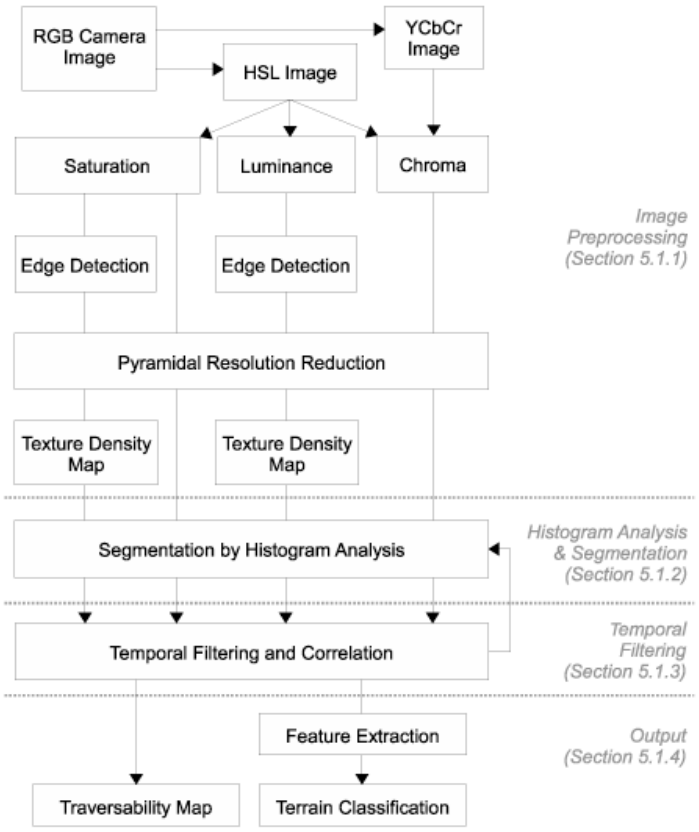


Figure 5.2: Methodology E - Algorithm Overview

- **Saturation** (based on the S channel of the *HSL* colourspace)

  It describes the whiteness or purity of a colour [27,90]. Since the luminance is all embedded in the L channel of the HSL colourspace, the S channel is mostly light-invariant. However, in the case of hard reflections and shadows, the colour properties of a surface may change, which in turn alters the saturation level of that surface. Nevertheless, this change is still small when compared to the H or L channels.

- **Saturation-based texture** (SbT)

  It can be derived by applying an edge detector on the S channel of the *HSL* colourspace. Then the texture is defined as the density of edges in different

parts of the image. Practically, this is achieved by applying pyramidal resolution reduction to the output of the *Sobel* edge detector (*Section 3.1.1*) as illustrated in Figure 5.3.

- **Luminance-based texture** (LbT, based on the L channel of the *HSL* colourspace)

  This is a standard grayscale image, where the Sobel edge detector is applied and texture density is calculated in exactly the same way as in SbT. An example is illustrated in *Figure 5.3*.

- **Chroma** (based on combining the Cb and Cr components of $YCbCr$ colourspace)

  Chroma provides luminance-independent colour information in the $YCbCr$ colourspace. As with the S channel of the *HSL* colourspace, hard shadows and reflections alter the chroma level making their detection difficult. To solve this problem Wu et al. [11] propose the combination of the two chroma components (Cb and Cr) in order to detect features that are entirely light invariant. However, the Cb and Cr components have a relatively small variation range when compared to the Y component. Based on this observation, the Cb and Cr values are scaled to fit the 0 - 255 range and subsequently their mean value is derived. This way all the low-chrominance features are eliminated including shadows, reflections and non-coloured road markings. An example is illustrated in *Figure 5.4*.

Each of these four features is represented by an image array with dimensions equal to those of the input image (in this case 320×240). The next step is to remove random noise that is generated by the imaging sensor. This is achieved by performing pyramidal resolution reduction to derive lower resolution arrays. In these arrays, the value of each element is equal to the average sum of the corresponding pixels at the higher resolution image. At this stage, the full resolution images are no longer required, since all the remaining processing can be performed on the condensed arrays. The next paragraph describes how these features are used to segment the image.

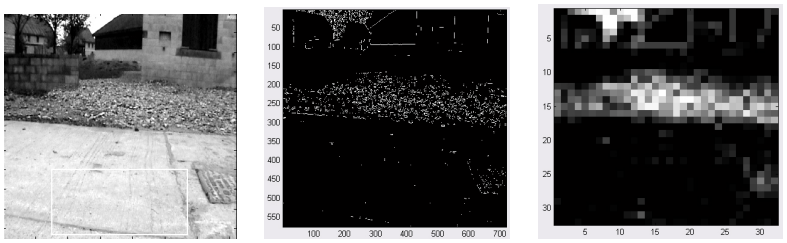


Figure 5.3: Calculating texture density by pyramidal resolution reduction on the output of the *Sobel* edge detector

Figure 5.4: Chroma-based Analysis: Areas with low chrominance are eliminated. Besides the wall on the left of the original image is also eliminated, however this information is later recovered through other channels (e.g. texture).

5.1.2 Segmentation by Histogram Analysis

Several prior terrain detection techniques have been developed around the assumption that the area immediately in front of the vehicle is initially traversable and thus they identify the pathway by comparison to a "safe" window near the bottom of the image [15]. The current approach also adopts this idea. A histogram of the "safe window" is drawn for each of the Saturation, SbT, LbT and Chroma image arrays as illustrated in *Figure 5.5*. Furthermore, the histogram resolution is reduced (using pyramidal reduction) in order to simplify its processing and improve performance. Then, by detecting the histogram peaks, the main image features are extracted based on the assumption that each surface is characterised by its saturation level, chrominance level and texture density. At this point it is also important to note that each peak is considered as a feature with five attached properties, namely:

- **Left histogram peak edge**: The point where the left slope meets the "average level"[1] line

[1]Defined as the mean of all the histogram values

- **Right histogram peak edge**: The point where the right slope meets the "average level" line
- **Histogram peak value**: The peak value of the low-resolution histogram
- **Average segment value**: A segment is defined by the left and right edges of the peak, which are used to derive the average segment value.
- **Age**: Each peak is assigned an age (or confidence level), meaning the time that the peak has remained consistent. A peak is then considered as a valid feature only if its age is above a certain threshold.

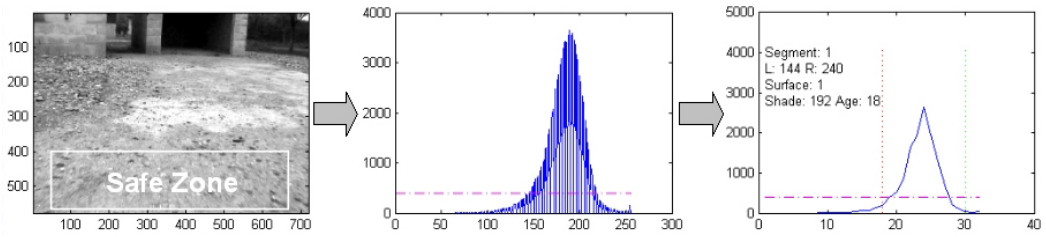


Figure 5.5: Histogram calculation for the area inside the "Safe Zone", followed by pyramidal resolution reduction for identifying the histogram peak. The dashed blue line corresponds to the "average level line" and the vertical coloured lines correspond to the edges of the segment.

The next step is to segment the entire image area below the horizon level. In the histogram of *Figure 5.5*, there is only one main segment thus the image will essentially be thresholded. Each cell of the low-resolution images is provisionally marked as "traversable" or "non-traversable", depending on whether its 8-bit value falls within the histogram segment or not. However, it should be highlighted that more complex surfaces may result in two or more histogram peaks and thus two or more segments. This feature makes the current approach suitable for identifying simple as well as composite traversable surfaces. Finally, the output of segmentation is fed into the temporal memory model as described in the next paragraph.

5.1.3 Temporal Filtering and Correlation

Creating high-level representations of complex raw data can be improved by introducing a temporal memory structure as a way of reducing noise and increasing system accuracy and reliability. This approach proposes the use of temporal behaviour analysis on the output of the segmentation as a top-level filter before correlation. Specifically, the segments identified by histogram analysis in *Section 5.1.2* are tracked over a series of video frames in order to check their consistency. This is done by assigning a confidence level to each type of surface, which adjusts depending on whether a similar surface appears repeatedly or not. In this way, the system compensates for noise and vibrations on a frame-by-frame basis. Similarly, each grid cell of the segmented images is also assigned a confidence level, which increases if its status as "traversable" or "non-traversable" remains unchanged over time. The final output consists of four new "*traversability*" maps based on the Saturation, SbT, LbT and Chroma analysis over time. The next step is to correlate the available information into a single array which fuses the data from the four different maps. At this stage, a traversable area is simply defined as the intersection of the individual traversable areas as derived from each of the four images. Although, more sophisticated techniques could have been implemented, this specific one was preferred as the best compromise between robustness and real-time performance. *Figure 5.6* summarises this approach.

5.1.4 Traversability Map and Terrain Classification

The output of *Section 5.1.3* is a "*traversability*" map which indicates the likelihood of a surface being drivable. Four different levels of traversability are possible for each grid cell. *Figure* 5.7 illustrates such examples, where the darker shades of gray indicate less traversable areas. The next step is to identify the terrain type (e.g. tarmac, soil, paved pathway, concrete). As mentioned in *Section 5.1.2*, each surface is characterised by a set of features, which can be used to train a classifier on a database of common terrain types. Subsequently, the features of each new terrain are extracted and compared to the database before picking the surface with the

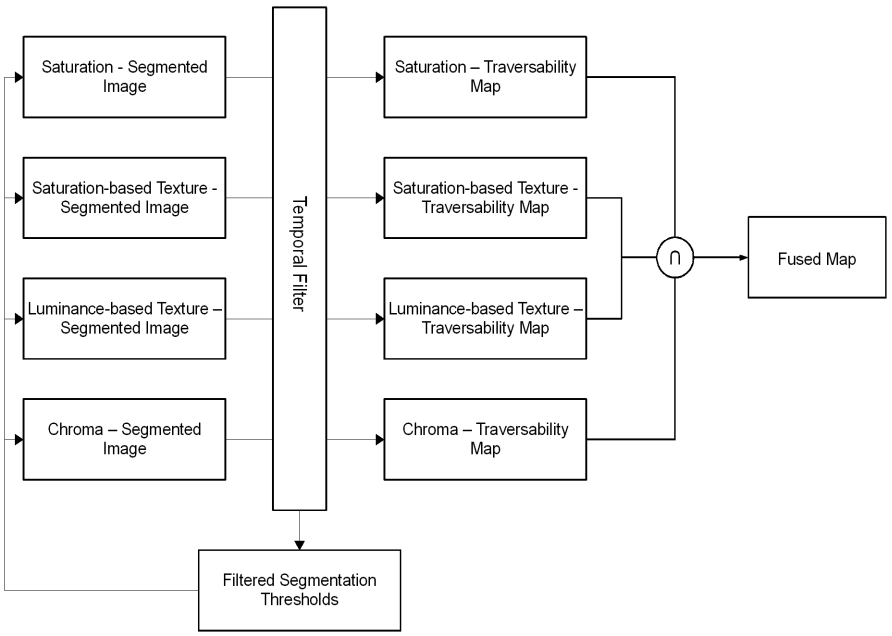


Figure 5.6: Temporal Filtering Process

highest similarity index. Visually, the feature vectors are represented by a spider diagram as illustrated in *Figure 5.8*. Each axis in this diagram corresponds to one of the surface features namely:

- Saturation
- TextureS: Saturation-based Texture (SbT)
- TextureV: Luminance-based Texture (LbT)
- Chroma

Each of these features is mentioned with a H, L or R subscript character, where:

- H: histogram peak value
- L: left histogram peak edge
- R: right histogram peak edge

Although the spider diagrams of different surfaces may look similar at first sight, a closer investigation reveals significant variations that enable robust classification.

5.2 Results

The approach presented in this chapter has been evaluated using video data captured from an unmanned ground vehicle under a wide range of weather and lighting conditions, on different terrain types. Although a quantitative analysis of the performance of this methodology has not been completed yet, the algorithm has generally been robust in predicting the traversability of an area regardless of the image quality from the camera and the vibrations. *Figure 5.8* provides some characteristic examples of the system output for different types of terrain.

The evaluation was done using a standard consumer camcorder and a computer notebook, where the algorithm was able to execute in real-time at a frame rate of 25 frames per second. Additionally, the processing overhead was low meaning that the same platform could be used for performing more autonomous navigation related tasks such as vehicle control and data fusion. Obstacles could also be detected as non-traversable areas except in situations were they were indistinguishable from the underlying surface. Regarding changing weather and lighting conditions, the performance was good, although reflections were sometimes detected as non-traversable areas. However, this is a weakness that could be resolved through data fusion with other sensing technologies.

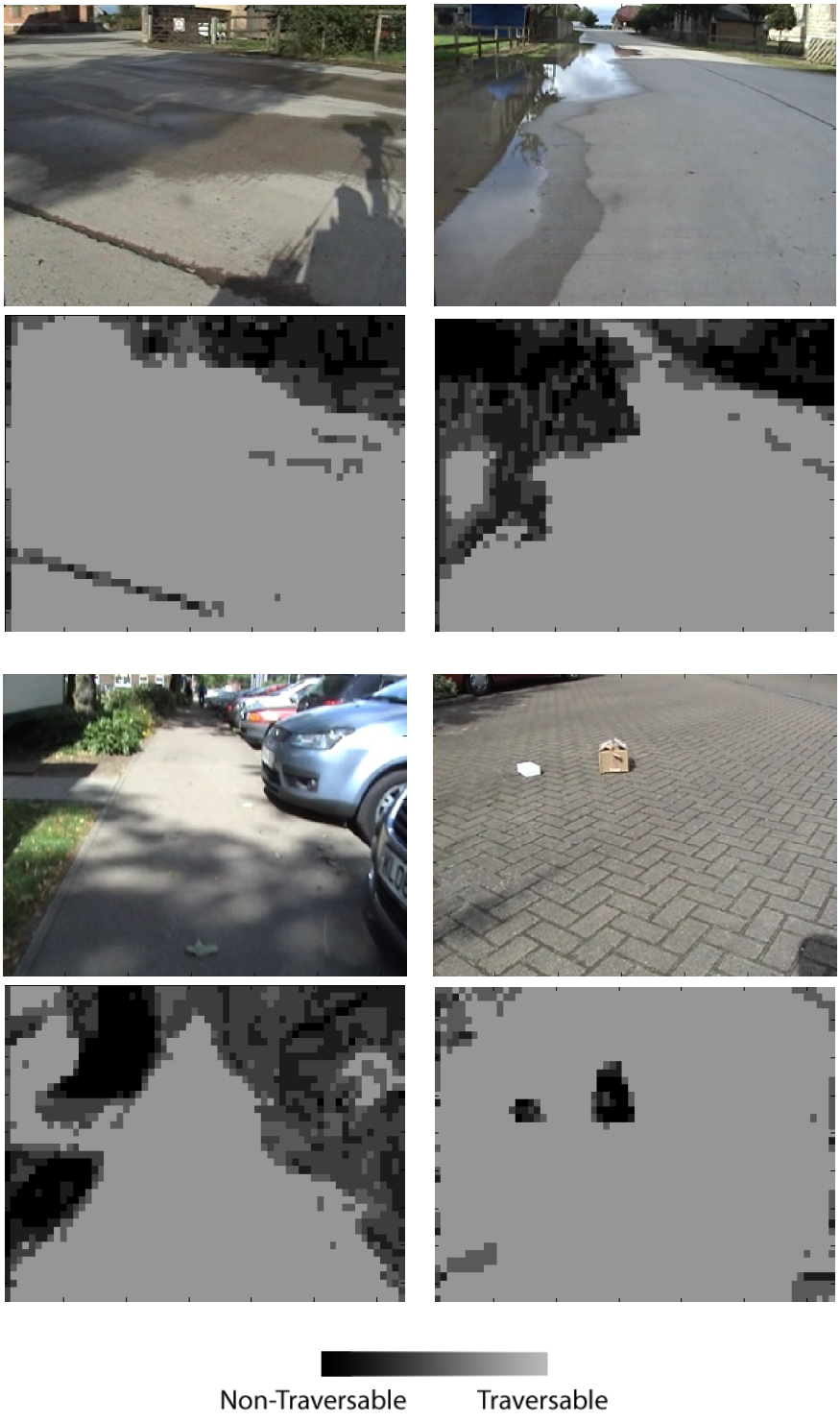


Figure 5.7: Segmentation results after temporal filtering

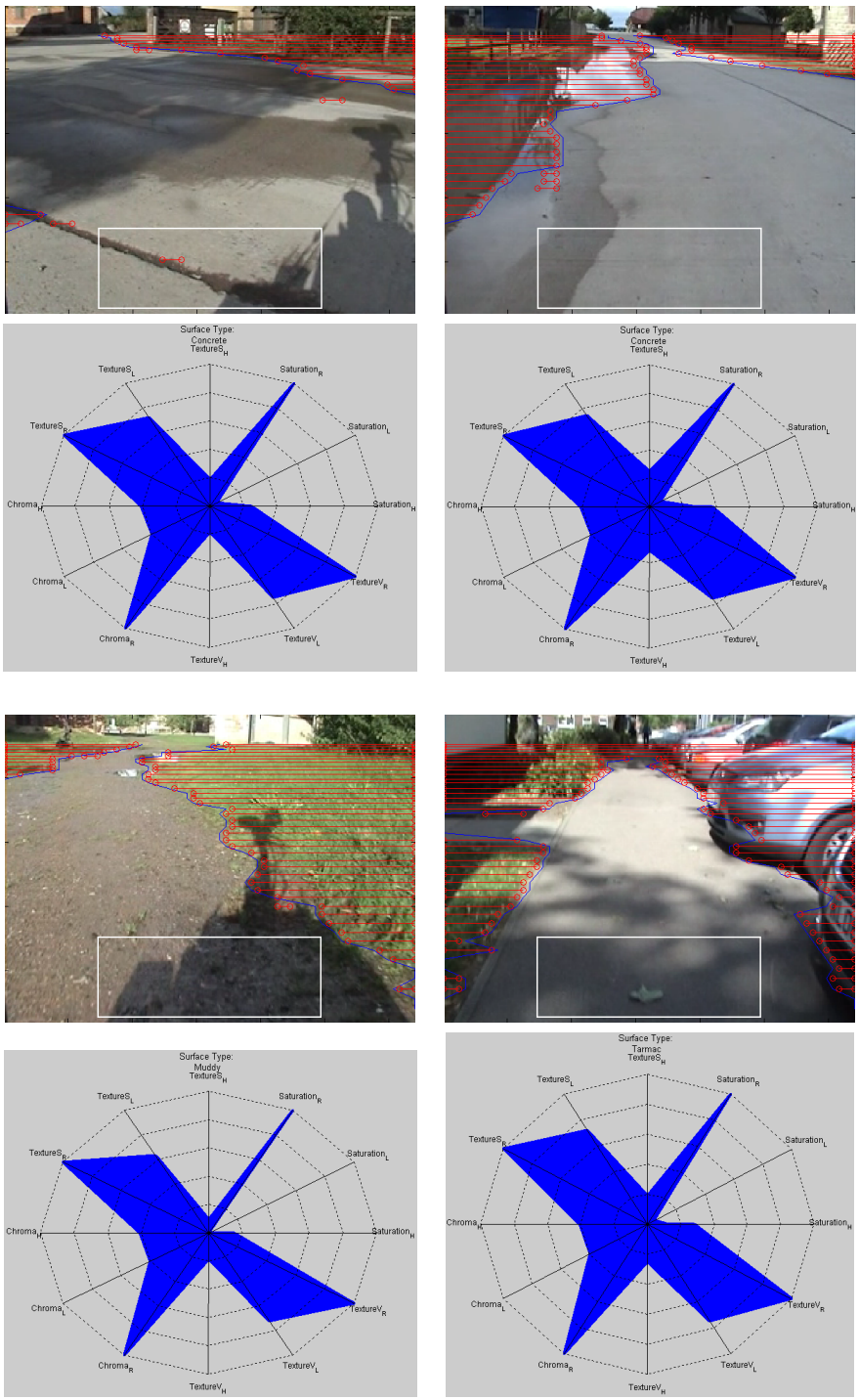


Figure 5.8: Traversable Terrain Detection & Classification (*white rectangle outlines the area used as the sample for terrain detection & classification; blue line indicates terrain boundaries; red lines indicate off-road areas*)

Bibliography

[1] Mobileye, "Driver assistance vision applications, vehicle, pedestrian, lane detection," *http://www.mobileye-vision.com/default.asp?PageID=202, [Last visited on: 10/04/2008].*

[2] V. Kastrinaki, M. Zervakis, and K. Kalaitzakis, "A survey of video processing techniques for traffic applications," *Image and Vision Computing*, vol. 21, pp. 359–381, April 2003.

[3] P. Newman, J. Leonard, J. Tardos, and J. Neira, "Explore and return: experimental validation of real-time concurrent mapping and localization," *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on*, vol. 2, pp. 1802–1809 vol.2, 2002.

[4] M. Bosse, P. Newman, J. Leonard, M. Soika, W. Feiten, and S. Teller, "An atlas framework for scalable mapping," *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, vol. 2, pp. 1899–1906 vol.2, Sept. 2003.

[5] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part i," *Robotics & Automation Magazine, IEEE*, vol. 13, pp. 99–110, June 2006.

[6] T. Bailey and H. Durrant-Whyte, "Simultaneous localization and mapping (slam): part ii state of the art," *Robotics & Automation Magazine, IEEE*, vol. 13, pp. 108–117, Sept. 2006.

[7] A. Davison, I. Reid, N. Molton, and O. Stasse, "Monoslam: Real-time single camera slam," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, pp. 1052–1067, June 2007.

[8] D. Hahnel, R. Triebel, W. Burgard, and S. Thrun, "Map building with mobile robots in dynamic environments," *IEEE International Conference on Robotics and Automation, 2003. Proceedings. ICRA '03*, vol. 2, pp. 1557–1563 vol.2, Sept. 2003.

[9] C.-C. Wang, C. Thorpe, and S. Thrun, "Online simultaneous localization and mapping with detection and tracking of moving objects: theory and results from a ground vehicle in crowded urban areas," *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, vol. 1, pp. 842–849 vol.1, Sept. 2003.

[10] R. Cucchiara, C. Grana, M. Piccardi, A. Prati, and S. Sirotti, "Improving shadow suppression in moving object detection with hsv color information," *Intelligent Transportation Systems, 2001. Proceedings. 2001 IEEE*, pp. 334–339, 2001.

[11] J. Wu, Z. Yang, J. Wu, and A. Liu, "Virtual line group based video vehicle detection algorithm utilizing both luminance and chrominance," *Industrial Electronics and Applications, 2007. ICIEA 2007. 2nd IEEE Conference on*, pp. 2854–2858, May 2007.

[12] A. Talukder, R. Manduchi, R. Castano, K. Owens, L. Matthies, A. Castano, and R. Hogg, "Autonomous terrain characterisation and modelling for dynamic control of unmanned vehicles," *Intelligent Robots and System, 2002. IEEE/RSJ International Conference on*, vol. 1, pp. 708–713 vol.1, 2002.

[13] P. Jansen, W. van der Mark, J. van den Heuvel, and F. Groen, "Colour based off-road environment and terrain type classification," *Intelligent Transportation Systems, 2005. Proceedings. 2005 IEEE*, pp. 216–221, 13-15 Sept. 2005.

[14] R. Manduchi, A. Castano, A. Talukder, and L. Matthies, "Obstacle detection and terrain classification for autonomous off-road navigation," *Auton. Robots*, vol. 18, no. 1, pp. 81–102, 2005.

[15] C. Thorpe, M. Hebert, T. Kanade, and S. Shafer, "Vision and navigation for the carnegie-mellon navlab," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 10, pp. 362–373, May 1988.

[16] M. Bertozzi, A. Broggi, and A. Fascioli, "Vision-based intelligent vehicles: State of the art and perspectives," *Robotics and Autonomous Systems*, vol. 32, pp. 1–16, July 2000.

[17] G. Desouza and A. Kak, "Vision for mobile robot navigation: a survey," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, pp. 237–267, Feb 2002.

[18] DARPA, "Urban grand challenge (2007)," tech. rep., http://www.darpa.mil/grandchallenge, [Last visited: 15/04/2008].

[19] A. Broggi, C. Caraffi, P. Porta, and P. Zani, "The single frame stereo vision system for reliable obstacle detection used during the 2005 darpa grand challenge on terramax," *Intelligent Transportation Systems Conference, 2006. ITSC '06. IEEE*, pp. 745–752, 2006.

[20] A. Nefian and G. Bradski, "Detection of drivable corridors for off-road autonomous navigation," *Image Processing, 2006 IEEE International Conference on*, pp. 3025–3028, 8-11 Oct. 2006.

[21] Y. Alon, A. Ferencz, and A. Shashua, "Off-road path following using region classification and geometric projection constraints," *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, vol. 1, pp. 689–696, 17-22 June 2006.

[22] S. Goldberg, M. Maimone, and L. Matthies, "Stereo vision and rover navigation software for planetary exploration," *Aerospace Conference Proceedings, 2002. IEEE*, vol. 5, pp. 5–2025–5–2036 vol.5, 2002.

[23] TRW-Conekt, "Lane departure warning system," *http://www.conekt.net/*, [Last visited: 23/05/2008].

[24] P.-Y. Hsiao and C.-W. Yeh, "A portable real-time lane departure warning system based on embedded calculating technique," *Vehicular Technology Conference, 2006. VTC 2006-Spring. IEEE 63rd*, vol. 6, pp. 2982–2986, 7-10 May 2006.

[25] M. Nixon and A. Aguado, *Feature Extraction & Image Processing*. Butterworth Heinmann Newnes, January 2002.

[26] R. Marfil, L. Molina-Tanco, A. Bandera, J. A. Rodriguez, and F. Sandoval, "Pyramid segmentation algorithms revisited," *Pattern Recogn.*, vol. 39, no. 8, pp. 1430–1451, 2006.

[27] B. Jahne, *Digital Image Processing (5th Edition)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2002.

[28] C. Rasmussen and T. Korah, "On-vehicle and aerial texture analysis for vision-based desert road following," *Computer Vision and Pattern Recognition, 2005 IEEE Computer Society Conference on*, pp. 66–66, June 2005.

[29] M. Tuceryan, "Moment based texture segmentation," *Pattern Recognition, 1992. Vol.III. Conference C: Image, Speech and Signal Analysis, Proceedings., 11th IAPR International Conference on*, pp. 45–48, 30 Aug-3 Sep 1992.

[30] M. Tuceryan and A. K. Jain, *Texture Analysis, In The Handbook of Pattern Recognition and Computer Vision (2nd Edition)*, pp. 207–248. World Scientific Publishing Co., 1998.

[31] L. Lorigo, R. Brooks, and W. Grimsou, "Visually-guided obstacle avoidance in unstructured environments," *Intelligent Robots and Systems, 1997. IROS '97., Proceedings of the 1997 IEEE/RSJ International Conference on*, vol. 1, pp. 373–379 vol.1, 7-11 Sep 1997.

[32] P. Batavia and S. Singh, "Obstacle detection in smooth high curvature terrain," *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on*, vol. 3, pp. 3062–3067, 2002.

[33] R. Manduchi, "Bayesian fusion of color and texture segmentations," *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, vol. 2, pp. 956–962 vol.2, 1999.

[34] V. Marion, "Method of color image processing to eliminate shadows and reflections," *Word Intellectual Property Organisation*, vol. WO 2004/027710 A1, 2002.

[35] C. Fredembach and G. Finlayson, "Simple shadow removal," *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, vol. 1, pp. 832–835, 0-0 2006.

[36] A. Prati, I. Mikic, M. Trivedi, and R. Cucchiara, "Detecting moving shadows: algorithms and evaluation," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 25, pp. 918–923, July 2003.

[37] L. Xu, F. Qi, and R. Jiang, "Shadow removal from a single image," *Intelligent Systems Design and Applications, 2006. ISDA '06. Sixth International Conference on*, vol. 2, pp. 1049–1054, Oct. 2006.

[38] X. Tao, M. Guo, and B. Zhang, "A neural network approach to the elimination of road shadow for outdoor mobile robot," *Intelligent Processing Systems, 1997. ICIPS '97. 1997 IEEE International Conference on*, vol. 2, pp. 1302–1306 vol.2, Oct 1997.

[39] K. Dickinson and C. Wan, "Road traffic monitoring using the trip ii system," *Road Traffic Monitoring, 1989., Second International Conference on*, pp. 56–60, Feb 1989.

[40] A. Houghton, G. Hobson, N. Seed, and R. Tozer, "Automatic vehicle recognition," *Road Traffic Monitoring, 1989., Second International Conference on*, pp. 71–78, Feb 1989.

[41] R. Siegwart and I. R. Nourbakhsh, *Introduction to Autonomous Mobile Robots*. New York, NY, USA: MIT Press, 2004.

[42] C. Wohler and J. Anlauf, "An adaptable time-delay neural-network algorithm for image sequence analysis," *Neural Networks, IEEE Transactions on*, vol. 10, pp. 1531–1536, Nov 1999.

[43] P. Batavia and S. Singh, "Obstacle detection using adaptive color segmentation and color stereo homography," *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, vol. 1, pp. 705–710 vol.1, 2001.

[44] D. Feiden and R. Tetzlaff, "Obstacle detection in planar worlds using cellular neural networks," *Cellular Neural Networks and Their Applications, 2002. (CNNA 2002). Proceedings of the 2002 7th IEEE International Workshop on*, pp. 383–390, 22-24 July 2002.

[45] A. Talukder, R. Manduchi, A. Rankin, and L. Matthies, "Fast and reliable obstacle detection and segmentation for cross-country navigation," *Intelligent Vehicle Symposium, 2002. IEEE*, vol. 2, pp. 610–618 vol.2, 17-21 June 2002.

[46] D. Ferguson, M. Darms, C. Urmson, and S. Kolski, "Detection, prediction, and avoidance of dynamic obstacles in urban environments," *Intelligent Vehicles Symposium, 2008 IEEE*, pp. 1149–1154, June 2008.

[47] C. Caraffi, S. Cattani, and P. Grisleri, "Off-road path and obstacle detection using decision networks and stereo vision," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 8, pp. 607–618, Dec. 2007.

[48] G. Zong, L. Deng, and W. Wang, "A method for robustness improvement of robot obstacle avoidance algorithm," *Robotics and Biomimetics, 2006. ROBIO '06. IEEE International Conference on*, pp. 115–119, Dec. 2006.

[49] Z. Xiang, Z. Xu, and J. Liu, "Small obstacle detection for autonomous land vehicle under semi-structural environments," *Intelligent Transportation Systems, 2003. Proceedings. 2003 IEEE*, vol. 1, pp. 293–298 vol.1, 12-15 Oct. 2003.

[50] B.-F. Wu and C.-T. Lin, "A fuzzy vehicle detection based on contour size similarity," *Intelligent Vehicles Symposium, 2005. Proceedings. IEEE*, pp. 496–501, 6-8 June 2005.

[51] A. Broggi, C. Caraffi, S. Cattani, and R. Fedriga, "A decision network based frame-work for visual off-road path detection problem," *Intelligent Transportation Systems Conference, 2006. ITSC '06. IEEE*, pp. 951–956, 2006.

[52] G.-S. Young, T.-H. Hong, M. Herman, and J. Yang, "Obstacle detection for a vehicle using optical flow," *Intelligent Vehicles '92 Symposium., Proceedings of the*, pp. 185–190, Jun-1 Jul 1992.

[53] S. Thrun, "Probabilistic algorithms in robotics," April 2000.

[54] M. Bertozzi, A. Broggi, D. Colla, and R. Fascioli, "Sensing of automotive environments using stereo vision," in *In 30th International Symposium on Automotive Technology and Automation (ISATA), Special Session on Machine Vision and Intelligent Vehicles and Autonomous Robots*, pp. 187–193, 1997.

[55] C. Brenneke, O. Wulf, and B. Wagner, "Using 3d laser range data for slam in outdoor environments," *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, vol. 1, pp. 188–193 vol.1, Oct. 2003.

[56] T. Lemaire, C. Berger, I. Jung, and S. Lacroix, "Vision-based slam: Stereo and monocular approaches," *International Journal of Computer Vision (IJCV)*, vol. 74, pp. 343–364, September 2007.

[57] D. Yuen and B. MacDonald, "An evaluation of the sequential monte carlo technique for simultaneous localisation and map-building," *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, vol. 2, pp. 1564–1569 vol.2, Sept. 2003.

[58] S. Thrun, "Robotic mapping: A survey," in *Exploring Artificial Intelligence in the New Millenium* (G. Lakemeyer and B. Nebel, eds.), Morgan Kaufmann, 2002. to appear.

[59] T. A. V. Calleja, *Visual Navigation and Environment Modeling for Wheeled Mobile Robots*. PhD thesis, Institut de Robòtica i Informàtica Industrial, Universitat Politècnica de Catalunya, 2007.

[60] D. Fox and W. Burgard, "Markov localization for mobile robots in dynamic environments," *Journal of Artificial Intelligence Research*, vol. 11, pp. 391–427, 1999.

[61] A. Davison, Y. G. Cid, and N. Kita, "Real-time 3d slam with wide-angle vision," July 2004.

[62] O. Stasse, A. Davison, R. Sellaouti, and K. Yokoi, "Real-time 3d slam for humanoid robot considering pattern generator information," pp. 348–355, September 2006.

[63] B. Williams, G. Klein, and I. Reid, "Real-time slam relocalisation," *ICCV 2007. IEEE 11th International Conference on Computer Vision*, pp. 1–8, Oct. 2007.

[64] J. Guivant and E. Nebot, "Optimization of the simultaneous localization and map-building algorithm for real-time implementation," *IEEE Transactions on Robotics and Automation*, vol. 17, pp. 242–257, Jun 2001.

[65] J. M. M. Montiel, J. Civera, and A. J. Davison, "Unified inverse depth parametrization for monocular slam," in *Proceedings of Robotics: Science and Systems II*, pp. 16–19, 2006.

[66] J. Civera, J. A. M. Andrew J. Davison, and J. M. M. Montiel, "Drift-free real-time sequential mosaicing," *International Journal of Computer Vision*, vol. Online, 2008.

[67] J. Civera, A. J. Davison, and J. M. M. Montiel, "Inverse depth parametrization for monocular slam," *Robotics, IEEE Transactions on*, vol. 24, pp. 932–945, Oct. 2008.

[68] J. Civera, A. Davison, and J. Montiel, "Inverse depth to depth conversion for monocular slam," *Robotics and Automation, 2007 IEEE International Conference on*, pp. 2778–2783, April 2007.

[69] R. Smith, M. Self, and P. Cheeseman, "A stochastic map for uncertain spatial relationships," in *Fourth International Symposium on Robotics Research*, (Cambridge, MA, USA), pp. 467–474, MIT Press, 1987.

[70] J. Shi and C. Tomasi, "Good features to track," *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94., 1994 IEEE Computer Society Conference on*, pp. 593–600, Jun 1994.

[71] T. Bailey, J. Nieto, J. Guivant, M. Stevens, and E. Nebot, "Consistency of the ekf-slam algorithm," *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pp. 3562–3568, Oct. 2006.

[72] J. Adkisson, *Lost Eye: Coping with Monocular Vision After Enucleation Or Eye Loss from Cancer, Accident, Or Disease*. iUniverse, 2006.

[73] A. Saxena, M. Sun, and A. Ng, "Make3d: Learning 3d scene structure from a single still image," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. Accepted for future publication, 2008.

[74] S. Glanville, "Anim8or, "http://www.anim8or.com/main/index.html", [last visited on: 20/11/2008]."

[75] A. J. Davison, "Reserach website, "http://www.doc.ic.ac.uk/ ajd/", [last visited on: 20/11/2008]."

[76] ASUS, "Eeepc, "http://eeepc.asus.com/global/index.html", [last visited on: 20/11/2008]."

[77] Intel, "Opencv library, "http://sourceforge.net/projects/opencvlibrary/" [last visited on: 20/11/2008]."

[78] A. Lee, "Virtualdub, "http://www.virtualdub.org/", [last visited on: 20/11/2008]."

[79] Eclipse-Foundation, "Eclipse c/c++ development tooling - cdt, "http://www.eclipse.org/cdt/", [last visited on: 20/11/08]."

[80] C.-C. Wang and C. Thorpe, "A hierarchical object based representation for simultaneous localization and mapping," *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, vol. 1, pp. 412–418 vol.1, Sept.-2 Oct. 2004.

[81] C. Bibby and I. Reid, "Simultaneous localisation and mapping in dynamic environments (slamide) with reversible data association," in *Proceedings of Robotics Science and Systems*, 2007.

[82] P. Gemeiner, W. Ponweiser, P. Einramhof, and M. Vincze, "Real-time slam with a high-speed cmos camera," *Image Analysis and Processing, 2007. ICIAP 2007. 14th International Conference on*, pp. 297–302, Sept. 2007.

[83] C. Laugier, D. A. Vasquez Govea, M. Yguel, T. Fraichard, and O. Aycard, "Geometric and bayesian models for safe navigation in dynamic environments," *Intelligent Service Robotics*, 2007.

[84] C.-C. Wang, C. Thorpe, M. Hebert, S. Thrun, and H. Durrant-Whyte, "Simultaneous localization, mapping and moving object tracking," *The International Journal of Robotics Research*, vol. 26, June 2007. (to appear).

[85] Z. Xu, Y. Zhuang, and H. Chen, "Obstacle detection and road following using laser scanner," *Intelligent Control and Automation, 2006. WCICA 2006. The Sixth World Congress on*, vol. 2, pp. 8630–8634, 21-23 June 2006.

[86] Y. Wang, E. K. Teoh, and D. Shen, "Lane detection and tracking using b-snake," *Image and Vision Computing*, vol. 22, no. 4, pp. 269 – 280, 2004.

[87] Y. Shan, F. Yang, and R. Wang, "Color space selection for moving shadow elimination," *Image and Graphics, 2007. ICIG 2007. Fourth International Conference on*, pp. 496–501, Aug. 2007.

[88] P. Bellutta, R. Manduchi, L. Matthies, K. Owens, and A. Rankin, "Terrain perception for demo iii," *Intelligent Vehicles Symposium, 2000. IV 2000. Proceedings of the IEEE*, pp. 326–331, 2000.

[89] M. of Defence, "Mod grand challenge (2008)," tech. rep., http://www.challenge.mod.uk/, [Last visited: 15/04/2008].

[90] W. K. Pratt, *Digital Image Processing: PIKS Scientific Inside*. Wiley-Interscience, 2007.