CRANFIELD UNIVERSITY

HANI ALQAAN

AUTOMATIC PIPELINE SURVEILLANCE AIR-VEHICLE

SCHOOL OF AEROSPACE, TRANSPORT SYSTEMS & MANUFACTURING

Doctorate of Philosophy Degree

Supervisor: Dr. Al Savvaris

February 2016

CRANFIELD UNIVERSITY

SCHOOL OF AEROSPACE, TRANSPORT SYSTEMS & MANUFACTURING

Doctorate of Philosophy Degree

HANI ALQAAN

Automatic Pipeline Surveillance Air-Vehicle

Supervisor: Dr. Al Savvaris

February 2016

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

ABSTRACT

This thesis presents the developments of a vision-based system for aerial pipeline Right-of-Way surveillance using optical/Infrared sensors mounted on Unmanned Aerial Vehicles (UAV). The aim of research is to develop a highly automated, on-board system for detecting and following the pipelines; while simultaneously detecting any third-party interference. The proposed approach of using a UAV platform could potentially reduce the cost of monitoring and surveying pipelines when compared to manned aircraft. The main contributions of this thesis are the development of the image-analysis algorithms, the overall system architecture and validation of in hardware based on scaled down Test environment.

To evaluate the performance of the system, the algorithms were coded using Python programming language. A small-scale test-rig of the pipeline structure, as well as expected third-party interference, was setup to simulate the operational environment and capture/record data for the algorithm testing and validation.

The pipeline endpoints are identified by transforming the 16-bits depth data of the explored environment into 3D point clouds world coordinates. Then, using the Random Sample Consensus (RANSAC) approach, the foreground and background are separated based on the transformed 3D point cloud to extract the plane that corresponds to the ground. Simultaneously, the boundaries of the explored environment are detected based on the 16-bit depth data using a canny detector. Following that, these boundaries were filtered out, after being transformed into a 3D point cloud, based on the real height of the pipeline for

fast and accurate measurements using a Euclidean distance of each boundary point, relative to the plane of the ground extracted previously. The filtered boundaries were used to detect the straight lines of the object boundary (Hough lines), once transformed into 16-bit depth data, using a Hough transform method. The pipeline is verified by estimating a centre line segment, using a 3D point cloud of each pair of the Hough line segments, (transformed into 3D). Then, the corresponding linearity of the pipeline points cloud is filtered within the width of the pipeline using Euclidean distance in the foreground point cloud. Then, the segment length of the detected centre line is enhanced to match the exact pipeline segment by extending it along the filtered point cloud of the pipeline.

The third-party interference is detected based on four parameters, namely: foreground depth data; pipeline depth data; pipeline endpoints location in the 3D point cloud; and Right-of-Way distance. The techniques include detection, classification, and localization algorithms.

Finally, a waypoints-based navigation system was implemented for the air-vehicle to fly over the course waypoints that were generated online by a heading angle demand to follow the pipeline structure in real-time based on the online identification of the pipeline endpoints relative to a camera frame.

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my God for everything. I would then like to express my sincere gratitude to my advisor, Dr. Al Savvaris, for the continuous support of my PhD study and related research, for his motivation, patience, and immense knowledge. His guidance helped me for the entire research and writing of this thesis and helped achieve this work. I could not have imagined having a better advisor and mentor for my PhD study.

Besides my advisor, I would like to thank the rest of staff in the Autonomous Systems group, for their insightful comments and encouragement, but also for the hard questions which gave me incentive to widen my thesis and research from various perspectives.

Also, I would like to thank my colleagues for their much-appreciated involvement in helping me complete this project.

I would like to thank the pipeline's monitoring company for providing the aerial video capture of the pipeline structure.

Last, but not the least, I would like to thank my family: my mother, wife and my brothers and sister for supporting me spiritually and morally throughout my studies and the writing of this thesis and in my life.

TABLE OF CONTENTS

ABSTRACT i
ACKNOWLEDGEMENTS iii
TABLE OF CONTENTS v
LIST OF FIGURES ix
LIST OF TABLES xvii
LIST OF ABBREVIATIONS xix
LIST OF SYMBOLS xxi
Chapter 1 Introduction 1
1.1 Background 1
1.2 Project Aim & Objectives 2
1.3 Contribution to Knowledge 3
1.4 Publication 4
1.5 Thesis Outline 4
Chapter 2 Literature Review 5
2.1 Introduction 5
2.2 Pipeline Integrity Overview 5
2.3 Pipeline Infrastructure Characteristics 8
2.4 Pipeline Monitoring Systems 12
2.4.1 Built-On/In Monitoring Systems 12
2.4.2 Aerial Monitoring Systems 14
2.5 Aerial Platforms 18
2.6 Computer Vision Techniques 20
2.6.1 Pipeline Detection & Tracking 21
2.6.2 Third-Party Interference Monitoring 23
2.6.3 Oil Leak Monitoring 29
2.7 Wireless Communications 30
2.8 Pipeline Monitoring Costs 31
2.9 Chapter Summary 32
Chapter 3 General System Overview and Experimental Setup 33
3.1 Introduction 33
3.2 System Requirements and Limitations 34
3.2.1 The Operational Requirements 34
3.2.2 Limitations 35
3.3 System Architecture 35
3.4 Aerial Platform 37
3.4.1 Hardware Components 37

3.4.2 Software Development 55
3.5 Ground Control Station (GCS) 57
3.5.1 Hardware Components 57
3.5.2 Software Development 58
3.6 Test Site 62
3.6.1 A Small-Scale Prototype of a Survey Site 63
3.7 System Integration 64
3.8 Chapter Summary 65
Chapter 4 Pipeline Endpoints Identification 67
4.1 Introduction 67
4.2 Visible-Based Pipeline Endpoints Identifications 68
4.2.1 Image Acquisition 68
4.2.2 Pipeline Detection & Localization Algorithm 69
4.2.3 Experimental Setup 81
4.2.4 Experimental Results 83
4.3 Depth-Based Pipeline Endpoints Identification 87
4.3.1 Requirements 88
4.3.2 Identification algorithm 90
4.3.3 Performance 105
4.4 Chapter Summary 106
Chapter 5 Third-Party Interference Detection 109
5.1 Introduction 109
5.2 Third-Party Objects Detection 111
5.2.1 Objects Detection 111
5.2.2 Third-Party Objects Classification 114
5.2.3 Third-Party Objects Localization 121
5.3 Performance Evaluation 121
5.4 Chapter Summary 123
Chapter 6 Air-Vehicle Autopilot Waypoints Navigation 125
6.1 Introduction 125
6.2 UAV Platform 126
6.3 Waypoints Navigation Algorithm 127
6.3.1 Real-time Course Waypoints Generation 129
6.3.2 Change Estimation of the Course Angle 131
6.3.3 Turn Anticipation Distance Estimation 132
6.3.4 Proximity Distance 133
6.3.5 Heading Demand 133
6.3.6 Loitering 133
6.4 Platform Flight Control System 134
6.4.1 Platform 6DOF Dynamic Model 134
6.4.2 Stability Augmentation System Model 135

6.4.3 Autopilot Model 136
6.5 Simulation Results 136
6.5.1 Waypoints Navigation Performance 136
6.5.2 Heading Performance 138
6.5.3 Roll SAS Performance 138
6.5.4 Altitude Performance 140
6.5.5 Pitch SAS Performance 140
6.5.6 Speed Performance 141
6.5.7 Loitering Performance 142
6.6 Chapter Summary 143
Chapter 7 Performance and Evaluation of the Vision-Based Aerial Pipeline Surveillance System 145
7.1 Introduction 145
7.2 Test-Rig 147
7.3 Endpoints Identification of the Pipeline 148
7.3.1 Capability 149
7.3.2 Accuracy 154
7.4 Third-Party Interference Detection and Classification 160
7.4.1 Capability 161
7.5 Pipeline Following/Tracking 167
7.5.1 Course Waypoint Generation 168
7.5.2 Waypoints Navigation 175
7.6 Chapter Summary 191
Chapter 8 Discussions and Conclusions 195
8.1 Introduction 195
8.2 Discussion of Research Results 195
8.3 Fulfilment of Research Aim and Objectives 198
8.4 Research Limitations 199
8.5 Conclusion 200
8.6 Future Work 200
REFERENCES 203
APPENDICES 215

LIST OF FIGURES

Figure 2-1: Causes of Significant Pipeline Incidents on Hazardous Liquid Transmission Pipelines from 1988-2008 (Baker, 2009) 6

Figure 2-2: Transmission Pipeline in the Americas (Hopkins, 2002) 9

Figure 2-3: Trans-Alaska Pipeline System (TAPS) (Welch, 2010) 10

Figure 2-4: Oil and Gas Pipeline Infrastructure - the Middle East (U.S. Energy Information Administration (eia), 2010) 11

Figure 2-5: Environments of pipeline infrastructure, Ref: Image collection was produced from reference (Oil & Gas Pipeline, 2011) 11

Figure 2-6: Common remote sensing platforms (NASA tutorial on remote sensing, 2011) 18

Figure 2-7: [Left] Breguet-Richet Gyroplane No.1, 1907 [Right] Quadrotor (Devaud et al., 2012) 19

Figure 2-8: Unmanned Air-Vehicles (UAVs) autonomy level 20

Figure 2-9: [Left]: Image Space, [Right]: Parameter space or Hough space (Antolovic, 2008) 22

Figure 2-10: Semantic concept of parallel piped classifying in 3D feature space (Rahman and Afroz, 2013) 26

Figure 2-11: Concept of minimum distance classifier (Rahman and Afroz, 2013) 27

Figure 2-12: Types of Haar-like features (Viola and Jones, 2004) 28

Figure 2-13: Directions of wireless technologies (Toshihiro, 2010) 31

Figure 3-1: General system architecture of the vision-based aerial pipeline surveillance system 36

Figure 3-2: GAUI 500X Quadrotor platform (Multi-Rotor Technology (MRT), 2012) 39

Figure 3-3: ASUS Xtion Pro Live Sensor (ASUS, 2012) 40

Figure 3-4: Asus Xtion Pro Live weight measurement 40

Figure 3-5: Sensor projection of infrared structured light pattern 42

Figure 3-6: Gaui Crane II Gimbal 43

Figure 3-7: Samples of calibration images at various poses 44

Figure 3-8: Checkerboard corner detection in a colour image 44
Figure 3-9: Plane's corner detection in depth image (red boxes) 45
Figure 3-10: Nitrogen 6x embedded board (Boundary Devices, 2012) 45
Figure 3-11: Nitrogen 6X Add-on Ti-Wi Module (Boundary Devices, 2012) 46
Figure 3-12: OpenNI Framework (OpenNI, 2012) 48
Figure 3-13: ArduPilot Mega Autopilot board (ArduPilot, 2013) 51
Figure 3-14: ArduPilot Mega onboard firmware (ArduPilot, 2013) 51
Figure 3-15: 3DR uBlox GPS with ArduPilot Mega hardware interface (ArduPilot, 2013) 53
Figure 3-16: XBee transceiver (DIGI, 2014) 54
Figure 3-17: XBee & APM hardware interface (ArduPilot, 2013) 54
Figure 3-18: Li-Po 1300mAh battery & DC/DC converter 55
Figure 3-19: The aerial platform software process 56
Figure 3-20: XBee module & FTDI cable for interface with the PC 58
Figure 3-21: GCS software processing 59
Figure 3-22: ArduPilot Mega GCS application (ArduPilot, 2013) 59
Figure 3-23: Server/Client Scheme 60
Figure 3-24: Flow chart of communication 61
Figure 3-25: Streaming a video wirelessly 61
Figure 3-26: Pipeline structures samples, Ref: Image collection was produced from reference (Oil & Gas Pipeline, 2011) 62
Figure 3-27: Third-party interference samples, Ref: Image collection was produced from reference (Pipeline's Third-Party Interference, 2011) 62
Figure 3-28: Small-scale of the survey sites 63
Figure 3-29: Experimental setup layout 64
Figure 3-30: Aerial system integration in Catia design 65
Figure 3-31: Aerial platform & payloads integration 65
Figure 4-1: Two images of a pipeline at various environments (Pipeline Surveillance Company - Not for Public Release, 2012) 69
Figure 4-2: Camera-based pipeline detection algorithm 69

Figure 4-3: (Left): Intensity (Grey colour) of RGB colour space, (Right): Value (V) component of HSV colour space 71

Figure 4-4: The edges detected as a binary image based on the RGB intensity (Left), and the HSV value component (Right) 72

Figure 4-5: (Left) Image space and (Right) Parameter space or Hough space (Antolovic, 2008) 73

Figure 4-6: The longest line detection using Hough Transform 74

Figure 4-7: True & false detection of multi-lines detection using Hough Transform 74

Figure 4-8: Filtering & enhancement of pipeline detection 75

Figure 4-9: Real pipeline endpoint identification 76

Figure 4-10: Geometric calculations for pipeline position estimation 80

Figure 4-11: Experimental setup 82

Figure 4-12: Camera sensor (Canon, 2011) 82

Figure 4-13: Numbers of the detected pipeline segments 83

Figure 4-14: Position error percentage 84

Figure 4-15: Pipeline structure 85

Figure 4-16: Results of pipeline detection 85

Figure 4-17: Pipeline segments endpoints history 86

Figure 4-18: Depth-based pipeline endpoints identification 88

Figure 4-19: Kinect greyscale image of a depth array 89

Figure 4-20: Depth-based pipeline endpoints identification algorithm 91

Figure 4-21: 3D point cloud mapping 93

Figure 4-22: RGB of expected sample of pipeline and third-party objects 95

Figure 4-23: Coplanar & non-coplanar filtration in the 3D point cloud 96

Figure 4-24: Object boundary detection 97

Figure 4-25: 3D point cloud of objects boundary's height filtration 99

Figure 4-26: Depth image of objects boundary's height filtration 100

Figure 4-27: Geometry of the distance from a point to a line 102

Figure 4-28: Pipeline identification in 3D point clouds 104

Figure 4-29: Pipeline identification in RGB image (for demonstration) 104
Figure 5-1: Third-party interference detection algorithm..... 110
Figure 5-2: 3D point cloud of the detected objects 112
Figure 5-3: RGB data of the detected objects 114
Figure 5-4: Some of the positive samples used to train the classifier..... 115
Figure 5-5: Some of the negative samples used to train the classifier 116
Figure 5-6: Basic types of Haar-like features (Viola and Jones, 2004)..... 117
Figure 5-7: Integral image (Viola and Jones, 2004)..... 117
Figure 5-8: Cascade classifier structure (Viola and Jones, 2004) 120
Figure 5-9: Demonstration of the pipeline's third-party interference detection 123
Figure 6-1: Piper J-3 Cub 40 aerial platform..... 127
Figure 6-2: High-level block diagram of waypoints navigation..... 128
Figure 6-3: Waypoints navigation block diagram..... 129
Figure 6-4: The reference ground course configuration 130
Figure 6-5: Air-vehicle course projected on reference path (Start waypoint A; end waypoint J) 137
Figure 6-6: Heading virus course 138
Figure 6-7: Bank angle history..... 139
Figure 6-8: Altitude history 140
Figure 6-9: Longitudinal angles history..... 141
Figure 6-10: Airspeed history 142
Figure 6-11: Loitering Mode 142
Figure 7-1: Indoor test-rig general configuration of the proposed flight-tests scenarios of the pipeline surveillance system..... 148
Figure 7-2: Demonstration of the pipeline endpoints identification capabilities at the presence of the pipeline structure (sensitivity rate), (a) captured at 100 cm altitude with present of objects (interruption) and (b) captured at different altitude 120 cm (resolution variety) without objects presented .. 150
Figure 7-3: Demonstration of the pipeline endpoints identification error at the present of the pipeline structure (false negative rate) captured at 100 cm altitude..... 151

Figure 7-4: Demonstration of the pipeline endpoints identification capabilities at the absence of the pipeline structure (Specificity rate) captured at 100 cm altitude 152

Figure 7-5: Demonstration of the pipeline endpoints identification error at the absent of the pipeline structure (false positive rate) captured at 120 cm altitude 153

Figure 7-6: behaviour of the estimated backward ($XEPb$) and forward ($XEPf$) north position of the pipeline segment endpoints relative to the camera frame, and the desired north position of the backward in ($XEPbd$) and forward in ($XEPfd$) endpoints, while, [Top]: captured at 100 cm altitude, [Bottom]: captured at 120 cm altitude 155

Figure 7-7: Behaviours of the estimated backward ($YEPb$) and forward ($YEPf$) east position of the pipeline segment endpoints relative to the camera frame, and the desired east position of the backward in ($YEPbd$) and forward in ($YEPfd$) endpoints, [Top]: captured at 100 cm altitude, [Bottom]: captured at 120 cm altitude 158

Figure 7-8: Behaviours of the estimated backward ($hEPb$) and forward ($hEPf$) height of the pipeline segment endpoints relative to the ground, their mean values ($hEPb$) and ($hEPf$), their standard deviations ($hEPbstd$) and ($hEPfstd$), and the ground-truth height of both of them ($hEPd$), [Top]: captured at 100 cm altitude, [Bottom]: captured at 120 cm altitude 159

Figure 7-9: Demonstration of third-party interference detection capabilities at the presence of them (sensitivity rate), (a) captured at 100 cm altitude and at present of other objects (interruption effects) and (b) captured at different altitude 120 cm (resolution effects) 162

Figure 7-10: Demonstration of third-party interference detection error at the presence of them (false negative rate), (a) captured at 100 cm altitude and at presence of other objects (interruption effects) and (b) captured at different altitude 120 cm (resolution effects) 163

Figure 7-11: Demonstration of third-party interference detection capabilities at the absence of them (Specificity rate), (a) captured at 100 cm altitude and at presence of other objects (interruption effects) and (b) captured at different altitude 120 cm (resolution effects) 165

Figure 7-12: Demonstration of third-party interference detection error at the absence of them (false positive rate), (a) captured at 100 cm altitude and at presence of other objects (interruption effects) and (b) captured at different altitude 120 cm (resolution effects) 166

Figure 7-13: Cases results of the real-time configuration of the air-vehicle course waypoints (WPs) based on the visual information of the pipeline segment endpoints (EPs) 169

Figure 7-14: Behaviour representation of the east position of the generated course waypoints relative to the camera frame includes pipeline's relative point ($YPrelative$), current waypoint ($YWPcurrent$), target waypoint ($YWPtarget$), and future waypoint ($YWPfuture$), while, [Top]: captured at 100 cm altitude, [Bottom]: captured at 120 cm altitude 172

Figure 7-15: Behaviour representation of the north position of the generated course waypoints relative to the camera frame includes pipeline's relative point ($XPrelative$), current waypoint ($XWPcurrent$), target waypoint ($XWPtarget$), and future waypoint ($XWPfuture$), while, [Top]: captured at 100 cm altitude, [Bottom]: captured at 120 cm altitude 174

Figure 7-16: Behaviours of the forward-follow (dff) and cross-follow (dcf) distances, where, ($dffd$) and ($dcfd$) are the desired values of the forward and cross follow distances, respectively, while, [Top]: captured at 100 cm altitude, [Bottom]: captured at 120 cm altitude..... 177

Figure 7-17: Behaviours of the errors of the pipeline forward-follow (δff) and cross-follow (δcf), [Top]: captured at 100 cm altitude, [Bottom]: captured at 120 cm altitude 179

Figure 7-18: Behaviours of the pipeline cross-follow error based on the pipeline infinite (δcfI) and pipeline segment (δcfS), [Top]: captured at 100 cm altitude, [Bottom]: captured at 120 cm altitude..... 180

Figure 7-19: Behaviours of the 2D position of the air-vehicle (PA) relative to the camera frame coordinates, involved the desired 2D position (PAd), the mean value (PA), the standard deviation ($PAstd$), while, [Top]: captured at 100 cm altitude, [Bottom]: captured at 120 cm altitude 182

Figure 7-20: The estimated altitude of the air-vehicle (h) relative to the camera frame coordinates, the desired altitude of the air-vehicle (hd), the mean value of the estimated altitude of the air-vehicle (h), the standard deviation of the estimated altitude of the air-vehicle ($hstd$), [Top]: captured at 100 cm altitude, [Bottom]: captured at 120 cm altitude..... 184

Figure 7-21: The estimated 3D position displacements of the air-vehicle ($\Delta x, \Delta y, \Delta z$), their main values, and standard deviations, [Top]: captured at 100 cm altitude, [Bottom]: captured at 120 cm altitude 186

Figure 7-22: The estimated air-vehicle's pitch angle (θ), the desired pitch angle (θd), the mean value (θ), the standard deviation (θstd), [Top]: captured at 100 cm altitude, [Bottom]: captured at 120 cm altitude 188

Figure 7-23: The estimated air-vehicle's roll angle (ϕ), the desired roll angle (ϕd), the mean value (ϕ), the standard deviation (ϕstd), [Top]: captured at 100 cm altitude, [Bottom]: captured at 120 cm altitude 189

Figure 7-24: The estimated air-vehicle's yaw angle (ψ), the desired yaw angle (ψd), the mean value (ψ), the standard deviation (ψstd), [Top]: captured at 100 cm altitude, [Bottom]: captured at 120 cm altitude 190

LIST OF TABLES

Table 2-1: Onshore Oil Pipeline Incidents in Western Europe in 1995 (Hopkins et al., 1999)..... 7

Table 2-2: Onshore Gas Pipeline Incidents in Western Europe in 1970-1997 (Hopkins et al., 1999) 7

Table 2-3: A summary of cost-benefit comparison result (Palmer, 2002)..... 32

Table 3-1: Hardware constraints 35

Table 3-2: Weight details of the payload components..... 38

Table 3-3: GAUI 500X Quadrotor specifications (Multi-Rotor Technology (MRT), 2012) 39

Table 3-4: ASUS Xtion Pro Live sensor specifications (ASUS, 2012)..... 41

Table 4-1: Performance of Hough transforms detection rates..... 80

Table 4-2: Camera specifications (Canon, 2011) 82

Table 4-3: Performance results of pipeline endpoints identification..... 106

Table 5-1: Performance results of third-party interference detection..... 122

Table 6-1: Piper J-3 Cub 40 general specifications..... 127

Table 7-1: Flight tests description..... 147

Table 7-2: Performance results of the pipeline endpoints identification algorithm 154

Table 7-3: Results of the 3D position of the pipeline's endpoints and errors relative to the camera frame coordinates..... 160

Table 7-4: Performance results of the third-party interference detection algorithm..... 167

Table 7-5: Results of 3D position of the air-vehicle and their errors relative to the camera frame coordinates at different altitudes..... 183

Table 7-6: performance results of 3D position displacement and errors relative to the previous position at two altitudes scenarios..... 185

Table 7-7: Results of air-vehicle orientations and their errors relative to the camera frame at different altitudes 191

LIST OF ABBREVIATIONS

APM	ArduPilot Mega
CARS	Coherent Anti-Raman Spectroscopy
CCD	Charge-Coupled Device
CEPA	Canadian Energy Pipeline Association
CMY	Cyan, Magenta and Yellow
CP	Cathodic Protection
DEM	Digital Elevation Model
eia	Energy Information Administration
EMR	Electro-Magnetic Radiation
EMS	Electro-Magnetic Spectrum
EO	Electro-Optical
EP	Endpoint
ESC	Electronic Speed Controller
FAA	Federal Aviation Administration
FOV	Field-Of- View
FPV	First-Person View
FTIR	Fourier Transform Infrared Spectroscopy
GBRA	Guadalupe-Blanco River Authority
GCS	Ground Control Station
GPS	Global Positioning System
GUI	Graphical User Interface
HIS	Hue, Intensity and Saturation
HT	Hough Transform
IACC	Impressed Alternating Cycle Current
IMU	Inertial Measurement Unit
IDE	Integrated Development Environment
IR	Infra-Red
LIDAR	Light Detection and Ranging
LIF	Laser Induced Fluorescence
LUV	Luminance, Chrominance U and Value
ML	Machine Learning
mmW	milli-meter Wave
MRT	Multi-Rotor Technology
MVD	Minimum Vector Dispersion

OPS	Office of Pipeline Safety
PHMSA	Pipeline & Hazardous Materials Safety Administration
PID	Proportional, Integral, Derivative
QVGA	Quarter Video Graphics Array
RANSAC	Random Sample Consensus
RC	Remote Control
RF	Radio Frequency
RGB	Red, Green, and Blue colours
RGBD	Red, Green, Blue, and Depth
ROS	Robot Operating System
ROW	Right-of -Way
SAR	Synthetic Aperture Radar
SAS	Stability Augmentation System
SXGA	Super Extended Graphics Array
TAD	Turn Anticipation Distance
TAPS	Trans-Alaska Pipeline System
TCP	Transmission Control Protocol
TDLAS	Tunable Diode Laser Absorption Spectroscopy
ToF	Time of Flight
UAV	Unmanned Air-Vehicle
UDP	User Datagram Protocol
UV	Ultra-Violet
VGA	Video Graphics Array
VTOL	Vertical Take-Off & Landing
WP	Waypoint

LIST OF SYMBOLS

$\boldsymbol{\rho}$	The shortest distance from the corner of the image space to the straight line
$\boldsymbol{\theta}$	The angle between the shortest distance ρ and the horizontal axis (width) in image space
$\boldsymbol{x, y}$	Width and height of the image (pixels)
$\boldsymbol{X, Y, Z}$	3D-world coordinates
$\boldsymbol{x_{p_{(1)}}, y_{p_{(1)}}}$	Width and height of the first endpoint of the real pipeline segment (pixels)
$\boldsymbol{x_{p_{(2)}}, y_{p_{(2)}}}$	Width and height of the second endpoint of the real pipeline segment (pixels)
$\boldsymbol{x_{UR}, y_{UR}}$	Width and height of the upper-right corner of the image frame (pixels)
$\boldsymbol{x_{UL}, y_{UL}}$	Width and height of the upper-left corner of the image frame (pixels)
$\boldsymbol{x_{DR}, y_{DR}}$	Width and height of the down-right corner of the image frame (pixels)
$\boldsymbol{x_{DL}, y_{DL}}$	Width and height of the down-left corner of the image frame (pixels)
$\boldsymbol{F_{x_{max}}, F_{y_{max}}}$	Maximum width and height of the boundary frame
$\boldsymbol{F_{x_{min}}, F_{y_{min}}}$	Minimum width and height of the boundary frame
$\boldsymbol{P_{x_U}, P_{y_U}}$	Width and height of the line segment interception with the upper line of the image frame (pixels)
$\boldsymbol{P_{x_D}, P_{y_D}}$	Width and height of the line segment interception with the down line of the image frame (pixels)
$\boldsymbol{P_{x_L}, P_{y_L}}$	Width and height of the line segment interception with the left line of the image frame (pixels)
$\boldsymbol{P_{x_R}, P_{y_R}}$	Width and height of the line segment interception with the right line of the image frame (pixels)
$\boldsymbol{P_{X_{max(1)}}, P_{Y_{max(1)}}}$	Maximum width and height of the correspond endpoint in the image frame (pixels)
$\boldsymbol{P_{X_{min(1)}}, P_{Y_{min(1)}}}$	Minimum width and height of the correspond endpoint in the image frame (pixels)
$\boldsymbol{l}$	Length of the real pipeline segment
$\boldsymbol{x_{p_1}, y_{p_1}}$	Width and height of the first endpoint of the real pipeline segment relative to the image frame
$\boldsymbol{x_{p_2}, y_{p_2}}$	Width and height of the second endpoint of the real pipeline segment relative to the image frame
$\boldsymbol{l_{11}}$	The length between the first endpoint of real pipeline segment to the first endpoint of the Hough line segment
$\boldsymbol{l_{12}}$	The length between the first endpoint of real pipeline segment

	to the second endpoint of the Hough line segment
l_{21}	The length between the second endpoint of real pipeline segment to the first endpoint of the Hough line segment
l_{22}	The length between the second endpoint of real pipeline segment to the second endpoint of the Hough line segment
$x_{H_{(1)}}, y_{H_{(1)}}$	Width and height of the first endpoint of the Hough line segment (pixels)
$x_{H_{(2)}}, y_{H_{(2)}}$	Width and height of the second endpoint of the Hough line segment (pixels)
α_1	The angle between the real pipeline segment and the connected line between the first endpoint of real pipeline segment and the second endpoint of the Hough line segment
α_2	The angle between the real pipeline segment and the connected line between the second endpoint of real pipeline segment and the first endpoint of the Hough line segment
ρ_1	The shortest distances of the first endpoints the pair of segments
ρ_2	The shortest distances of the second endpoints the pair of segments
ρ_a	The average distance between a pair of segments
ρ_p	The parallelism factor or difference between a pair of segments
u, v	Width and height of the depth image, respectively
d_{raw}	Raw of the depth value
I_d	Depth matrix
I_{dh}	Depth homogenous matrix
P_{3D}	3D point cloud matrix
P_{h3D}	Homogenous 3D point cloud matrix
K_{IR}	Intrinsic calibration matrix of the IR sensor
X_w	x-axis of the 3D point cloud coordinates
Y_w	y-axis of the 3D point cloud coordinates
Z_w	z-axis of the 3D point cloud coordinates
W_w	Homogenous factor of the 3D point cloud coordinates
α	Minimum probability of finding at least one good set of observations in (***N)*** trials.
$P_{3D_{edges}}$	3D point cloud of the edges
A, B, C, D	Plane parameters
$H_{max_{Pipeline}}$	Maximum height of the existing pipeline structure
S	endpoints list of the detected Hough lines
t_w	maximum width of the actual pipeline in meter
t_l	Minimum acceptable length of the pipeline in meter
N_{v_m}	Minimum number of 3D point cloud in vicinity of pipeline structure

N_v	number of 3D point cloud in vicinity of pipeline structure
l_M	Centreline segment length
P_{s_M}	Start-point of the centreline segment
P_{e_M}	End-point of the centreline segment
$L(P_s, P_e)$	Line segment
$L_M(P_{s_M}, P_{e_M})$	Centreline segment
P_s	Start-point of the line segment
P_e	End-point of the line segment
$d(P, L)$	Distance between point to line
$\vec{V}_L$	Direction vector of the line segment
$\vec{V}_P$	Direction vector of the point to the line segment
P_{s_1}	Start-point of the first line segment
P_{e_1}	End-point of the first line segment
P_{s_2}	Start-point of the second line segment
P_{e_2}	End -point of the second line segment
x_{s_M}	x-axis of start-point of the centreline segment in 3D point cloud coordinates
y_{s_M}	y-axis of start-point of the centreline segment in 3D point cloud coordinates
z_{s_M}	z-axis of start-point of the centreline segment in 3D point cloud coordinates
x_{e_M}	x-axis of end-point of the centreline segment in 3D point cloud coordinates
y_{e_M}	y-axis of end-point of the centreline segment in 3D point cloud coordinates
z_{e_M}	z-axis of end-point of the centreline segment in 3D point cloud coordinates
P_{In}	Inlier 3D point cloud
P_{Out}	Outlier 3D point cloud
R	Matrix rotations
T	Matrix translations
I_{RGB}	Matrix of the RGB data
Ih_{RGB}	Homogenous matrix of the RGB data
u', v'	Width and height of the RGB image
ii	The value of the integral image
i	The value of the original image
s	the sum of the cumulative row
$w_{t,i}$	Weight of image at each boosting trail and each image frame
t	Sequence of boosting trial
n	Sequence of image frame

h_j	Weak learner
ns	Number of the negative sample in Haar classifier
ps	Number of the positive sample in Haar classifier
x_{sw}	Width and height of the sub-window of the image
f	Feature
p	Polarity or the direction of the inequality sign
θ	Threshold
j	Feature number
ϵ_j	Error of each feature
x_{3rd}	Third-party interference location at x-axis
y_{3rd}	Third-party interference location at y-axis
h_d	Desired altitude
ψ_d	Heading demand
ϕ_d	Bank angle demand
x_a	Position of the air-vehicle at x-axis
y_a	Position of the air-vehicle at y-axis
z_a	Position of the air-vehicle at z-axis
V_A	Speed of the air-vehicle
v_x	Speed of the air-vehicle at x direction
v_y	Speed of the air-vehicle at y direction
v_z	Speed of the air-vehicle at z direction
WP_c	Current waypoint of the course
WP_t	Target waypoint of the course
WP_f	Future waypoint of the course
EP_c	Current endpoint of the pipeline segment
EP_t	Target endpoint of the pipeline segment
EP_f	Future endpoint of the pipeline segment
$dx_{c,n}$	North vector of the current endpoint
$dy_{c,n}$	East vector of the current endpoint
$\Delta\chi_T$	Course angle change
χ_c	Current segment angle
χ_n	Next segment angle
$WP_{t_{north}}$	North coordinate of target waypoint
$WP_{c_{east}}$	East coordinate of current waypoint
$WP_{f_{north}}$	North coordinate of future waypoint
$WP_{c_{north}}$	North coordinate of current waypoint
$WP_{t_{east}}$	East coordinate of target waypoint
$WP_{t_{north}}$	North coordinate of target waypoint

$\bar{\dot{\chi}}$	Air-vehicle average rate of turn
P_{A_n}	North coordinate of the aircraft position
P_{A_e}	East coordinate of the aircraft position
WP_{t_n}	North coordinate of the target waypoint
WP_{t_e}	East coordinate of the target waypoint
ψ_l	Loitering heading demand
C_p	Power coefficient
C_T	Thrust coefficient
J	Advance ratio

Chapter 1
Introduction

1.1 Background

Maintaining the integrity of oil and gas pipelines, that may run hundreds of miles through desolate terrain, is a highly demanding and manpower-intensive proposition. An unchecked leak can result in environmental disasters and costly disruptions to business. It is in the interest of any company to maintain the value of its pipelines and guard them effectively against any damage caused by third-parties or any other defects.

Today, the pipeline corridors are surveyed by regular foot, vehicle patrols or using fixed-wing aircraft or helicopters. These patrols prevent developments and events, which could place the pipelines, the surroundings of pipelines or security of supplies at risk.

As a result of global progress in high-resolution remote sensing and computer vision technology, it is now possible to design a highly automated airborne surveying system with remote sensors and context-oriented image processing software, to carry out these types of surveillance and monitoring missions.

The main aim of this project is to develop a vision based low-cost aerial technology for monitoring and routine inspections of pipeline’s rights-of-way. The technology is envisaged to compose a suite of sensors (for example Optical/Infrared, 3D LIDAR). In addition, computer vision techniques integrated

onboard the UAV platform, which comprise of; detection and visual tracking of the pipeline structure, anomalies detection algorithms and video data relay to the ground station. The development of these algorithms is performed in this research by designing and programming methods in terms of mathematical, physical and statistical functions for acquiring, processing, analysing, and understanding the images.

The developed system is envisaged to provide efficient and continuous support to pipeline infrastructures. The technology can be adjusted to the particular requirements to define the most suitable cost package depending on fuel, communication support, embedded sensors, etc. It is envisaged that the data could then be fused with other vital information (for example, internal erosion) obtained from other sensors and fed into the operating surveying system to define the status of the infrastructure, as well as of the surrounding area (for example environmental risks caused by leakages).

1.2 Project Aim & Objectives

The main aim of this research is to develop a vision based system for low-cost aerial technology for surveillance and routine inspections of lengthy pipelines and their Rights-of-Way.

The system includes a suite of remote sensors (such as, Optical/Infrared, 3D scanner system) integrated on-board a UAV platform. In addition to that, a developed computer vision algorithms for image processing in real-time for detection and visual following of the pipeline structure, anomalies detection, and video data relay to the ground station, were developed as part of the.

The key objectives of this project are summarized as follows:

- Develop a computer vision algorithm to estimate the position of the endpoints of the pipeline segments in the frames sequence in near real-time to automatically keep track/follow the pipeline.

- Develop a vision-based pipeline auto tracking algorithm, able to operate without GPS, based on the detected pipeline endpoints information in real-time.

- Develop a computer vision algorithm to detect anomalies and third-party activities in the vicinity of the pipeline structure and relay this information to the ground operator for visualisation and analysis.

- Integrate a video transmission datalink to transmit the data from the aerial platform to the Ground Control Station (GCS) with minimum latency.

- Test and validate the developed pipeline surveillance system using a scaled-down experimental setup in the laboratory.

1.3 Contribution to Knowledge

The contribution to knowledge of the project can be divided into two parts. First, the author had to develop the computer vision algorithms for detecting the pipeline and the third party interference. This included the testing and modifications of several image processing algorithms. Secondly, once the developed algorithms were optimised to run at sufficiently high update rate (i.e. near real-time), since this was a key success criteria in the project. They had to be integrated with the UAV autopilot system to create the complete aerial surveillance and monitoring system. In summary the contribution to knowledge can be summarised as follows:

1. The development of the computer vision approach for implementation on an on-board UAV embedded system for pipeline detection. The algorithms are able to run at a high update rate (near real-time), thus enabling the tracking and following of the pipeline structure by the UAV.

2. The video/images from the UAV platform are transmitted via a datalink to the ground control station to detect any third party activities and

infringements of the pipeline right-of-way (e.g. by JCBs, trucks etc.). The information are then processed on the workstation by the developed detection algorithms (using Haar classifier) for alerting the ground operator.

3. The pipeline detection algorithms are integrated with the UAV autopilot system to enable the UAV aerial platform to follow the pipeline and operate autonomously. Therefore, minimising the operator workload.

1.4 Publication

Alqaan, H., Mannberg, M. & Savvaris, A., 2012. Automatic Pipeline Detection for UAVs, Infotech@ Aerospace 2012, Orange County, USA.

1.5 Thesis Outline

This thesis is composed of eight chapters. *Chapter One* introduces the background regarding the project aims, objectives, and the contributions to knowledge. *Chapter Two* presents the literature review carried out covering previous and related works. This is followed by *Chapter Three*, which presents the overall concept of monitoring the pipeline's Right-of-Way (ROW) in real-time using a depth sensor mounted on a UAV platform. In addition, it explains and demonstrates the hardware, communications and the experimental setup of the system. *Chapter Four* describes the computer vision algorithm developed for the pipeline segment endpoint identification. Next, *Chapter Five* presents the computer vision algorithm for detecting third-party interference. *Chapter Six* presents the air-vehicle autopilot waypoint navigation system. Then, *Chapter Seven* covers performance results and evaluation of this research. Finally, *Chapter Eight* discusses, confirms, concludes and remarks on the automatic pipeline surveillance air-vehicle research carried-out in the project.

Chapter 2
Literature Review

2.1 Introduction

This chapter presents the review of related research and work that were carried-out in this project as part of the literature review. This review starts with an overview of a pipeline's integrity. Then, moves to the characteristics of the existing pipeline infrastructure. Following that, the current systems that have already been developed to monitor the pipeline, in general, are discussed. Then, the types of suitable aerial platforms that could be employed are described. The "second" part of the literature review covers existing computer vision techniques and applications. In addition, there is a brief survey of existing wireless communications systems and technology. Finally, a brief overview of the costs of current monitoring systems is covered at the end of the chapter.

2.2 Pipeline Integrity Overview

Hazardous transmission pipeline failures are an occasional occurrence but have the potential to cause major risks to population, properties and the environment, besides economic costs. During the last decades, pipelines have caused fires and explosions that killed more than 200 people and injured more than 1,000 people nationwide in the USA (Guadalupe-Blanco River Authority (GBRA), 2010). For example, In Dec. 2010, (San Martin Texmelucan, 2010), Mexico reported a massive oil pipeline explosion that laid waste to parts of a central Mexican city, incinerating people, cars, houses and trees as gushing

crude turned streets into flaming rivers. At least 28 people were killed, 13 of them children, in a disaster that authorities blamed on oil thieves. Also, on April 2010, another pipeline ruptured (Anthony, 2012), near Solomon, Kansas, US, because of previous excavation damage and about 1,659 barrels of natural gasoline were lost. Furthermore, the Saudi petroleum pipeline and export network and energy sector, in general (U.S. Energy Information Administration (eia), 2010), has been a terrorist target in the past. In February 2006, Saudi security prevented an attempted suicide bomb attack on the Abqaiq petroleum processing facility, after Al-Qaeda leadership called for renewed attacks against the country's economic backbone. So, many companies have recorded many incidents throughout their pipeline's route. Their defects are analysed and reported quantitatively that were either because of physical or operational causes.

The Pipeline and Hazardous Materials Safety Administration (PHMSA) (Baker, 2009) reported primary causes of the significant incidents of potentially hazardous liquid transmission pipeline with percentages between 1988 to 2008 as shown in Figure 2-1.

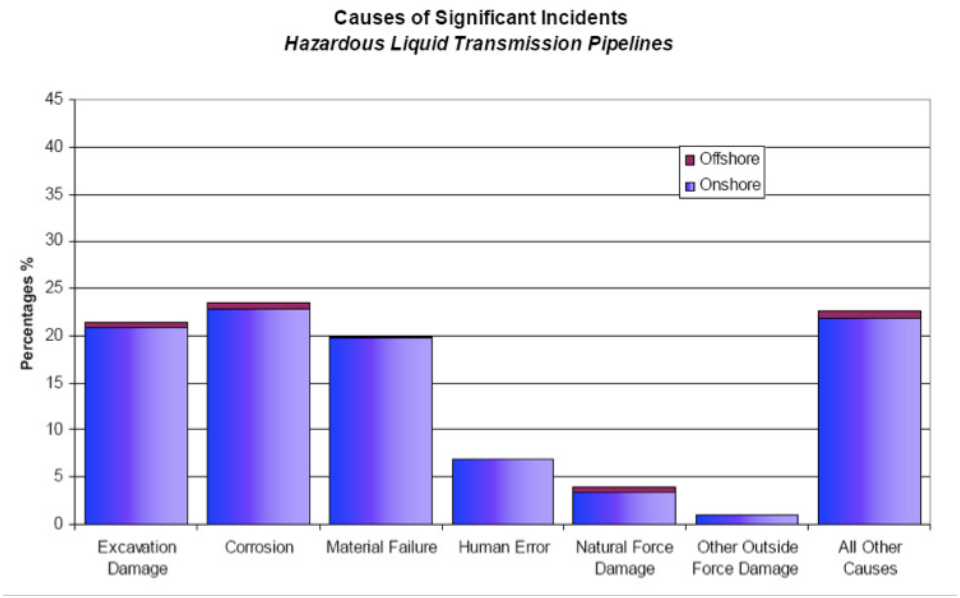


Figure 2-1: Causes of Significant Pipeline Incidents on Hazardous Liquid Transmission Pipelines from 1988-2008 (Baker, 2009)

Also, (Palmer et al., 2004) reported the quantities of causes of onshore oil pipeline incidents in Western Europe between 1971 and 1995 as shown in Table 2-1.

Table 2-1: Onshore Oil Pipeline Incidents in Western Europe in 1995 (Hopkins et al., 1999)

Cause	Number	Annual Average (1991-95)		Annual Average (1971-95)	
Mechanical Failure	4	5.2	38%	3.5	25%
Operational	1	1	7%	1	7%
Corrosion	1	2.6	19%	4.1	30%
Natural hazard	0	0.4	3%	0.6	4%
Third-party activity	4	4.2	31%	4.5	33%

According to Table 2-1, third-party interference (such as JCBs, gouges, dents, and so forth) has been reported at high percentage as a cause of incidents in comparison to the other threats in the oil pipeline as well as in the gas pipeline as shown in Table 2-2.

Table 2-2: Onshore Gas Pipeline Incidents in Western Europe in 1970-1997 (Hopkins et al., 1999)

Cause	rate
Hot tap by error	5%
Ground movement	6%
Corrosion	15%
Construction/material defect	18%
Third-party activity	50%
Other/unknown	6%

Because of these failures in the transmission pipelines, it is necessary to increase attention of public safety, also the direct cost of the failures affecting

the pipeline operators, and lessen these risks as much as possible by surveying the integrity of these pipelines and their Right-of-Way. Based on these results obtained from (Baker, 2009; Hopkins et al., 1999), it is essential to find a reliable technique that is capable of improving the integrity of these pipelines by early detection of third-party activities, or any leak already occurring in real-time and reporting their positions. So, some issues are required to be identified to help design such a system, which includes the necessary specifications of main pipeline networks (width, length, materials, surrounding environment, and so forth) and the possible methods that could be used to survey the pipelines and its corridor.

For pipelines that are deemed dangerous, such as those carrying gas and high-pressure oil, preventive measures to detect potential threats are more important than measures to detect real damages. In actual practice, dangerous pipelines are regularly patrolled by specialised personnel, either on foot, by vehicles, or even by helicopters. However, this kind of manual checking is laborious, economically expensive and is not seen to be efficient or necessarily useful. Therefore, there is an increasing necessity for automatic, continuous, and low-cost pipeline monitoring systems.

2.3 Pipeline Infrastructure Characteristics

Initially, some of the pipeline specifications are required to be known to design such a surveillance system. Based on the reports carried-out by (Canadian Energy Pipeline Association (CEPA), 2010; Hopkins, 2002; U.S. Energy Information Administration (eia), 2010), there are many types of pipelines utilised to transport oil and gas around the world which are listed below:

i. Flow lines & Gathering Lines – These short distance lines gather variety products in an area and move them to process facilities. They are usually small diameter from 50 mm (2”) to 305 mm (12”).

ii. Feeder Lines - These pipelines move the oil and gas fluids from processing facilities, storage, and so forth, to the main transmission lines. They can be up to 508 mm (20”) in diameter.

iii. Transmission Lines – These are the main conduits of oil and gas transportation as shown in Figure 2-2. They can be very large in diameter as, for example, in Russia, where they are around 1422 mm (56”) in diameter; or very long, such as in the case in the USA’s liquid pipeline system that is over 250,000 km (155,000 miles) in length. Crude oil transmission lines carry different types of product, to refineries or storage facilities.

iv. Product Lines - Pipelines carrying refined petroleum products from refineries to distribution centres are called product pipelines.

v. Distribution Lines - These allow local, low-pressure, allocation from a transmission system. Distribution lines could have in some cases large diameter, but most are under 152 mm (6”) diameter.

Figure 2-2: Transmission Pipeline in the Americas (Hopkins, 2002)

One of the largest transmission pipelines systems in the world is the Trans-Alaska Pipeline System (TAPS) (Welch, 2010). It was designed and constructed to move oil from the North Slope of Alaska to the northern most ice-free port-Valdez in Alaska as shown in Figure 2-3. The pipeline route covers 800 miles from Prudhoe Bay to the port of Valdez. It has a 48 inch diameter particular cold-weather steel material. Pump stations are located every 50 to 200 miles along the pipeline. They contain centrifugal pumps used to maintain movement

of oil within the pipeline. Pump station sites, which may cover on area that exceed 25 acres, may also include large liquid storage tanks.

Saudi Aramco operates more than 9,000 miles of petroleum pipelines throughout the country based on (U.S. Energy Information Administration (eia), 2010). Two major pipelines are 745 miles long and are called Petrolane, also known as the East-West Pipeline, which runs across Saudi Arabia from its Abqaiq complex to the Red Sea, as shown in Figure 2-4. The Petrolane system consists of two pipelines 56 inch and 48 inch diameter, respectively. Moreover, in the Eastern province, the pipeline between Riyadh and Dhahran is about 236 miles long; and a smaller 220 miles long between Riyadh and Qassim to the north.

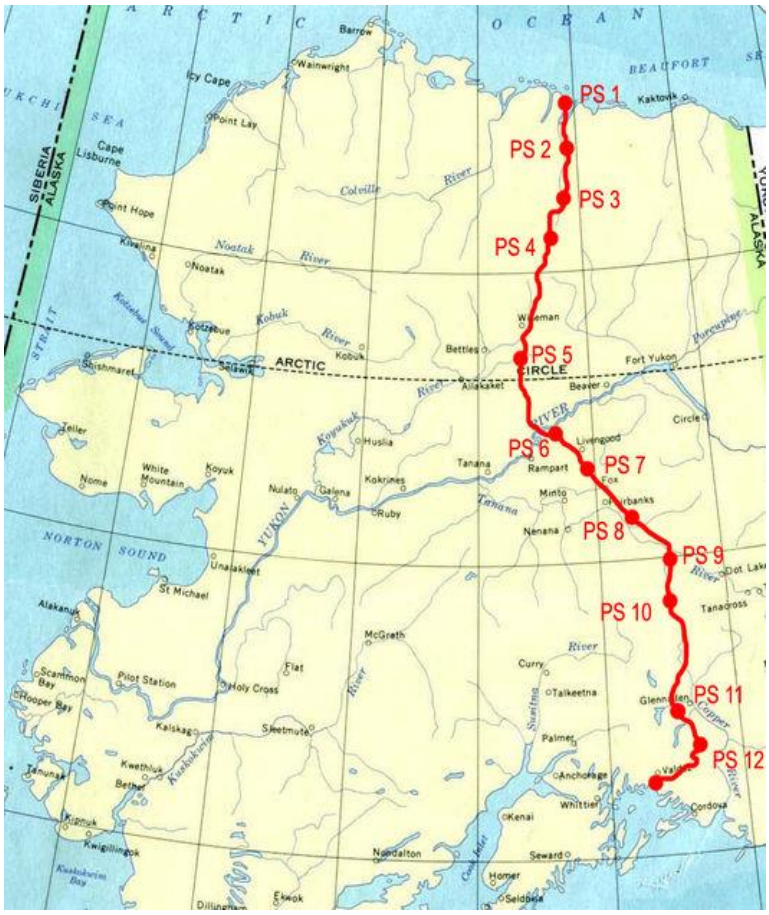


Figure 2-3: Trans-Alaska Pipeline System (TAPS) (Welch, 2010)

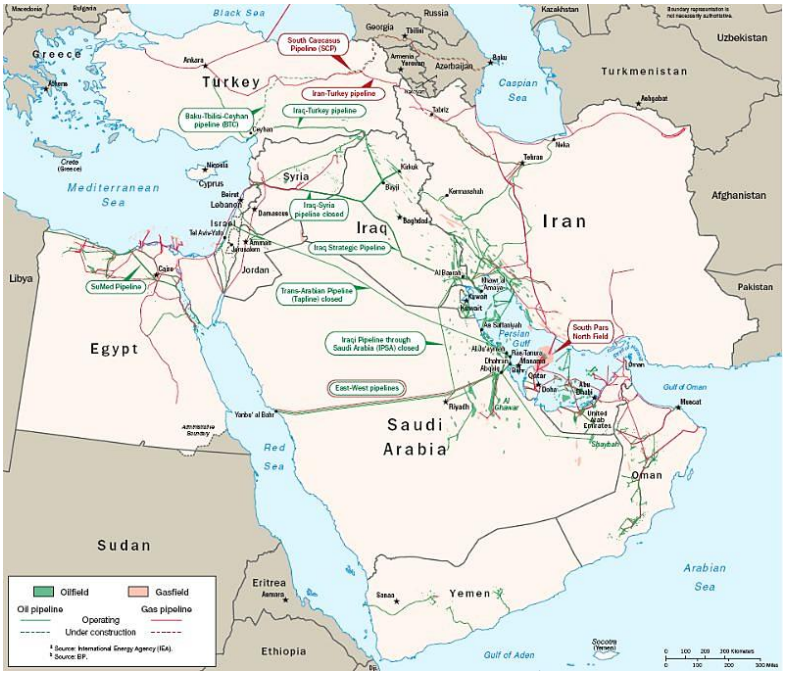


Figure 2-4: Oil and Gas Pipeline Infrastructure - the Middle East (U.S. Energy Information Administration (eia), 2010)

Examples of the type of environments where pipeline infrastructure is built/pass through are illustrated in Figure 2-5. As shown, the environments are so diverse, ranging from arid terrains, forest to all year snow-covered areas.

Figure 2-5: Environments of pipeline infrastructure, Ref: Image collection was produced from reference (Oil & Gas Pipeline, 2011)

2.4 Pipeline Monitoring Systems

Several Systems are currently available or under development for observing and reporting third-party interference and activity in the vicinity of the pipeline. Some of them were already reviewed by (Burkhardt and Crouch, 2003), or reporting leaks from pipes as in (Murvay and Silea, 2012). These methods are classified in this review into aerial or built-on/in monitoring systems, within pipeline or Right-of-Way infrastructures.

2.4.1 Built-On/In Monitoring Systems

The built-on/in surveillance systems are technologies utilized for ground surveys of pipelines integrity, which are performed inside the pipeline or on its surface or its district. These methods include fiber optic (Huang et al., 2007), evanescent sensing (Culshaw and Dakin, 1996), acoustic sensing (Jin and Eydgahi, 2008; Park et al., 2007), Cathodic Protection (CP) (Joseph and Winslow, 1982), Impressed Alternating Cycle Current (IACC) (Anupama et al., 2014), flow monitoring (Murvay and Silea, 2012), and Software based dynamic modelling (Murvay and Silea, 2012).

The fibre optic system is typically utilized to generate a broadband acoustic signal to detect leaks or third-party intrusion. This acoustic pressure will induce an optical phase signal in the parallel optical fibre fixed on the surface of the pipes. The system can detect and locate several leaks or intrusions along the pipeline at the same time. On the other hand, this technology requires particular and complicated installation to enhance sensitivity in the pipeline (leak) and the Right-of-Way (intrusions).

The evanescent sensing is a kind of on-pipe monitoring system. It is an optical fibre buried along with the pipe to sense and monitor the local changes of pressure or concentration using lasers and optical detectors. Once natural gas leaks, the local pressure or density of the gas will change, which leads to change in the transmission characteristics of the optical fibre.

Acoustic or sonic monitoring system is a non-destructive method, which typically analyses pressure measurements throughout the pipeline. If there is a region at which a noise is generated, it indicates a leak. The technology could detect the location of any leak as well as third-party activities. However, to monitor long pipelines, many sensors need to be installed. One downside of this technology is that it has high false alarm rates when detecting small leaks.

Cathodic Protection (CP) monitoring system can be used for detection of corrosion, leaks, and third-party contacts. This technology works by measuring the variation in the current paths of cathodic protection once any interference is made. The detection range is short; and again it has a high false detection due to breaches in pipe coating and the influence from 60 Hz (and harmonics) signals from other sources.

The IACC technology consists of impressing electrical signals on the pipe by generating a time-varying voltage between the pipe and the soil at alternate locations where pipeline access is available. It has the same advantages and disadvantages of the CP method.

Flow monitoring devices measure the rate of change of pressure or the mass flow at different sections of the pipeline. If the rate of change of pressure or the mass flow at two locations in the pipe differs significantly, it could indicate a potential leak. The major advantages of the system include the small cost of the system as well as non-interference with the operation of the pipeline. The two disadvantages of the system include the inability to pinpoint the leak location, and the high rate of false alarms.

Software based dynamic modelling monitors various flow parameters at different locations along the pipeline for leak detection. These flow parameters are then included in the model to determine the presence of natural gas leaks in the pipeline. The major advantages of the system include its ability to monitor continuously, and also the non-interference with pipeline operations. However,

dynamic modelling methods have a high rate of false alarms and are expensive for monitoring an extensive network of pipes.

In general, the built-on/in systems are very effective in terms of, cost and effort for installation and maintenance and have difficulties in dealing with ageing and long pipelines and complex terrains.

2.4.2 Aerial Monitoring Systems

The aerial monitoring systems are a remote sensing technique that can automatically provide a real-time surveillance of pipelines and Right-of-Way at a distance; delivering optical, acoustical, or microwave information. These technologies have been widely used in the aerial surveillance field for over a decade now. As quoted by (Fung et al., 1998), the National Energy Board and the gas pipeline industry concluded, after several studies, that current remote sensing technologies have the potential to be applied to pipeline Right-of-Way monitoring. A project was launched to identify where the use of remote sensing may provide cost-effective solutions and reduce the current amount of ground surveys. However, some key elements are required to develop and build an aerial monitoring system, which include:

- Sensors (Hardware);
- Platform to carry the payload and perform the mission (Hardware);
- Data processing (Hardware & software);
- Communication technology for data rely (Hardware & software).

The remote sensing techniques are classified, based on the sensor used, into passive or active (Schowengerdt, 2006). The active sensor uses its energy source to illuminate the target and measure the reflected radiation. For leak detection, the active sensor illuminates the area around the pipeline with a laser or a broadband source. Monitoring the absorption or scattering that are caused due to the presence of substance molecules around the surface can be performed using a set of sensors operating at specific wavelengths. If there is a

notable absorption or scattering around the pipeline, then a leak could potentially exist. In recent years, many active monitoring systems have been developed to detect leaks from pipelines; these include, Tunable Diode Laser Absorption Spectroscopy (TDLAS) (Zhang et al., 2009), Laser Induced Fluorescence (LIF) (McStay et al., 2007), Coherent Anti-Raman Spectroscopy (CARS) (Eckbreth, 1977), and Fourier Transform Infrared Spectroscopy (FTIR) (Mahamuni and Adewuyi, 2009), diode laser absorption (Iseki et al., 2000), millimeter Wave (mmW) Radar systems (Gopalsami and Raptis, 2001), backscatter imaging (McRae and Kulp, 1993; Spoonhower et al., 2006), broadband absorption (Spaeth and O'Brien, 2003), and LIDAR systems (Owechko et al., 2010; Prasad and Geiger, 1996; Roper and Dutta, 2005; Tao and Hu, 2002).

Diode laser absorption uses the same technology as LIDAR with the crucial difference being that diode lasers are used instead of the more expensive pulsed lasers. However, one downside is that if only a single wavelength is used, the system can be prone to false alarms since the laser can be absorbed equally well by dust particles.

Broadband absorption systems utilize low-cost lamps as the source; thus, significantly reducing the cost of the active system. Monitoring is conducted at multiple wavelengths, so that the system is less prone to false alarms.

Millimeter wave radar systems have been used to detect the chemicals used in the pipeline leak detection system itself. It measures the variation of scattering properties of the radioactive substance leaks around the pipeline, which could potentially indicate a leak.

Backscatter imaging utilizes a carbon-dioxide laser to illuminate the area above the pipeline. They are mainly used in natural gas pipelines, since the gas scatters the laser light very strongly. This scattered signature is imaged using an infrared imager or an infrared detector in conjunction with a scanner.

LIDAR systems typically employ a pulsed laser as the illuminating source. It measures the molecules' energy absorption at specific wavelengths in the electromagnetic spectrum.

All the active systems described previously use a source and obtain either transmitted or scattered images to determine the presence of a leak. These systems can be mounted on moving vehicles, air-vehicle or on-location. The advantages of these systems include the capability to monitor over an extended range. Furthermore, the ability to monitor third-party interference or leaks even if there are no differences in temperature between the liquid and the surroundings. Also, under specific conditions, these techniques have a high spatial resolution and sensitivity. However, these systems still have high, false rate alarms.

Passive monitoring systems are similar to active ones in many aspects. However, the major difference between active and passive techniques is that passive techniques do not require a source. Either the radiation emitted by the natural gas or the background radiation serves as the source. It makes passive systems less expensive in some respects. However, since a strong radiation source is not used, far more expensive detectors and imagers have to be used with passive systems.

The two major types of passive systems used for monitoring leaks from natural gas pipelines are thermal imaging (Kulp, 1997; Weil, 1993) and multi-wavelength imaging (Althouse and Chang, 1995; Bennett et al., 1996; Marinelli and Green, 1996; Smith and Laubscher, 1999).

Thermal imaging detects natural gas leaks from pipelines due to the differences in temperature between the natural gas and the immediate surroundings. This method can be used in moving vehicles, helicopters or portable systems and can cover several miles or hundreds of miles of pipeline per day. Usually, expensive thermal imagers are required to pick up the small temperature differential between the leaking natural gas and the surroundings. Also, thermal

imaging will not be effective if the temperature of the natural gas is not different from that of the surroundings.

Multi-wavelength or hyperspectral imaging can be accomplished either in absorption mode or emission mode. For obtaining gas concentrations utilizing multi-wavelength emission, the gas temperatures have to be much higher than the surrounding air. Multi-wavelength emission measurements have been typically used in the past to obtain single point concentrations in hot combustion products (Sivathanu and Gore, 1991; Sivathanu et al., 1991).

Multi-wavelength absorption imaging utilizes the absorption of background radiation at multiple wavelengths to directly image the gas concentration, even in the absence of temperature gradients between the gas and the surrounding air. This technique has been used to monitor natural gas leaks in industrial settings very successfully. The advantage of employing multi-wavelength passive systems is that they are relatively immune to false alarms, and can be utilized for remote monitoring without being constantly watched over. The only drawback, however, of multi-wavelength or hyperspectral imaging is that it typically utilizes very sensitive sensors which are expensive.

The proposed approach in this project is similar to the use of an imagery system, such as the Light Detection And Ranging (LIDAR) system. A commercial manned air-vehicle monitoring pipeline Right-of-Way could need relatively minimum modifications for the installation of such a system on the aircraft. On operation, it could reveal any JCBs, trucks or other earth-moving equipment in the vicinity of the pipeline (Fung et al., 1998; Randell, 2010). The advantage of the proposed system is that it can detect the pipeline structure and Right-of-Way in real-time, provide the location of defects due to a third-party and reduce the amount of data transmission, i.e. required bandwidth. Thereafter, it can use computer vision techniques to detect and classify any object in the pipeline surrounding environment. The other advantage of this technology (Shaochuang et al., 1999), is that it has an active laser pulse that is not influenced by the shadow angle of the sun which, in turn, reduces their

influence on data acquisition. Moreover, when compared with Photogrammetry, it avoids loss of information when transformed from 3D to 2D, has an accurate elevation; has multi-beam echo acquisition to get high-density data, is suitable for a high level of automation, and has real-time capability to produce a digital elevation model (DEM).

To emulate the LIDAR system, the sensor proposed to be used in this project is “ASUS Xtion Pro Live”. It uses a projected, structured light pattern to estimate the depth from the sensor to the view of the environment. The depth sensor is proposed to optimize the description of objects based on elevation contrast.

2.5 Aerial Platforms

One of the main, aerial monitoring elements is the platform itself that will bear the sensor instruments. Figure 2-6, shows some common platforms used for aerial monitoring, for example, air-vehicle, balloons, satellites, spacecraft, probes, rovers, launch air-vehicles, etc. (NASA tutorial on remote sensing, 2011)

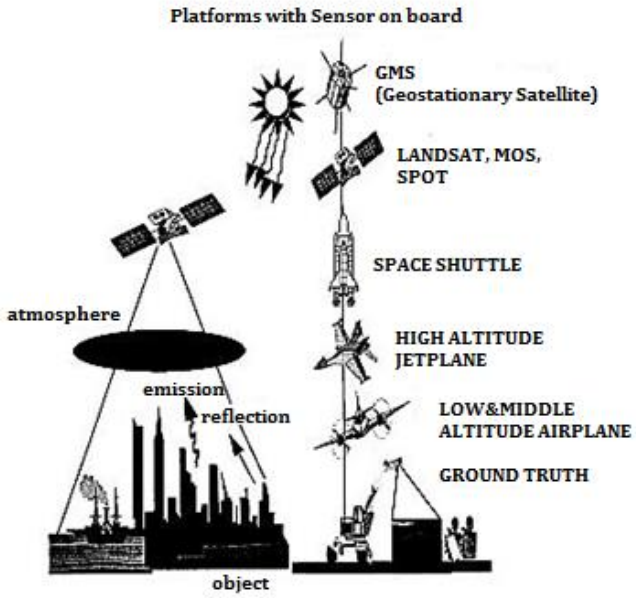


Figure 2-6: Common remote sensing platforms (NASA tutorial on remote sensing, 2011)

Nowadays, the Unmanned Aerial Air-vehicle (UAV) is an emerging technology being adapted for use in a wide range of applications in the field of remote sensing, and monitoring. The UAV can be remotely operated or, for more complicated and expensive systems, fly autonomously based on pre-programmed flight paths (Witayangkurn et al., 2011). This type of dull and repetitive mission makes UAVs an attractive solution for monitoring the length of pipeline routes and detecting any threats or leaks around it.

According to (Zongjian, 2008), the limitation of using satellites is due mainly to the high launch/flight costs, slow and weather-dependent data collection, restricted manoeuvrability, limited availability, flying time, and lower ground resolution. On the other hand, the UAV could operate relatively close to the pipeline, which in turn means it needs to use less expensive sensors to achieve the same high-resolution image as a satellite system. Also, it could provide real-time information and can cover the rights-of-way of a large segment of the pipeline quickly and efficiently (for example, when compared to foot patrols) (Roper and Dutta, 2006).

Following the recent San Bruno disaster, caused by a leaking gas pipeline, Mundus Group, a company that specializes in unmanned aerial air-vehicles (UAVs), suggested equipping UAVs with detector instrumentation to provide remote sensing of potentially leaking gas infrastructure in the US (The Business of Photonics, 2010).

Figure 2-7: [Left] Breguet-Richet Gyroplane No.1, 1907 [Right] Quadrotor (Devaud et al., 2012)

A quadrotor UAV platform will be used in the project to carry-out the tests. Recently, Many platforms such as this have been developed and built to achieve indoor/outdoor for search and rescue, or surveillance missions as shown in Figure 2-7 (Devaud et al., 2012; Gupte et al., 2012). The first quadrotor in history was designed by Louis Breguet in 1908 (Young, 1982) as shown on the left in Figure 2-7. This type of platform has the capability to follow a trajectory or waypoints of pipeline position, while ensuring stability, performing automatic vertical take-offs and landings, as well as hovering above identified, moving or stationary targets, such as threats around the pipeline or any other failure. The increased level of UAV autonomy will reduce the human operator's workload and increase the mission performance as shown in Figure 2-8.

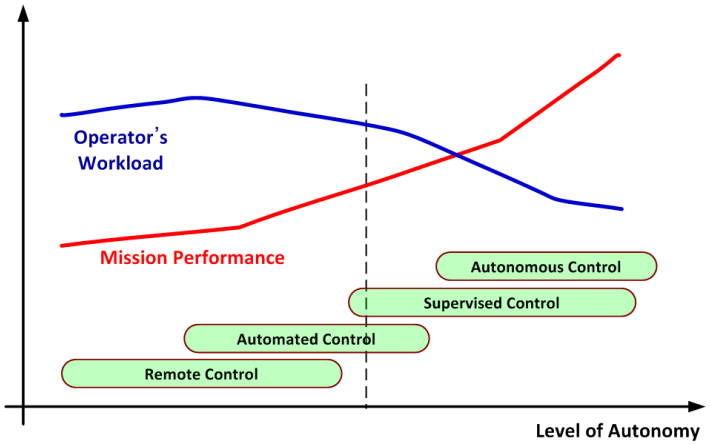


Figure 2-8: Unmanned Air-Vehicles (UAVs) autonomy level

2.6 Computer Vision Techniques

Many security and surveillance systems, that have already been developed and built, rely on the use of computer vision algorithms. Using image processing and analysis techniques, these monitoring applications automatically and remotely generate intelligent and useful descriptions of the monitoring environment (Koo et al., 2012; Nagai et al., 2009; Piciarelli et al., 2013). The descriptions in the computer vision field are interpreted based on features, boundaries, regions, and 3D models. Each technique is proposed based on the required description. However, in this project, four interpretations are needed to

achieve this goal, namely: extraction and tracking of the pipeline structure, third-party interference identification near the pipeline route, and leak detection.

2.6.1 Pipeline Detection & Tracking

In the open literature, work of several researchers and research centres focusing on the development of automatic target detection using computer vision for UAVs, can be found. For autonomous control purposes, several of the proposed methods are based on GPS (Dillman, 2002; Hasan et al., 2009; Low and Wang, 2008), vision (Dobrokhodov and Kaminer, 2006; Frew et al., 2004; Mondragón and Campoy, 2010; Mondragon et al., 2007; Rathinam et al., 2005), Digital Elevation Map (DEM) (Collins et al., 1998; Kanade et al., 2004; Nagai et al., 2009; Navarro-Serment, 2010), and a fusion of data (Du and Teng, 2007; Sohn et al., 2008). It is known that the GPS estimates the target position in the global frame while the other approaches estimate the position locally; therefore, some of the proposed approaches can follow the target objects locally as waypoints, landmarks or paths/trajectories.

In this project, the target required to be tracked is the pipeline, so in the computer vision field tracking the pipeline needs to be described first then its position can be localised. Due to the nature of the pipelines, the robust description that could be used as an indicator of pipeline structure is the linear segment. A number of vision-based techniques have been developed to extract this feature, such as combining the points of edges into lines (Cook and Delp, 1998; Nevatia and Ramesh Babu, 1980), Hough Transform (HT) (Agrawal et al., 1996; Antolovic, 2008; Gonzalez and Woods, 2003; Illingworth and Kittler, 1988), Random Sample Consensus (RANSAC) (Behrens et al., 2003; Nistér, 2005), vanishing point (Rathinam et al., 2008; Schindler et al., 2006; Tsai, Chang and Chen, 2006; Wang et al., 2004), learning algorithm (Rathinam et al., 2005), spline model (Wang et al., 2000).

Detecting pipelines by linking edge points can be achieved by determining positions and orientations of these points, then approximate the pipeline by

piecewise linear segments. However, it is a weak method to detect pipelines with the line because sometimes it is difficult to differentiate in colour.

Hough Transforms (HT) is one of the robust methods for finding the pipeline structure in an image. Once the feature points in the image space are detected, converting them into parameter space by using Hough transform can be done using the approach as shown in Figure 2-9. Then, the peak point in parameter space is evidence that there might be a line passing through that point that can be tuned by a number of points threshold.

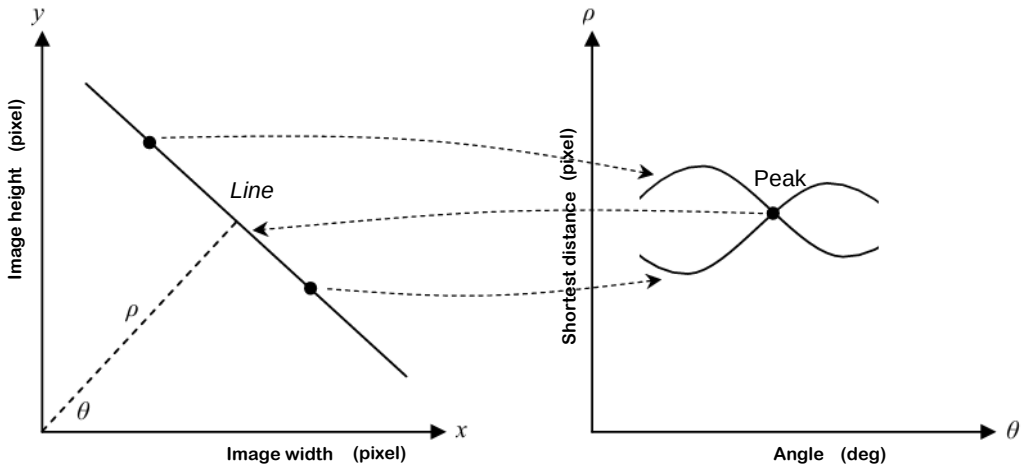


Figure 2-9: [Left]: Image Space, [Right]: Parameter space or Hough space
(Antolovic, 2008)

RANSAC is a voting algorithm that uses the minimum number observations (data points) required to estimate the underlying model parameters (Fischler and Bolles, 1981). It works by selecting a random sample of the minimum required size to fit the pipeline model, then it computes a putative pipeline model from the sample set. Following that, it detects the set of inliers to this pipeline model of the whole data set and loops the same process until a model with the most inliers over all samples is found. It is robust method for detecting pipelines. However, computational time grows quickly with a fraction of outliers and the number of parameters; moreover, it is not suitable for detecting multiple inlier structures.

The vanishing point is the candidate point based on the probability of at least two perspective projections of intersected line segments which are, in 3D, mutually parallel. The advantage of this approach is that it accurately detects the 3D linear structure. However, it is computationally complex and intensive.

Learning algorithms could be used to detect the pipeline using off-line modelling a cross-section profile of the pipeline at one raw of pixels and detect it by using the profile matching algorithm in real time. However, the algorithm fails with noisy boundaries of pipelines which is an issue considering that pipelines are built/found in different environments and terrains.

Spline modelling describes the prospective effect of parallel line boundaries of the pipeline by modelling piecewise polynomials of degree n with function values and the first n-1 derivatives that agree at the points where they join. Although it can detect pipeline structure, it cannot describe it in the case of any noise or broken boundaries line points; and, it also needs extra computations.

As a result, the proposed approach for this project to describe the pipeline structure as a combination of Hough transforms and RANSAC algorithms. These methods are robust and require lower computation than the others. The Hough transforms are used to describe the boundaries of the pipeline and RANSAC to emphasize that the boundaries are modelling the pipeline structure.

2.6.2 Third-Party Interference Monitoring

One of the main monitoring tasks is the detection and warning of any third-party, interference activities near the pipeline route in real-time. Since many sources of intrusions that potentially exist in the vicinity of the pipeline which include personnels, cars, trucks, and so forth, the proposed sensing technique is based on optical/point cloud data since the point cloud was already used in detecting the pipeline in the previous step. However, based on the point cloud data, there is different computer vision algorithms, can be used in this project to

determine the regions of interest, then analyse the synchronized optical data at that region to detect the targets such as third-parties interference.

Many algorithms have been developed to reconstruct a 3D point cloud model for segment regions of interest such as 3D region growing (Jarząbek-Rychard et al., 2010; Revol-Muller and Peyrin, 2000), 3D Hough transform (Borrmann et al., 2011; Tarsha-Kurdi, 2007), 3D recognizing structure (Vosselman and Gorte, 2004), and 3D RANSAC (Behrens et al., 2003; Huang et al., 2011; Kanade et al., 2004; Nagai et al., 2009; Rathinam et al., 2005).

A 3D, region growing, segmentation algorithm works on the principle of merging all neighbouring pixels to satisfy a homogeneity criterion starting from an initial set of seeds. The first process in an automated system is the unsupervised location of the seeds which is based on prior-knowledge of anatomical structures, such as humans, cars, trucks, and so forth; and their typical tracer uptake. The outcome of this data is taken into account to calculate certain parameters required for locating the seeds. This method is suitable to fit the high variability of the tracer uptake. Since both local and global parameters are taken into account in the merging process, this type of algorithms are competationally demanding.

The 3D Hough transform algorithm is often used to segment the 3D models of objects in point cloud sets. It is achieved by detecting the parameters of straight line segments, circular or elliptical cross sections that are subsequently tracked through the 3D point cloud. It maps the input sets of point cloud into zero-dimensional point sets in parameter space whose maxima represent object candidates. This approach is robust for 3D-dimensional parametric object detection. For a relatively large number of parameters it leads to extra space and time complexity.

A 3D recognizing structure algorithm has been used for object segmentation based on point cloud sets. This approach requires a predefined structure cue in the 3D point cloud set that is indicative of the object categories present in the

scene. Data training is then carried-out to match those cues for semantic segmentation.

A 3D RANSAC algorithm has been used to detect basic shapes in random point clouds. The algorithm decomposes the point cloud into an inliers specific structure of inherent shapes and outliers random residual points. Each detected inliers point serves as a representation of a set of corresponding points of the object shape. However, this algorithm is robust even in the presence of many outliers' points and in the presence of a high degree of noise. Moreover, this approach has lower computational load, compared to the other methods discussed previously.

Hence for this project, the proposed method to segment objects in the vicinity of pipeline routes will be based on RANSAC. It scales well to the size of the input point cloud, the number and size of the shapes within the data. Point sets with large samples are robustly decomposed within a reasonable computational time. Moreover, the algorithm is conceptually simple and easy to implement. Application areas include measurement of physical parameters, scan registration, surface compression, hybrid rendering, shape classification, meshing, simplification, approximation and reverse engineering.

Since it is difficult to classify the third-party activities using only point cloud data, the combination with optical data is proposed in this project. In computer vision there is a variety of algorithms, which have been proposed that are capable of assigning the segmented targets from the derived inliers reconstructed 3D models, using the alignment optical image data to obtain the desired category in order to consume the processing time and reduce the resulting classes. These supervised Machine Learning algorithms deal with single pixels or group of pixels in a segmented area. They include parallelepiped classifier (Rahman and Afroz, 2013), maximum likelihood classifier (Abkar et al., 2000), decision tree classifier (Pal and Mather, 2001), minimum distance classifier (Haala and Brenner, 1999; Wacker and Landgrebe, 1972), and multiple trained cascaded Haar classifier (Breckon and Barnes, 2009; Gaszczak et al., 2011).

The parallelepiped classifier is one of the most commonly used supervised classification algorithms for multispectral images, where the threshold of each class is defined in the training data to determine whether a given pixel is within the class or not (as shown in Figure 2-10). The interest region for each category is defined by a minimum and maximum pixel value on each axis of the segmented region. The accuracy of the classification depends on the selection of the minimum and maximum pixel values in lieu of the population statistics of each class. In this respect, it is very important that the distribution of the population of each category is well understood. This classifier is simple and easy to run having to make fewer assumptions regarding the character classes. In addition, the processing time will be minimum when compared to other classifiers. However, the accuracy will be low, especially when the distribution in feature space has covariance or dependency with oblique axes while the parallelepipeds are rectangular, which leads the classes to overlapping.

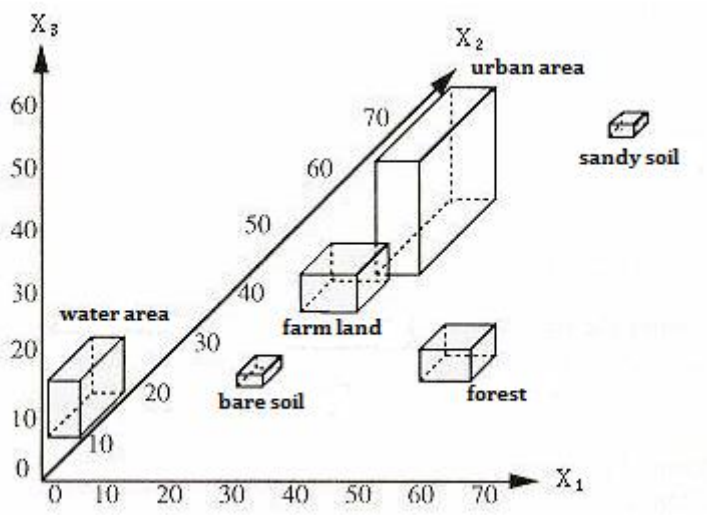


Figure 2-10: Semantic concept of parallel piped classifying in 3D feature space
(Rahman and Afroz, 2013)

The decision tree classifier is a multistage based classifier that compares the data with a range of properly selected features to break up a complex decision into simpler decisions. The selection of features is determined by an assessment of the spectral distributions of separability of the classes. The

procedure of this method, in general, involves the following steps: first splitting nodes, then determining which nodes are terminal; finally class labels are assigned to those terminal nodes based on a majority vote when it is assumed that certain categories are more likely than others. This classifier requires lower computing time than the other classifiers and, by comparison, the statistical errors are avoided. However, the accuracy depends completely on the design of the decision tree and the selected features.

The minimum distance classifier is used to classify new image data to classes that minimize the distance between the image data and the class in multi-feature space as shown in Figure 2-11. The distance is defined as an index of similarity so that the minimum distance is identical to the maximum similarity. The distances which are often used in this method include Euclidian distance when the variance of the population classes is different to each other; Normalized Euclidian distance when there is a difference in variance; and Mahalanobis distance when there is a correlation between axes in feature space.

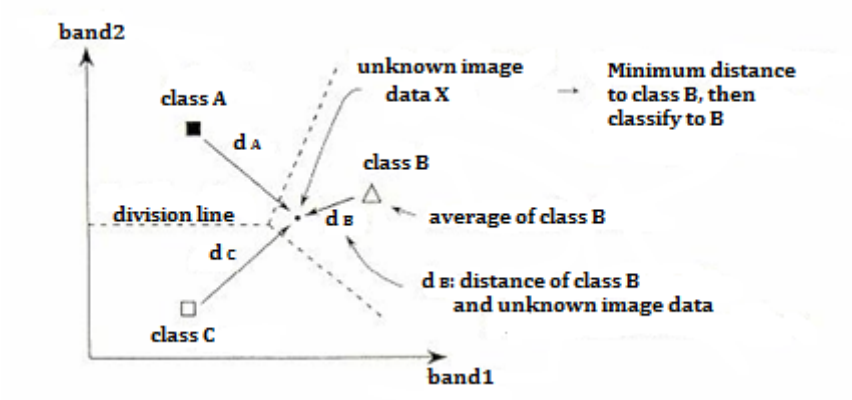


Figure 2-11: Concept of minimum distance classifier (Rahman and Afroz, 2013)

The maximum likelihood classifier is another popular method of classification in computer vision and remote sensing. It is based on a statistical decision criterion to assist in the classification of overlapping signatures where a pixel with the maximum likelihood is classified into the corresponding class of highest probability. When the variance-covariance matrix is symmetric, the likelihood is

the same as the Euclidian distance, while the determinants are equal to each other; the likelihood becomes the same as the Mahalanobis distances. The maximum likelihood method has the advantage of being more accurate methods, however, at the expense of extra computation processing time.

The multiple trained cascaded Haar classifier algorithms have been used in object detection and recognition applications. This classifier was originally developed for face detection, however, it is not restricted to just detecting faces. Therefore, as described by (Bradski and Kaehler, 2008), it can be used to classify any moving or stationary object that is typically rigid and has blocky features that make it distinct. Based on the work published by (Monteiro et al., 2006) to detect pedestrians, it proves that this classifier is a robust and rapid approach for object detection. It forms a copulative set of weak classifiers into a strong classifier. Each weak classifier uses rectangular areas, called Haar-like features as shown in Figure 2-12, to compare with the trained features of the desired object in a given orientation in the image. The Haar feature values are computed as the sum of differences between differing rectangular sub-regions at a localized scale which, although limited in scope as individual features, can be computed extremely efficiently. Individually, they are weak discriminative classifiers, but when combined as a conjunctive cascade, a powerful discriminative classifier can be constructed, capable of recognizing common structures over varying illumination, base colour and scale.

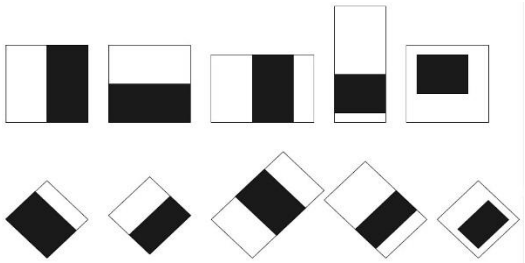


Figure 2-12: Types of Haar-like features (Viola and Jones, 2004)

The classifier is trained using a set of a few hundred positive and negative objects training images (separate set of positives and another for the

negatives). The use of boosting techniques then facilitates classifier training to select a maximal discriminant subset of these Haar-like features, from the exhaustive and over the complete set, to act as a multi-stage cascade. In this way, the final cascaded Haar classifier consists of several key simpler (weak) classifiers that all forms a stage in the resultant complex (strong) classifier. These simpler classifiers are essentially degenerative decision-tree classifiers that take the Haar-like feature responses as input to the weak classifiers and return a Boolean pass/reject response. A given region within the image must then achieve a pass response from all of the weak classifiers in the cascade to be successfully classified as an instance of the object that the strong overall classifier has been trained upon. The classifier is then evaluated over a query image at multiple scales and multiple positions using a search window approach. Despite this apparent exhaustive search element of the classifier, the nature of the cascade (sorted in order of most discriminating features) allows earlier rejection of the majority of such windows with a minimal evaluator (and hence computational) requirement. In this way, the Haar cascade classifier thus successively combines more complex classifiers in a cascade structure which eliminates negative regions, as early as possible, during detection but focuses attention on promising regions of the image. This detection strategy dramatically increases the speed of the detector, provides an underlying robustness to changes in scale and maintains achievable real-time performance. Moreover, it is capable of detecting both static and moving objects within the scene.

The algorithm that is proposed in this project to identify and recognize third-party interference is Haar classifier. Where each one classifies specifically trained object features of the known third-party activities at different categories of orientations and then uses a matching algorithm in the test image.

2.6.3 Oil Leak Monitoring

Another important factor that could help with increasing the integrity of the pipeline route is a vision based real-time, oil leak detection system. Leaks, which may be caused by any defects on the pipeline, can be detected using

RGB imaging. The leak is detected as a slick of petroleum around the pipeline. In the optical image scene, the visual descriptions of this oil slick could be identified, in general, by using a surface texture or feature colour variation. However, these may benefit from other cues such as shape or time coherence.

The surface texture is defined by three characteristics of surface roughness, waviness, and form. It could be used to describe the surface properties of an oil slick. Many applications have been developed to describe the surface texture in computer vision. They include road texture (Paquis et al., 2000), human skin texture (Fiedler et al., 1995; Kenet et al., 1998), fabric texture (Kumar, 2008), and composite material texture (Chen et al., 2011).

Colour variation is based on the wavelength variance of the reflected, emitted, or transmitted light. This information could be used efficiently to extract the oil slick from the surrounding pipeline. In computer vision objects can be identified by using colour variation analysis, which is much faster than using texture analysis. Again in the literature, different applications can be found that addresses how to extract the objects based on the colour variation; such as roads (He et al., 2004), human skin (Hjelmås and Low, 2001; Kovac et al., 2003), and fires (Ahuja, 2004).

In this project, the same approach of classification of the colour variation to extract the oil slick around the pipeline is proposed.

2.7 Wireless Communications

Today, wireless communication has become the most popular method for data and information exchange technology. It provides a real-time data transmission over a distance without the need to use any wires or cables. The data from the platform is received at the ground control station in near real-time. Recently, many standard Wi-Fi technologies have been developed (Toshihiro, 2010) to transmit and receive the data wirelessly using different standards such as IEEE 802.11b, IEEE 802.11g, and IEEE 802.11a.

Some factors were taken into account in this project when it came to selecting the most efficient Wi-Fi method. These include coverage range, transmission speed, and power consumption to trade-off between them. As shown in Figure 2-13, the coverage ranges vary from 50 to 150 meters. The rate of data transmitted (bandwidth efficiency) range is from 1.6 Mbps to 54 Mbps.

The proposed Wi-Fi method in this project is IEEE 802.11g. This method is an extension to IEEE 802.11b. It transmits the data at 54Mbps and covers up to 150 meters within 2.4 GHz. It has lower power consumption than the other, and the devices are physically smaller. It is also worth mentioning that, it is cheaper than the other options.

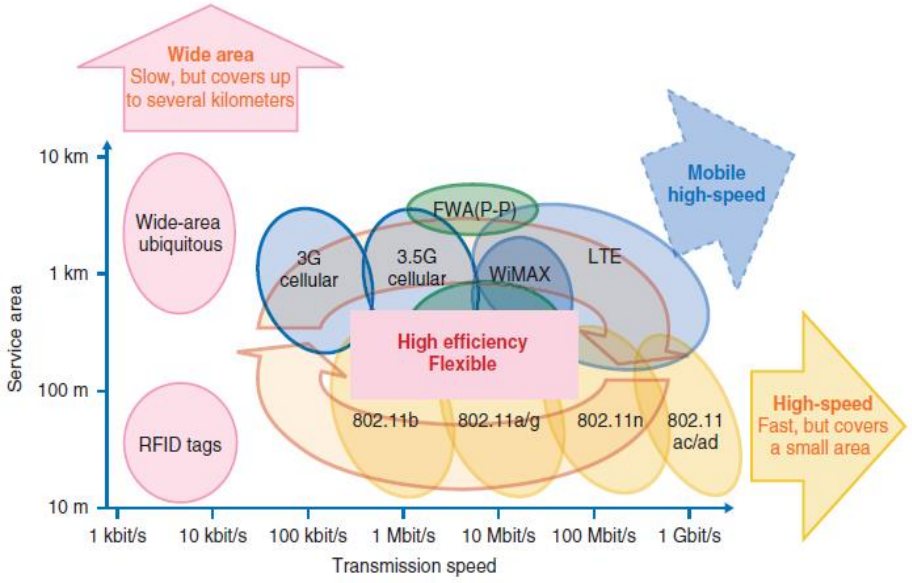


Figure 2-13: Directions of wireless technologies (Toshihiro, 2010)

2.8 Pipeline Monitoring Costs

The cost-benefit examples are considered based on the Gasunie pipeline system (Palmer, 2002). The study, which was sponsored by seven companies, shows that using satellites for surveillance costs approximately 3.5 times as much as using helicopters (manned air-vehicle), for obtaining the same benefit, as shown in Table 2-3.

Table 2-3: A summary of cost-benefit comparison result (Palmer, 2002)

	Satellite	Helicopter
Frequency of survey (days)	14	14
Basic cost per km ($)	16	3.5
Fraction of activities detected	0.41	0.39
Cost benefit ($)	6,336,221	1,771,411

2.9 Chapter Summary

This literature review chapter presented a review of related research and work found in the open literatures that were carried-out by other researchers and institutes. The major cause of defects in the oil and gas pipeline is third-party interference. The main characteristics of some of the existing pipeline infrastructure in the world were outlined and presented, such as configurations, dimensions, and diversity of the environments. The existing built-on/in monitoring systems are effective in terms of cost and effort of installation and maintenance but have difficulties to deal with ageing and long pipelines in complex terrains. It may be concluded that the most suitable aerial techniques are to be used in this project visual depth and colour information. However, as for the type of platform, Unmanned Air-vehicle (UAV) was found to be suitable platform that could be used for remote sensing applications, with many benefits compared to the other aerial platforms. For following and tracking the pipeline, a waypoints navigation technique will be used.

The technique that was proposed to identify the pipeline endpoints is based on vision information, and it is a combination of Hough transforms and RANSAC algorithms. As for the proper technique to detect third-party activities Haar classifier is proposed. To detect oil leaks, colour variation and classification technique is proposed, to extract the oil slicks around the pipeline. Finally, to relay the video data from the aerial platform to the GCS, the IEEE 802.11g technique is proposed.

Chapter 3
General System Overview and Experimental Setup

3.1 Introduction

This chapter presents an overview of the vision-based aerial pipeline surveillance system structure. This system was proposed based on the research requirements and limitations which are also described. In general, the architecture of this system includes both the hardware and software components. These components are distributed into three parts; namely, the aerial platform, Ground Control Station (GCS), and the test-rig. The aerial platform is proposed to carry out the hardware components, which also includes the embedded systems board to run the software that was developed to carry-out the surveillance mission. The Ground Control Station (GCS) is also proposed to assist in performing the pipeline structure surveillance mission by executing part of the image processing algorithm onsite. Finally, the test-rig is used to validate the system in real-time running both the pipeline detection and monitoring algorithms along with third-party detection and classification. The proposed hardware components in each sub-system are described in detail, including the specifications and the general layout of these main parts.

3.2 System Requirements and Limitations

The Unmanned Aerial Vehicle (UAV) is proposed to be used as the primary aerial platform for monitoring and surveying the pipeline structure in this research. However, this platform must be able to perform the following tasks along the pipeline Right-of-Way (when it is scaled in the test-rig setup, it is assumed to be around 10 m on both sides of the pipeline centreline).

a) Identifying the endpoints of the pipeline structure, in real-time, based on the visual depth information, relative to the depth sensor frame. To localize the pipeline segment endpoints, online for the purposes of auto tracking and detecting the objects in vicinity of the pipeline.
b) Tracking the pipeline structure without using a GPS, in real-time, based on the localization of the pipeline segment endpoints to keep monitoring the route of the pipeline structure.
c) Detecting the third-party activites within the Right-of-Way of the pipeline route, in real-time, based on the localization of the pipeline segment endpoints to monitor the pipeline structure from any threat in the vicinity.
d) Relaying the data into the ground station in near real-time to alert the ground operator of any issues/third-party interference.

3.2.1 The Operational Requirements

The operational requirements of the system are listed below:

1) Feature identification and detection should be at sensitivity and specificity rates of at least 75%.
2) Feature identification and detection should have false negative and false positive rates of no more than 25%.
3) Localization accuracy of the features should be within a few meters at full scale.
4) 10 m data coverage is required at each side of the pipeline corridor.

5) It must be able to monitor the pipeline infrastructure at least once a day.
6) Data transmission must be in near real-time.

3.2.2 Limitations

The selection of the complete system (i.e. ground and air) components and equipment was made taking into account the limitations and constraints listed in Table 3-1.

Table 3-1: Hardware constraints

Objectives	**Measure for effectiveness of solution**
Max payload	~ 1000 g
Operation	Indoor at low-level (1 m – 4 m)
Power capacity	~ 4900 mAH
Flight time	A total flight time of at least 10 minutes for the quadcopter with the entire required payload.
Pipeline structure	5 cm overground 3-5 m length, 1.2 cm width. (1m:50m scale)
Communication range	~ 10 meters for indoor test purpose
OS support	Linux Ubuntu 10.10, X86, 32/64 bit
Depth Image	VGA (640x480): 30 fps
GCS-PC processor	Intel Core i5 2.4 GHz processor and 8 GB RAM memory.
Embedded processor	Quad-Core ARM® Cortex A9 processor at 1GHz RAM memory of 1GBytes of 64-bit wide DDR3 @ 532MHz

3.3 System Architecture

This section presents the proposed high-level system architecture of the visual-based, pipeline structure surveillance as shown in Figure 3-1. This architecture is divided into three parts, which are the aerial platform, ground control station, and router.

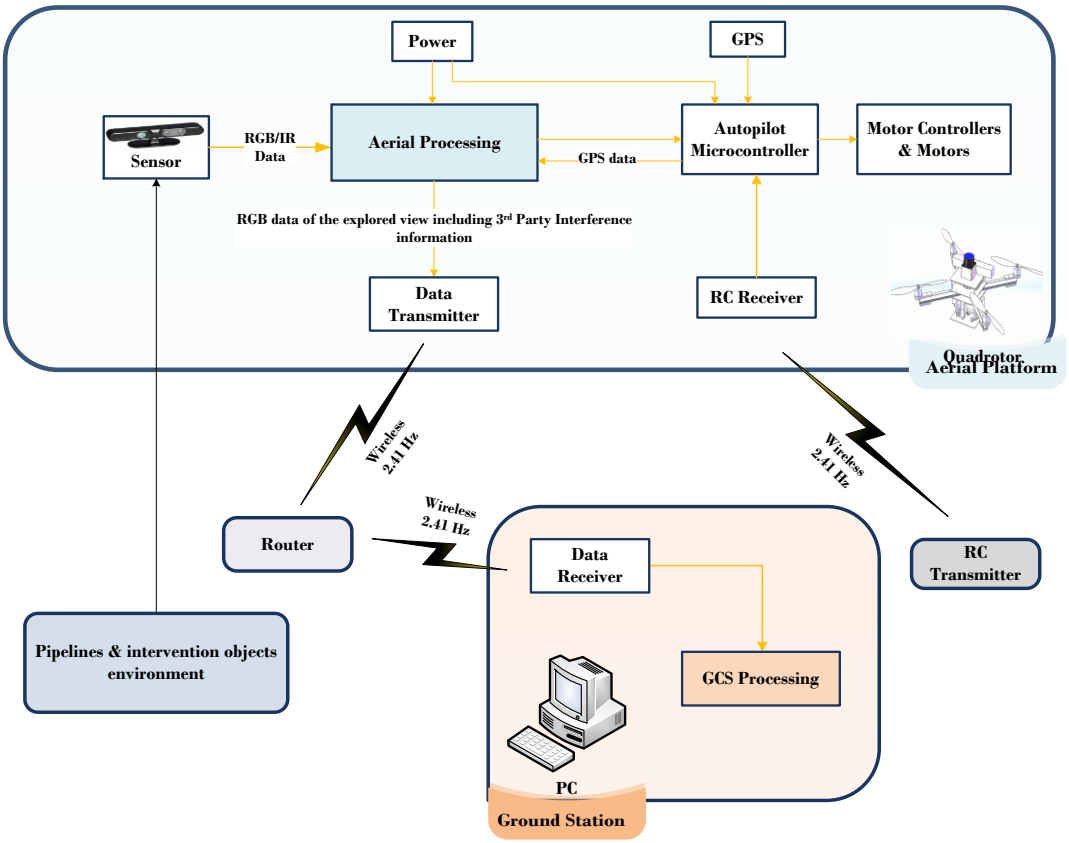


Figure 3-1: General system architecture of the vision-based aerial pipeline surveillance system

The aerial platform payload includes the remote sensor that provides visual data of the pipeline structure at high resolution. On-board data processing is carried-out and the data is transmitted to the ground station in real-time, so to track the pipeline structure automatically. The aerial platform consists of an EO/IR camera for image capture, a data recording, an on-board embedded processor for algorithms processing, autopilot controller to navigate the platform, power supply and, finally, a data link for the UAV command and control and data transmission. The ground control station receive/transmit the recorded data, process the computationally demanding parts of the developed algorithms, and stream the video data of the explored environment to the ground operator. The ground station consists of a workstation to process the algorithms and a data link for the UAV command, control and data transmission. The router is proposed to simply pass the data between the aerial platform and

the ground station. The test-rig is proposed to evaluate and validate the system which involves a small-scale pipeline structure and small samples of third-party interference objects.

3.4 Aerial Platform

This section describes, in detail, the integrated hardware components and the developed software on-board the aerial platform. Also, it describes how to setup and interface them together.

3.4.1 Hardware Components

This section presents the hardware components that are on-board the aerial platform as part of the surveillance system. The hardware includes the following components:

1) The Gaui 500X Quadrotor (aerial UAV platform).
2) Asus Xtion Pro Live (RGB/depth vision sensor).
3) Nitrogen x6 board (processor board to run a computer vision algorithm).
4) ArduPilot Mega (microcontroller board to control the dynamics of the platform with assisting of the built-in autopilot, IMU, and GPS).
5) 3DR uBlox (GPS Module).
6) XBee module (Wi-Fi RF Module for wireless telemetry data transmissions).
7) Li-Po battery (Power supply for the embedded boards).

More information is described in the following subsections.

3.4.1.1 Gaui 500X Quadrotor

The Gaui 500X Quadrotor is a light weight UAV platform as shown in Figure 3-2 that is proposed to carry out the imagery sensor payload and the other corresponding components to perform in real-time the pipeline surveillance mission automatically (Multi-Rotor Technology (MRT), 2012).

The efficiency and load of the propeller are optimized, for improved wind resistance over other quads. It utilizes GU-344 (three-axis stabilizing system) for both beginners and professionals in combination with other electronics for FPV flying that makes it a very stable air-vehicle. The maximum flying weight is between 1100g to 2000g, depending on payloads used, such as batteries, 500X GAUI Motors/ESCs, cameras, processing boards and other related equipment. The main reason for selecting this platform is the maximum take-off weight, which is the total weight of the payload components and the UAV platform itself with all the systems as shown in Table 3-2.

Table 3-2: Weight details of the payload components

Item	**Weight**
Battery 3S	323g
Crane II	96g
ASUS Xtion Pro Live	218g
Battery 2S	83g
Nitrogen 6x board	88g
Mount	150g
TOTAL	**1000g**

Operation mode and flight characteristics are similar to those of helicopters without complex transmissions. It has a collapsible body design that greatly reduces crash damage and allows for easy repairs. With four brushless motors and 18A Electro Speed Controller (ESC), the 500X is really powerful and responsive. The specifications are listed in Table 3-3 below.

Table 3-3: GAUI 500X Quadrotor specifications (Multi-Rotor Technology (MRT), 2012)

Item	Specification
Weight (g)	670
Max Flying weight (g)	Up to 2000
Platform diameter (mm)	500
Motors	960 kV, scorpion model
Electro Speed Controller (ESC)	Brushless 18A
Flight efficiency	Standard battery (2S 2000mAh), for flight times longer than 12min. With a high-capacity battery, the flight time will be 20 min or longer.

Figure 3-2: GAUI 500X Quadrotor platform (Multi-Rotor Technology (MRT), 2012)

3.4.1.2 ASUS Xtion Pro Live

Two vision data are acquired as main inputs to provide the function of colour (RGB) and the depth (elevation at down projection) information of the vision target scene. So, in the market, many sensors exist to provide them either, individually, using two separate sensors or together as one sensor. However, using one platform that combines all of them is superior in terms of cost, weight, interface, integration and operation. Hence, the proposed sensor used in this project is the ASUS Xtion Pro Live. This sensor provides combined functions of

RGB colour information and per-pixel depth information. The sensor includes RGB camera; IR structured light source, and a second camera dedicated to IR detection that provides depth information as shown in Figure 3-3.

Figure 3-3: ASUS Xtion Pro Live Sensor (ASUS, 2012)

It was chosen for its low price £138 and low weight of 218 g, as shown in Figure 3-4 that guarantees that the Quadrotor payload limit remains at around of 1kg.

Figure 3-4: Asus Xtion Pro Live weight measurement

The technology of obtaining the depth is based on the structured light technique. Precision is similar to that of Time of Flight (ToF) cameras, 1cm more or less, with a limited range between 0.8-3.5 m, but they tend to have trouble seeing small objects and have to be indoors. The resolution and speed of this sensor are similar to the conventional VGA cameras, usually 640 by 480 at 30 fps. The technical specifications are shown in Table 3-4.

Table 3-4: ASUS Xtion Pro Live sensor specifications (ASUS, 2012)

Item	**Specifications**
Power consumption	< 2.5W
Depth range	0.8m to 4m
Field of view	58° H, 45° V, 70° D (Horizontal, Vertical, Diagonal)
Sensors	RGB & Depth & Microphone (x2)
Depth Image Size	VGA (640x480): 30 fps QVGA (320x240): 60 fps
Resolution	SXGA (1280*1024)
Platform	Intel X86 & AMD
OS support	Windows 7 32/64, XP, Vista Linux Ubuntu 10.10, X86, 32/64 bit Android
Interface	USB 2.0
Software	OpenNI SDK bundled
Programming language	C/C++ (Windows), C++ (Linux), and JAVA
Dimensions	180mm x 35mm x 50mm
Operating Environment	Indoor
Weight	218g

The IR camera and the IR projector form a stereo pair with a known baseline. Hence, a structured light technique works by projecting a fixed pattern of infrared light from the IR projector such as a grid of lines, or a constellation of points or dark speckles on top of the scene's objects as shown in Figure 3-5. This pattern is seen distorted when looked at from a perspective different from the projector. By analysing this distortion, information about the depth can be retrieved, and the surface reconstructed.

Depth is calculated by triangulation against a known pattern from the projector. The pattern is memorized at a known depth. For a new image, the depth is calculated at each pixel, and a small correlation window (9x9 or 9x7) is used to compare the local pattern at that pixel with the memorized pattern at that pixel and 64 neighbouring pixels in a horizontal window. Then, the best match gives

an offset from the known depth, regarding a pixel which is called disparity. The device performs other interpolation of the best match to get the sub-pixel accuracy of 1/8 pixels that provide the known depth of the memorized plane, and the disparity. However, the estimated depth of each pixel is calculated by triangulation.

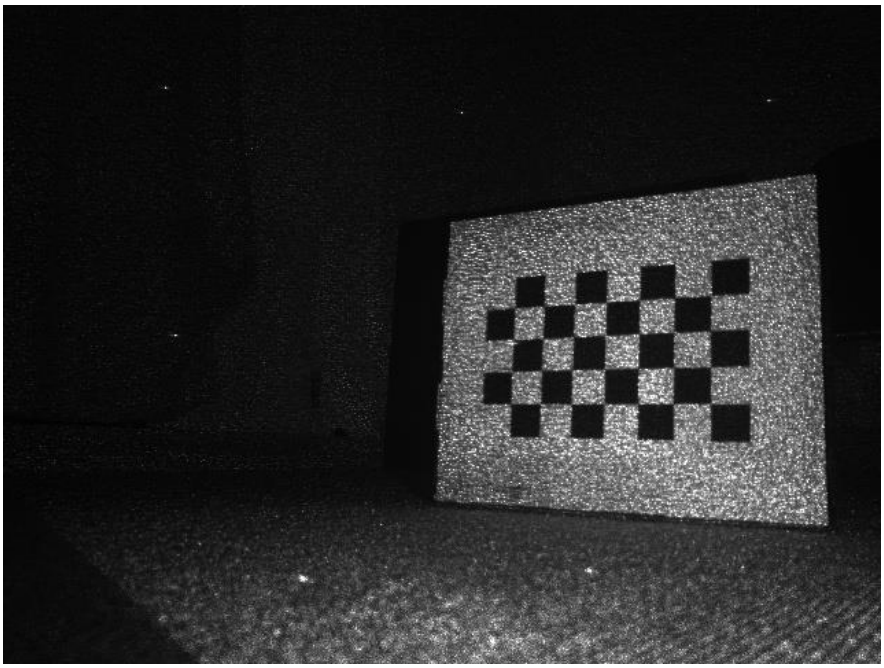


Figure 3-5: Sensor projection of infrared structured light pattern

Then, the depth of each pixel produces 3D data in the form of a point cloud. A point cloud is a set of points in three-dimensional space, each with its XYZ coordinates. So, every point corresponds to exactly one pixel of the captured images in the case of stereo, ToF or structured light cameras.

To mount the Asus Xtion Pro Live on the Gaui 500X Quadrotor platform, a step-down configuration Gimbal was selected to integrate them which is called Gaui Crane II, as shown in Figure 3-6. Moreover, the Asus Xtion Pro Live hardware is connected to the Embedded Board using a USB cable.

Figure 3-6: Gaui Crane II Gimbal

To interact between the pixels of the depth and colour images with their corresponded real world coordinates, a 3D real world model needs to be constructed. To interact with the depth and colour images, an accurate camera calibration for depth and colour images is carried-out offline. This calibration is a process performed to compute the true intrinsic parameters of the camera, such as focal length, centre image position, and lens distortion. Besides, computing the extrinsic parameters (relative transform between the depth and the colour cameras), it also includes rotations and translation parameters.

The calibration was performed offline on the colour and depth cameras to find the intrinsic and extrinsic parameters by using the same approach as (Herrera et al., 2012). This approach was designed to calibrate simultaneous multicolour cameras, a depth camera, and the relative transform between them. First, the intrinsic parameters of both colour and depth cameras are calibrated. Then, the relative transform between the cameras (extrinsic calibration) was calibrated by computing the rotation and translation parameters to enable the alignment between them.

The approach requires only a planar surface with a simple checkerboard pattern to be imaged from various poses as shown in Figure 3-7. The checkerboard corners provide suitable constraints for the colour images while the planarity of the points provides constraints on the depth images. The pixels at the borders

of the calibration object can be ignored and, thus, depth discontinuities are not used. For this purpose, Kinect Calibration Toolbox for Matlab was used.

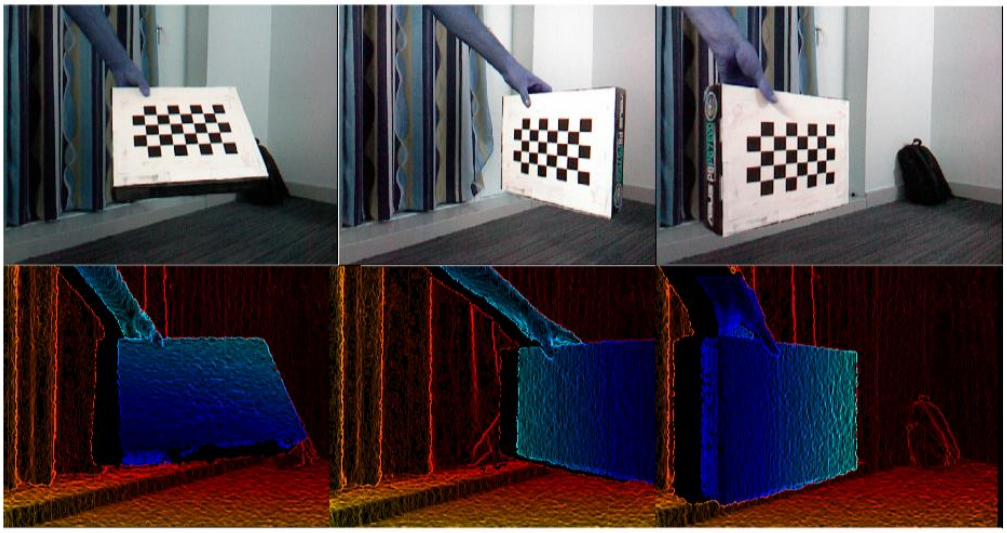


Figure 3-7: Samples of calibration images at various poses

For the colour camera, the checkerboard corners are extracted from the colour intensity image using a corner detection algorithm in image processing to obtain the colour intrinsic parameters as shown in Figure 3-8. Then, a homography is computed for each colour image using the known positions of the corners in world coordinates and the relative pixels in the colour image coordinates. After that, each homography imposes constraints on the intrinsic parameters that are solved with a linear system of equations. The distortion coefficients are initially set to zero.

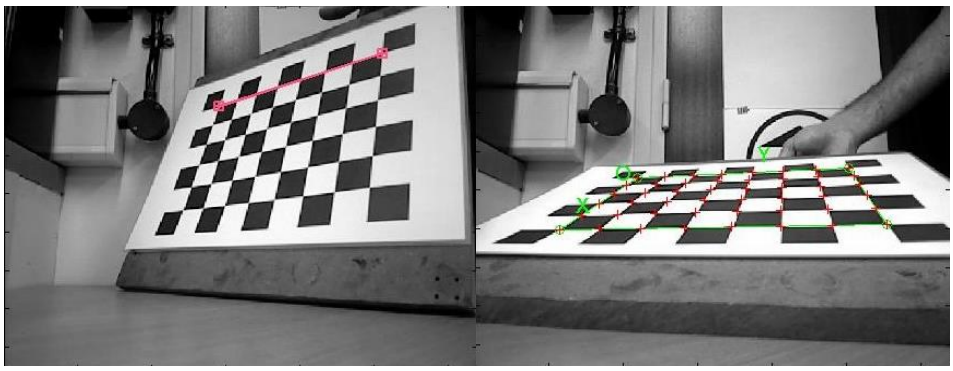


Figure 3-8: Checkerboard corner detection in a colour image

The same method is applied to the depth camera to obtain the depth intrinsic parameters, in which the noisy four corners of the plane are extracted manually in the depth image as an initial guess because the checkerboard is not visible, as shown in Figure 3-9. Then, a homography is computed for each depth image using the corner positions in plane coordinates and the relative pixels in the image depth coordinates. This computation will produce the focal lengths, centre points, and the transformation parameters.

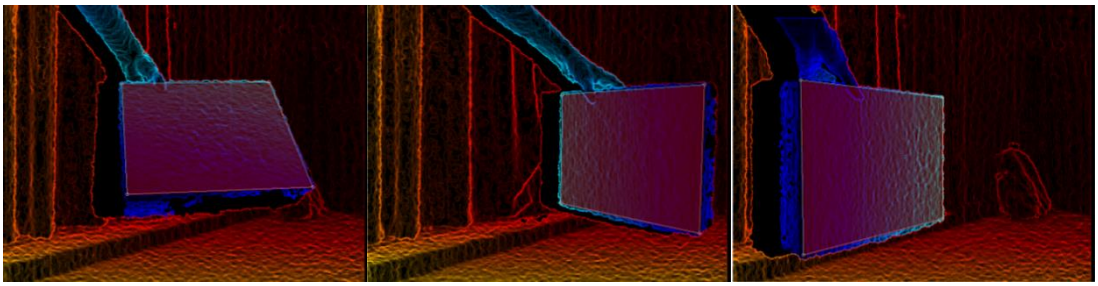


Figure 3-9: Plane's corner detection in depth image (red boxes)

3.4.1.3 Nitrogen 6x Embedded Board

In order to increase the level of autonomy at the quadrotor, an embedded board was added on-board the platform to process the computer vision algorithms that control the dynamic of the platform in real-time. The embedded processor board that is proposed in this project is based on the Nitrogen 6x as shown in Figure 3-10.

Figure 3-10: Nitrogen 6x embedded board (Boundary Devices, 2012)

This specific embedded board was chosen for several reasons, such as the limited size and weight and the built-in Wi-Fi module, as shown in Figure 3-11 that was used to communicate with the ground control station (GCS).

Figure 3-11: Nitrogen 6X Add-on Ti-Wi Module (Boundary Devices, 2012)

The Nitrogen 6X is a highly integrated development system based on the next generation Quad-Core ARM-Cortex A9 processor from Freescale, the i.MX6. This processor supports a wider and faster memory bus (64-bit DDR3 1066 MHz), integrated HDMI, Gigabit Ethernet and additional display channels with a high level of integration. It is a low-cost development platform. Additional highlights of the board are listed below (Boundary Devices, 2012):

- Quad-Core ARM® Cortex A9 processor at 1GHz.
- RAM memory of 1GBytes of 64-bit wide DDR3 @ 532MHz.
- Board Dimensions: 4.5″ x 3″.
- 2MB Serial Flash.
- Three display ports (PRGB, LVDS, HDMI).
- Parallel camera port with OV5642 Interface.
- Multi-stream-capable HD video engine delivering 1080p60 decode, 1080p30 encode and 3D video playback in HD.
- Superior 3D graphics performance with quad shaders for up to 200 Mt/s.
- Separate 2-D and/or Vertex acceleration engines for an optimal user interface experience.
- Serial ATA (SATA).
- Dual SDHC card slots.
- PCI express port.
- Analog (headphone/mic) Audio.

- 10/100/1G Ethernet with Power over Ethernet support.
- 2 RS-232 Serial ports.
- 10-pin JTAG interface.
- I2C/GPIO/SPI.
- High-speed USB ports (2xHost, 1xOTG).
- CAN port.
- Real Time Clock.
- Ti-Wi 802.11 b/g/n Wi-Fi+BT.
- Supports Android 4.3, Embedded Linux, and WinCE7.0 Operating Systems.

The Nitrogen 6x board runs a Linux-based operating system. The computer vision algorithms were coded in Python with the aid of OpenCv and OpenNi libraries.

The Linux operating system installed on the Nitrogen6x is Debian GNU-Linux. Debian is a Linux distribution with access to online repositories hosting software packages. Debian officially hosts free software in its repositories but also allows commercial 3rd party software to be installed. Furthermore, it gives the possibility to install all the dependencies and libraries required to use the Asus Xtion Pro Live.

The Open Natural Interaction (OpenNI) is a multi-language framework and an open source application that is used to interface the Xtion Pro Live sensor on this system by providing functions capable of acquiring depth data from the Asus Xtion Pro Live. It could be used with any depth sensor such as Xbox Kinect or Asus Xtion.

The OpenNI framework gives the interfaces either for the physical apparatus or the middleware components, as seen in Figure 3-12, by compiling it with OpenCV library (OpenNI, 2012).

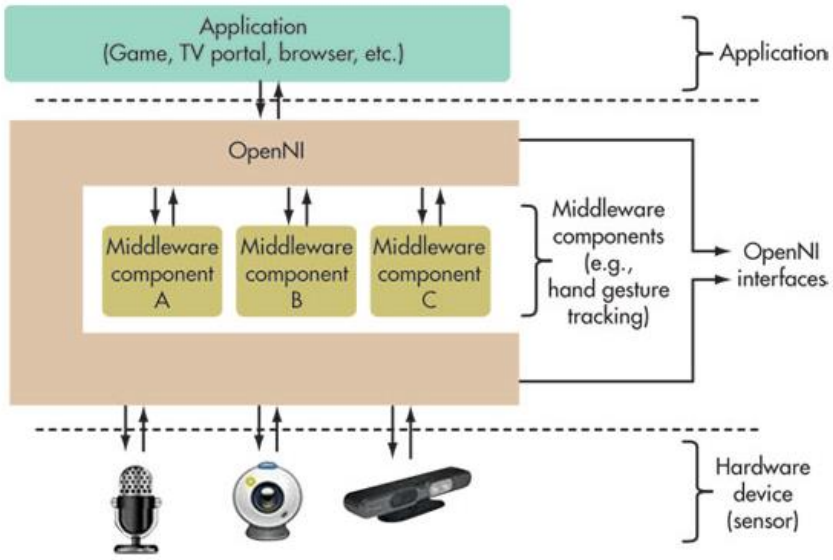


Figure 3-12: OpenNI Framework (OpenNI, 2012)

OpenCV is an Open Source Computer Vision library that includes many built-in algorithms, as described by (Opencv dev team, 2012). It is free for both academic and commercial use. It has C++, C, Python and Java interfaces and supports Windows, Linux, Mac OS, iOS and Android. OpenCV was designed for computational efficiency and has a strong focus on real-time applications. Written in optimized C/C++, the library can take advantage of multi-core processing. Enabled with OpenCL, it can benefit from the hardware acceleration of the underlying heterogeneous compute platform.

OpenCV has a modular structure, which means that the package includes several shared or static libraries. The following modules are available (Opencv dev team, 2012):

- Core - a compact module defining basic data structures, including the dense, multi-dimensional array Mat and basic functions used by all other modules.

- imgproc - an image processing module that includes linear and nonlinear image filtering, geometrical image transformations (resize, affine and perspective warping, generic table-based remapping), colour space conversion, histograms, and so on.
- Video - a video analysis module that includes motion estimation, background subtraction, and object tracking algorithms.
- calib3d - basic multiple-view geometry algorithms, single and stereo camera calibration, and object pose estimation, stereo correspondence algorithms, and elements of 3D reconstruction.
- features2d - salient feature detectors, descriptors, and descriptor matchers.
- Objdetect - detection of objects and instances of the predefined classes (for example, faces, eyes, mugs, people, cars, and so on).
- Highgui - an easy-to-use interface to video capturing, image and video codecs, as well as simple UI capabilities.
- gpu - GPU-accelerated algorithms from different OpenCV modules.

To interface the Nitrogen 6X embedded board to the Asus Xtion Pro live hardware, the attached Xtion's USB cable is plugged into one of the USB ports onboard. Furthermore, the Nitrogen6x is interfaced to the Ardupilot Mega via the serial port UART1.

3.4.1.4 ArduPilot Mega Microcontroller Board

The proposed hardware to control the Quadrotor platform is the ArduPilot Mega (APM). APM is a fully programmable open source autopilot system that requires a GPS module and sensors to create a functioning Unmanned Air-vehicle (UAV) (ArduPilot, 2013). It can control many platforms, such as fixed-wing air-vehicle, multi-rotor helicopters, traditional helicopters, as well as ground rovers. It has full autopilot capability for autonomous stabilization, waypoint based navigation and two-way telemetry using XBee wireless modules. Also, it supports 8 RC channels with four serial ports.

This controller consists of the main processor board as shown in Figure 3-13, and the APM firmware that is a code runs onboard to control the chosen UAV platform as shown in Figure 3-14.

The ArduPilot Mega was selected due to the following features (ArduPilot, 2013):

- Controller designed to be used with autonomous air-vehicle, multicopters (tri, quad, hex, oct, and so forth), traditional helicopters, car or boat.
- Based on a 16MHz Atmega2560 processor.
- Built-in hardware failsafe that uses a separate circuit (multiplexer chip and ATMega328 processor) to transfer control from the RC system to the autopilot and back again. Includes ability to reboot the main processor in mid-flight.
- Dual-processor design with 32 MIPS of on-board power.
- Supports of 3D waypoints and mission commands (limited only by memory).
- It comes with a 6-pin GPS connector (EM406 style).
- It has 16 spare analog inputs (with ADC on each) and 40 spare digital input/outputs to add additional sensors.
- Four dedicated serial ports for two-way telemetry and in-flight command using the powerful MAVLink protocol.
- It can be powered by either the RC receiver or a separate battery.
- Hardware-driven servo control, which means less processor overhead, tighter response and no jitters.
- The autopilot can process eight RC channels (including the autopilot on/off channel).
- LEDs for power, failsafe status, autopilot status and GPS lock.
- 4MB of on-board data logging memory. Missions are automatically data-logged and can be exported to KML.
- It has full autopilot software, including IMU and ground station/mission planning code.

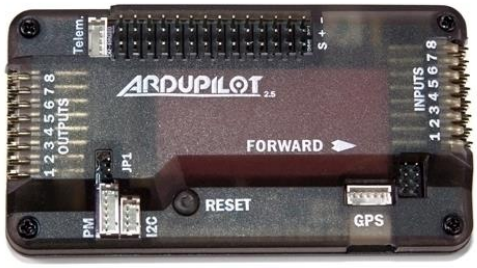


Figure 3-13: ArduPilot Mega Autopilot board (ArduPilot, 2013)

Figure 3-14: ArduPilot Mega onboard firmware (ArduPilot, 2013)

The communication between the Nitrogen 6x and the ArduPilot has been obtained using a serial port UART with the following configuration parameters:

- Baud rate=9600
- Stop bit=1
- Parity=None

The software has been developed for both the Nitrogen 6x and the ArduPilot. Concerning the embedded board, which works with a Linux-based OS, the “Termios” library was used for the ArduPilot, while the Arduino integrated

development environment (IDE) provides the main tools for serial port programming.

The anatomy of a program performing serial I/O is as follows:

- An open serial device with standard system called open.
- Configure communication parameters and other interface properties with the help of specific functions and data structures.
- Use system calls read and write for reading from, and writing to the serial interface.
- Close device with the system called close when done.

3.4.1.5 3DR uBlox GPS

The 3DR uBlox GPS module has an on-board compass kit, and it is the most recommended and compatible module to be used with the APM mounted on the aerial platform. This module provides the telemetry data of the aerial platform to the Ground Control Station (ArduPilot, 2013).

Features and Specifications:

- ublox LEA-6H module.
- 5 Hz update rate.
- 25 x 25 x 4 mm ceramic patch antenna.
- LNA and SAW filter.
- Rechargeable 3V lithium backup battery.
- Low noise 3.3V regulator.
- I2C EEPROM for configuration storage.
- Power and fix indicator LEDs.
- Protective case.
- APM compatible 6-pin DF13 connector.
- Exposed RX, TX, 5V and GND pad.
- 38 x 38 x 8.5 mm total size, 16.8 grams.

This GPS is connected with the ArduPilot Mega flight controller Board using two cables, as shown in Figure 3-15, which are:

1. Four-position cable to connect the GPS MAG port to the APM I^2C port.
2. Five-position-to-six-position cable to connect the GPS port to the APM GPS port.

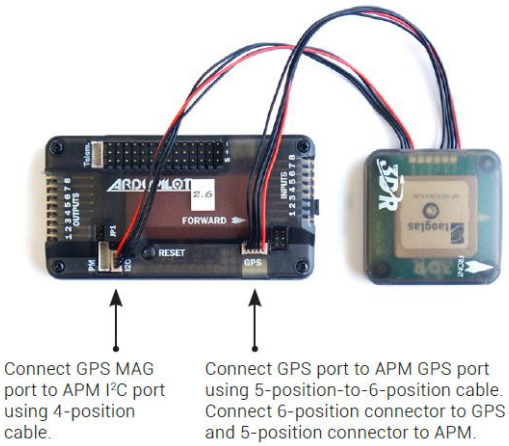


Figure 3-15: 3DR uBlox GPS with ArduPilot Mega hardware interface (ArduPilot, 2013)

The 3DR uBlox GPS with Compass is already pre-configured for compatibility with APM autopilot.

3.4.1.6 Xbee 802.15.4 Transceiver

The XBee-PRO® OEM RF module is an IEEE 802.15.4 compliant solution, that satisfies the unique needs of low-cost, low-power wireless sensors (DIGI, 2014). This module when mounted on the aerial platform can send/receive the telemetry data wirelessly, as shown in Figure 3-16.

The advantages and limitations of this module are as follows:

- Operate at ISM 2.4 GHz frequencies.
- Easy-to-use.

- Requires minimal power and provides reliable delivery of critical data between devices.
- It can send a live telemetry data as the airframe progresses through the mission
- A new mission could be sent while the UAV flies without having to land.
- The range of operation has been tested out for half a mile with no loss in connection; the connection has been found to drop off at 3/4 of a mile.
- It is capable of operating at a temperature rating (-40 to 85°C).
- Approved for use in Europe, Canada, Australia, and the United States.
- It supports advanced networking & low-power modes.

Figure 3-16: XBee transceiver (DIGI, 2014)

Onboard the aerial platform, the XBee module (pins) is connected with the ArduPilot Mega (Teleport) using the 3DR four-wire XBee cable and XBee adapter to interface them together as shown in Figure 3-17.

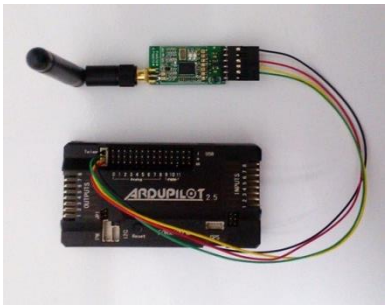


Figure 3-17: XBee & APM hardware interface (ArduPilot, 2013)

Then, to interface them together, it is required to set up the right port and the baud rate (APM default is 57600 bps) by using the moltosenso Network Manager or using the GCS mission planner directly.

3.4.1.7 Li-Po Battery

The Gaui 500X Quadrotor has a standard battery (2S 2000mAh), for flight times longer than 12 min. With a high-capacity battery, the flight time will be 20 min or longer. The Nitrogen 6X board requires a 5V/3A max power source. However, a high discharge Li-Po 1300mAh battery has been chosen using three Cells at 11.1V. A DC/DC converter to 5V has been integrated to provide up to 60 minutes of practical work as shown in Figure 3-18.

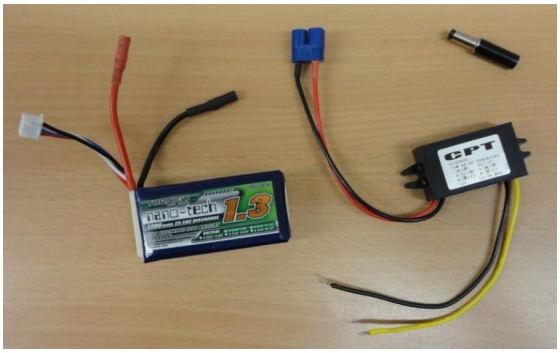


Figure 3-18: Li-Po 1300mAh battery & DC/DC converter

3.4.2 Software Development

This section presents the software development structure for the embedded board on-board processing on the aerial platform. This structure consists of five algorithms as shown in Figure 3-19. At the beginning, pre-processing algorithms are proposed to acquire the optical/depth data from the sensor, and transform the depth from IR (pixels) into 3D point cloud (meter). Then, the second part of the algorithm is required to identify the pipeline segment endpoints using the depth information of the explored environment as an IR (pixels) and 3D point cloud (meters) formats. Once the endpoints of the pipeline segment are identified, the auto tracking algorithm for the pipeline structure is proposed to

navigate the UAV into the course waypoints that are generated online based on the pipeline endpoints and send the commands into the autopilot processor. Simultaneously and finally, an algorithm is proposed to transmit visual information of the explored view into the Ground Control Station (GCS) including IR image, RGB image, 3D point cloud data, plane parameters (A, B, C, and D), and endpoints (EPs) locations of the pipeline segment. Samples of the approach have been coded in this research to develop the algorithms as mentioned in Appendix A using the Python programing language.

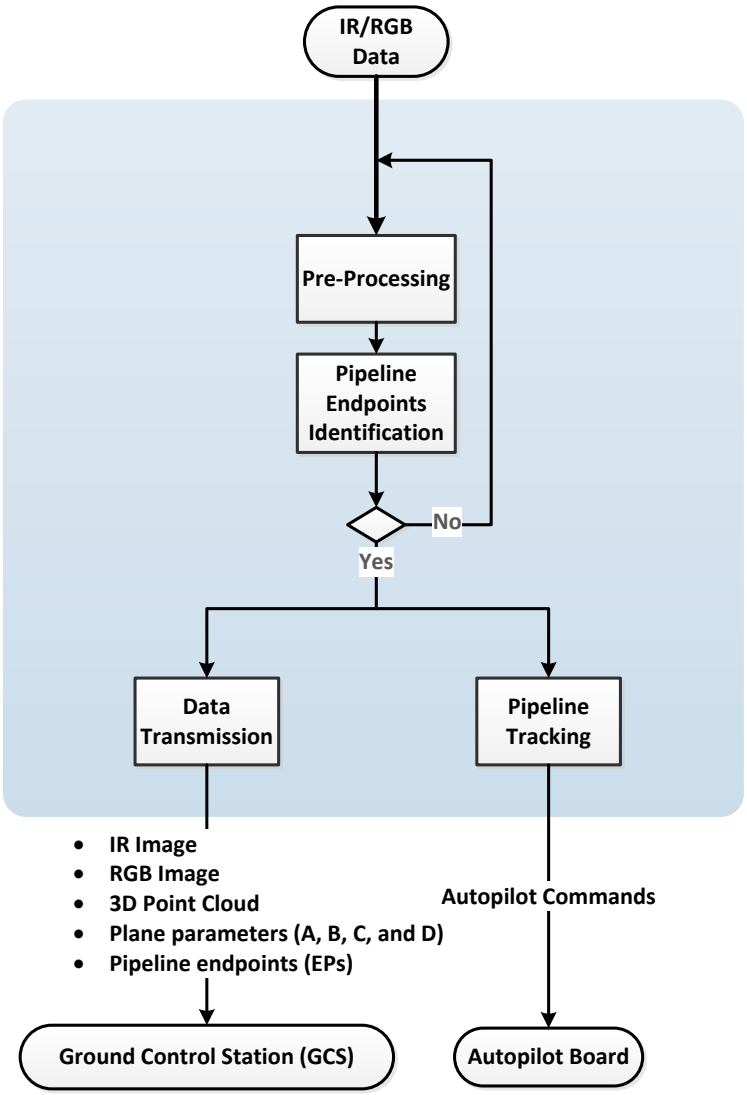


Figure 3-19: The aerial platform software process

3.5 Ground Control Station (GCS)

This section describes the hardware components and their integration and the software development that is used to build and develop the compatible Ground Control Station (GCS) for this project.

3.5.1 Hardware Components

This section presents the hardware components that are built in the ground control station platform to integrate and build the surveillance system. The proposed GCS's platform consists of the following hardware components, as shown in Figure 3-1:

1. Workstation to process the data receiving from the aerial platform and stream them in RGB format in near real-time.
2. XBee transceiver to transmit/receive the telemetry data.
3. Wi-Fi module and Router to pass the data between the aerial platform and ground station.

More information is detailed in the following subsections.

3.5.1.1 Workstation

The workstation of the Ground Control Station can be any of a variety of laptops or desktops. In this project, the PC that was used is a 64-bit Linux operation system that has an Intel Core i5 2.4 GHz processor and 8 GB RAM memory.

3.5.1.2 XBee 802.15.4 Transceiver

The XBee module used in the GCS is similar to the one described in (Section 3.4.7). The only difference is that one was interfaced with the ArduPilot Mega, but here it interfaces with a Notebook PC. However, the easiest way to interface the XBee module with the Notebook PC (USB port) is by using the FTDI cable with FTDI adapter as shown in Figure 3-20.

Figure 3-20: XBee module & FTDI cable for interface with the PC

3.5.1.3 Wi-Fi Module and Router

To communicate with the aerial Quadrotor platform and the GCS wirelessly, it was proposed to use a built-in Wi-Fi module at each one of them with a 2.4 GHz frequency band. So, a hardware router was proposed to link between them which is a networking device used to simply pass data between the wireless networks.

3.5.2 Software Development

In the Ground Control Station (GCS), the proposed software development structure processes are implemented on the ground workstation and consist of three algorithms that are data receiving, third-party interference detection, and data streaming in near real-time as shown in Figure 3-21. The data receiving algorithm receives the data from the aerial platform, such as IR image, RGB image, 3D point cloud data, plane parameters (A, B, C, and D), and endpoints (EPs) locations of the pipeline segment. Then, an algorithm is proposed to detect the third-party interference based on the received information. Finally, the explored view, in near real-time, is streamed to the ground operator, including the detection information of any activity of third-party interference using ground station mission planner software.

Mission Planner is a full-featured ground station application for the APM open-source autopilot project as shown in Figure 3-22. It is used in this project to control the aerial platform and provide the telemetry and video data in real time. The GCS with the Mission Planner allows the UAV Pilot to plan the automated segment of their flight visually. This segment plan is done in a point-and-click

fashion, placing waypoints (WPs) on a satellite map displayed in the Mission Planner. The mission is uploaded to the APM and assigned to a programmable switch on the DX7s Transmitter. During the flight, the Pilot only has to press the switch to initiate the automated mission.

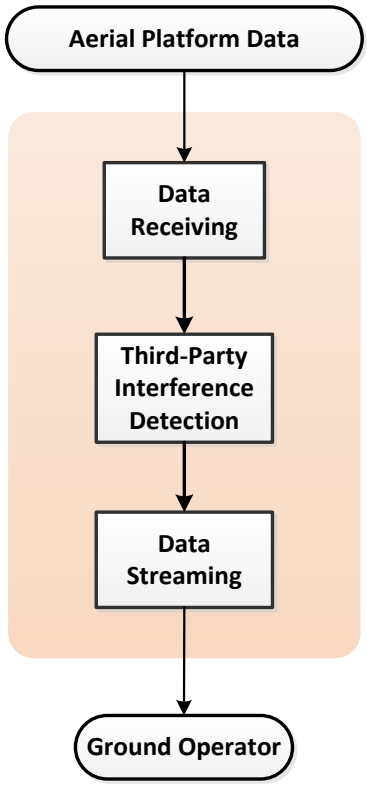


Figure 3-21: GCS software processing

Figure 3-22: ArduPilot Mega GCS application (ArduPilot, 2013)

This application has many advantages such as the following:

- Easy point-and-click waypoint entry, using Google Maps/Bing/Open street maps/Custom WMS.
- Select mission commands from drop-down menus.
- Download mission logs files and analyse them.
- It configures the APM settings of the Quadrotor airframe.
- Interface with a PC flight simulator to create a full hardware-in-the-loop UAV simulator.
- It shows the output from APM's serial terminal.

Due to the high data rate required to be transmitted, such as video and telemetry data, a User Datagram Protocol (UDP) has been selected to perform an efficient and simple connectionless transmission. This protocol does not establish a prior communication when it sends data to the other hosts. To use UDP, an application uses a datagram socket, which binds a combination of an IP address and a service port on both ends, and, as such, establishes host-to-host communication. Data sent to a given socket can be read on a matching socket on the receiving side, this way of working is the so called Server/Client model.

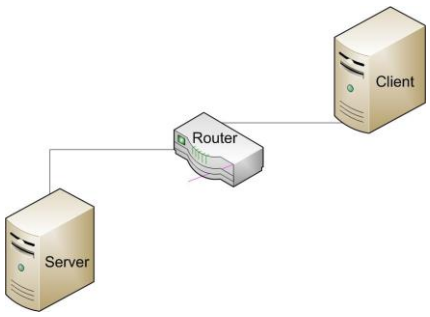


Figure 3-23: Server/Client Scheme

The client/server describes the communication mode of cooperating programs in an application on separate hardware. The server provides a function or service for one or more clients, simultaneously, which initiates communication to requests for such services as shown in Figure 3-23. Two scripts have been

written in Python to achieve this method, one for the Client and one for the Server. The block scheme is shown in Figure 3-24. Figure 3-25 demonstrates the wireless streaming of the depth and RGB images using the server/client networking method.

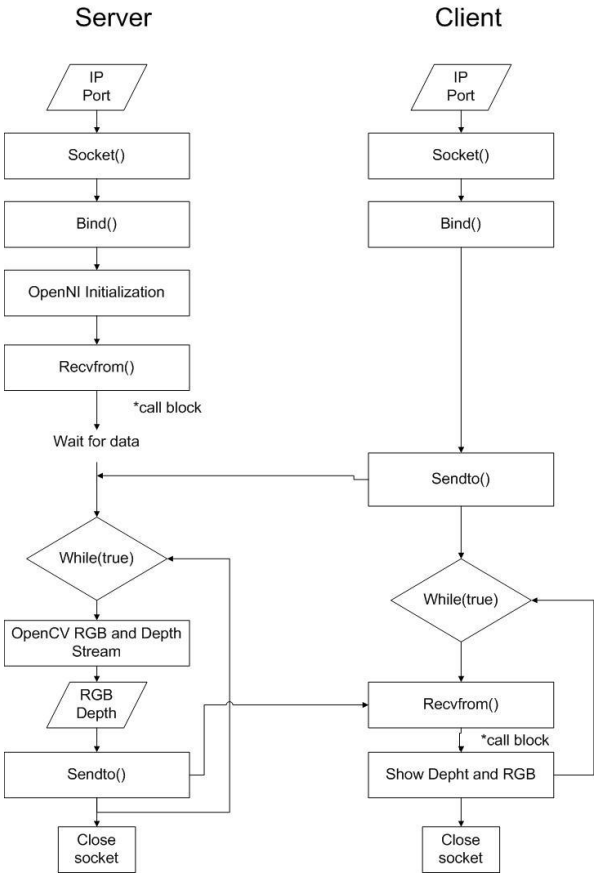


Figure 3-24: Flow chart of communication

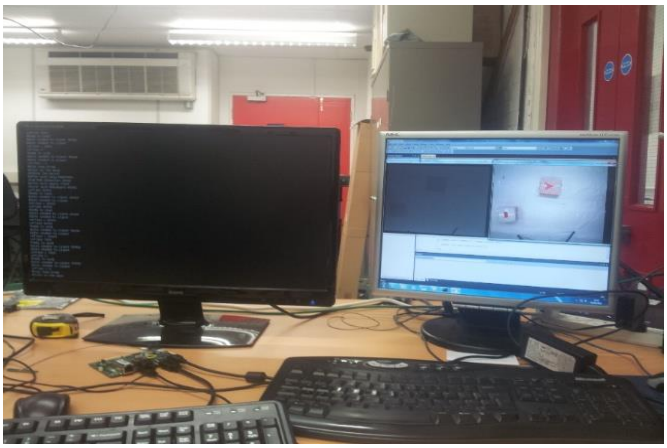


Figure 3-25: Streaming a video wirelessly

3.6 Test Site

The test-rig developed in this project is aimed at testing and validiating the vision based pipeline surveillance system. This rig should include a real or sample components of the following:

- Pipeline structure to identify the endpoints of the segment and track it as shown in Figure 3-26.
- Third-party objects for detecting them as shown in Figure 3-27.
- Moreover, miscellaneous objects to validate the detection.

Figure 3-26: Pipeline structures samples, Ref: Image collection was produced from reference (Oil & Gas Pipeline, 2011)

Figure 3-27: Third-party interference samples, Ref: Image collection was produced from reference (Pipeline's Third-Party Interference, 2011)

3.6.1 A Small-Scale Prototype of a Survey Site

The survey sites in this project are a set of images that involve a pipeline structure and, at least, one third-party interference objects within a small set of that imagery. Due to the struggle of finding a real pipeline structure in the UK and third-party interference, in addition to the vision sensor capabilities and limitations, the set imagery of the survey sites was captured indoors with a small-scale prototype, containing a pipeline structure and samples of third-party interference, as shown in Figure 3-28.

Figure 3-28: Small-scale of the survey sites

3.7 System Integration

However, the full system proposed to monitor the pipeline Right-of-Way, is fully integrated in this project, which includes both the hardware components and software development as shown in Figure 3-29.

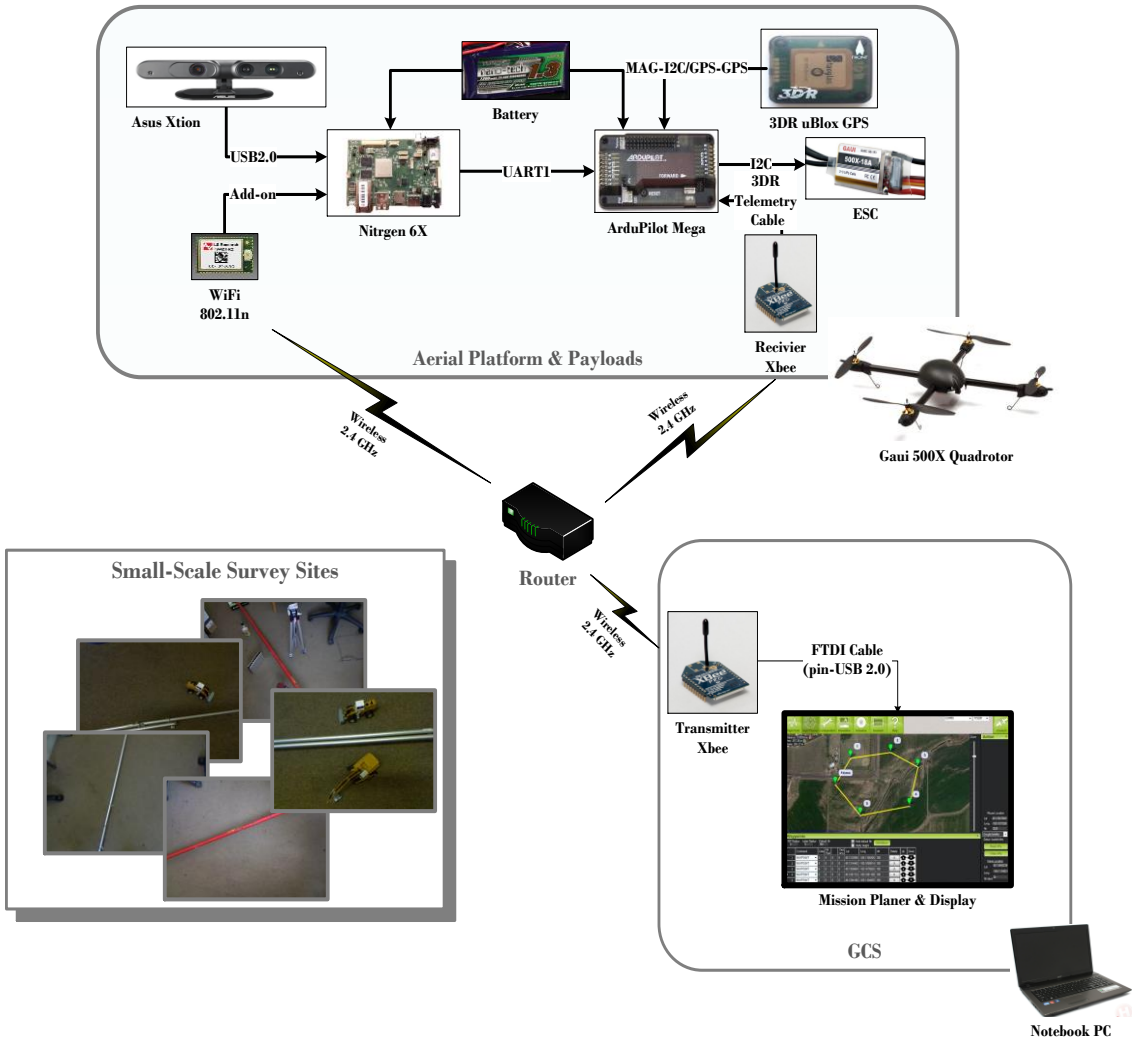


Figure 3-29: Experimental setup layout

The aerial platform for surveying the pipeline and its Right-of-Way is fully integrated and includes the proposed hardware components design, as shown in Figure 3-30, and the developed software, as demonstrated in Figure 3-31.

Figure 3-30: Aerial system integration in Catia design

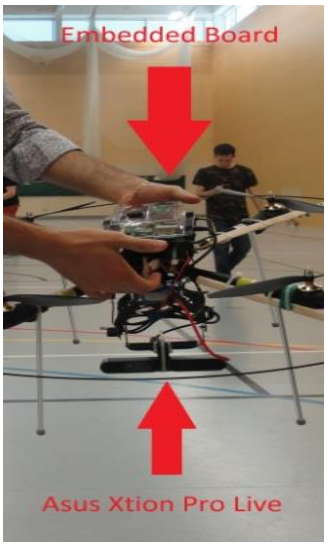


Figure 3-31: Aerial platform & payloads integration

3.8 Chapter Summary

In this chapter, the overall architecture of the automatic pipeline aerial surveillance system, requirements and limitations were presented and described. The architecture describes the high-level design concept of the system that involves both the hardware and software components of the aerial platform, ground staion and test-rig. The system requirements were described in terms of the payload and operational requirements. The limitations of the

system that led to the choice of equipment and sub-systems to meet the requirements, were listed as well. The system/platform limitations such as weight, processing speed, power, communication range and speed were also addressed. The overall proposed experimental setups, required to be implemented to test and validate the performance of the aerial pipeline Right-of-Way surveillance system, were also described in this chapter.

Chapter 4
Pipeline Endpoints Identification

4.1 Introduction

This chapter presents the development of the vision-based algorithm that is used onboard the Unmanned Aerial Air-vehicles (UAV), to identify and localize the endpoints of the pipeline structure in real-time. The first objective of this chapter is to develop a reliable algorithm capable of estimating the endpoints position of the pipeline segments throughout the frame sequences in near real-time. The aims of this development are to keep track/follow the pipeline online, down-sample the data in each frame by filtering the region that is an outlier of the Right-of-Way to focus the processing on the data located within the Right-of-Way region. This leads to reducing the computational cost to detect any third-party interference and any defects.

In this project, two methods are proposed to identify and localize the over-ground pipeline structure. One is based on the standard camera (RGB intensities) and the other is based on the IR sensor (depth intensities). Different computer vision techniques were used with each sensor. The proposed techniques are capable of detecting and localizing the pipeline with high detection and processing rates. The techniques are described in detail in the following sections. Furthermore, the experimental setup, simulation, analysis and performance evaluations are performed for each method and are described in this chapter.

4.2 Visible-Based Pipeline Endpoints Identifications

This section describes the proposed visible-based algorithm used to detect and localize the pipeline structure using a UAV mounted with a standard visible camera. This algorithm is proposed to detect and localize the endpoints of the pipeline structure, efficiently in real time, based on the colour intensity in order to keep track of the pipeline and maintain monitoring and surveying.

Image processing techniques were used to develop this algorithm. The first step of the processing involves enhancing the image data to improve the edge detection performance. Then, the boundaries of the features are detected in the sense of the image by using a canny edge detector. Afterward, the boundary candidates are extracted to represent a straight line using the Hough Transform method. Following this, the straight lines are filtered by comparing them with known pipeline segments in the area, allowing fast and accurate pipeline identification. This step enables the system to reject with high degree of reliablity inaccurate measurements while retaining the correct pipeline detections and location.

Several image sets and two aerial videos, captured by the standard camera, were used to test and analyse the system. The performance of this algorithm was evaluated using a Matlab/Simulation environment to verify that the system is capable to reliably detect and localize the pipeline structure with a low cost aerial platform, with high detection rate and fast processing time.

4.2.1 Image Acquisition

Two images are used to demonstrate how the proposed algorithm detects the pipeline as shown in Figure 4-1. These images were captured by (Pipeline Surveillance Company - Not for Public Release, 2012) from a manned air-vehicle from two different views and in various environments, at a resolution estimated to be greater than 0.5 m per pixel. It is worth mentioning that the two images were captured at different altitudes.

Figure 4-1: Two images of a pipeline at various environments (Pipeline Surveillance Company - Not for Public Release, 2012)

4.2.2 Pipeline Detection & Localization Algorithm

Since the most recognizable feature of the pipeline structure that is easily detectable in the standard RGB image is the shape, the development commenced with detection of the line as a shape. However, there are many different methods to detect line features in image processing applications. In this project, the proposed algorithm to detect the pipeline is demonstrated in Figure 4-2.

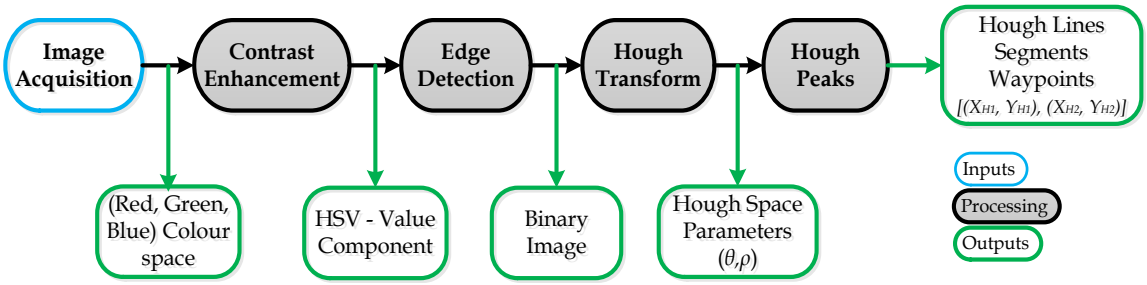


Figure 4-2: Camera-based pipeline detection algorithm

At first, the RGB data is acquired from the sensor. Then, those RGB colours are converted into HSV colours to simplify and enhance the analysis. After that, a canny edge detector is implemented to detect the edges as points based on the differentiation of the colour intensity. Then, based on those points of the edges, a Hough Transform method is implemented to construct a solid line configuration that potentially refers to the pipeline in the image. Finally, the lines constructed are represented as waypoints.

4.2.2.1 Contrast Enhancement

In order to obtain optimum edge detection, the contrast is enhanced. This enhancement can be done by improving the intensity (luminance) which is the total amount of light passing through a particular area. The RGB colour space describes colours based on the intensities of Red, Green, and Blue channels, respectively, which are not always suitable for colour based applications. Therefore, it is useful to transform it into Hue, Saturation & Value (HSV) colour space. It describes colours in the same way as the human eye senses colour, where a colour is represented by its Hue (H) and Saturation (S). The saturation adds white light to distinguish between the components of each surface by its lightness (such as metallic for pipe, asphalt for the road, water in a lake or sea, and on forth). Value (V) describes the overall intensity or strength of the light. A detailed description of the HSV colour space can be found in Gonzalez and Woods (2003).

Figure 4-3 demonstrates the difference between the intensity of RGB colour space and value components of HSV colour space in two images with different environments. Figure 4-4 shows the effect of this improvement when an edge detector has been applied.

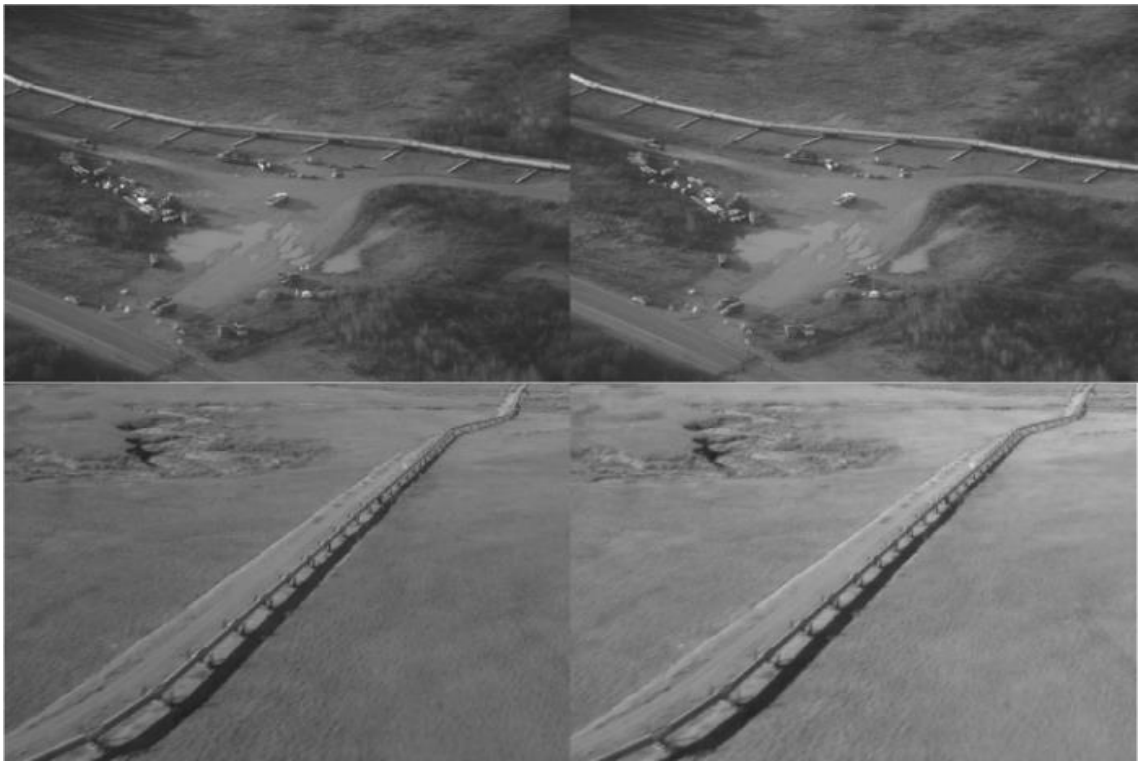


Figure 4-3: (Left): Intensity (Grey colour) of RGB colour space, (Right): Value (V) component of HSV colour space

4.2.2.2 Edge Detector

The second step in the algorithm of line detection is finding the edges for the Hough Transform. However, the optimal method used in this research to detect the edges is the canny edge detector which is developed by Canny (1986). The output of this detector is a binary image represented by points based on the differentiation of the colour intensity as demonstrated in Figure 4-4.

The process of this method starts by smoothing (blurring) the image to eliminate noise. Then, it finds where the high magnitude of image gradients is to highlight regions that have a high spatial derivative. The algorithm then tracks through these regions and suppresses any point that is not at the maximum region (non-maximum suppression) and marks only local maxima as edges. The gradient array is now further decreased by hysteresis. Hysteresis is used to track throughout the remaining points that have not been suppressed. Hysteresis uses two thresholds to the gradient to determine the potential edges. When the

gradient magnitude is below the lower threshold, it is set to zero (no edge). If the gradient magnitude is above the high threshold, it makes a strong edge. If the gradient magnitude is between them, it is set to zero, unless it connects to a strong edge with a gradient above the high threshold.

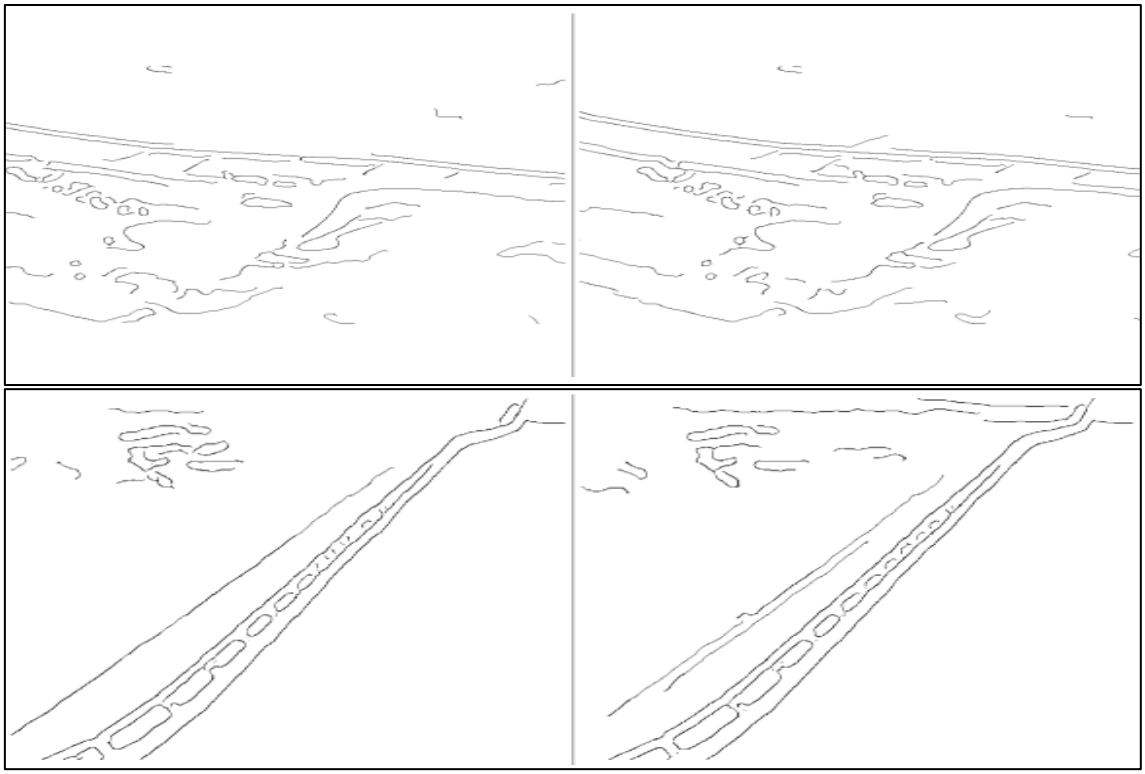


Figure 4-4: The edges detected as a binary image based on the RGB intensity (Left), and the HSV value component (Right)

4.2.2.3 Hough Transform

Hough Transforms (HT) is a robust method for finding lines in the image that was developed by (Hough, 1962). The image features (points) that are detected by the canny edge detector at particular (x, y) coordinates is evidence that there might be a line passing through some of those points.

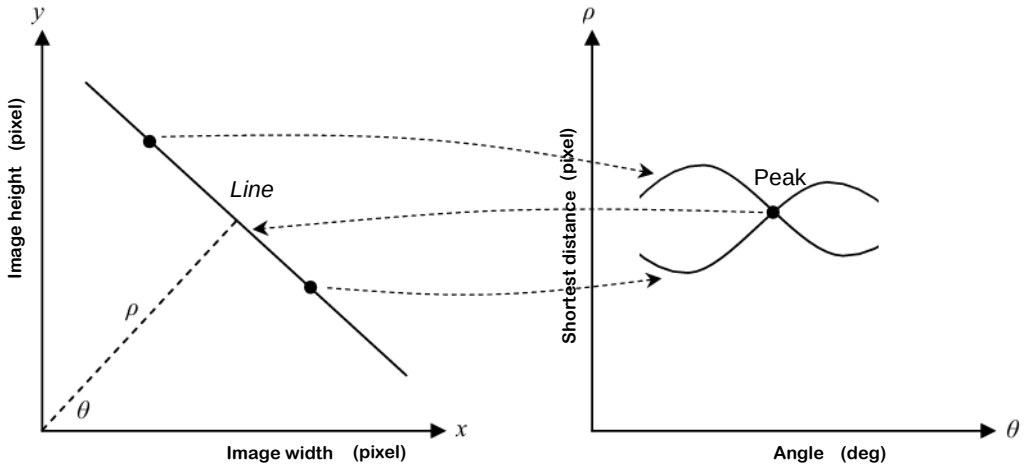


Figure 4-5: (Left) Image space and (Right) Parameter space or Hough space
(Antolovic, 2008)

The process steps of achieving Hough Transform method are as follows:

- Line detection: Accumulate tracings of the parametric sinusoid (ρ, θ) in Hough space for each pixel (x, y) in image space by using the Equation (4-1) from (Antolovic, 2008) as shown in Figure 4-5.

$$\rho = x\,cos\,\theta + y\,sin\,\theta \tag{4-1}$$

  Where: ρ is the shortest distance from the corner of the image space to the straight line, θ is the angle between the shortest distance ρ and the horizontal axis (width) in image space, and (x, y) represents the width and height of the pixel in image space.

- Peaks detection: the intersection points in Hough space between the curves which represent a straight line at parameter values (ρ, θ) as shown at the right graph in Figure 4-5. However, as more curves are intersected at that point, the more the line becomes solid, which represents the number of points corresponded.

- Linking the lines: Once a set of peaks has been detected in Hough space, it remains to be determined if there is a line segment associated with those

peaks, as well as start and ending points. For each peak, the first step is to find the location of all non-zero pixels in the image space that contributed to that peak and construct line segments based on those pixels.

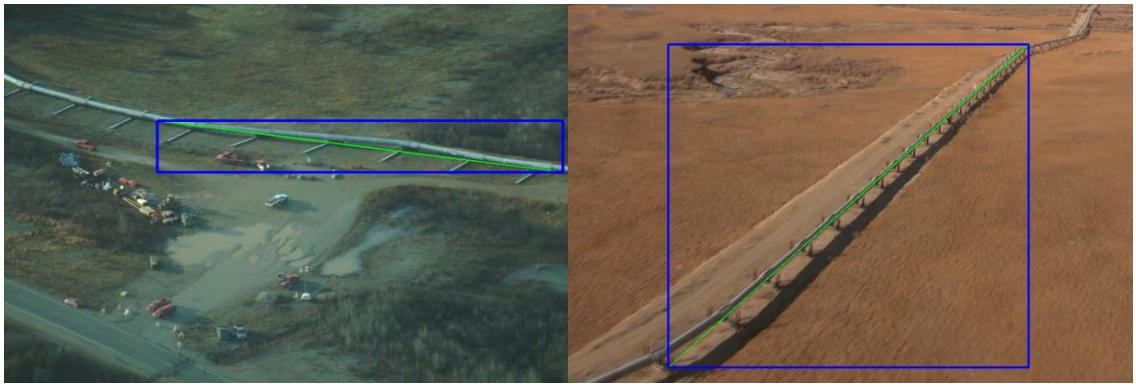


Figure 4-6: The longest line detection using Hough Transform

4.2.2.4 Filtering & Enhancement

Finally, based on the lines detected by Hough Transform; the longest line was identified as a pipeline candidate. This assumption is based on the idea that there will only be one main line feature within the image, Figure 4-6. However, if there are many features in a line, such as multi-segments, then it will not be suitable to just choose the longest line which will lead to missing the other lines, as shown in Figure 4-7.

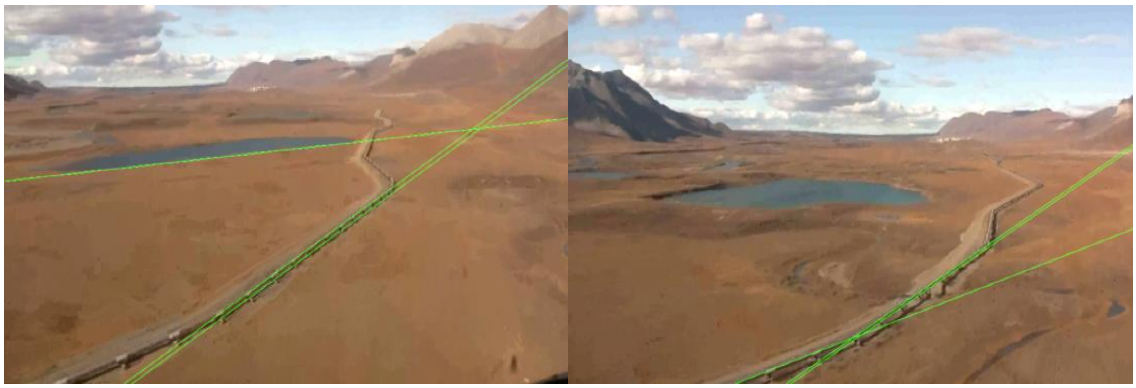


Figure 4-7: True & false detection of multi-lines detection using Hough Transform

However, to identify the right line segments and improve the performance of the detection algorithm, the pipeline candidates are projected from the image frame to the world by transforming the pixel coordinates (x_i, y_i) into 3D-world coordinates (X, Y, Z). These line segments are then compared to the existing pipeline waypoints within each frame to determine the correct detection as described in Figure 4-8.

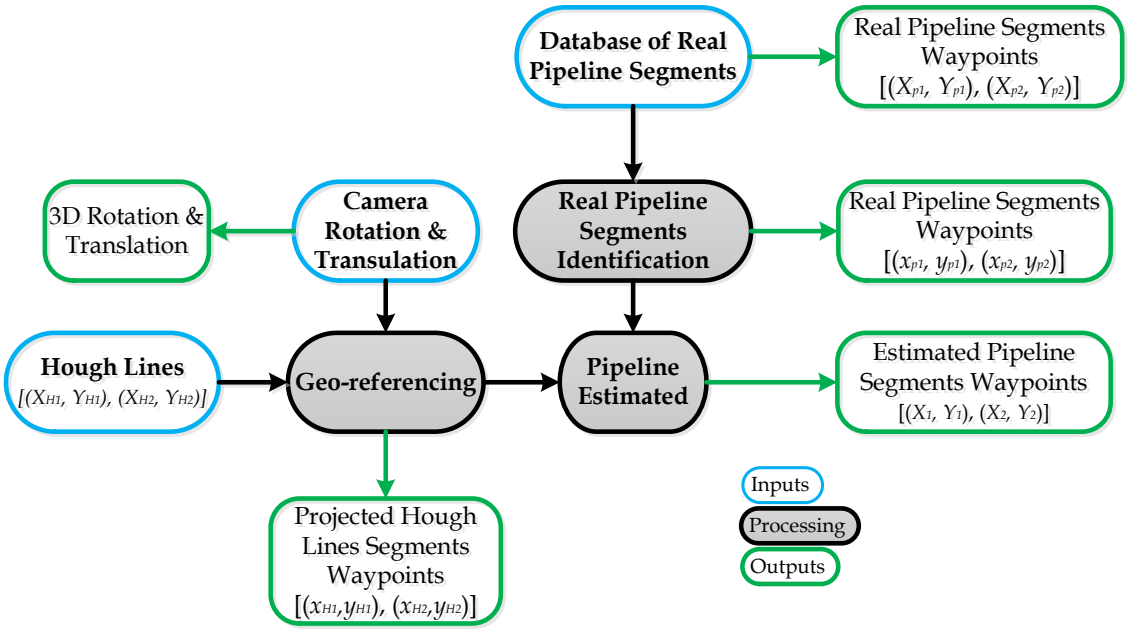


Figure 4-8: Filtering & enhancement of pipeline detection

4.2.2.5 Geo-Referencing

In order to calculate the errors between the known pipeline segments and the lines detected by the vision algorithm, the first step is to geo-reference the line features found in the image. It is an important issue to project the camera measurements into measurements in the real 3D world because scenes are not only 3D; they are also physical spaces with physical units. Consequently, the relation between the camera's coordinate frame and the world coordinate system is a critical component in any attempt to reconstruct a 3D scene.

In order to implement this, the camera is calibrated to model the camera's geometry and to estimate the distortion properties of the lens. These measurements define the intrinsic parameters of the camera. When these factors are known it is possible to use homography transform to project points from the image plane to the ground plane (assuming the terrain is flat).

Once the Hough lines are projected into the 3D world coordinate system, the endpoints of the pipe are estimated based on the distance between the existing pipeline, Hough lines, and their parallelism.

4.2.2.6 Real Pipeline Endpoints Identification and Localization

Before estimating the position of the pipeline endpoints, the endpoints of the real pipeline segment that pass through the image need to be identified, to avoid matching incorrect areas as described in Figure 4-9.

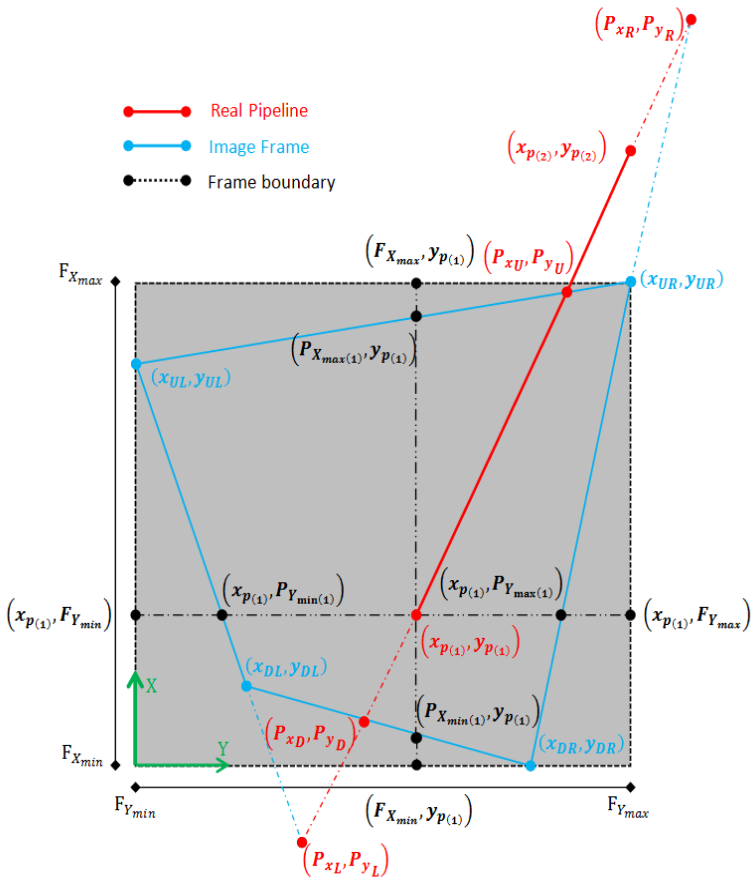


Figure 4-9: Real pipeline endpoint identification

To identify the endpoints of this pipeline segment, the following steps were developed as follows:

1) Predefine the entire real pipeline endpoints for each segment $[(x_{p_{(1)}}, y_{p_{(1)}}), (x_{p_{(2)}}, y_{p_{(2)}})]$.

2) Calculate the image corners in the real world based on the resolution of the original image frame $[(x_{UR}, y_{UR}), (x_{UL}, y_{UL}), (x_{DR}, y_{DR}), (x_{DL}, y_{DL})]$.

3) Determine the maximum and minimum frame boundaries $(F_{x_{max}}, F_{x_{min}}, F_{y_{max}}, F_{y_{min}})$.

4) Calculate the interception of each line segment with the edges of the frame $[(P_{x_U}, P_{y_U}), (P_{x_D}, P_{y_D}), (P_{x_L}, P_{y_L}), (P_{x_R}, P_{y_R})]$.

5) Determine the maximum and minimum boundaries for each predefined endpoint within the edges of the frame $(P_{X_{max(1)}}, P_{X_{min(1)}}, P_{Y_{max(1)}}, P_{Y_{min(1)}})$.

6) Decide if the pipeline segment falls within the image:

    a. If the entire segment falls within the frame, confirm these endpoints.

    b. If one endpoint of the predefined line segment meets the condition, while the other endpoint falls outside the frame, confirm the acceptable point and calculate the second endpoint from the interception endpoint.

    c. If neither of the pipeline endpoints falls within the frame, but the segment is crossing the image, identify the interception points at the edges of the image.

4.2.2.7 Pipeline Position Estimation

Estimating the position of the pipeline is achieved by comparing a predefined number of lines detected by Hough Transform with the predefined real position of the pipeline endpoints at each frame and selecting the nearby one. The geometric calculations used in this project are shown in Figure 4-10, where the distance and parallelism between two lines are calculated. First, the length (l) of the real pipeline segment that is passing through the image is calculated using Pythagorean Theorem (Boljanovic, 2006) as shown in Equation (4-2):

$$l = \sqrt{\left(\mathrm{x}_{\mathrm{p}_1} - \mathrm{x}_{\mathrm{p}_2}\right)^2 + \left(\mathrm{y}_{\mathrm{p}_1} - \mathrm{y}_{\mathrm{p}_2}\right)^2} \tag{4-2}$$

Where: l is length of the real pipeline segment that is covered in the image frame. $(\mathrm{x}_{\mathrm{p}_1}, \mathrm{y}_{\mathrm{p}_1})$ and $(\mathrm{x}_{\mathrm{p}_2}, \mathrm{y}_{\mathrm{p}_2})$ are the width and height of the first and second endpoints of the real pipeline segment relative to the image frame, respectively.

Similarly, calculating the distances $(l_{11}, l_{12}, l_{21}, and\ l_{22})$, that are located between the endpoints of the real pipeline and the endpoints of the detected Hough lines as denoted in Figure 4-10 using again Pythagorean Theorem (Baer, 2005) as shown in Equation (4-3) to Equation (4-6).

$$l_{11} = \sqrt{\left(\mathrm{x}_{\mathrm{p}_1} - \mathrm{x}_{\mathrm{H}_1}\right)^2 + \left(\mathrm{y}_{\mathrm{p}_1} - \mathrm{y}_{\mathrm{H}_1}\right)^2} \tag{4-3}$$

$$l_{12} = \sqrt{\left(\mathrm{x}_{\mathrm{p}_1} - \mathrm{x}_{\mathrm{H}_2}\right)^2 + \left(\mathrm{y}_{\mathrm{p}_1} - \mathrm{y}_{\mathrm{H}_2}\right)^2} \tag{4-4}$$

$$l_{21} = \sqrt{\left(\mathrm{x}_{\mathrm{p}_2} - \mathrm{x}_{\mathrm{H}_1}\right)^2 + \left(\mathrm{y}_{\mathrm{p}_2} - \mathrm{y}_{\mathrm{H}_1}\right)^2} \tag{4-5}$$

$$l_{22} = \sqrt{\left(\mathrm{x}_{\mathrm{p}_2} - \mathrm{x}_{\mathrm{H}_2}\right)^2 + \left(\mathrm{y}_{\mathrm{p}_2} - \mathrm{y}_{\mathrm{H}_2}\right)^2} \tag{4-6}$$

Where: $l_{11},\ l_{12},\ l_{21},\ and\ l_{22}$ are the distances between each endpoint of the real pipeline segment with each endpoint of the detected Hough line segment. x and y refer to the north and east coordinates, respectively. p and H refer to the

real pipeline and the Hough line segments, respectively. $1 \text{ and } 2$ denote the first and second endpoints of each segment, respectively.

After that, the angles α_1 and α_2, represented in Figure 4-10, are calculated using the law of cosines formula (Boljanovic, 2006) as in Equation (4-7) and Equation (4-8):

$$l_{11}^2 - l_{12}^2 - l^2 + 2l_{12}l\,cos\,\alpha_1 = 0 \tag{4-7}$$

$$l_{22}^2 - l_{21}^2 - l^2 + 2l_{21}l\,cos\,\alpha_2 = 0 \tag{4-8}$$

Then, the shortest distances $\rho_1\,and\,\rho_2$ are determined that are located at the first and second endpoint of the Hough line segment and the real pipeline segment as denoted in Figure 4-10 using the law of sines formula (Boljanovic, 2006) as shown in Equation (4-9) and Equation (4-10).

$$\rho_1 = l_{12}\,sin\,\alpha_1 \tag{4-9}$$

$$\rho_2 = l_{21}\,sin\,\alpha_2 \tag{4-10}$$

Hence, the distance ρ_a between the Hough line segment and the real pipeline segment is calculated by averaging the shortest distances of each endpoint (ρ_1, ρ_2) as indicated in Figure 4-10 using Equation (4-11).

$$\rho_a = \frac{\rho_1 + \rho_2}{2} \tag{4-11}$$

In addition, the parallelism factor (ρ_p) used to measure the parallelism of any pair of straight lines is obtained based on the absolute difference between the normal lines of each endpoint using Equation (4-12):

$$\rho_p = |\rho_1 - \rho_2| \tag{4-12}$$

Finally, based on the obtained distance (ρ_a) and the parallelism factor (ρ_p), the line is selected to represent the candidate pipeline based on the limitations of the distance between any pair of lines as well the parallelism factor.

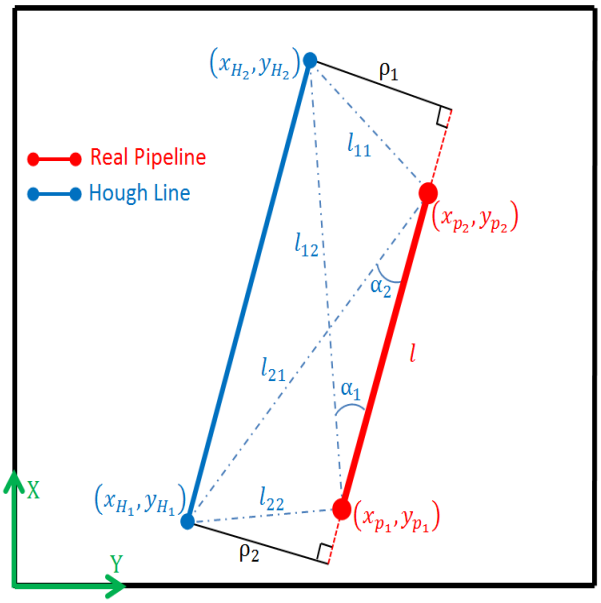


Figure 4-10: Geometric calculations for pipeline position estimation

4.2.2.8 Simulation Results

First, the Hough Transform algorithm was implemented in Matlab/Simulink to evaluate the performance of the system. Several image sets have been analysed, and two aerial videos have been used to test the system. The first video has been captured by an air-vehicle flying over the Trans-Alaska Pipeline in the United States while the second video shows a pipeline being constructed in the United Kingdom to allow testing of false positives.

Table 4-1: Performance of Hough transforms detection rates

Data Source	Number of Frames	Successful Detections	Detection Rate
1st Video	1200	1100	94%
2nd Video	45	30	66%

The detection rate of this system was measured by running the algorithm on the two image sets as in Table 4-1. In first video set, the system detects the pipeline with a success rate of 94% over 1200 frames. The second video is used to stress-test the system regarding false detection, which causes the detection rate to drop to 66%. This reduction comes from the fact that the system fails when there are no pipelines in a frame. In the second video, the pipeline is still being constructed leading to a large number of false detections where the pipeline segments have not been joined. This test was used to check the capability of detecting the objects in the image whatever they are true or false. So, later, the filtering was used to reduce the false detection.

4.2.3 Experimental Setup

In order to validate this pipeline detection approach, a lab experiment was performed as shown in Figure 4-11 which includes the following:

1. CCD camera sensor to provide the RGB images.
2. Small-scale pipeline structure.
3. Tripod to hold the camera sensor.
4. Two parallel channels to keep the heading.
5. Moving trolley (Plate).

The experimental investigations of detecting the pipeline structure were performed based on an optical camera sensor as shown in Figure 4-12 and the characteristics of which are tabulated in Table 4-2.

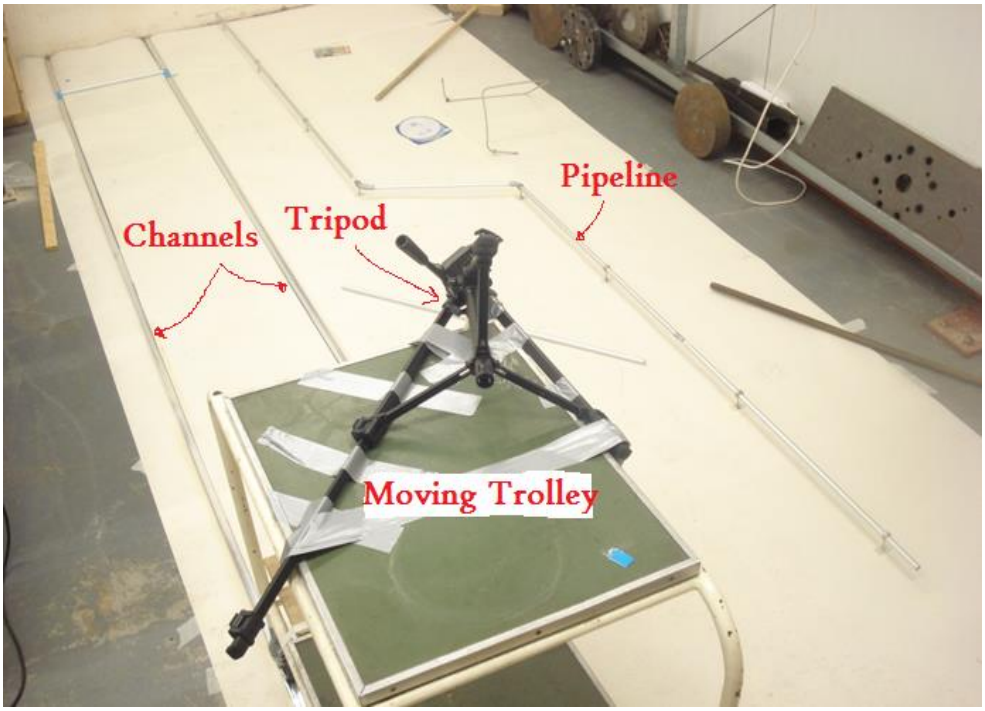


Figure 4-11: Experimental setup

Table 4-2: Camera specifications (Canon, 2011)

Type	1/2.3 type CCD
Focal Length	5.0 – 20.0 mm (35 mm equivalent: 28 – 112 mm)
FOV	± 47º
Frame size (resolution)	640 x 480 pixels
Frame Rate	25 fps
Weight	185 g (including battery/batteries and memory card)

Figure 4-12: Camera sensor (Canon, 2011)

4.2.4 Experimental Results

A camera tripod was mounted on a moving plate in the lab to simulate the footage obtained during steady flight over a pipeline. The camera was placed west of the pipeline and mounted at a height of 90 cm, looking down with a 30 degree roll angle. It was moved from south to north at a constant speed, thereby simplifying the calculation of the camera's current position. Two parallel channels are mounted on the ground at the linear path to keep the heading of the camera constantly. This gives an idealized vision of the scene that would be seen by the UAV.

A short video (25 seconds) was recorded along a small pipeline of 1.2 cm diameter and 4.5 m length as shown in Figure 4-11 to evaluate the algorithm off-line. The sensor used to record the video is a standard CCD digital camera with approximately 47 degree horizontal view angle. The frame rate and resolution are 25 fps and (640×480) pixels, respectively. The detection algorithm was implemented in Matlab/Simulink simulation environment.

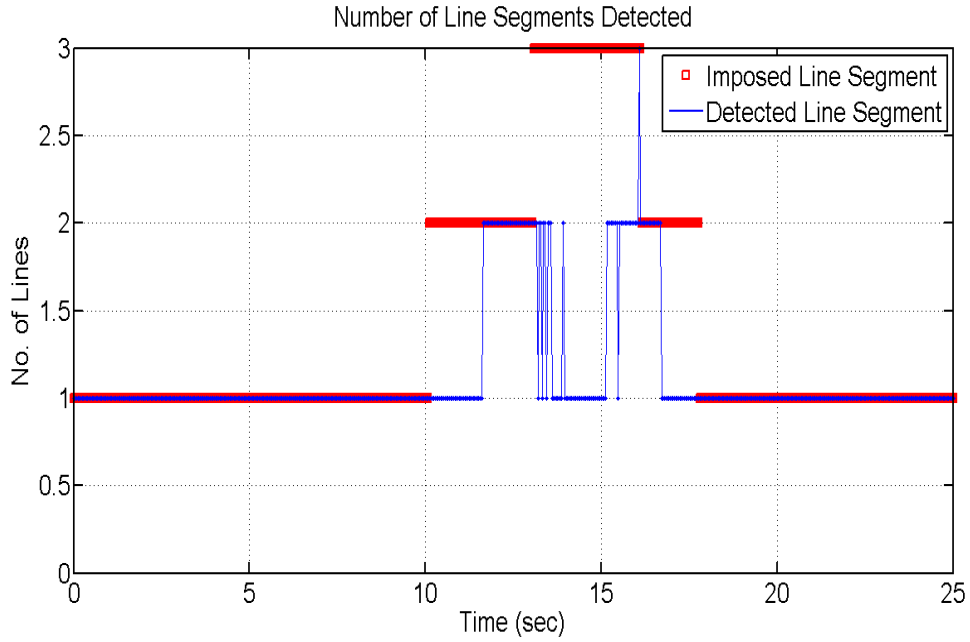


Figure 4-13: Numbers of the detected pipeline segments

Figure 4-13 shows the number of segments detected and compares it with the number of lines that should be detected at each frame along the pipeline's length.

The position error between the detected pipeline and the real one is shown in Figure 4-14. As illustrated in Figure 4-14 when the algorithm are tested in a controlled lab environment with the pipeline having only 1-segment in the captured image frame, it can achieve about 99% detection rate.

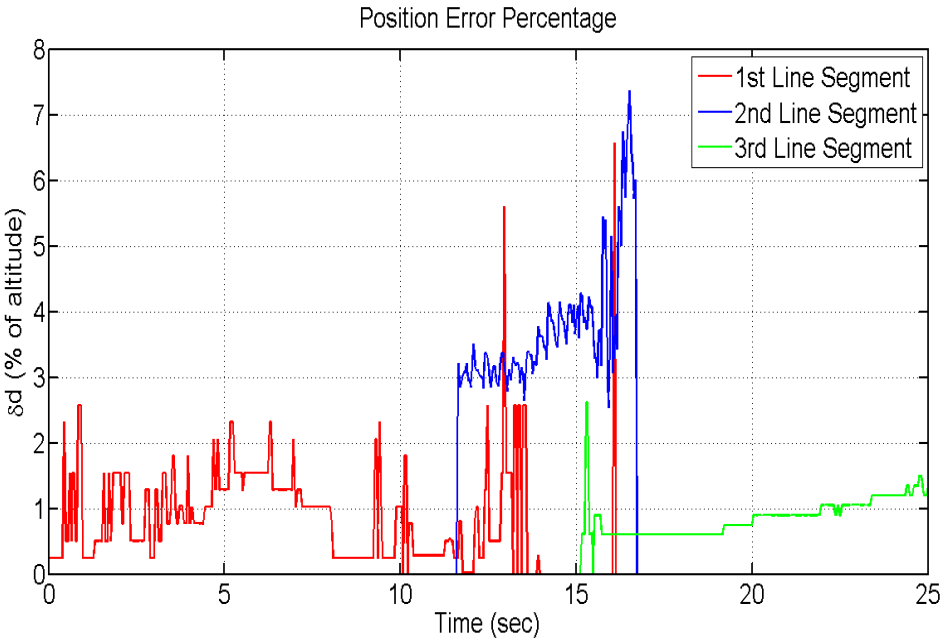


Figure 4-14: Position error percentage

However, when considering a pipeline consisting of 3-segments, as shown in Figure 4-15, the average detection rate slightly reduces to approximately 96%. This reduction comes from the number of Hough lines that are used to select the correct line in the image frame. Therefore, if the line segment in the image frame is long compared to the other segments, the detection algorithm fixes on that segment and hence improving the detection rate.

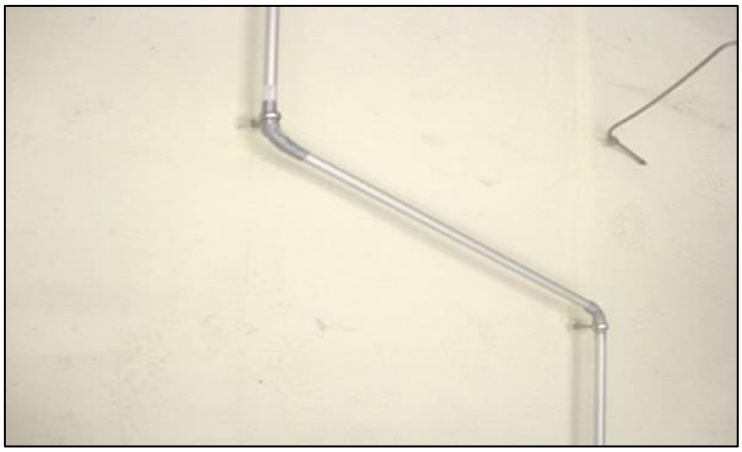


Figure 4-15: Pipeline structure

Figure 4-16 shows the initial robustness test results in which the algorithm is able to detect the pipeline and ignore the false detection of either a secondary pipe or other visually similar objects.

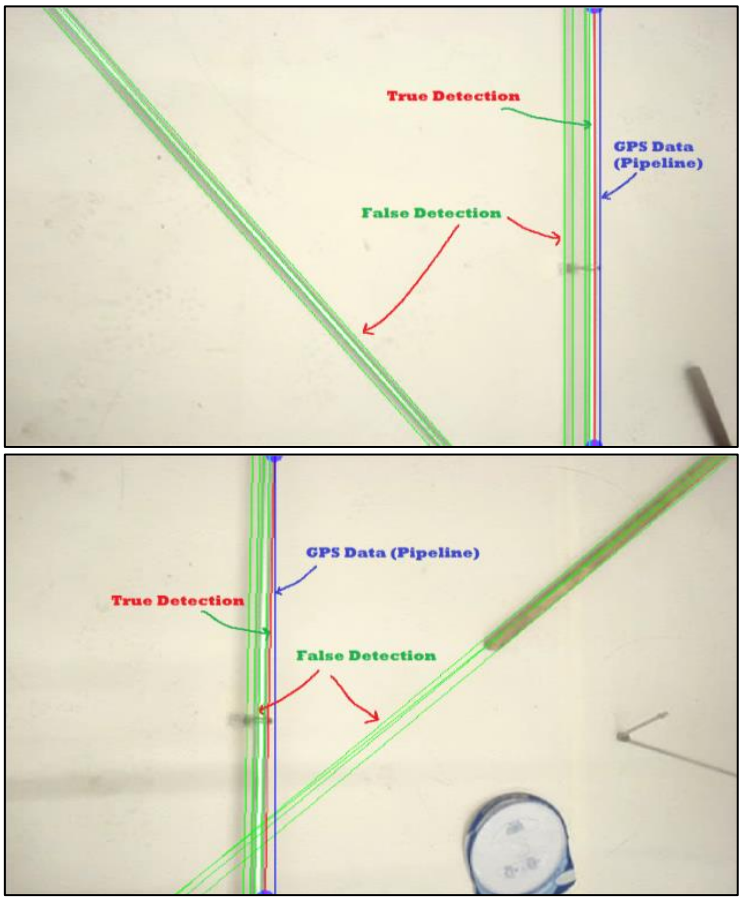


Figure 4-16: Results of pipeline detection

Figure 4-17 shows results of the pipeline structure and the endpoints history of the line segment at each image frame through the whole pipeline structure route.

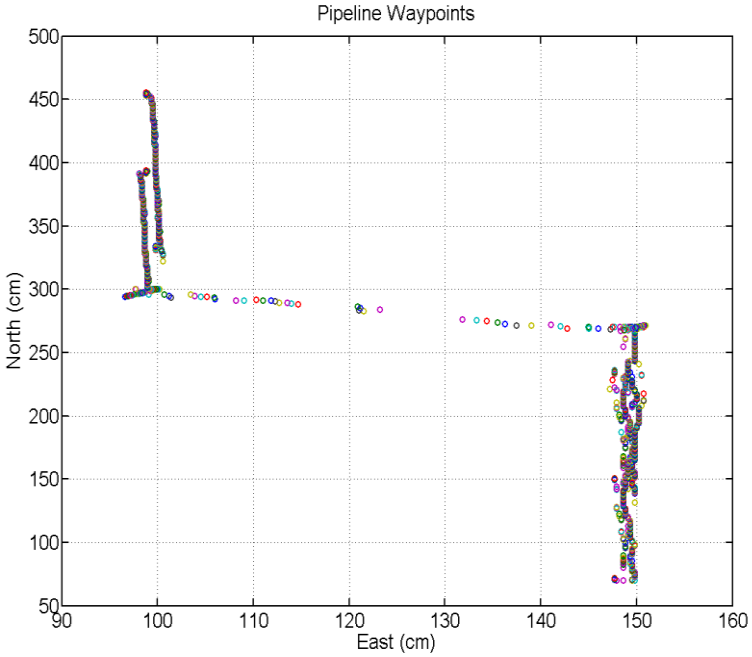


Figure 4-17: Pipeline segments endpoints history

As seen, the endpoints of the pipeline segments were identified successfully along the lengthy pipeline which represents the endpoints behaviours at each captured image frame (30 fps). Hence, this identification requires the GPS data of the air-vehicle in addition to the GPS database of the pipeline structure itself. So, another algorithm is required to be developed which can identify the endpoints of the pipeline without relying on the GPS data of the air-vehicle and the pipeline structure.

4.3 Depth-Based Pipeline Endpoints Identification

This section describes the proposed depth-based algorithm used to detect and localize the pipeline structure using an aerial platform equipped with a depth sensor. This algorithm was proposed to achieve an on-board reliable automatic system capable of identifying and localizing the pipeline structure, in real-time with a low cost aerial platform based on the depth intensity in order to keep track of the pipeline and maintain monitoring and surveying.

Once again use of computer vision techniques, were used to develop this algorithm. The processing of this algorithm commences with a transformation from 2D depth information into 3D point clouds. Following this, a RANSAC approach is used to detect the plane in the density of that 3D point cloud and to filter out the ground from non-ground points. Once the non-ground points have been found, they are then transformed back into the 2D depth matrix in order to implement the 2D edge detection. If the edges are, detected, then they are again convert into 3D points to geometrically filter out the points that are higher than a predefined height, relative to the plane of ground detected. It is performed based on the standard height of the existing pipelines with a hysterises margin, in order to remove points by that might due to buildings or any high objects. Again, once points are found, they are converted into a 2D depth array using 2D Hough transforms to extract the possible line structures. Then, the lines detected are converted into 3D points to geometrically check the parallelism and the distance between any two lines which refers to the possible structure of the pipeline. Finally, if the conditions are satisfied, the waypoints of the position of that pipeline's position are measured by picking up the endpoints of the pipeline segment.

The Kinect sensor was chosen in this project to provide the depth information of the scene in order to obtain more accurate detection than the colour information which has a high noise due to lighting effects.

The performance of this algorithm was tested indoors due to limitations of finding a real over-ground pipeline, and secondly due to the sensor limitations. Python and OpenCV library environment where used to verify that the system is capable of detecting and localizing reliably the pipeline structure from the air, with a high detection rate and fast processing time.

4.3.1 Requirements

In order to process the algorithms of pipeline detection in this project, two inputs are required to accomplish the processing as shown in Figure 4-18, namely they are:

a) A 2D depth array (data acquisition).

b) The intrinsic calibration model of the depth image.

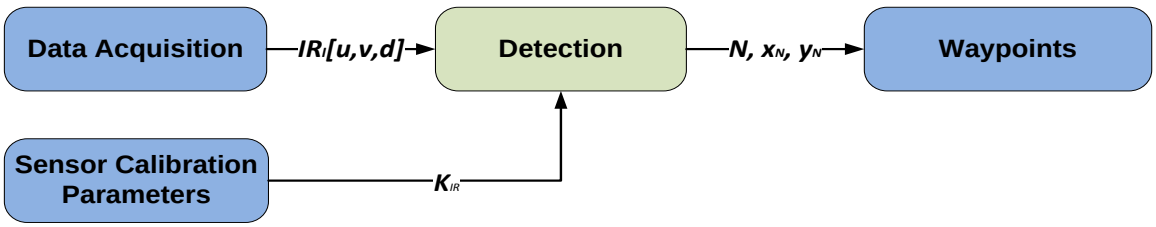


Figure 4-18: Depth-based pipeline endpoints identification

4.3.1.1 Data Acquisition

The first step of any computer vision system is the image acquisition. Once the image has been acquired, various techniques of processing can be applied to perform different computer vision tasks to the image. The sensor used in this stage of processing is the Kinect Xbox. This section describes how this sensor streams the depth data for computer vision analysis.

Two open source drivers/libraries are available to be used to stream this data in this project, which are Libfreenect (by OpenKinect community) and OpenNI

(Open Natural Interface). More information about those drivers is already described in chapter three.

By default, the depth image has a resolution of 640x480 pixels (u, v) where each pixel has 11-bit depth data. This data has a raw depth value (d_{raw}) varying from 0 to 2047 for each pixel which indicates the distance between the principal plane of the IR camera and a point in the scene, measured perpendicularly to the main plane through that pixel.

This depth map is provided by the Kinect as a greyscale image of an array as shown in Figure 4-19. In this figure, a darker coloured pixel represents a spot location nearer to the depth camera, while a brighter pixel locates farther to the camera depth. Moreover, the white regions are areas that the camera cannot see given the shooting angle. In a grayscale image, black is defined with a value of 0 and white is defined with a value of 256.

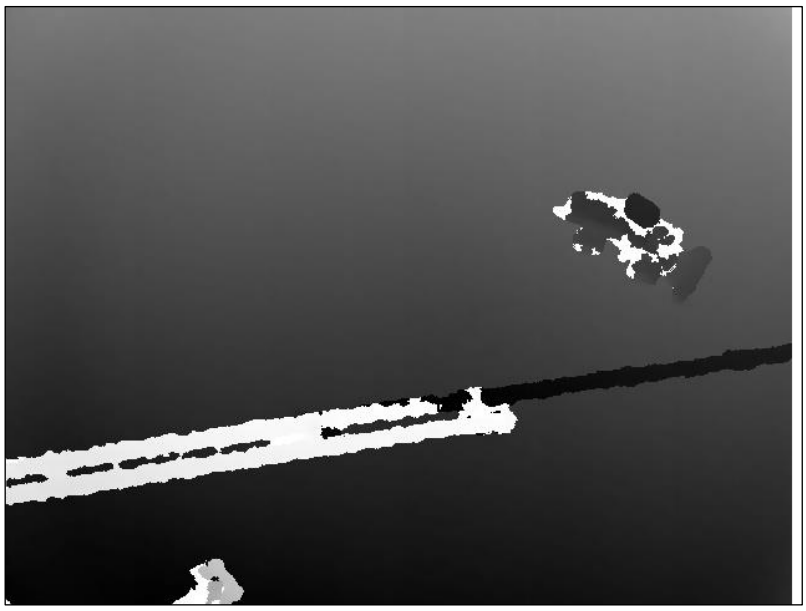


Figure 4-19: Kinect greyscale image of a depth array

4.3.1.2 IR Intrinsic Calibration Model

The intrinsic calibration model of the depth image is required in this chapter to assist processing of the pipeline detection algorithm as shown in Figure 4-18. The same model has already been performed in (section 4.2.2). The purpose of this model is to project the IR image coordinates into world coordinates (meter).

4.3.2 Identification algorithm

This section presents the overall design of the proposed identification algorithm of the pipeline structure endpoints based on the depth data as demonstrated in Figure 4-20. The processes used to develop this algorithm are listed as follows:

a) 3D point clouds mapping of the explored view based on the depth data.
b) Filtration of the coplanar and non-coplanar 3D point cloud and transform them back into 2D depth map.
c) Edge detection of the non-coplanar using 2D depth map and transform it again into 3D point clouds.
d) Thresholding the elevation of the non-coplanar edges using their 3D point cloud data, and then transform them into 2D depth map.
e) Detecting the possible line structures using 2D depth map of the thresholded edges and then transform them into 3D point cloud map.
f) Confirm if any pair of the lines detected in (e) satisfy the required geometric conditions of the pipeline configuration (such as parallelism, distance between them, corresponding to the non-coplanar 3D point cloud).
g) Finally, estimating the position of the endpoints of the pipeline segments (Endpoints) relative to the camera frame.

More details of these processes are described in the following sections of this chapter.

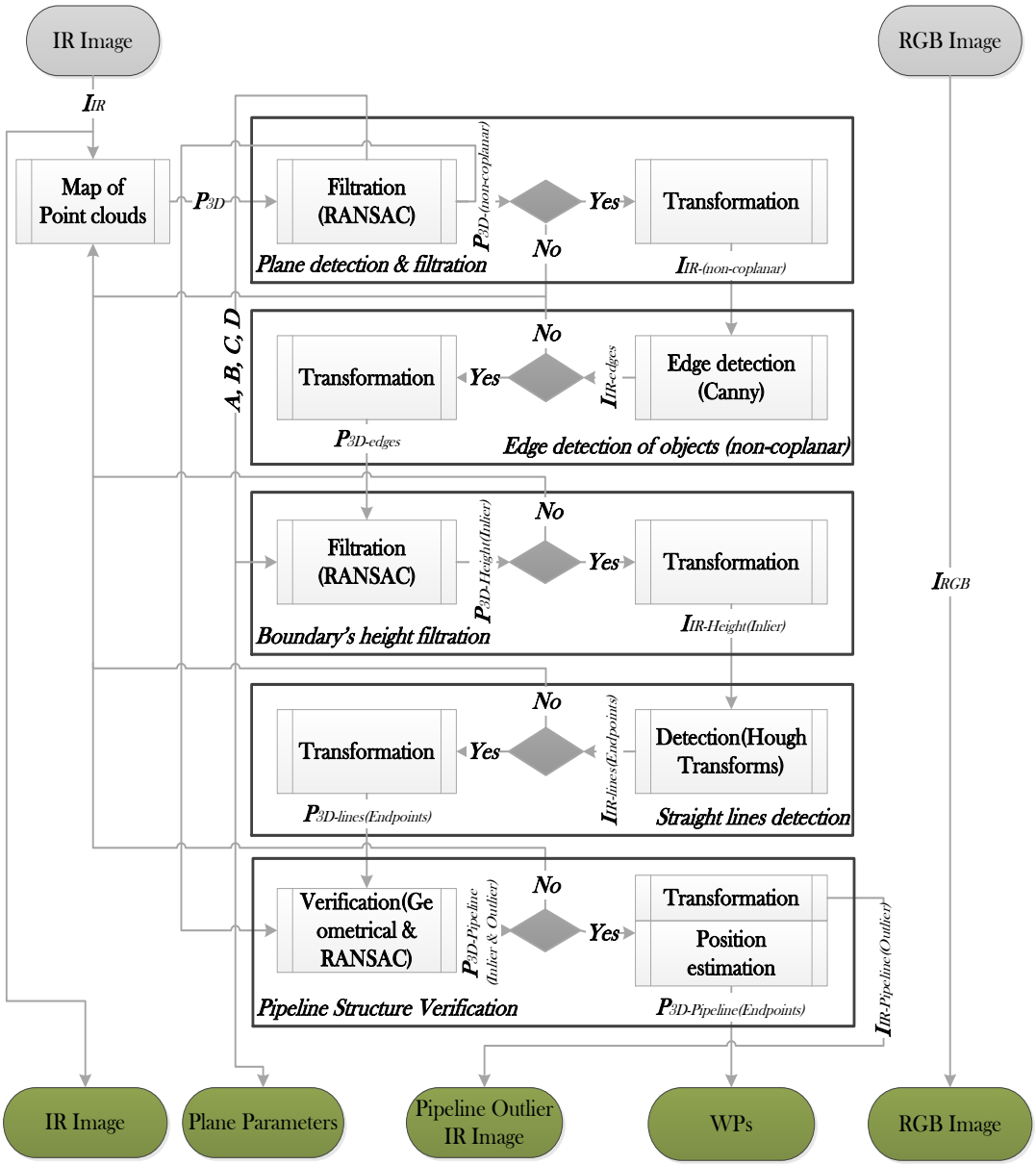


Figure 4-20: Depth-based pipeline endpoints identification algorithm

4.3.2.1 3D Point Cloud Mapping

A point cloud mapping is a 3D dataset of points, representing the scene's depth that is relative to the sensor's coordinates, in the real world's metric coordinates. The goal of this task is to develop an algorithm that can map and reconstruct the point cloud from the depth map. It is proposed in this project to assist detecting the pipeline structure geometrically by using the real spatial information about the objects in the scene.

In order to reconstruct the 3D point cloud in real time, the depth image is projected along the intrinsic calibration model of the depth sensor. This section describes the approach of how to transform the depth map into a dense 3D point cloud of the scenes in world space coordinates. So, let the depth map of size $u \times v$ pixels be I_d. Moreover, the corresponding 3D point cloud P_{3D} of each pixel is reconstructed using the depth value $I_d(u, v, d)$. First, rebuilding the $\mathrm{N} \times 3$ matrix of the depth map coordinates into a homogeneous scene coordinates $\mathrm{N} \times 4$ matrix as shown in form (4-13) below:

$$I_d = \begin{bmatrix} u \\ v \\ d \end{bmatrix} \rightarrow \begin{bmatrix} u \\ v \\ d \\ 1 \end{bmatrix} = I_{dh} \tag{4-13}$$

Where: $I_d(u, v)$ represents the matrix of the depth with a value d at each pixel location (u, v), $u, v\ and\ d$ represent pixel height, pixel width, and depth value of the image, respectively.

Then, the homogeneous depth matrix (I_{dh}) is projected into 3D world homogeneous scene coordinates (P_{h3D}) using the intrinsic calibration model of the depth sensor (K_{IR}) as shown in Equation (4-14):

$$P_{h3D} = \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ W_w \end{bmatrix} = K_{IR}.I_{dh} \tag{4-14}$$

Where: $X_w, Y_w,\ and\ Z_w$ are the 3D world coordinates, and W_w is the homogeneous factor.

After that, the 3D world homogeneous scene coordinates (P_{h3D}) are converted into inhomogeneous scene coordinates (P_{3D}) by dividing the fourth factor to get the 3D point cloud as in Equation (4-15):

$$P_{3D} = \begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} = \begin{bmatrix} X_w/W_w \\ Y_w/W_w \\ Z_w/W_w \end{bmatrix} \tag{4-15}$$

So, Figure 4-21 illustrates the construction of the 3D point cloud (P_{3D}) of the expected scene of this project that involves pipeline structure, third-party objects, and the ground plane. The processing rate of this approach is about 2 fps.

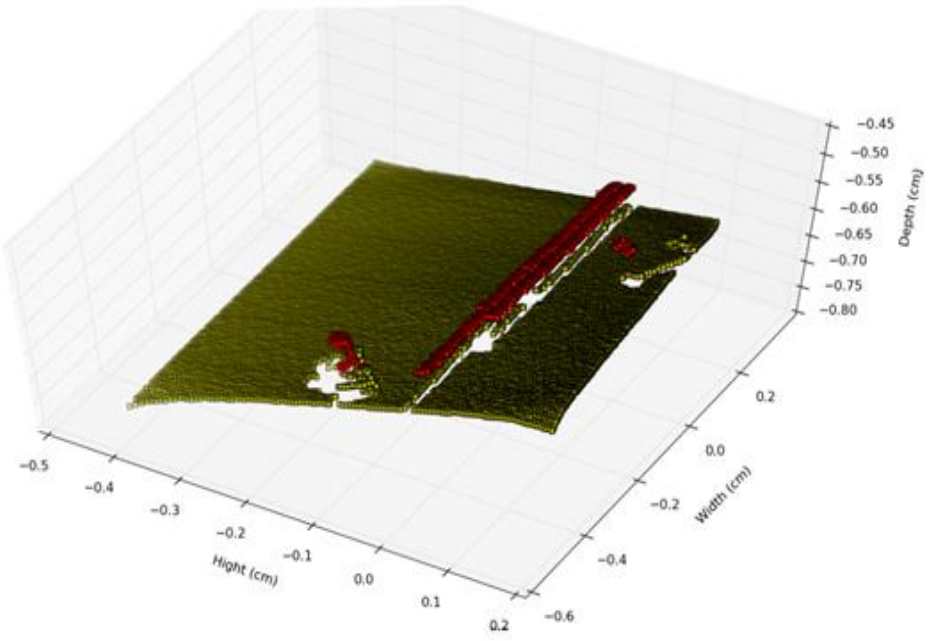


Figure 4-21: 3D point cloud mapping

4.3.2.2 Plane Detection & Filtration in a Point Cloud

The plane detection in point cloud is a kind of filtration used to separate between the objects and the ground in the scene based on the elevation data. The objective of this task in this project is to develop an algorithm capable of detecting a plane robustly in a dense 3D point cloud in a real-time. The approach proposed to process the plane detection in this project is based on a Random Sampling Consensus (RANSAC) algorithm that is widely used for this kind of detection in computer vision.

The principle of this algorithm is to find the best plane in a random and very large number of 3D point clouds. This principle is explained in details by (Derpanis, 2010; Fischler and Bolles, 1981).

First, it selects a three-point randomly and calculates the parameters of the corresponding plane. Then it filters out the coplanar point and non-coplanar point based on a given threshold. After that, it iterates these procedures a number of times; at each time, it compares the new count of the obtained coplanar point with the last highest one being saved. Once it is greater than the previous one, it replaces the count of the coplanar point. The inputs required for this algorithm are as follows:

- The 3D point cloud list.
- The tolerance threshold of the distance between the chosen plane and the other points.
- The foreseeable-support is the maximum feasible points belonging to the same plane. It is deduced from the point density and the maximum foreseeable ground plane surface.
- The probability ($\boldsymbol{\alpha}$) is a minimum probability of finding at least one good set of observations in (***N)*** trials. It usually lies between 0.90 and 0.99.

Algorithm 1 describes the plane extraction in detail.

Algorithm 1: RANSAC for plane detection

```
bestSupport = 0;
bestPlane(3,1) = [0, 0, 0];
bestStd = ∞;
i = 0;
e = 1 - forseeable-support/length(point-list)
N = round(log(1 - α)/log(1 - (1 -e))
while i ≤ N do
    j = pick 3 points randomly among (point-list)
    pl = pts2plane(j)
    dis = dist2plane(pl, point-list)
    s = find(abs(dis) ≤ t)
```

```
12:     st = Standard-deviation(s)
13:      if (length(s) > bestSupport) or (length(s) = bestSupport
and st <bestStd) then
14:           bestSupport = length(s)
15:           bestPlane = pl;
16:           bestStd = st
17:     end if
18:     i = i + 1
19: end while
```

The depth data used to validate the plane extraction consists of the pipeline structure and third-party interference object, as shown in RGB image data in Figure 4-22.

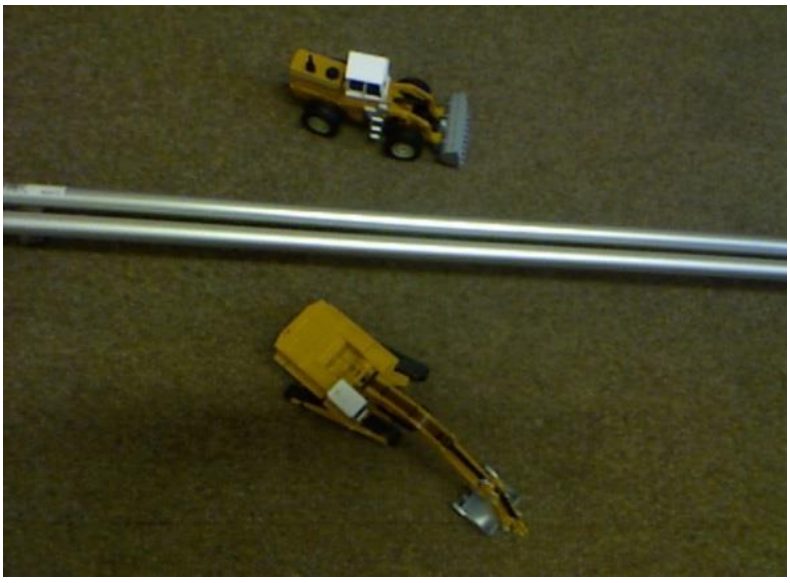


Figure 4-22: RGB of expected sample of pipeline and third-party objects

The result of the proposed plane detection algorithm shows how the coplanar points of the plane extracted efficiently, which represent the ground and the

remaining non-coplanar points represent the objects in the scene as shown in Figure 4-23.

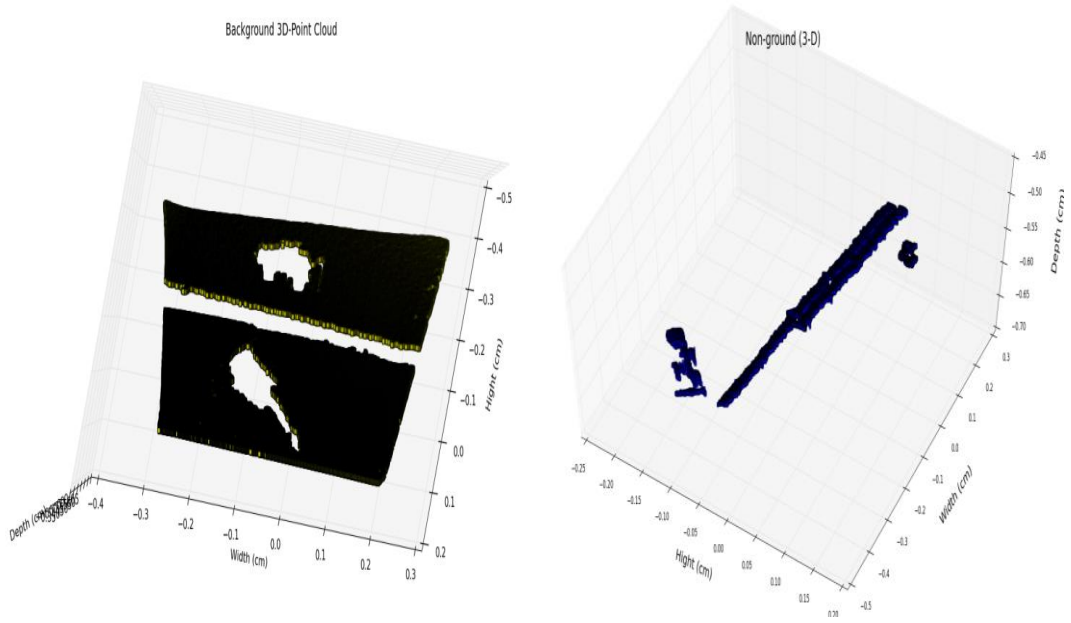


Figure 4-23: Coplanar & non-coplanar filtration in the 3D point cloud

4.3.2.3 Edge Detection of Objects (Non-Coplanar)

The edge detection is performed in the 2D depth image of the non-coplanar region of interest which represents the object boundaries. It is proposed to be used in the pipeline detection algorithm to map the boundaries of the objects based on their real elevation contrast rather than the colour contrast which is later used to extract the potential line structure automatically. The canny edge detector is proposed in this system to detect the boundaries of the objects which is one of the most robust methods for edge detection.

The process of this edge detector is already described in (Section 4.2.3.2) through multi-steps, where here it is based on the depth image rather than the colour image.

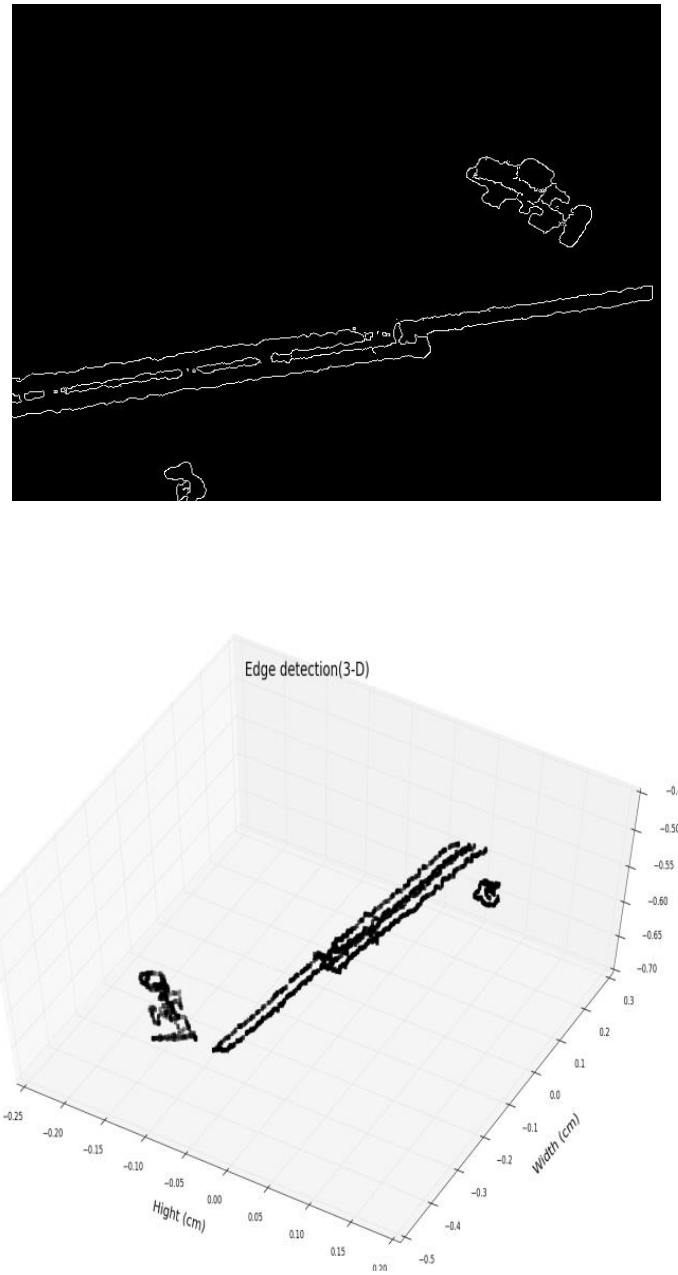


Figure 4-24: Object boundary detection

The visual performance of the proposed edge detection algorithm is represented in Figure 4-24 as a depth image and a 3D point cloud. The results verify the performance reliability of the object boundary detection algorithm.

4.3.2.4 Boundary's Height Filtration

This section presents the developed filtration algorithm of the boundary's height information of the objects (non-coplanar) relative to the ground detected in (section 4.3.3.2) to reduce the sample density, noisy points and computational process also increase the chance of finding the pipeline in those points. In this project, this algorithm is proposed to remove the boundaries (edges) that have an elevation higher than the maximum elevation of the existing pipelines (1.2 m based on the Alaskan pipeline), considering a small margin. The proposed approach used to perform this filtration is 3D RANSAC.

In order to perform this algorithm, three inputs are required which are:

- 3D point cloud of the edges ($P_{3D_{edges}}$)
- Plane parameters $[A, B, C, D]$
- Maximum height of the existing pipeline ($H_{max_{Pipeline}}$)

The RANSAC algorithm assumes the edge points are comprised of below the maximum height (inlier) and above it (outlier). The maximum height is predefined to match the highest elevation of the existing pipelines or the user requirement with a consideration of a small tolerance margin. First, as described in Algorithm 2, it initializes the inlier and outlier lists separately. Then, it calculates the distance of each point in the edges list to the candidate plane by using the standard equation of a plane in three-dimensional (Boljanovic, 2006) as in Equation (4-16).

$$d(Plane(A, B, C, D), Point(x, y, z)) = Ax + By + Cz + D \quad (4\text{-}16)$$

Where: d is the distance between the point and the plane. $A, B, C, and\ D$ are the plane parameters. $x, y, and\ z$ are the 3D world coordinates of the point.

Once the distance of the point is below or equals the threshold of predefined maximum height, it will append the point automatically into the inlier list, otherwise it will append it into the outlier.

Algorithm 2: RANSAC for boundaries height filtration

```
point-list = [P3D-edges];
plane = [A, B, C, D];
h-max = maximum height;
height-inlier = [ ];
height-outlier = [ ];
for p in point-list:
    dis = abs[A*p[1]+B*p[2]+C*p[3]+D]
    if (dis ⩽ h-max):
        height-inlier.append(p)
        height-inlier.append(p)
return height-inlier, height-outlier
```

The algorithm was tested on the 3D point cloud data of the object boundaries and is illustrated in Figure 4-25. The result shows that the performance of this algorithm is efficient for implementation it in this project. The processing rate of this algorithm is about 2 fps.

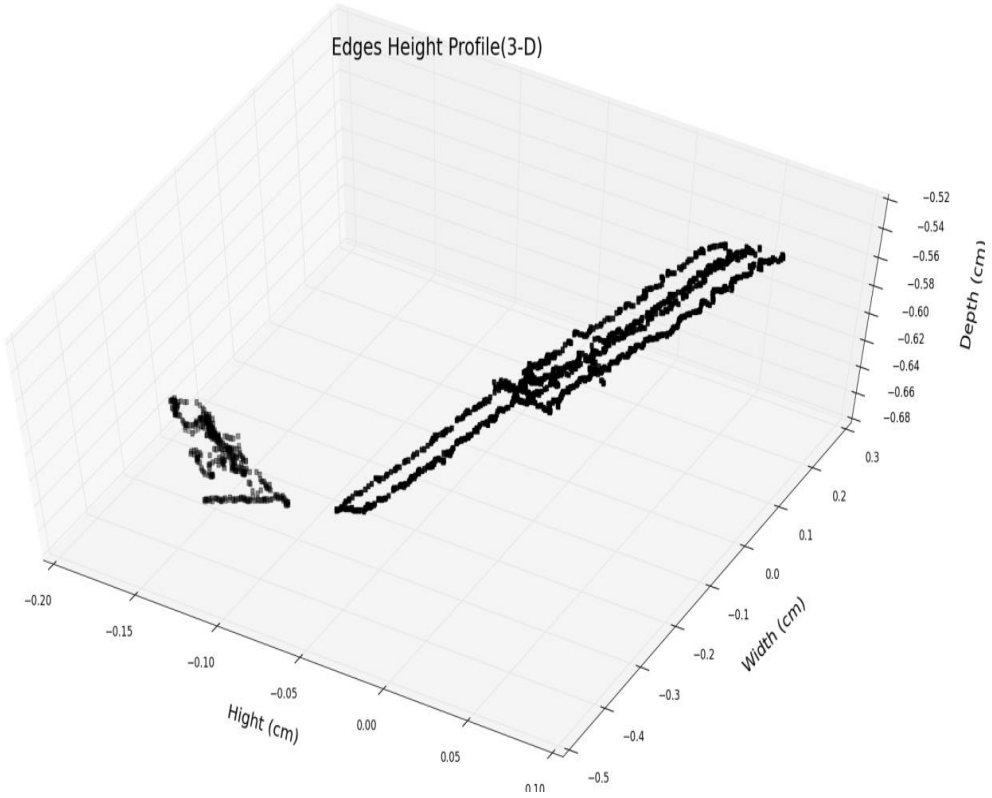


Figure 4-25: 3D point cloud of objects boundary's height filtration

Once the 3D point cloud of the object's boundary height was filtrated out, it is transformed into a raster in 2D depth image to prepare it for the Hough transforms in the next step, as shown in Figure 4-26.

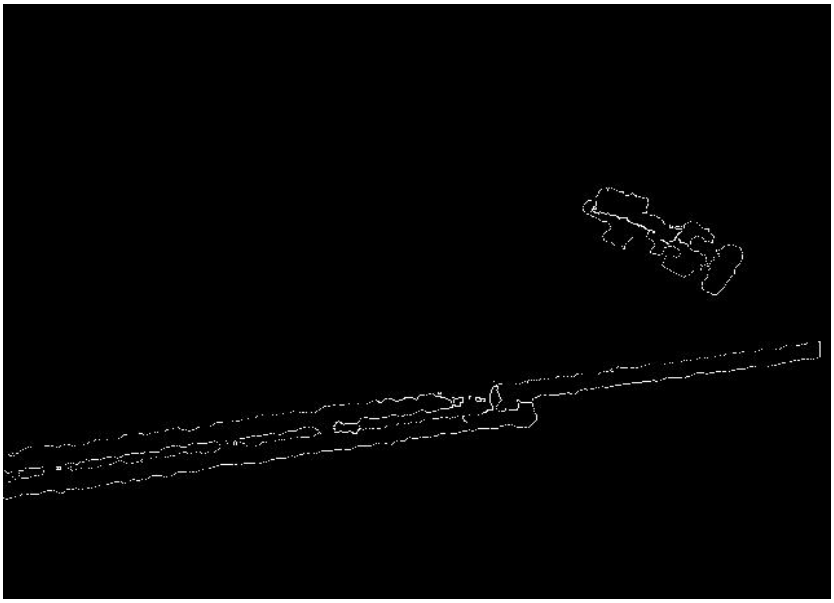


Figure 4-26: Depth image of objects boundary's height filtration

4.3.2.5 Straight Lines Detection

This section describes the proposed algorithm used to detect straight line structure based on the borders of the objects that are found and shown in Figure 4-26. The proposed method used in this step is the classical Hough transform, which is commonly proposed in computer vision applications to detect the straight line structure.

The requirements of this algorithm are:

- The binary image of the edge detection.
- The endpoints parameters $S(\mathbf{x}_{\mathbf{H}_{(1)}}, \mathbf{y}_{\mathbf{H}_{(1)}}, \mathbf{x}_{\mathbf{H}_{(2)}}, \mathbf{y}_{\mathbf{H}_{(2)}})$ of the detected lines.
- The shortest distance in Hough space $\boldsymbol{\rho}$ in pixels.
- The polarity of the shortest distance $\boldsymbol{\theta}$ in radians.
- The minimum number of intersections to detect a line.

- The minimum number of points that can form a line. Lines with less than this number of points are disregarded.
- Maximum gap between two points to be considered in the same line.

The details of how this algorithm works were presented in 4.2.3.3 in this thesis. The output of this algorithm is a 2D depth data of the detected line's endpoints. The expected output of this step is a 3D point cloud of the line's endpoints by performing a transformation.

4.3.2.6 Pipeline Verification

This section describes the approach developed to identify the potential line candidate as a real pipeline structure by using a standard geometric calculation and RANSAC. However, due to the probability that the straight lines detected in the previous section, could represent different objects than the pipeline or might be the result due to noise. So, the pipeline detection algorithm should be enhanced to identify the real pipeline structure more precisely.

The main requirements of this approach are as follows:

- The endpoint's list of the lines (S) already obtained in the previous section.
- A tolerance value (t_w) indicates the expected maximum width of the actual pipeline in meters.
- Another tolerance value (t_l) denotes to the minimum, acceptable length of the pipeline in meters.
- Vicinity's minimum number of 3D point cloud (N_{v_m}) required to accept the line segment as a real pipeline.

The outlined statements of the proposed problem's solutions are clarified as follows:

1) What is the distance between the parametric lines $d(P, L)$ with considering parallelism?
2) Is the distance $d(P, L)$ less or equal to the defined tolerance value (t_w)?

3) If (2) satisfied, what is the centre parametric line $(\mathrm{P_{s_M}}, \mathrm{P_{e_M}})$?
4) If (3) satisfied, what is the length of the centre line (l_M)?
5) Is the length of the new segments (l_s) equal or greater than the tolerance (t_l)?
6) If (5) satisfied, how many points lie in a vicinity of the line segments (N_v)?
7) Is the number of points (N_v) equal or greater than the vicinity's minimum number of points (N_{v_m})?

The proposed technique solving this problem consists of a geometrical calculation and RANSAC method. First, it selects one line $L(P_s, P_e)$ and checks its distance with the remaining lines by calculating the perpendicular distance between the points in the remaining lines and the selected line in order to avoid the nonparallel lines. This is done by using the formula (Boljanovic, 2006) in Equation (4-17) that computes the distance between a point and a line in n-dimensional to find the distance between the selected line $L(P_s, P_e)$ and the start points of the rest of the segments $d(P, L)$, as well the end points simultaneously. Then, the difference between them is checked to not exceed a specific margin in order to satisfy the semi-parallelism and if they satisfy the required tolerance (t_w).

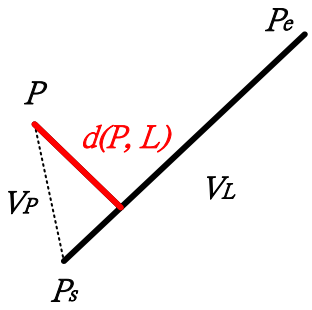


Figure 4-27: Geometry of the distance from a point to a line

$$d(P, L) = \frac{\left|\overrightarrow{\vec{V}_L} \times \vec{V}_P\right|}{\left|\vec{V}_L\right|} \tag{4-17}$$

$\vec{V}_L$ is the direction vector of the line $L(P_s, P_e)$ and $\vec{V}_P$ is the direction vector of the points (P) to the start point in the line (P_s). After that, the centre line segment $L_M(P_{s_M}, P_{e_M})$ is calculated by using the average formula as shown in Equation (4-18) to locate the midpoints that lie between the endpoints of any two parametric lines.

$$L_M(P_{s_M}, P_{e_M}) = \left(\frac{P_{s_1} + P_{s_2}}{2}, \frac{P_{e_1} + P_{e_2}}{2}\right) \quad (4\text{-}18)$$

Where: (P_{s_1}, P_{e_1}) and (P_{s_2}, P_{e_2}) refer to the start and end points of the first and second line segments, sequentially. Then, the standard Euclidean distance (Boljanovic, 2006) as shown in Equation (4-19) is used to calculate the length of the centre line (l_M) in order to make a decision either it is greater or equal to the predefined tolerance value (t_l), not lower.

$$l_M(P_{s_M}, P_{e_M}) = \sqrt{(x_{e_M} - x_{s_M})^2 + (y_{e_M} - y_{s_M})^2 + (z_{e_M} - z_{s_M})^2} \quad (4\text{-}19)$$

Where: l_M is the length of centre line segment. P_{s_M}, P_{e_M} are the start-point and end-point of the centre line segment, respectively. $(x_{s_M}, y_{s_M}, z_{s_M})\ and\ (x_{e_M}, y_{e_M}, z_{e_M})$ are the 3D world coordinates of the start-point and end-point of the centre line segment, respectively.

Once the centre lines satisfied the geometrical requirements, now one more step is required to verify the pipeline by calculating the weight of inlier points to the centre lines. So, the RANSAC method (Derpanis, 2010; Fischler and Bolles, 1981) is proposed to be used to compute the weight of the point clouds that are in-line with the centre line within a tolerance value referring to the maximum radius of the real pipeline.

The results in Figure 4-28 and Figure 4-29 show that the performance of this approach is robust and capable of detecting the over-ground pipeline structure in real-time based on the depth data, with a processing rate of about 2 fps.

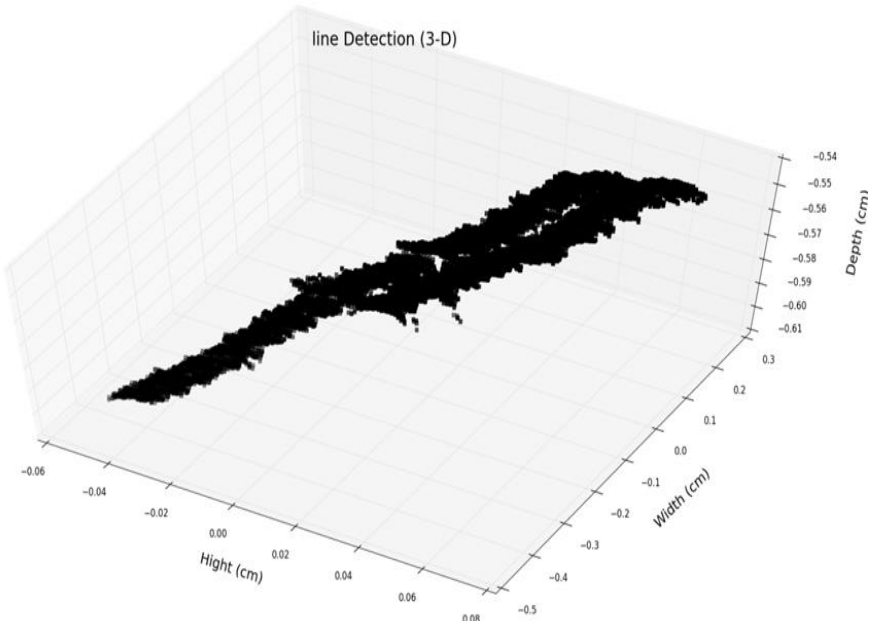


Figure 4-28: Pipeline identification in 3D point clouds

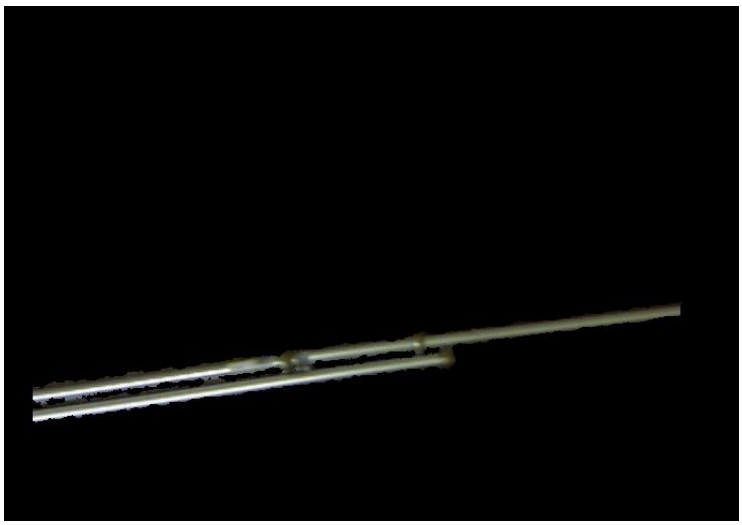


Figure 4-29: Pipeline identification in RGB image (for demonstration)

4.3.2.7 Pipeline Position Estimation

Estimating the position of the pipeline is the final step in the pipeline detection algorithm based on the depth information. It represents the position of the endpoints of the pipeline (EPs) relative to the camera reference to provide the location of the pipeline endpoints accurately, automatically, and near real-time. Those EPs are deducted to assist tracking the pipeline automatically, in addition, detecting any third-party interference around the pipeline location. Once the pipeline is already verified in the previous step, the endpoints of the centre line P_{s_M}, P_{e_M}, that is previously obtained, are representing the required EPs.

4.3.3 Performance

This section evaluates the performance of the depth-based aerial pipeline detection algorithm. To evaluate the performance of this algorithm, four indexes were considered to confirm the capabilities, which are sensitivity, specificity, false positive and false negative. Sensitivity and specificity relate to how likely the decision is correct while the false positive and false negative correspond to the errors.

- **Sensitivity** defined as the ratio of the number of the detected pipeline structure where the pipeline, in fact, relates to the total number of the present pipeline in the test.

- **Specificity** defined as the ratio of the number of the non-detected pipeline structure where the pipeline, in fact, does not exist to the total number of the absent pipeline in the test.

- **False positive,** defined as the ratio of the number of detecting pipeline structure where the pipeline, in fact, does not exist to the total number of the detected pipeline in the test.

- **False negative,** defined as the ratio of the number of not detected pipeline structure where the pipeline, in fact, exists to the total number of the not detected pipeline in the test.

In order to obtain an efficient performance, the algorithm should have high sensitivity and specificity and a low false positive and false negative ratio. Four tests were performed using this algorithm. The first one include a pipeline with a flat surface, the second includes a pipeline with a flat surface and objects, the third one includes a pipeline with the non-flat surface, and the last one includes a pipeline with non-flat surface and objects.

Table 4-3: Performance results of pipeline endpoints identification

	Test 1		Test 2		Test 3		Test 4	
	Detected	Not Detected	Detected	Not Detected	Detected	Not Detected	Detected	Not Detected
Presence	98	2	97	3	98	2	96	4
Absence	1	99	2	98	3	97	3	97
Sensitivity	98.00%		97.00%		98.00%		96.00%	
Specificity	99.00%		98.00%		97.00%		97.00%	
False Positive	1.01%		2.02%		2.97%		3.03%	
False Negative	1.98%		2.97%		2.02%		3.96%	

As results show in Table 4-3, the performance of the pipeline endpoints identification algorithm are capable efficiently of identifying the pipeline and estimate the position with a high detection rate under different circumstances.

4.4 Chapter Summary

In this chapter, two main algorithms were proposed and developed to identify the pipeline endpoints from the air and operate in near real-time, which are visible-based and IR-based data. Both of them were evaluated

experimentally and have confirmed the capabilities and reliabilities of identifying the pipeline endpoints in terms of identification decision, positioning accuracy, and processing load. The disadvantage of the visible-based algorithm is that it is dependent on the real database of the pipeline, which must be known, but the IR-based algorithm is capable of recognizing the pipeline without any external cues. The IR-based algorithm is capable of providing the position of the endpoints of the pipeline and the plane parameters of the ground in near real-time, which will be used in the auto tracking of the pipeline and the third-party detection.

Chapter 5
Third-Party Interference Detection

5.1 Introduction

Today, one of the main defects of pipeline safety in the world is due to third-party interference, almost involving 40% of the pipeline integrity defects. So, to reduce this kind of problem, it is necessary to develop and build a reliable algorithm, capable of detecting and localizing any third-party interference, automatically in real-time. Therefore, integrating a small aerial platform such as a UAV equipped with a vision sensor and in the appropriate computer vision algorithms is one of the promising solutions to accomplish this mission and that is part of the focus of this project. Hence, the aim of this chapter is to develop and build an efficient algorithm based on aerial IR and RGB vision data, capable of automatically detecting any third-party interference and instantly alarming the operation centre with vision and location evidence.

The algorithm proposed in this project contains three aspects, namely: detection, classification, and localization. The detection algorithm task is to detect the regions of interest of the objects geometrically based on the remaining point cloud obtained from the pipeline detection algorithm in the previous chapter after extracting the ground and the pipeline regions. While, the classification part was proposed to recognize third-party objects by using a Machine Learning (ML) technique after being filtered into objects Inlier the ROW and then transformed into the RGB image. Additionally, the localization was proposed to locate the centre position of the region of the third-party object

once classified using the measurements of the relative centre position of the area to the sensor reference.

The reliability of this algorithm is evaluated indoors in this chapter due to the sensor and the pipeline limitations. The evaluation is estimated using a small scale sample (scene) of a pipeline and some third-party objects. The results demonstrate the capability and accuracy of recognizing the third-party objects with a high degree of detection rate and efficient processing speed. A complete overview of the algorithm is illustrated in Figure 5-1 and addressed in detail (section 5.2) below.

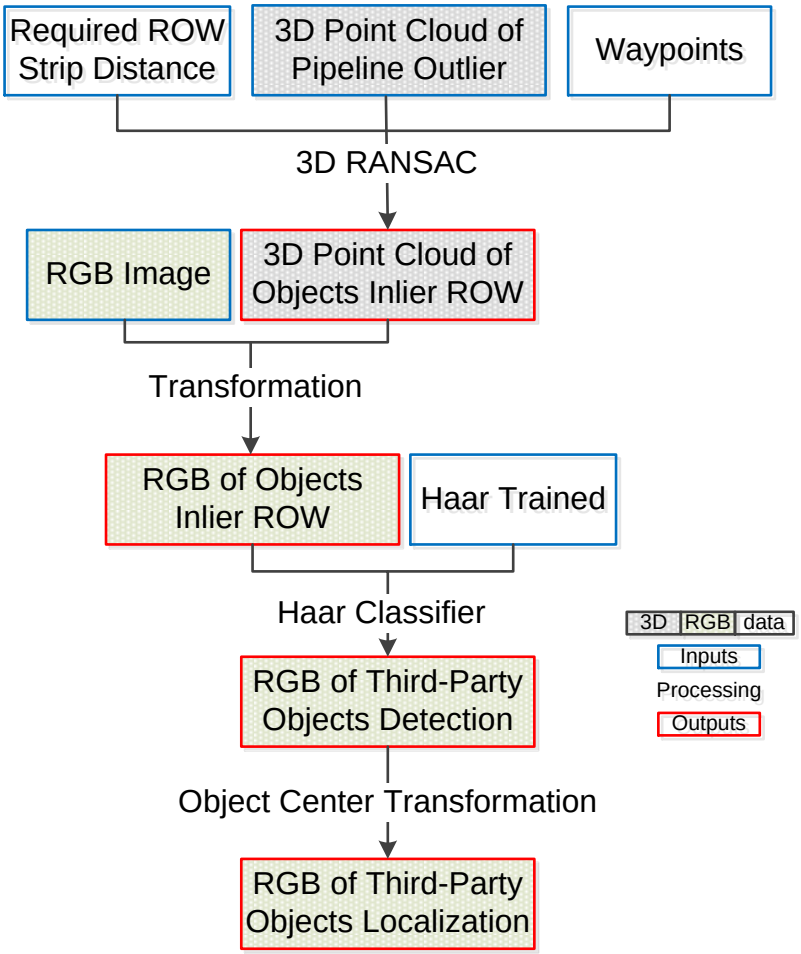


Figure 5-1: Third-party interference detection algorithm

5.2 Third-Party Objects Detection

This section describes the approach used in this project to automatically detect and estimate the position of the third-party interference objects in the vicinity of the pipeline structure in real-time using a fusion of depth and RGB images.

5.2.1 Objects Detection

The detection of objects in this project is to find any object that is found only around the Right-of-Way of the pipeline. The algorithm used is a 3D RANSAC in 3D point clouds to find the objects Inlier the ROW of the pipeline and a transformation to align those data with the RGB data. The details of this algorithm are explained in the following section.

5.2.1.1 Point Cloud of the Objects Inlier the ROW

This section describes the filtration process of the objects that lie within the pipeline Right-of-Way strip based on the filtered Outlier point cloud of the pipeline to concentrate only on the regions of interest over the image scene. The proposed algorithm used in this project to filter those point clouds is a simple Outlier removal algorithm that removes the points that are farther than a predefined threshold distance from a line segment. This threshold refers to the strip distance of the pipeline Right-of-Way. The line segment is known by the endpoints of the pipeline already obtained in the previous chapter.

The simple Outlier removal algorithm is an iterative process which is described in the following steps:

- Computing the corresponding linearity of the 3D non-ground point cloud by calculating the shortest distance $d(P_{3D}, L)$ of each point cloud (P_{3D}) into the pipeline segment (L) using ecludean distance fourmuls (Boljanovic, 2006) as in Equation (5-1).

$$d(P_{3D}, L) = \frac{\left|\overrightarrow{\vec{V}_L} \times \vec{V}_P\right|}{\left|\vec{V}_L\right|} \quad (5\text{-}1)$$

Where $\vec{V}_L$ is the direction vector of the line $L(P_s,\ P_e)$ and $\vec{V}_P$ is the direction vector of the point (P) to the start-point in the line (P_s).

- Proving if this distance $d(P, L)$ is less than or equal to the predefined threshold (t_l), it will then append its corresponding point (P) as Inlier (P_{In}), otherwise it will be Outlier (P_{Out}) using Equation (5-2).

$$P = \begin{cases} P_{In} & if\ d(P, L) \leq t \\ P_{Out} & Otherwise \end{cases} \quad (5\text{-}2)$$

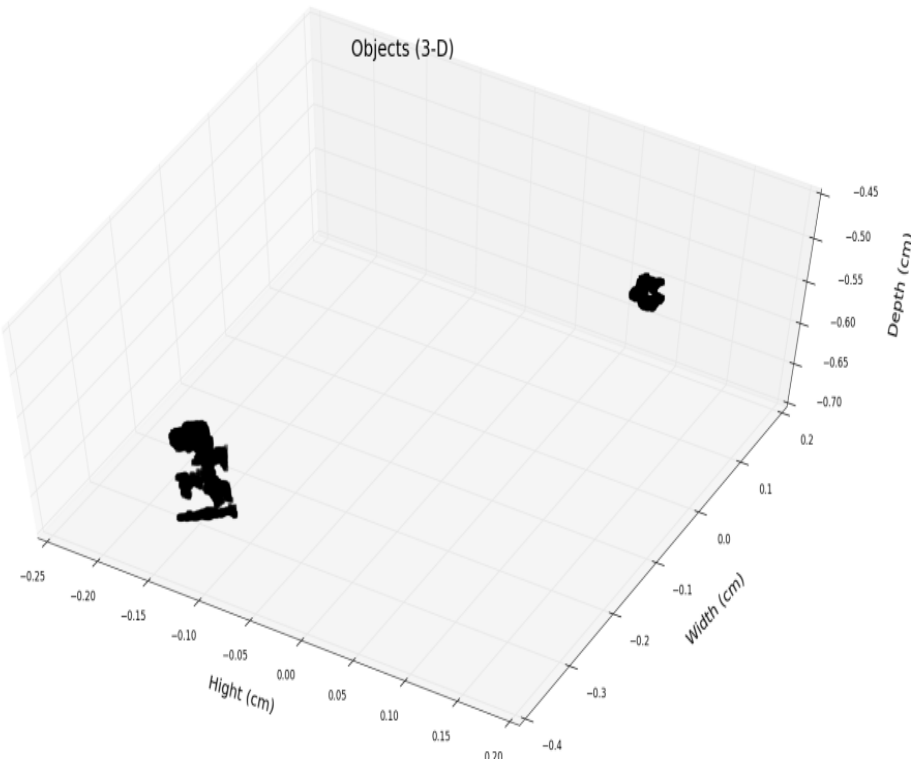


Figure 5-2: 3D point cloud of the detected objects

Figure 5-2 represents the 3D point cloud construction of the detected objects that are corresponded to the predefined Right-of-Way strip.

5.2.1.2 RGB of the Objects Inlier the ROW

Projection of the 3D point cloud map into the RGB map is used in the project to align the interested region in the IR map after being processed with the RGB map in order to achieve more processing on the RGB map or for exposing in colour. Now, the 3D point cloud map only contains geometry information while the RGB has colour information which all of them require for the purpose of this project. However, the problem is that there is a difference between the corresponding pixels in the 3D point cloud and RGB map. So, in this section, a projection of each pixel in the 3D point cloud map into its corresponding pixel of the RGB map is proposed and described.

Building a 4xN homogeneous matrix (Theoharis et al., 2008) of the given IR map (I_{dh}) from the original 3xN matrix (I_d) is shown in Equation (5-3).

$$I_d = \begin{bmatrix} u \\ v \\ d \end{bmatrix} \rightarrow \begin{bmatrix} u \\ v \\ d \\ 1 \end{bmatrix} = I_{dh} \tag{5-3}$$

After that, the 3D rotation (R) with 3D translation (T) matrices in one matrix are combined. Then, their homogenous matrix is built by adding the last raw data to multiply it with the homogenous matrix (Theoharis et al., 2008) of the IR data as shown in Equation (5-4) in order to transform it into a homogenous matrix for the RGB matrix.

$$Ih_{RGB} = \begin{bmatrix} \mathrm{u}' \\ \mathrm{v}' \\ \mathrm{d}' \\ 1 \end{bmatrix} = \begin{bmatrix} \mathrm{R}_{11} & \mathrm{R}_{12} & \mathrm{R}_{13} & \mathrm{T}_{\mathrm{x}} \\ \mathrm{R}_{21} & \mathrm{R}_{22} & \mathrm{R}_{23} & \mathrm{T}_{\mathrm{y}} \\ \mathrm{R}_{31} & \mathrm{R}_{32} & \mathrm{R}_{33} & \mathrm{T}_{\mathrm{z}} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ d \\ 1 \end{bmatrix} \tag{5-4}$$

Finally, the transformed RGB data $(\mathrm{u}', \mathrm{v}')$ is extracted from the homogenous RGB matrix (Ih_{RGB}) by dividing the fourth element and eliminating the third element (d') to get the default matrix of the RGB (I_{RGB}).

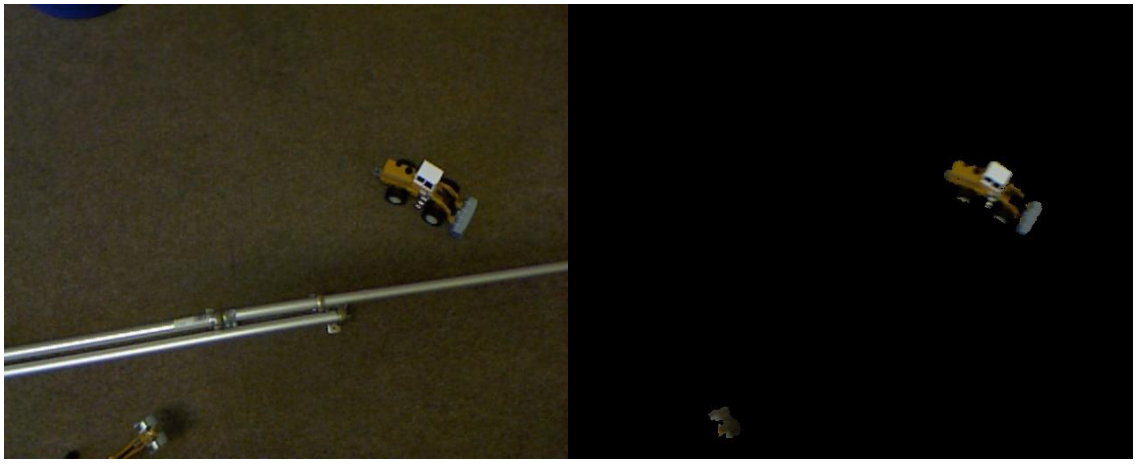


Figure 5-3: RGB data of the detected objects

As shown in Figure 5-3, it represents the result of the object detection in an RGB image after filtering out the background, the pipeline, and the Right-of-Way Outlier objects. These results will be used to classify the objects as an intrusion or not.

5.2.2 Third-Party Objects Classification

The classification of the third-party is to recognize the objects that have already been detected in (Section 5.2.1.2) based on its features. To recognize the third-party objects in real time, the Haar classifier algorithm, firstly published by Viola and Jones (2004), to detect face's features in images in real-time, will be used. The Haar classifier is one of the supervised Machine Learning classification techniques. This classifier is capable of running online and at a high detection rate based on real-time video footage.

Haar-like features consider adjacent rectangular regions at a particular location in a detection window, then sum up the pixel intensity values in each region and compute the difference between these sums (Viola and Jones, 2004). This difference is then used to categorize subsections of an image.

This classifier works as follows:

1) Create sample images (offline)
2) Haar Training (offline)
3) Performance testing of the classifier (Online).

5.2.2.1 Collect & Create Samples

First, this classifier requires collecting a set of positive and negative sample images to be trained with a few illuminations and pose variations that are undistorted. As much as the number of samples is increased, the performance increases. The positive samples only contain the target object and, in this project, some of the third-party objects are selected. Also, it is essential to provide some negative specimens, which do not contain the target object being trained for, to supply the training. A small set of positive and negative samples are shown below in Figure 5-4 and Figure 5-5, respectively.

Figure 5-4: Some of the positive samples used to train the classifier

Figure 5-5: Some of the negative samples used to train the classifier

After collecting the samples, the target object is cropped to get the positive training samples from the positive image with a proper size sub-window of the image with sensible widths and heights to include only the object. Cropping is achieved manually for each sample by using any photo editing tool; clipper image software was used for this purpose.

5.2.2.2 Haar Training

This section describes the Haar training algorithm based on the integral image in order to process the Haar features of the object candidate in constant time. The cascade of stages is proposed to eliminate non-object candidates quickly. Each stage consists of many different Haar features. Each of them is classified by a Haar-feature classifier to generate an output to the stage comparator. The stage comparator sums up the outputs of the Haar feature classifiers and compares this value with a stage threshold to determine if the stage should be passed or not. If all stages are passed the object candidate is concluded to be third-party interference. These terms will be discussed in more detail in the following sections

5.2.2.2.1 ***Haar-Like Features***

Haar-like features are a rectangular group of pixels representing the contrast variances between their adjacent instead of using the intensity values of the pixels that determined their relative dark and light area as shown in Figure 5-6.

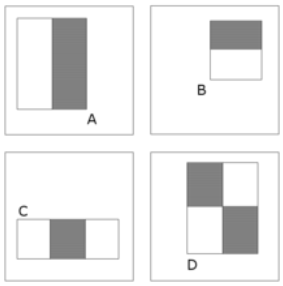


Figure 5-6: Basic types of Haar-like features (Viola and Jones, 2004)

Those rectangle features can be computed rapidly and in constant time with any size of rectangular pixel group in order to detect any objects with various sizes by using the integral image algorithm, first used by Viola and Jones (2004), which is defined in Equation (5-5):

$$ii(x, y) = \sum_{\acute{x} \le x} \sum_{\acute{y} \le y} i(\acute{x}, \acute{y}) \tag{5-5}$$

Where: (ii) represents the value of the integral image at any pixel (x, y) for the given original image (i) at the pixel $(\acute{x}, \acute{y})$ as shown in Figure 5-7 by summing up the intensity values of all the pixels starting from the top left location at point (*0, 0*) to the location of the target point (x, y).

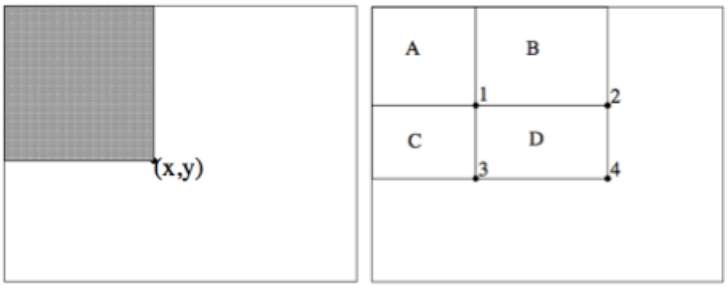


Figure 5-7: Integral image (Viola and Jones, 2004)

Moreover, the integral image can be computed in one pass through the total pixel intensity values of the image using Equation (5-6):

$$ii(x, y) = ii(x - 1, y) + s(x, y) \tag{5-6}$$

Where (s) is the sum of the cumulative row as defined in Equation (5-7) with the initial conditions $s(x,-1) = 0$, and $ii(-1,y) = 0$.

$$s(x,y) = s(x,y-1) + i(x,y) \tag{5-7}$$

Once the integral image of the original image has been obtained, it is used to extract any rectangle of the described Haar-like features in Figure 5-6. Hence, the value of any rectangle sum is simply computed in four array references as obvious in the notation in Figure 5-7 at right. For instance, the value of the integral image at location 1 is the sum of pixels in rectangle A, at location 2 is (A+B), at location 3 is (A+C), and at location 4 is (A+B+C+D). So, the sum of the original image (i) over the rectangle D can be defined in Equation (5-8):

$$\sum_{x_0 \le x \le x_1, y_0 \le y \le y_1} i(x,y) = ii(D) + ii(A) - ii(B) - ii(C) \tag{5-8}$$

5.2.2.2.2 ***Learning Algorithm***

Because there are too many rectangular features expected in each standard sub-window it would be computationally expensive to evaluate all of them. Fortunately, there is a small number of features and it is efficient enough to represent the target object. So, an AdaBoost learning algorithm is employed to select both the efficient feature and train its corresponded strong classifier as related to the work of Viola and Jones (2004). This algorithm constructs a strong classifier as a weighted linear combination of some weak classifiers weighted based on their accuracy. Hence, each single rectangle of the Haar-like feature could be considered as a weak classifier once it has the least weighted error which means it is efficient to reject regions that are highly unlikely to contain the target object.

An AdaBoost algorithm performs a sequence of boosting trial *t* on the giving set of sample images $(x_1, y_1), \ldots\ldots, (x_n, y_n)$, where n is the number of image sample. At the beginning, weights are initialized over the given set of sample images

using $w_{1,i} = \frac{1}{2ns}, \frac{1}{2ps}$ for $y_i = 0,1$ where ns and ps is the number of negative and positive samples, respectively. At each trial, each image receives a weight determining its importance by normalizing the weights in Equation (5-9).

$$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}} \tag{5-9}$$

Then, for each feature, train the weak classifiers (h_j) represented in Equation (5-10) which involves width and height pixel of the sub-window of the image (x_{sw}), a feature (f_j), a threshold (θ_j) and a polarity (p_j) indicating the direction of the inequality sign.

$$h_j(x_{sw}, f, p, \theta) = \begin{cases} 1 & if\ p_j f_j(x_{sw}) < p_j \theta_j \\ 0 & otherwise \end{cases} \tag{5-10}$$

The error (ϵ_j) is evaluated with respect to w_t as in Equation (5-11):

$$\epsilon_j = \sum_i w_i \left| h_j(x_{sw_i}) - y_i \right| \tag{5-11}$$

However, the classifier that has a lowest weighted error is selected as the most efficient weak classifier. After that, the weights are updated to emphasize the examples that were misclassified by using Equation (5-12).

$$w_{t+1,i} = w_{t,i} \beta_t^{1-\epsilon_i} \tag{5-12}$$

Where $\epsilon_i = 0$ if example x_{sw_i} is classified correctly, $\epsilon_i = 1$ otherwise, and $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.

Finally, the strong classifier (h) as shown in Equation (5-13) is a weighted combination of the *T* weak classifiers that are weighted according to their accuracy.

$$h(x_{sw}) = \begin{cases} 1 & if \sum_{t=1}^{T} \alpha_t(x_{sw}) h_t(x_{sw}) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t(x_{sw}) \\ 0 & otherwise \end{cases} \tag{5-13}$$

Where *T* is the number of weak classifiers h_t, $\alpha_t = log\frac{1}{\beta_t}$.

5.2.2.2.3 ***Cascade Classifier Structure***

This section describes the cascade classifier structure that is used to increase the detection performance of the target objects while radically reducing computation time. The overall structure of the cascade classifier consists of a sequence of strong classifiers arranged in a degenerate decision tree as shown in Figure 5-8. Each strong classifier involves a boosted set of weak classifiers. A positive decision from the first strong classifier triggers the evaluation of a second strong classifier that has also been adjusted to achieve very high detection rates, and so on for each strong classifier. So, the target object is detected once all the strong classifiers have been passed. A negative decision at any strong classifier leads to the immediate rejection of the sub-window before more complex classifiers are called upon to achieve low, false positive rates.

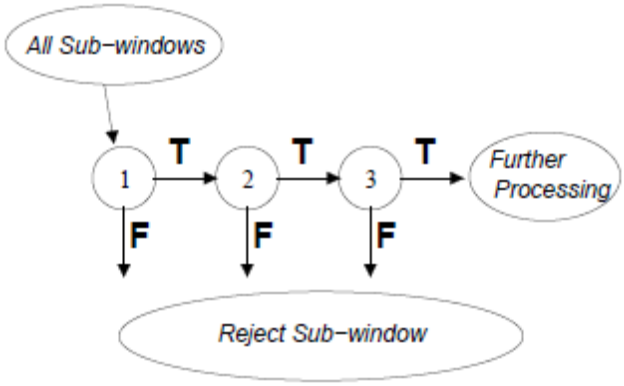


Figure 5-8: Cascade classifier structure (Viola and Jones, 2004)

5.2.3 Third-Party Objects Localization

This section describes the approach used to estimate the position of the detected third-party interference objects relative to the sensor reference. Once the third-party candidate objects are classified using the Haar classifier approach, the pixel located at the centre of the outcome sub-window of the Haar classifier, is converted into its corresponding point cloud to represent the locations of the third-party objects that relate to the sensor reference as 2D data (x_{3rd}, y_{3rd}).

5.3 Performance Evaluation

This section represents and evaluates the performance of the proposed algorithm used to detect the pipeline third-party interference objects, aerially and in real time. Four statistical performance indexes are considered to evaluate and estimate the accuracy of this proposed algorithm which are sensitivity, specificity, false positive and false negative. Sensitivity and specificity relate to how likely the decision is correct; either the third-party objects exist or do not exist respectively while the false positive and false negative correspond to the errors.

- **Sensitivity** defined as the ratio of the number of the detected pipeline third-party interference objects where the third-party objects, in fact, exist to the total number of the present third-party objects in the test.

- **Specificity**, defined as the ratio of the number of the not detected pipeline third-party interference objects where the third-party objects, in fact, doesn't exist to the total number of the absent third-party objects in the test.

- **False positive,** defined as the ratio of the number of detecting pipeline third-party interference objects where the third-party objects, in fact, doesn't exist to the total number of the detected third-party objects in the test.

- **False negative,** defined as the ratio of the number of not detected pipeline third-party interference objects where the third-party objects, in fact, exists to the total number of the not detected third-party objects in the test.

The algorithm should have high sensitivity and specificity and low false positive and false negative ratios to confirm the performance capability. Two sets of indoor frames were captured from the air using the pipeline detection algorithm that was described in Chapter Four to evaluate the performance of this algorithm which comprises trained third-party objects and the second set includes a mix of empty (Black) and not trained objects framers. The captured frames represent the remaining data (region of interest) after removing the background and the pipeline regions. The algorithm was tested online by sliding a search window through each frame image and checking whether an image region at a certain location is classified as third-party objects or not. The performance result of this test is presented in Table 5-1.

Table 5-1: Performance results of third-party interference detection

	Detected	**Undetected**
Presence	1826	174
Absence	32	1968
Sensitivity	91.30%	
Specificity	98.40%	
False Positive	1.72%	
False Negative	8.12%	
Processing Rate (f/s)	2	

According to Table 5-1, the performance of the third-party interference detection algorithm shows that it has the capability to detect correctly at a high rate of 91% and reject correctly at 98%. Also, it has an acceptable low detection error rate of 1.7% of incorrectly detecting and 8% of missed detection. So, the overall

performance and accuracy of the algorithm is efficient to detect the pipeline third-party interference objects and estimate its position with reliable performance.

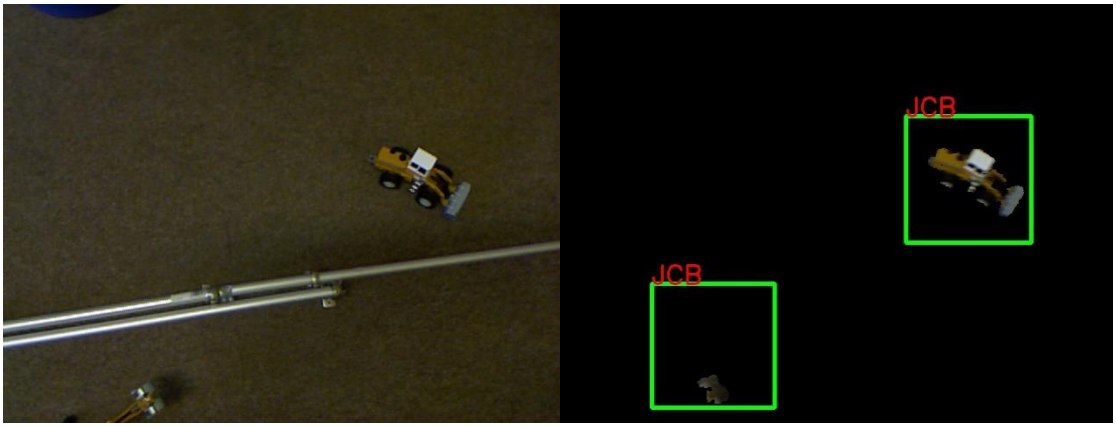


Figure 5-9: Demonstration of the pipeline's third-party interference detection

Figure 5-9 shows an image where the algorithm is employed and it correctly detects the third-party intervention.

5.4 Chapter Summary

In this chapter, filtrations and Haar classifier algorithms were proposed and developed to automatically detect and localize the pipeline third-party interference objects within the Right-of-Way in real-time. The performance outcome of this method was proofed experimentally using an indoor set of frames captured from above to emulate an aerial platform. The result shows that it is capable of efficiently detecting the pipeline third-party interference objects and, additionally, it can perform operations in real-time with an update rate of about 8 frames per second.

Chapter 6
Air-Vehicle Autopilot Waypoints Navigation

6.1 Introduction

The aerial following of the pipeline is one of the main requirements that could assist in providing an automatic real-time monitoring of the pipeline Right-of-Way integrity. In this project, a 2D waypoints tracking algorithm capable of following the pipeline structure accurately and in real-time was proposed and developed. While simultaneously maintaining the desired altitude; this tracking algorithm produces heading and altitude demands to control the attitude errors of the air-vehicle. This technique is mainly based on the real-time vision position estimation of the pipeline segment endpoints that were described previously in chapter four. Those endpoints represent the pipeline segment at each image scene. The position and speed of the UAV platform also need to be taken into account with regards to the acceptable speed of running the algorithm.

For security and safety reasons, the air-vehicle cannot fly directly over the pipeline. So, the first step in this algorithm is configuring the waypoints of the reference air-vehicle's course based on the estimated endpoints of the pipeline segments to produce target, past, and future course waypoints. Then, it calculates the path angles change between the current and next courses. After that, based on the configured target waypoint of the course, the algorithm computes how far the air-vehicle from that waypoint (proximity distance) is.

Simultaneously, it computes the turn anticipation distance (TAD) that corresponds to the course target waypoint. Once the proximity distance becomes equal to or lowers than the TAD, then the air-vehicle starts turning by updating the target waypoint and computing the desired heading. Also, due to safety considerations, as well as further maintaining autonomy of the system and keeping tracking the pipeline, it was decided that the system must have an autonomous capability that commands the air-vehicle to initiate loiter phase once it lost the vision-based EPs or detects some threats around the pipeline.

The piper cub flight control system developed in Matlab/Simulink environment is proposed to evaluate the tracking algorithm performance. This model was upgraded to be convenient with the requirements of this project. It comprises the piper cub dynamic model, its stability augmentation system, and the autopilot controllers. For the evaluation purpose in this chapter, the input waypoints representing the pipeline position were predefined offline. The performance of this algorithm presents an acceptable result that is reliable to track the pipeline robustly and in real time.

6.2 UAV Platform

The Piper J-3 Cub 40 platform as shown in Figure 6-1 was used in this chapter as a prototype to develop the tracking algorithm. This platform is a lightweight fixed-wing UAV that has three conventional control surfaces, which are the aileron, elevator and rudder. Hitec servo actuates each control surface.

The propulsion system is made up of a 1.29 KW Electric-Brushless-Dualsky Xmotor Series. The main platform specifications are detailed in Table 6-1.

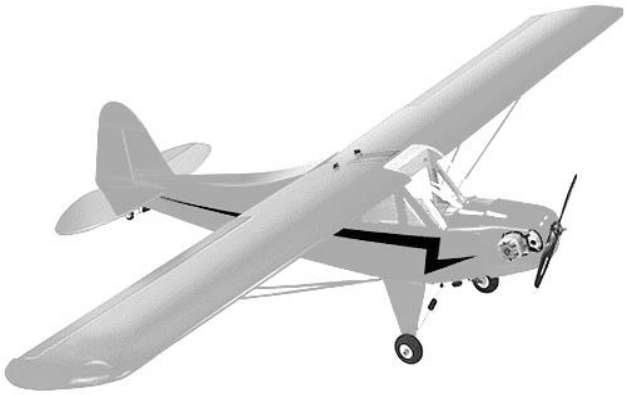


Figure 6-1: Piper J-3 Cub 40 aerial platform

Table 6-1: Piper J-3 Cub 40 general specifications

Wing Span	2.040 m
Profile chord	0.3070 m
Wing reference area	0.6290 m^2
Length	1.25 m
Weight (includes Payloads)	5.65 Kg
Engine	Electric-Brushless-Dualsky Xmotor Series
Power Unit	Li-Po 4-Cell pack 8000mAh, 14.8 V
Engine power	1.29 KW

6.3 Waypoints Navigation Algorithm

This section describes the proposed waypoints navigation algorithm in this project, as shown in Figure 6-2. The aim of this algorithm is to enable 2D waypoints tracking by navigating the air-vehicle through a reference course of

waypoint coordinates during the required altitude (h_d) which is acquired and maintained within the flight control system already developed by Moghimi (2009). The waypoints could be predefined offline prior the flight or online in real time. This algorithm produces the heading demand (ψ_d), that maintains the UAV platform tracking the pipeline and turns smoothly between the WPs transition. The heading demand is then sent into the flight control system which goes through the heading controller and ultimately translates into bank demand (ϕ_d).

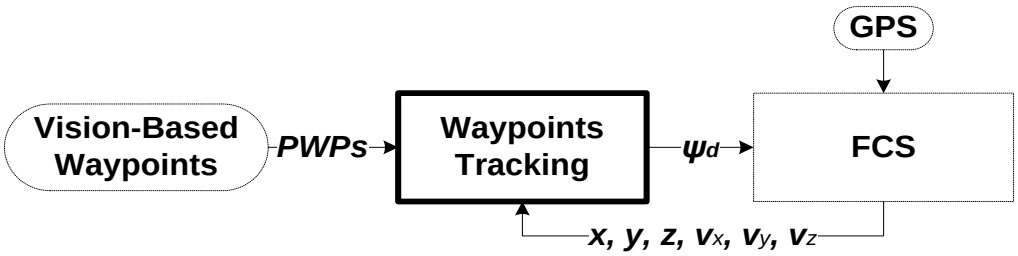


Figure 6-2: High-level block diagram of waypoints navigation

The proposed waypoints tracking algorithm requires two inputs to process the tracking/following algorithm that are:

1) The vision-based endpoints (EPs) that represent the pipeline segments that are already described in the pipeline detection algorithm in chapter four.

2) The UAV platform position ($\mathrm{x}_a, \mathrm{y}_a, \mathrm{z}_a$) and velocity ($\mathrm{v}_x, \mathrm{v}_y, \mathrm{v}_z$) data that can be obtained from the flight control model.

The processing structure of the tracking algorithm is presented in Figure 6-3 and described in more detail below.

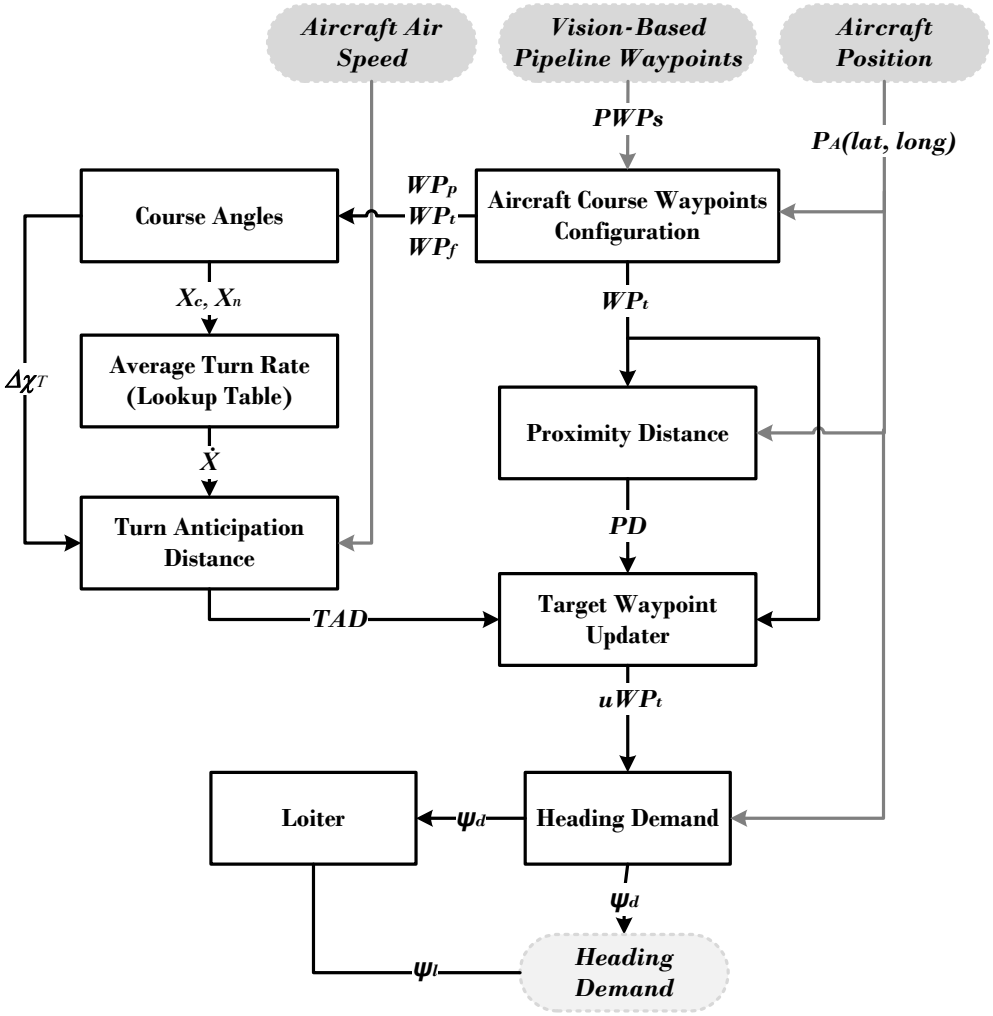


Figure 6-3: Waypoints navigation block diagram

6.3.1 Real-time Course Waypoints Generation

This section describes the real-time air-vehicle's course waypoints generation algorithm to keep the air-vehicle following the pipeline, taking the pipeline integrity issue as shown in Figure 6-4 into consideration. To preserve the pipeline integrity, the air-vehicle should not fly over the pipeline directly, so a

reference ground course waypoints (current (WP_c), target (WP_t) and future (WP_f)) should be generated for the air-vehicle at the right-side of the pipeline segment with a particular distance based on the online sequences of the pipeline segment endpoint (EPs) and the air-vehicle position (P_A).

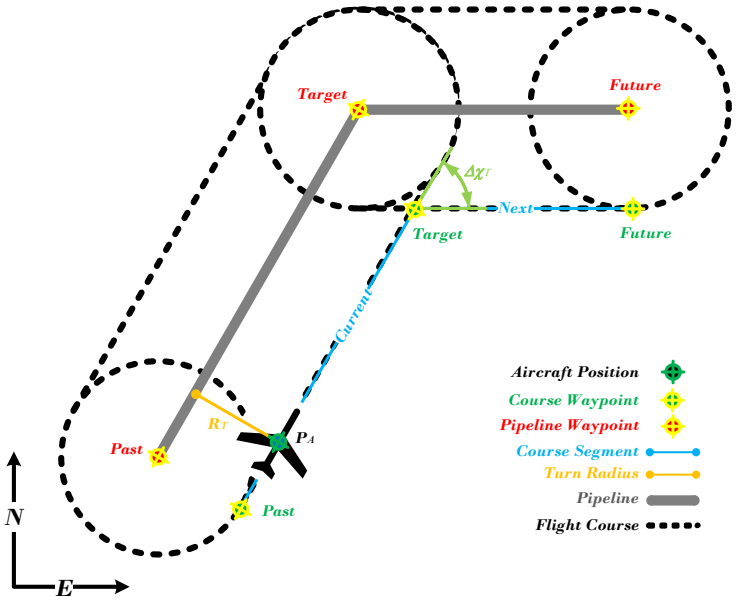


Figure 6-4: The reference ground course configuration

This configuration determines the structure of the pipeline waypoints, which first selects the nearest pipeline endpoint to the air-vehicle position as a current pipeline endpoint (EP_c) which also determines the current segment of the pipeline then the other corresponding endpoint of the current pipeline segment is considered as a target pipeline endpoint (EP_t) and the corresponding endpoint of the next pipeline segment is considered as a future pipeline endpoint (EP_f). Notice, when there is just one pipeline segment, the pipeline target (EP_t) and future (EP_f) endpoints' coordinates become equivalents. After that, the algorithm computes the required course waypoints (WPs) for the air-

vehicle based on the pipeline endpoints (EPs). If there is just one pipeline segment available it is easy to compute the course waypoints (WPs) directly by using Equation (6-1) to find the parallel segment which represents the direct course segment.

$$WP_{c,t,f}(x,y) = EP_{c,t,f}(x,y) - \left(R \times \frac{(-dy_{c,n}, dx_{c,n})}{\sqrt{dx_{c,n}^2 + dy_{c,n}^2}}\right) \quad (6\text{-}1)$$

Where: $dx_{c,n} = EP_{t,f}(x) - EP_{c,t}(x)$ and $dy_{c,n} = EP_{t,f}(y) - EP_{c,t}(y)$ represent north and east vectors of either the current or the next segments. R represents the offset distance between the pipeline and the air-vehicle's course. c, t, f denotes the current, target, and future waypoints, respectively.

However, if there are more than one pipeline segments, the target and future course waypoints are computed based on the interactions between the current course segments already computed unless the current waypoint is considered equal to the one already calculated in Equation (6-1).

6.3.2 Change Estimation of the Course Angle

In order to manoeuvre the camera view to follow the changeable pipeline segment vectors with a steady and uninterrupted view, the course angle change ($\Delta\mathcal{X}_T$) is estimated based on the difference between the current segment angle ($\mathcal{X}_c$) and the next segment angle ($\mathcal{X}_n$) using Equation (6-2). Those angles are computed based on the corresponding north and east of the configured course waypoints (WPs) relative to the image frame as denoted in Figure 6-4 by using the arctangent formula (Boljanovic, 2006) shown in Equation (6-3) and Equation (6-4), respectively.

$$\Delta\mathcal{X}_T = \mathcal{X}_n - \mathcal{X}_c \quad (6\text{-}2)$$

$$\mathcal{X}_c = \arctan\left(\frac{WP_{t_{north}} - WP_{c_{east}}}{WP_{t_{north}} - WP_{c_{north}}}\right) \quad (6\text{-}3)$$

$$\mathcal{X}_n = \arctan\left(\frac{WP_{f_{north}} - WP_{t_{east}}}{WP_{f_{north}} - WP_{t_{north}}}\right) \quad (6\text{-}4)$$

Where: $\Delta\mathcal{X}_T$ is the estimated change of course angles. $\mathcal{X}_c$ is the current segment angle relative to the image frame. $\mathcal{X}_n$ is the next segments angle relative to the image frame. $WP_{t_{north}}$ is the north coordinate of the target waypoint. $WP_{c_{east}}$ is the east coordinate of the current waypoint. $WP_{f_{north}}$ is the north coordinate of the future waypoint. $WP_{c_{north}}$ is the north coordinate of the current waypoint. $WP_{t_{east}}$ is the east coordinate of the target waypoint. Finally, $WP_{t_{north}}$ is the north coordinate of the target waypoint.

6.3.3 Turn Anticipation Distance Estimation

Turn Anticipation Distance (*TAD*) (Moghimi, 2009) represents the distance from the air-vehicle to the course target waypoint (WP_t), that is required to change the course into a circular trajectory tangent to the current course segment and the next course segment to smooth the transition from one course to another while keeping the deviation from reference path as small as possible. TAD is computed based on three parameters once the course angle changes, which are air speed of the air-vehicle (V_A), change of course angles ($\Delta\mathcal{X}_T$), and the air-vehicle average rate of turn ($\bar{\dot{\mathcal{X}}}$) using Equation (6-5).

$$TAD = \frac{V_A}{\bar{\dot{\mathcal{X}}}} \tan\left(\frac{\Delta\mathcal{X}_T}{2}\right) \quad (6\text{-}5)$$

Where: $\bar{\dot{\mathcal{X}}}$ is the air-vehicle average rate of turn which is estimated based on a constructed lookup table with assuming the air-vehicle maximum turn rate (at a maximum bank angle). V_A is the air speed of the air-vehicle.

6.3.4 Proximity Distance

The proximity distance is the distance from the air-vehicle position (P_A) to the course target waypoint (WP_t) as shown in Equation (6-6) (Moghimi, 2009), to assist making a turn decision and update the target waypoint (uWP_t) of the course once the air-vehicle has reached close enough to the target waypoint (WP_t) and becomes equal or more than the obtained turn anticipation distance (TAD).

$$proximity\ distance = \sqrt{\left(P_{A_n} - WP_{t_n}\right)^2 + \left(P_{A_e} - WP_{t_e}\right)^2} \tag{6-6}$$

Where: $P_{A_n}\ and\ P_{A_e}$ are the north and east coordinates of the aircraft position with relative to the image frame. $WP_{t_n}\ and\ WP_{t_e}$ are the north and east coordinates of the target waypoint relative to the image frame.

6.3.5 Heading Demand

The algorithm estimates the heading demand that is required to follow the course waypoints based on the corresponded air-vehicle position.

6.3.6 Loitering

Due to safety considerations, as well as further maintaining autonomy of the system and keeping tracking the pipeline, it was proposed that the system must have an autonomous capability that commands the air-vehicle to initiate loiter phase once a waypoint WP is lost for any reason or when it detect any defects around the pipeline or at the end of the flight when reaching the last, predefined WP. In the loiter phase, the air-vehicle start a steady and stable, continuous banking.

$$\psi_l = \psi + 5 * sample\ time \tag{6-7}$$

The heading of the loitering (ψ_l) algorithm works by increasing the input heading command (ψ), continuously at each sample time, by an increment value of five times the sample time as in Equation (6-7).

6.4 Platform Flight Control System

In this section, the flight control system of the Piper Cub platform is introduced to assist in evaluating the performance of the proposed tracking algorithm. It is the latest model that was developed for this platform at Cranfield University by (Moghimi, 2009). This model was constructed using Matlab/Simulink environment. The model comprises three sub-models that are:

1) 6DOF dynamics model;
2) Stability Augmentation System (SAS);
3) Autopilot controllers.

Each one of them is described more details in the following sub-sections.

6.4.1 Platform 6DOF Dynamic Model

The linear and nonlinear 6DOF dynamics models of the Piper Cub platform were developed in a Matlab/Simulink environment. The nonlinear dynamic model consists of the following sub-models:

1. Aerodynamic model;
2. Propulsion model;
3. Total forces and moments model;
4. Equations of motion model;
5. Mass and inertia model;
6. ISA atmosphere model.

The block diagram of this model is presented in Appendix B. The linear model acquires elevator, aileron, rudder, and throttle as inputs and produces 12 state space variables as outputs.

6.4.2 Stability Augmentation System Model

The Piper Cub platform has a conventional high-wing configuration with zero sweep angles, which means it is naturally stable. However, since it is relatively lightweight and supposed to be flown outdoors and exposed to wind effects, it will have weak stability and control issues that could lead to not following the waypoints properly unless it has a Stability Augmentation System (SAS) on-board. So, this system was already developed by (Moghimi, 2009) to prevent excessive control commands and unstable flight conditions. Also, it enables the air-vehicle to have acceptable flight handling qualities according to the international UAV standards and assures stable flight throughout the flight mission.

The classic PID controller was used to develop this system. The general architecture is basically based on the standard model that was developed by (Stevens and Lewis, 1992). However, changes have been made in the architecture wherever necessary to improve the handling qualities response of the air-vehicle. The developed system consists of three controllers that are:

1) Pitch SAS;
2) Roll SAS;
3) Yaw SAS.

Pitch SAS has pitch angle and pitch rate feedback; roll SAS contains rolling angle and rolling rate feedback while yaw SAS only benefits from yaw rate feedback. All feedback loops have their particular gains that were initially obtained from linear SAS design that's already been designed by (Saban, 2006), and was re-tuned manually for the Piper Cub model during the simulations. Those gains that were re-tuned for each type of the SASs are presented in the table in Appendix C.

6.4.3 Autopilot Model

The autopilot model of this platform is designed to be capable of automatically tracking/following the pipeline structure in real time, while taking into account the platform limitations. This model was developed based on the standard PID controller by (Moghimi, 2009). It consists of three controllers, which are:

1) Altitude holds controller;
2) Heading holds controller;
3) Auto-throttle (Speed) controller.

Those controllers were successfully designed and their capabilities were proofed to automatically acquire the desired altitude, heading, and speed to assist the air-vehicle's in following the course waypoints properly.

6.5 Simulation Results

This section represents the simulation results of the 2D waypoints tracking/following algorithm that was designed and developed to evaluate its performance using Matlab/Simulink environment in this chapter. The performance is assessed in the following subsections.

6.5.1 Waypoints Navigation Performance

This section presents the performance evaluation of the proposed tracking/following algorithm and how it is capable of keeping track of the pipeline at the flight level and loitering phases. The pipeline waypoints were generated offline to match the required missions that were proposed. The platform speed and altitude were initialized as 15 m/s and 400 ft (100 m), respectively. Wind speed is considered as zero in this simulation.

Figure 6-5 shows that the heading controller performance is acceptable, though there was a small deviation from the reference when the air-vehicle flew

between the waypoints, which can be reduced by further tuning of the heading controller. Also, it shows the stability of the loitering behaviour.

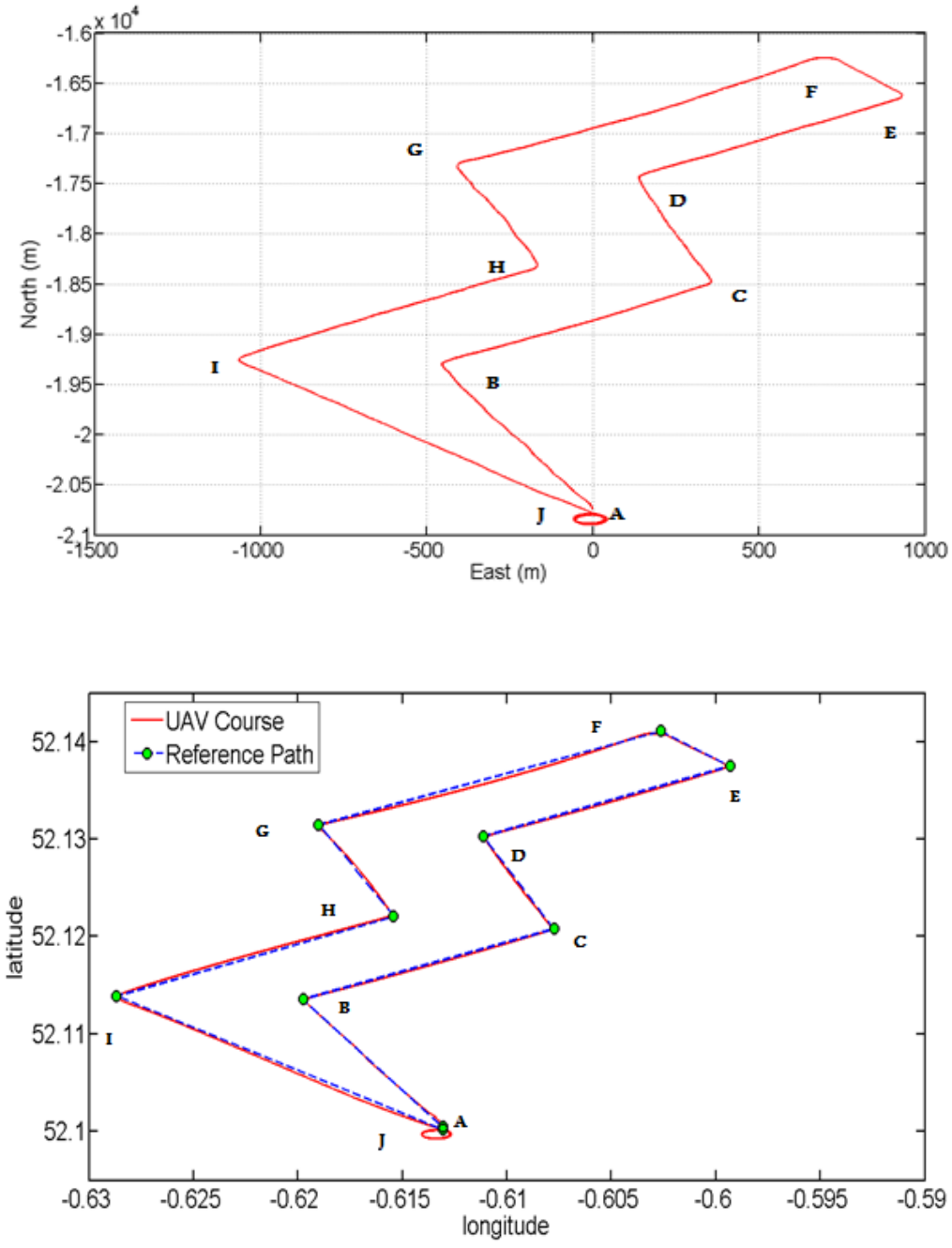


Figure 6-5: Air-vehicle course projected on reference path (Start waypoint A; end waypoint J)

When air-vehicle reaches the final waypoint (J), it starts a loitering manoeuvre.

6.5.2 Heading Performance

This section presents the performance of the heading controller and how the demand heading angle keeps track of or follows the desired course in cruise and loitering modes.

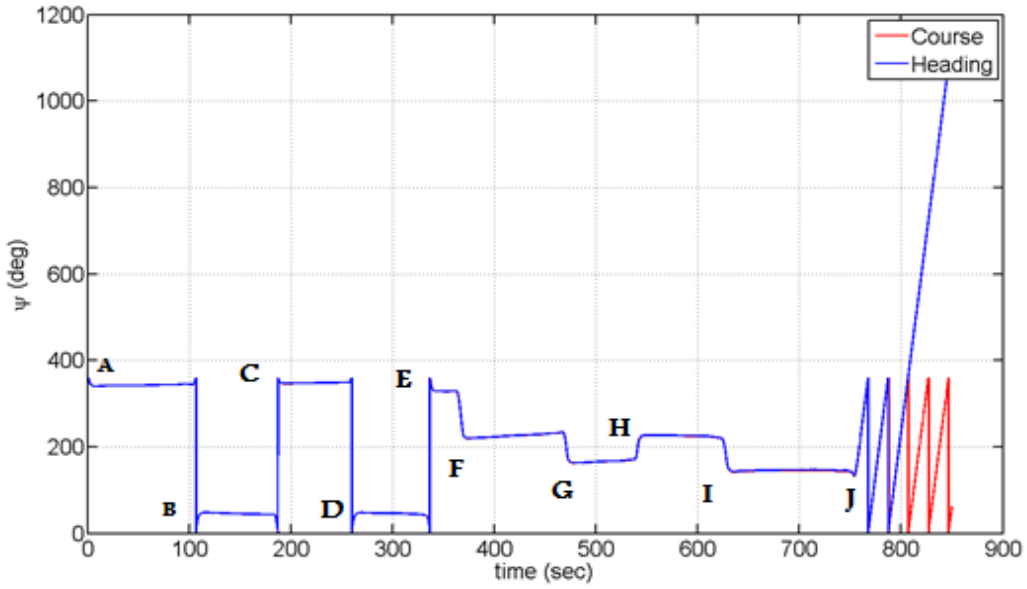


Figure 6-6: Heading virus course

Figure 6-6 shows the performance of the heading controller and how it follows the course of the cruise flight (from the beginning to 760 sec) and loitering mode (from 760 sec to the end).

6.5.3 Roll SAS Performance

This section demonstrates the dynamic behaviour of the bank angle along the flying mission at cruise and loitering modes to represent the performance of the roll SAS and the heading autopilot model.

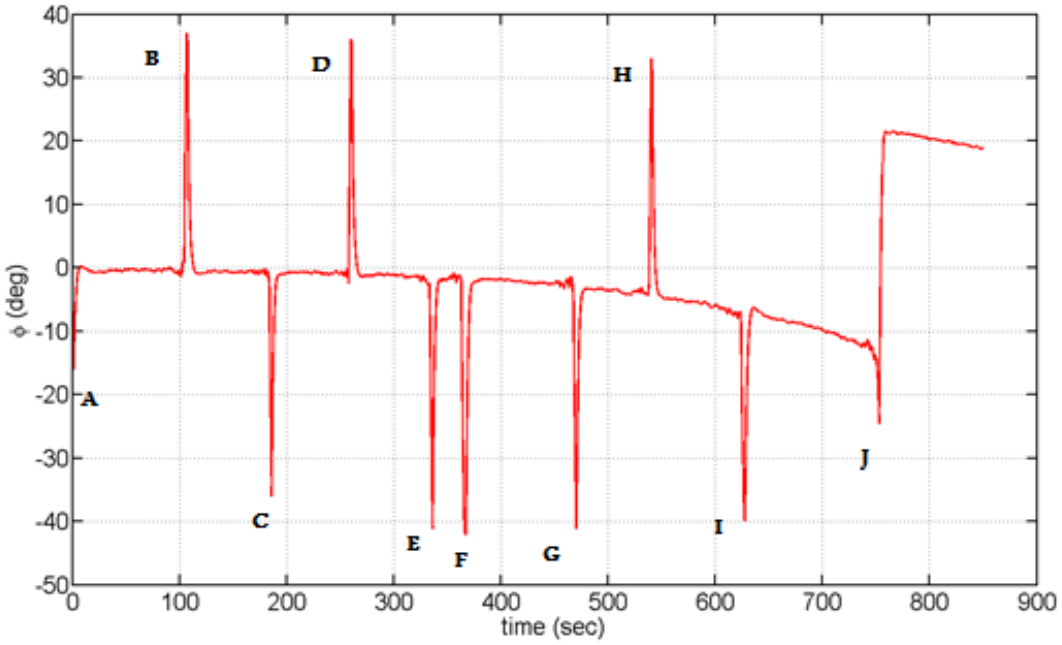


Figure 6-7: Bank angle history

The history of the bank angle variation is shown in Figure 6-7. Roll SAS is responsible for preventing very rapid changes in bank angle, and limiting its magnitude within -60 and +60 degree range. If the bank angle change rate is too high, it could result in a bank angle magnitude going beyond the specified limit, and therefore severely endangering flight stability and safety. As seen in Figure 6-7, the SAS is able to keep the bank angle within the desired limits. However, the bank angle experiences slight undesirable low and high-frequency oscillations after each turn phase. Undesirable high-frequency oscillations are also observed when the air-vehicle is in loitering phase. However, this gets suppressed automatically afterwards as shown in the figure above. Both of the addressed undesirable phenomena can be alleviated by further tuning of heading autopilot and Roll SAS controller and feedback gains. As a conclusion, the Roll SAS and heading autopilot performance are acceptable, having the potential to be further tuned for improved performance.

6.5.4 Altitude Performance

This section demonstrates how the air-vehicle is capable of keeping the desired altitude while moving throughout the waypoints.

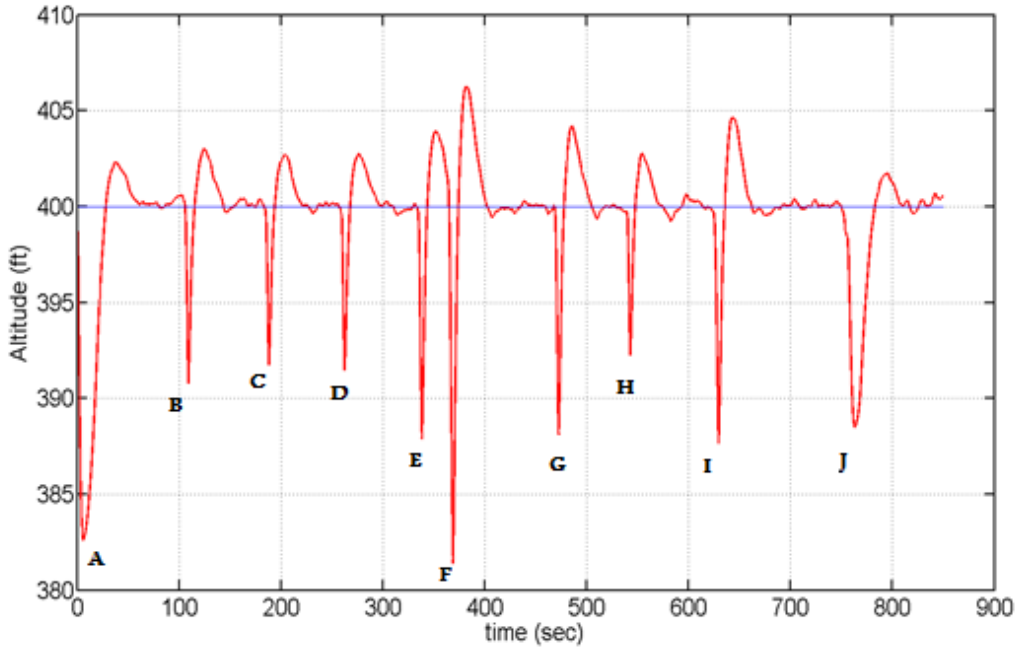


Figure 6-8: Altitude history

Figure 6-8 shows that the altitude hold controller is able to maintain the commanded altitude at the required 400 ft during cruise flight, the oscillations are observed due to the turn to follow the required course.

6.5.5 Pitch SAS Performance

This section presents the dynamic behaviour of the longitudinal attitudes that evaluate the performance of the pitch SAS throughout the cruise and loitering modes.

Figure 6-9 illustrates the air-vehicle, longitudinal angles, and variation history. The pitch angle (θ) is maintained within -8 to +8 degrees during most of the flight time, with a maximum value of 18 degrees at the beginning. The angle of attack (α) is kept well within -4 to +4 degree range at all times, which will ensure

the air-vehicle is far away from the stall condition. The path angle (γ) mostly remains within -6 to +6 degrees throughout the flight. The overall assessment of air-vehicle longitudinal dynamic behaviour concludes that pitch SAS is well able to ensure the stability of the air-vehicle in cruise flight and loitering mode.

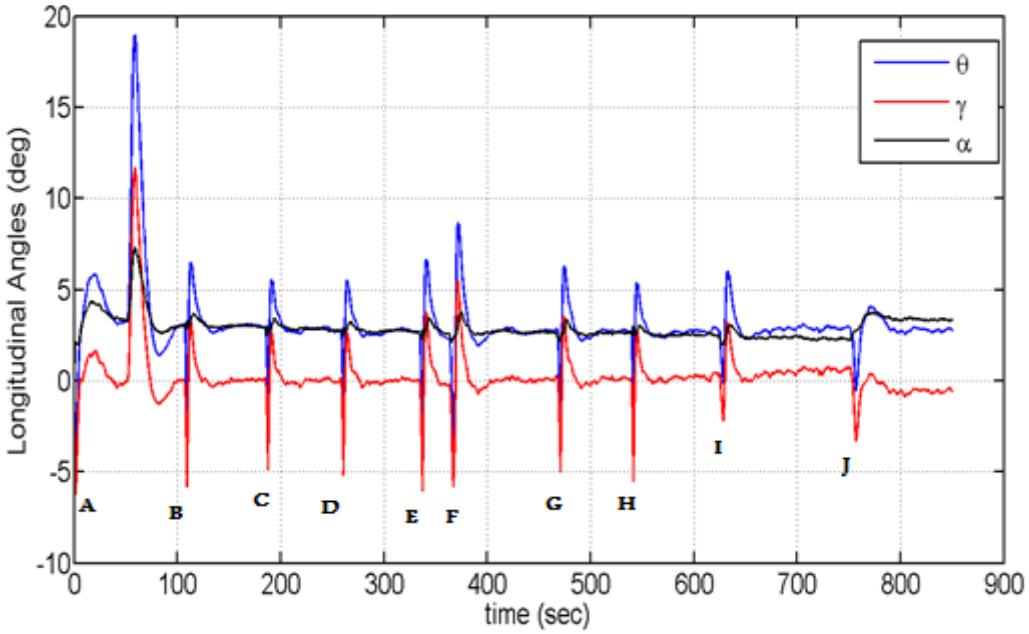


Figure 6-9: Longitudinal angles history

6.5.6 Speed Performance

This section presents the air speed performance throughout the cruise and loitering modes. Figure 6-10 shows the airspeed history, having high, oscillatory behaviour and relatively consistent moderate deviation from the reference value (15 m/s). These oscillations are mainly because of imperfectness of the engine propeller geometry, which makes the propeller efficiency, low and causes undesirable engine model outputs. The solution to this problem is selecting a more efficient propeller for the engine that will deliver higher efficiency, as well as higher propeller power coefficient (C_p) and propeller thrust coefficient (C_T) values at lower advance ratio (J).

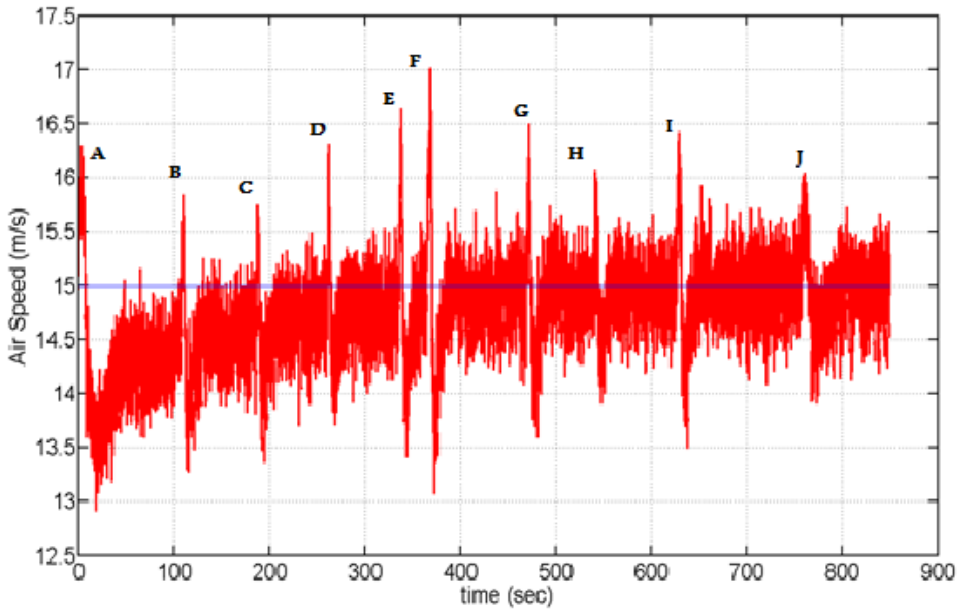


Figure 6-10: Airspeed history

6.5.7 Loitering Performance

This section presents the aerial platform loitering mode performance of the pipeline structure tracking/following algorithm.

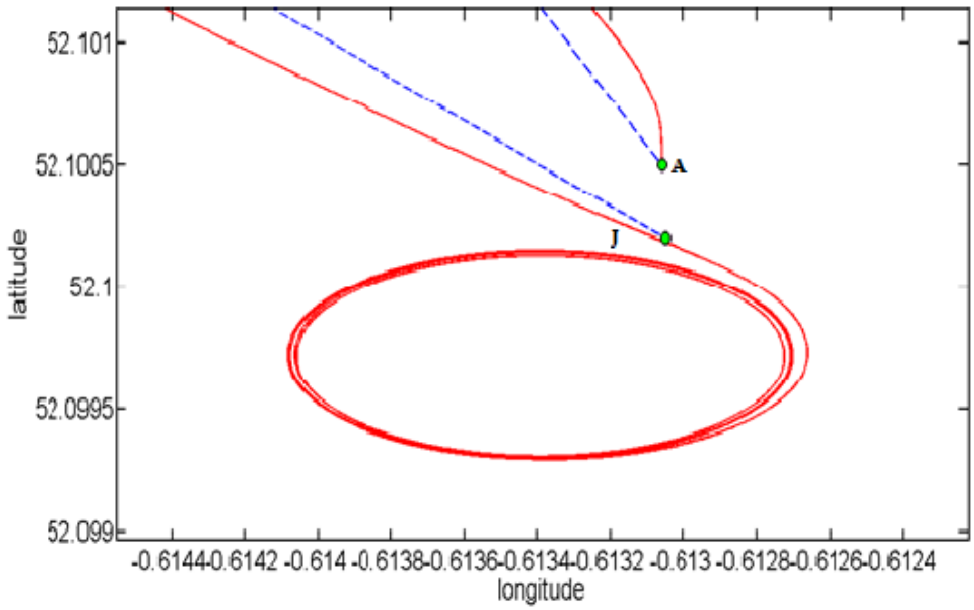


Figure 6-11: Loitering Mode

Figure 6-11 demonstrates the performance of the loitering mode of the UAV once it reaches the last waypoint, where it keeps turning until it gets the commands to land or go back again through the waypoints.

6.6 Chapter Summary

In conclusion, automatic tracking/following the pipeline structure using aerial platform equipped with vision sensor was presented in this chapter by implementing a vision-based 2D waypoints tracking algorithm. This algorithm was designed and developed to keep the aerial platform track/follow the pipeline structure automatically. The pipeline structure is represented by its endpoints, which are acquired online using the vision-based automatic pipeline detection algorithm which was presented earlier in chapter four.

The performance of the algorithm was evaluated in Matlab/Simulink simulation environment. The input waypoints that represent the pipeline structure were predefined offline due to some limitations. The results show a robust and adequate performance of the proposed algorithm for automatically tracking/following the pipeline structure.

Chapter 7

Performance and Evaluation of the Vision-Based Aerial Pipeline Surveillance System

7.1 Introduction

This chapter presents the results and evaluates the performance of the complete integrated automatic pipeline surveillance system. This representations and evaluations were implemented in each of the following, developed algorithms:

- Identification of the pipeline's segment endpoints;
- Detection of third-party interference;
- Following/tracking the pipeline structure.

As mentioned previously, in order to represent and evaluate those algorithms, indoor flight tests were performed to produce the required data due to the difficultly and lack of aerial ability (in the public domain) of real pipeline data. In addition to the limitations the data/video obtained, do not have IMU and depth information. The experimental data sets, produced in these tests, consists of synchronized depth (16-bit) and RGB (8-bit) images that have a resolution of 480X640 pixels and were captured at 30 fps using depth/optical sensor (ASUS Xtion) mounted on a Gaui 500X quadrotor platform. The aerial platform to fly

over the field test at a “scaled” average speed of 4 cm/s and at altitudes range of 100 cm to 120 cm. The experimental setup for those field test data sets includes a small-scale prototype of the pipeline structure (around 3 m length and 1.2 cm width), as well as expected small third-party interference objects.

The pipeline’s endpoints identification was computed and evaluated using statistical parameters such as sensitivity, specificity, false negative and false positive rates. While, the accuracy of the position of the identified endpoints, was represented and evaluated based on the ground-truth position and the behaviour of the position at each frame, relative to the camera frame.

Likewise, the performance of detecting the third-party interference was represented and assessed in terms of the rates of the detection and the processing speed. The detection rate was estimated by using sensitivity, specificity, false negative and false positive rates. The processing speed of the algorithm was estimated based on the average processing time of each frame.

Finally, the behaviour performance of following/tracking the pipeline structure was represented and evaluated based on the following list:

- 2D positions of the generated waypoints of the course, relative to the camera frame.
- 3D Positions of the air-vehicle, relative to the camera frame.
- 3D Orientation of the air-vehicle, relative to the camera frame
- Longitudinal distance of the air-vehicle, relative to the front-endpoint of the current pipeline segment.
- Lateral distance of the air-vehicle, relative to the pipeline segment.
- Lateral distance of the air-vehicle, relative to the pipeline as a line.
- Processing speed.

Comparison with ground-truth was made to validate the results. More detail and description is given in the following sub-sections.

7.2 Test-Rig

Different tests were performed to represent and evaluate the performance of each algorithm, which include the pipeline segment endpoints identification, third-party interference objects detection, and pipeline following algorithms. In those tests, each data set consists of depth (16-bit) and RGB (8-bit) images with a resolution of 480x640 pixels and were captured indoors at 30 fps using an ASUS Xtion sensor mounted on a Quadrotor. The Quadrotor flies over a small scale-pipeline at an average speed of 5 cm/s and altitudes of 100 cm, to 120 cm. The sensor angular Field-of-View (FOV) as specified by the manufacturer are 57° horizontally and 43° vertically. The proposed Right-of-Way distance (ROW) is 15 cm on each side, to match the real Right-of-Way distance of 30 meters. The sighting area comprises of a small-scale pipe prototype (around 3 m length and 1.2 cm width) as well as expected small third-party interference objects as described in Table 7-1, below.

Table 7-1: Flight tests description

	Frames No	Altitude (cm)	Pipeline	Third-party interference
Test 1	1450	100	absent	absent
Test 2	1480	120	absent	absent
Test 3	1410	100	present	absent
Test 4	1610	120	present	absent
Test 5	1420	100	present	present
Test 6	1450	120	present	present

These tests were performed at the presence and away from the pipeline, at different altitudes, to assess the effect of pixels resolution change and the performance of the pipeline endpoints identification. To evaluate the performance of the pipeline detection, the tests were performed at the presence of the pipeline at different altitudes. Moreover, to assess the performance of the third-party interference detection, the proposed tests are performed with the presence and absence of the third-party interference at the pipeline, presence and absence of the other objects, and at a variation of altitudes. The proposed

scenario of those indoor flight tests has a unique measurement and configuration for all of them as shown in Figure 7-1 with just playing with the pipeline, objects and third-party interference objects and changing the altitudes.

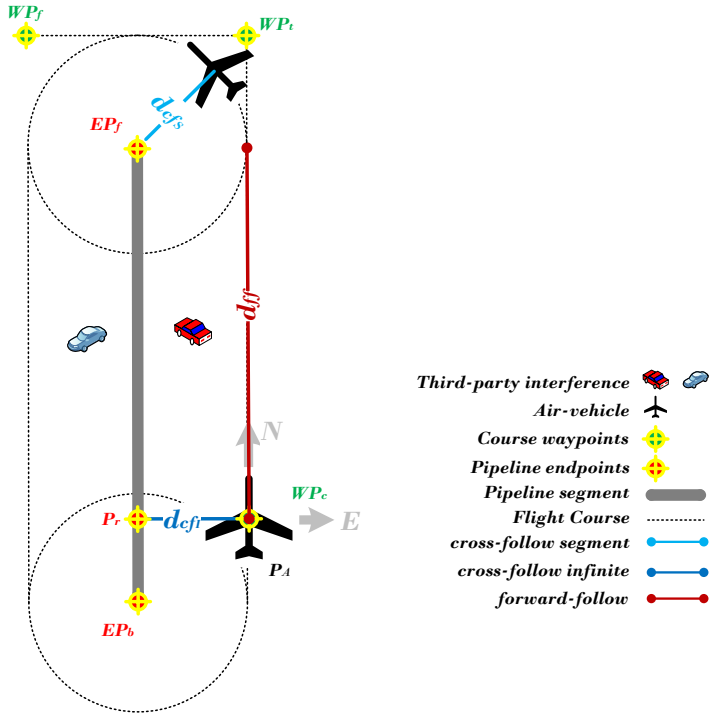


Figure 7-1: Indoor test-rig general configuration of the proposed flight-tests scenarios of the pipeline surveillance system

7.3 Endpoints Identification of the Pipeline

This section represents and evaluates the performance of the identification algorithm of the pipeline's endpoints while the air-vehicle follow/track the pipeline structure as shown in Figure 7-1. This algorithm was evaluated in terms of the capablity, accuracy and compatational load. The capablity was represented and evaluated based on the results of the following statistics factors, which are sensitivity, specificity, false positive and false negative ratios. While, the accuracy is based on the ground-truth data and the position behaviour of the pipeline's endpoints, relative to the camera frame. The computation was assessed by estimating the processing speed (frame per second). However, sensitivity and specificity relate to how likely the decisions of

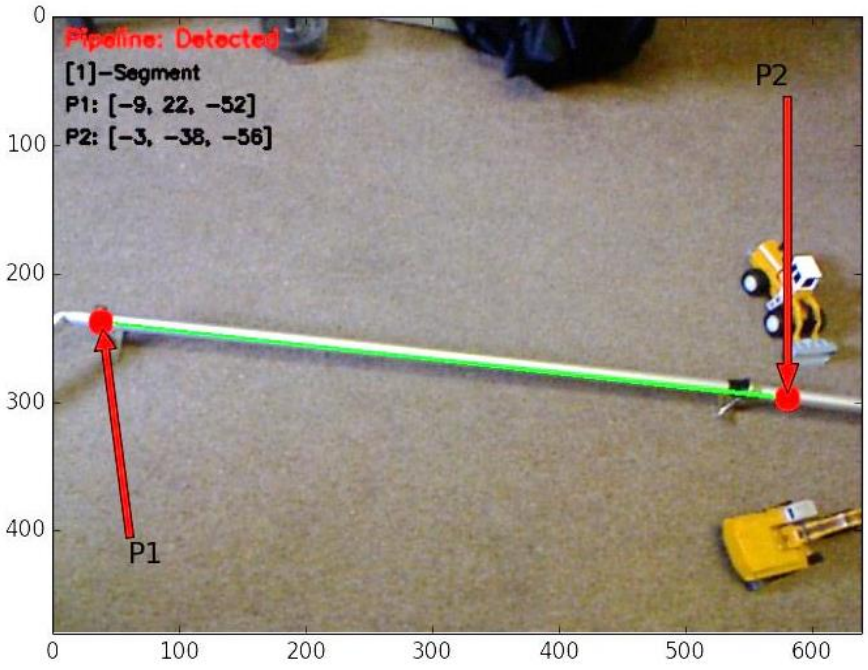
identifying the endpoints are correct when there is a pipeline or not, respectively. While the false positive and false negative estimate the errors of that decisions. So, to obtain a reliable and capable identification performance, the algorithm should have high sensitivity and specificity ratios, simultaneously, low false positive and false negative ratios. To obtain an accurate identification, the position of the identified endpoints was validated with the ground-truth data as well as the behaviour.

Four flight tests were prepared and performed to identify the endpoints of the pipeline structure and provide the required data sets to represent and evaluate the performance of this algorithm. The information of each flight test is described in Table 7-1.

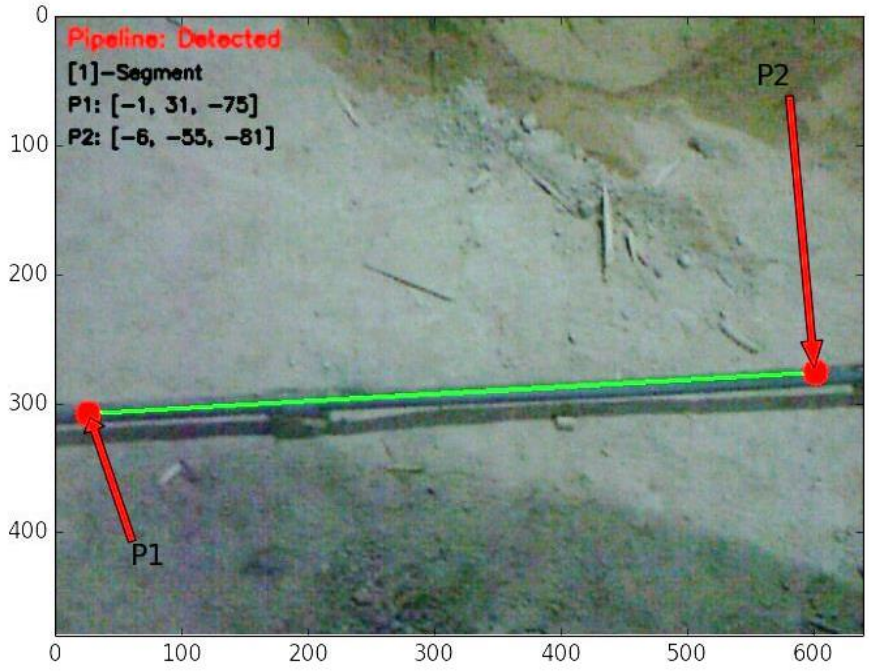
7.3.1 Capability

This section represents and evaluates how much the algorithm is capable of identifying the endpoints of the pipeline structure using sensitivity, specificity, false positive, and false negative ratios. However, the sensitivity ratio of the pipeline endpoints' identification is the percentage of the correctly identified endpoints of the pipeline at each frame to the total number of frames that involve a pipeline. This ratio is proposed to estimate the correct decision of the identification at the presence of the pipeline. Two flight tests were performed, to represent these ratios, which are test 1 and 3, described in Table 7-1. Figure 7-2 demonstrates the sensitivity performance of pipeline endpoints identification at different altitudes.

The performance results of this ratio are shown in Table 7-2. The sensitivities of identifying the endpoints, at 100 cm and 120 cm altitudes, are 90.57% and 90.81%, respectively, which means, at those roughly high rates, the algorithm is capable of correctly and efficiently identifying the endpoints.Hence, evaluating the developed algorithms at different altitudes when the pipeline is present in the frame, tests the algorithm ability and performance in dealing with variations in pixel resolution.



[a] Captured at 100 cm height and present of different objects



[b] Captured at 120 cm height and without objects

Figure 7-2: Demonstration of the pipeline endpoints identification capabilities at the presence of the pipeline structure (sensitivity rate), (a) captured at 100 cm altitude with present of objects (interruption) and (b) captured at different altitude 120 cm (resolution variety) without objects presented

The false negative ratio of the pipeline endpoints' identification is the percentage of the unidentified endpoints of the pipeline in the presence of the pipeline into the total number (absence and presence) of the unidentified endpoints of the pipeline. This ratio is to estimate the error of the endpoints' identification in the presence of the pipeline. To represent the performance of this ratio, flight tests 1 and 3, which are described in Table 7-1, were performed. Figure 7-3 demonstrates the false negative performance of pipeline endpoints identification at different altitudes.

The performance results of the false negative ratio are shown in Table 7-2. The estimation results of the false negative ratio of the third-party interference detection, at 100 cm, 120 cm height, are 8.69%, and 9.4%, respectively, which means, at those roughly low rates, the algorithm is capable of efficiently identifying the endpoints correctly in the presence of the pipeline with small errors in case of the variation in pixel resolution.

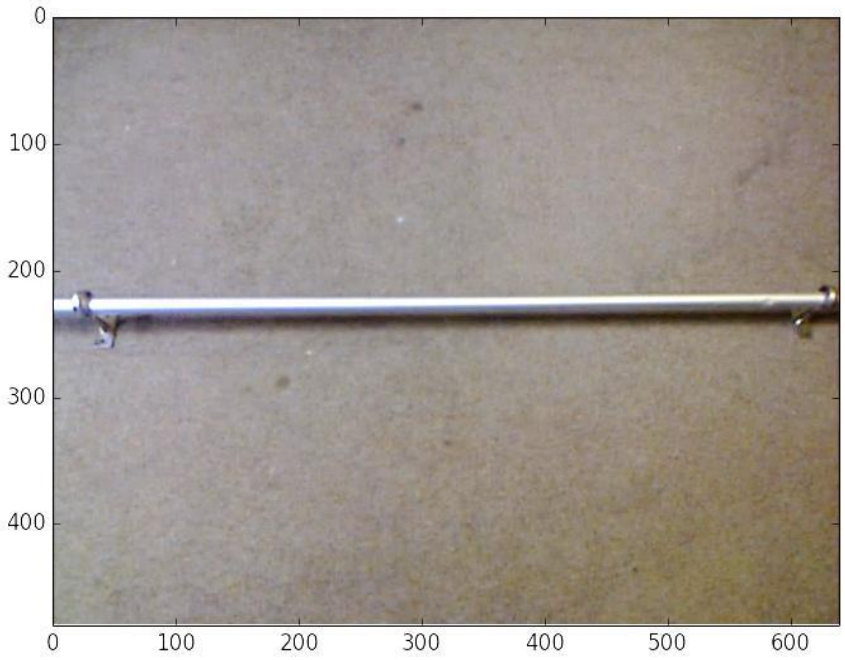


Figure 7-3: Demonstration of the pipeline endpoints identification error at the present of the pipeline structure (false negative rate) captured at 100 cm altitude

The specificity ratio of the pipeline endpoints identification is the percentage of the unidentified endpoints of the pipeline at the absence of the pipeline into the total number of the absent pipeline. This ratio is proposed to confirm the capability of the unidentified endpoints of the pipeline correctly. Two flight tests were performed to represent this ratio. Tests 2 and 4 described in Table 7-1. Figure 7-4 represents the specificity ratio behaviour at different altitudes.

The results of this ratio are shown in Table 7-2. The specificities of the unidentified endpoints, at 100 cm, 120 cm altitudes are 96.41%, and 96.42%, respectively, which means the correct decision of not identifying the endpoints is effective when there is a variation in the pixel resolution.

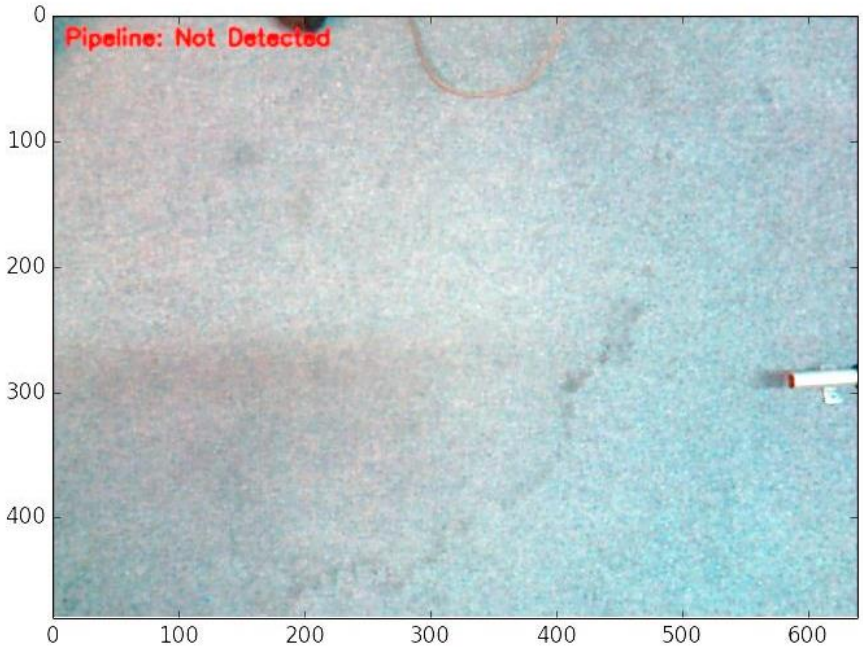


Figure 7-4: Demonstration of the pipeline endpoints identification capabilities at the absence of the pipeline structure (Specificity rate) captured at 100 cm altitude

The false positive ratio of the pipeline endpoints identification is the percentage of the identified endpoints at the absence of them into the total number (absence and presence) of the identified endpoints. This percentage is proposed to estimate the error of the identification in the absence of the

pipeline. Flight tests 2 and 4 are used to represent this ratio, which are described in Table 7-1. Figure 7-5 demonstrates the false positive performance of the endpoints' identification at different altitudes.

The performance results of this ratio are summarized in Table 7-2. The estimations of the false negative ratio of the pipeline endpoints identification, at 100 cm, 120 cm height are 3.91%, and 3.5%, respectively. That means, at those low rates, the algorithm is capable of efficiently identifying the third-party interference correctly in the absence of the pipeline and with the variations of pixel resolution.

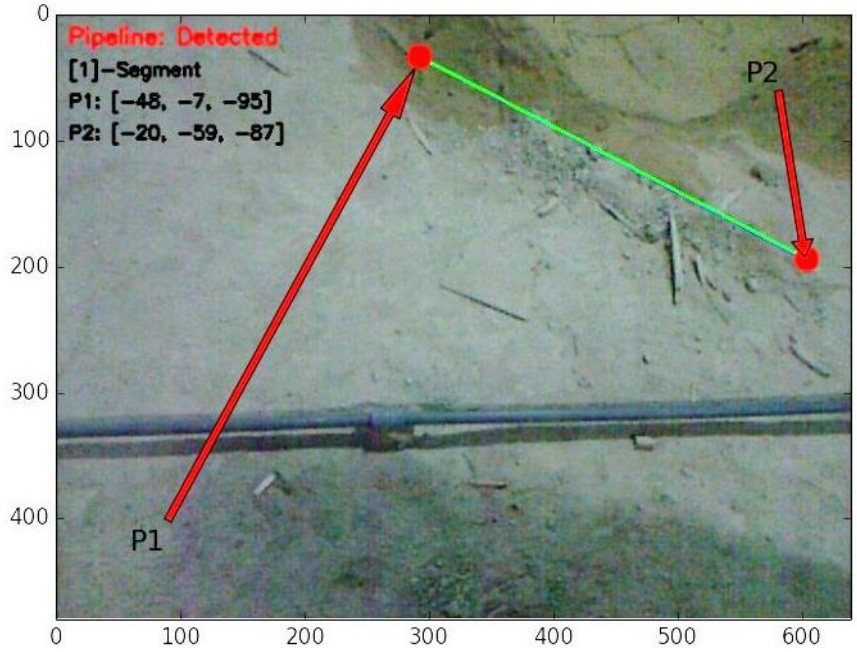


Figure 7-5: Demonstration of the pipeline endpoints identification error at the absent of the pipeline structure (false positive rate) captured at 120 cm altitude

As the results show in Table 7-2, the performance of the pipeline endpoints' identification algorithm confirms the capability of identifying the endpoints of the pipeline correctly with a high sensitivity rate of 90% and low error rate of 5%. Simultaneously, it is capable of discriminating the pipeline from the other objects at a high specificity rate of 96% and low error rate at 9%.

Table 7-2: Performance results of the pipeline endpoints identification algorithm

Pipeline Endpoints Identification	Identified	Unidentified	Identified	Unidentified
Presence	**Test 1**		**Test 3**	
	1277	133	1462	148
Absence	**Test 2**		**Test 4**	
	52	1398	53	1427
Sensitivity	90.57%		90.81%	
Specificity	96.41%		96.42%	
False Positive	3.91%		3.50%	
False Negative	8.69%		9.40%	

7.3.2 Accuracy

This section represents the position performance of the identified endpoints of the pipeline's segment, relative to the camera frame. This representation is undertaken to assess the quality of the endpoints identification and confirm its position accuracy. The data sets of test 3 and 4, described above in Table 7-1, were used to perform this assessment. Each endpoint of the pipeline segment is represented in terms of the north (X_{EP}), east (Y_{EP}), and height (h_{EP}) positions, which (X_{EP}) and (Y_{EP}) relative to the camera frame, while (h_{EP}) position relatives to the estimated ground plane. Figure 7-6 illustrates the estimated north positions behaviour of the backward (X_{EP_b}) and forward (X_{EP_f}) endpoints of the identified pipeline segment, relative to the origin of the camera frame throughout frames' sequence of each flight test that was proposed to confirm the quality performance of the identification. Additionally, the north ground-truth position of the backward ($X_{EP_{b_d}}$) and forward ($X_{EP_{f_d}}$) endpoints, shown in red and blue respectively, is also represented for the validation purpose of the algorithm. As seen, the grey and black dots represent the estimated north position of the identified backward (X_{EP_b}) and forward (X_{EP_f}) endpoints throughout each frame of the flight tests, which also represent the length of coverage of the pipeline segment.

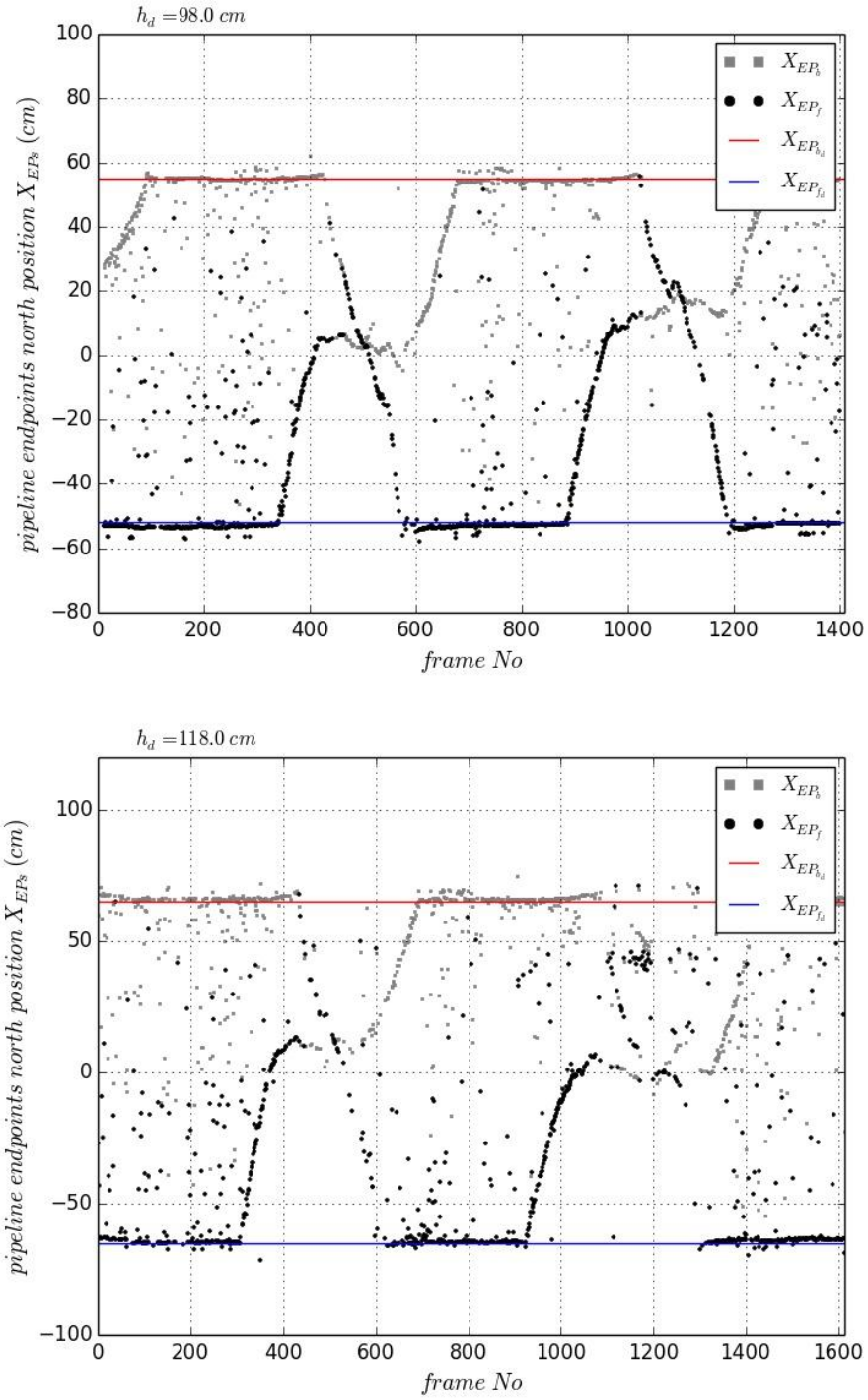


Figure 7-6: behaviour of the estimated backward (X_{EP_b}) and forward (X_{EP_f}) north position of the pipeline segment endpoints relative to the camera frame, and the desired north position of the backward in ($X_{EP_{b_d}}$) and forward in ($X_{EP_{f_d}}$) endpoints, while, [Top]: captured at 100 cm altitude, [Bottom]: captured at 120 cm altitude

While following the pipeline, the behaviour of the estimated North position of each identified endpoint should be constant throughout each frame. Several cases potentially affect the performance, that are:

1 Variation of air-vehicle position;
2 Variation of air-vehicle orientation;
3 Camera vibrations;
4 Proximity to the pipeline endpoint;
5 False positive identifications of the pipeline endpoints;
6 Partial detection of the pipeline segment.

However, as seen in Figure 7-6, the estimations of the north positions $[X_{EP_b}, X_{EP_f}]$ of the endpoints approximately keep the expected behaviour for different heights.

Figure 7-7 shows the estimated east position (Y_{EP}) behaviour of the identified forward (Y_{EP_f}) and backward (Y_{EP_b}) endpoints of the pipeline, relative to the origin of the camera frame throughout each frame sequence of the flight tests, which were proposed for the position evaluation. Additionally, for the validation purpose of the algorithm the East ground-truth positions of the backward ($Y_{EP_{b_d}}$) and forward ($Y_{EP_{f_d}}$) endpoints are shown in red and blue, respectively. As seen, the grey and black dots represent the estimated east position of the identified backward (Y_{EP_b}) and forward (Y_{EP_f}) endpoints at each frame. While following the pipeline, the east position (Y_{EP}) should always be constant, except in the following cases:

1 Variation of altitude;
2 Variation of roll angle;
3 Variation of heading angle;
4 The proximity of pipeline endpoint;
5 Partial detection of the pipeline segment.

However, as can be seen in Figure 7-7, under the conditions of flight tests 1 and 3, which are [-11.8 cm, -11.8 cm] and [-9.06 cm, -9.06 cm], respectively; the

east positions' estimation [Y_{EP_b}, Y_{EP_f}] of the endpoints keeps the same location, relative to the camera's origin.

Figure 7-8 shows the estimated height position behaviour of the identified forward (h_{EP_f}) and backward (h_{EP_b}) endpoints of the pipeline, relative to the origin of the camera frame throughout the frames' sequence of each flight tests, which were proposed for the position evaluation. Additionally, the ground-truth positions of the height of the backward ($h_{EP_{b_d}}$) and forward ($h_{EP_{f_d}}$) endpoints are shown as solid lines in grey and black, respectively, for the validation purpose of the algorithm. As seen, the grey and black dots represent the estimated height position of the identified backward (h_{EP_b}) and forward (h_{EP_f}) endpoints at each frame. While following the pipeline, the height position should always be constant, except in the following cases:

1) Variation of altitude;
2) Variation of pitch angle;
3) Variation of roll angle;
4) Ground extraction error.

However, as seen in Figure 7-8, the height positions estimation [h_{EP_b}, h_{EP_f}] of the endpoints keep the same location, relative to the ground, under conditions of flight tests 1 and 3 which are [3.48 cm, 2.02 cm] and [4.33 cm, 1.3 cm], respectively.

Table 7-3 summarizes the performance results of the 3D position of the identified endpoints of the pipeline. Based on that, the proposed algorithm is capable and valid to identify the endpoints of the pipeline with good performance.

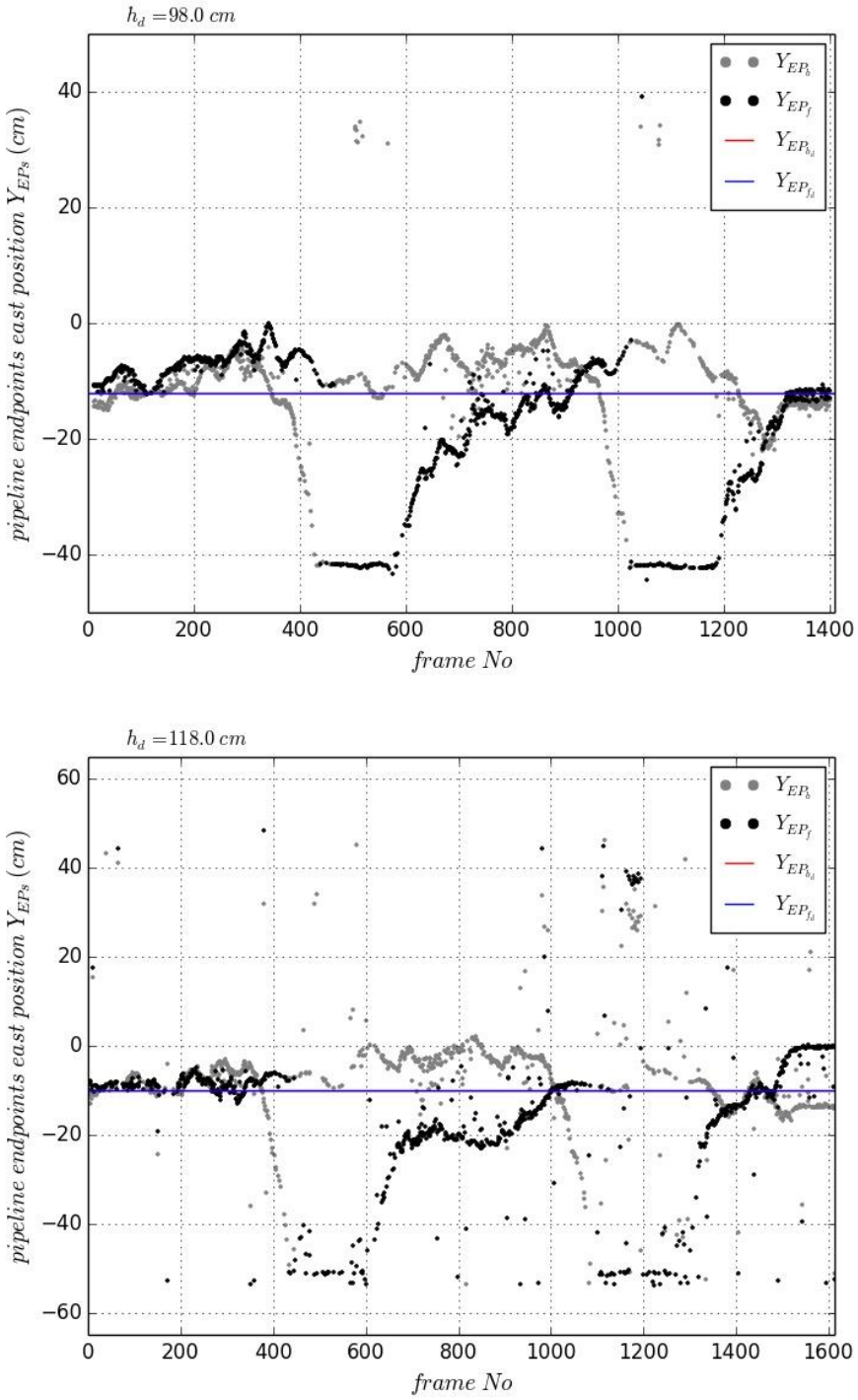


Figure 7-7: Behaviours of the estimated backward (Y_{EP_b}) and forward (Y_{EP_f}) east position of the pipeline segment endpoints relative to the camera frame, and the desired east position of the backward in ($Y_{EP_{b_d}}$) and forward in ($Y_{EP_{f_d}}$) endpoints, [Top]: captured at 100 cm altitude, [Bottom]: captured at 120 cm altitude

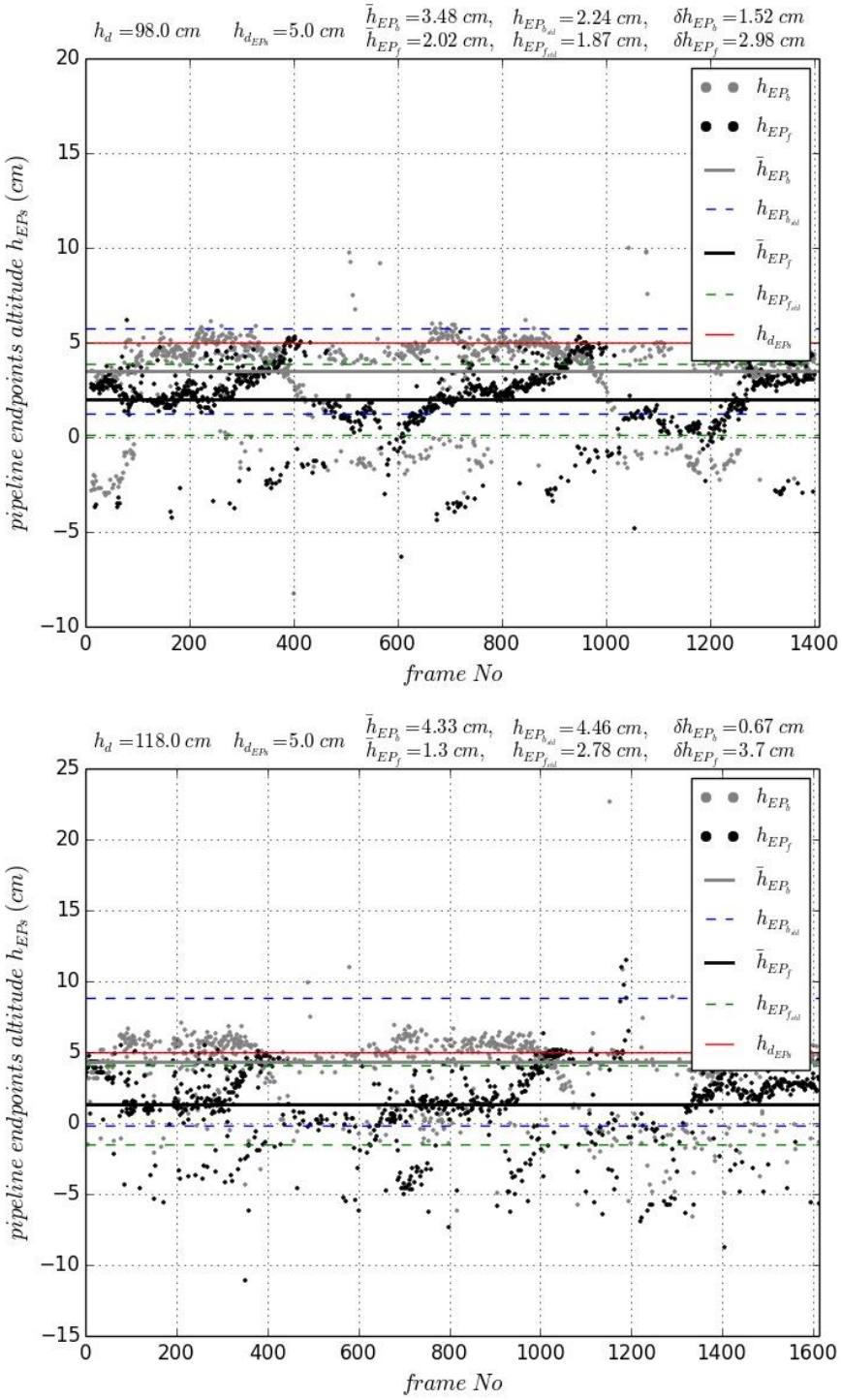


Figure 7-8: Behaviours of the estimated backward (h_{EP_b}) and forward (h_{EP_f}) height of the pipeline segment endpoints relative to the ground, their mean values ($\bar{h}_{EP_b}$) and ($\bar{h}_{EP_f}$), their standard deviations ($h_{EP_{b_{std}}}$) and ($h_{EP_{f_{std}}}$), and the ground-truth height of both of them (h_{EP_d}), [Top]: captured at 100 cm altitude, [Bottom]: captured at 120 cm altitude

Table 7-3: Results of the 3D position of the pipeline's endpoints and errors relative to the camera frame coordinates

Test	**Endpoint**	***Ground-truth***			***Mean-estimated***			***std***			***error***		
		X_{EP_d} (cm)	Y_{EP_d} (cm)	h_{EP_d} (cm)	$\bar{X}_{EP}$ (cm)	$\bar{Y}_{EP}$ (cm)	$\bar{h}_{EP}$ (cm)	$X_{EP_{std}}$ (cm)	$Y_{EP_{std}}$ (cm)	$h_{EP_{std}}$ (cm)	δX_{EP} (cm)	δY_{EP} (cm)	δh_{EP} (cm)
1	backward	55	-12	5	54	-11.8	3.48	3.21	4.8	2.24	1	-0.2	1.52
	forward	-51	-12	5	-52	-11.8	2.02	2.68	5.5	1.87	1	-0.2	2.98
3	backward	65	-10	5	66	-9.06	4.33	2.9	5.1	4.46	-1	-0.94	0.67
	forward	-65	-10	5	-65.5	-9.06	1.3	2.43	6.5	2.78	0.5	-0.94	3.7

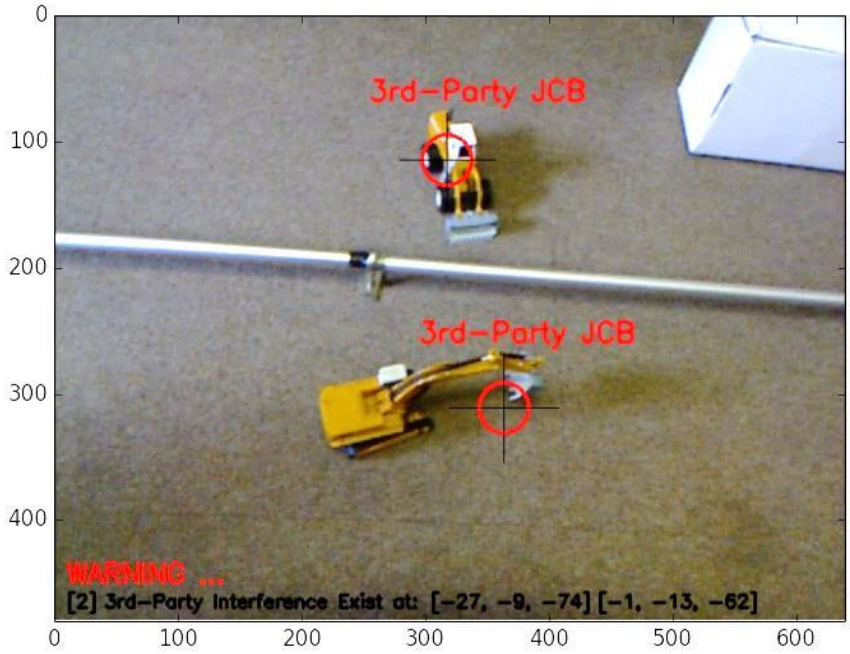
7.4 Third-Party Interference Detection and Classification

This section represents and evaluates the performance of the proposed algorithm used to detect the third-party interference in near real-time while the air-vehicle follow/track the pipeline structure as shown in Figure 7-1. This algorithm was represented and evaluated in terms of the capablity and compatational load. Four statistical performance indexes were considered to assess the capability of the detection and classification, namely: sensitivity, specificity, false positive and false negative ratios. Sensitivity and specificity relate to how likely the decision of the detection is correct; either the third-party objects are present or absent, respectively; while the false positive and false negative corresponds to the rate of the detection error. The algorithm should have high sensitivity and specificity ratios, while keeping low false positive and false negative ratios in order to have a good detection performance. However, the estimation of the processing speed (frame per second) was used to represent and evaluate the computational load of the algorithm. Four flight tests were carried-out to detect and classify the third-party interference and provide the required data sets to represent and evaluate the performance of the algorithm. These flight tests involve test 3, 4, 5, and 6 that are described in Table 7-1.

7.4.1 Capability

This section represents and evaluates how much this algorithm is capable to detect and classify the third-party interference using sensitivity, specificity, false positive, and false negative ratios. The sensitivity ratio of the third-party interference objects detection is the proportion of the correctly detected third-party interference when present to the total number (detected and undetected) third-party intervention. This proportion is used to estimate the correct decision of the detection in the presence of the third-party interference objects. Two flight tests were made (test 5 and 6) to evaluate this, as described in Table 7-1. Figure 7-9 demonstrates the performance of the sensitivity rate of the third-party interference detection at different altitudes. The performance results of the sensitivity ratio are summarized in Table 7-4. The estimated sensitivities rates of detecting third-party interference, at 100 cm and 120 cm altitudes are 88.5% and 87.5%, respectively; which means, at those high rates, the algorithm is capable of efficiently detecting third-party interference correctly. The tests also validate the performance of the developed algorithms in the case where there are variations in the pixel resolution.

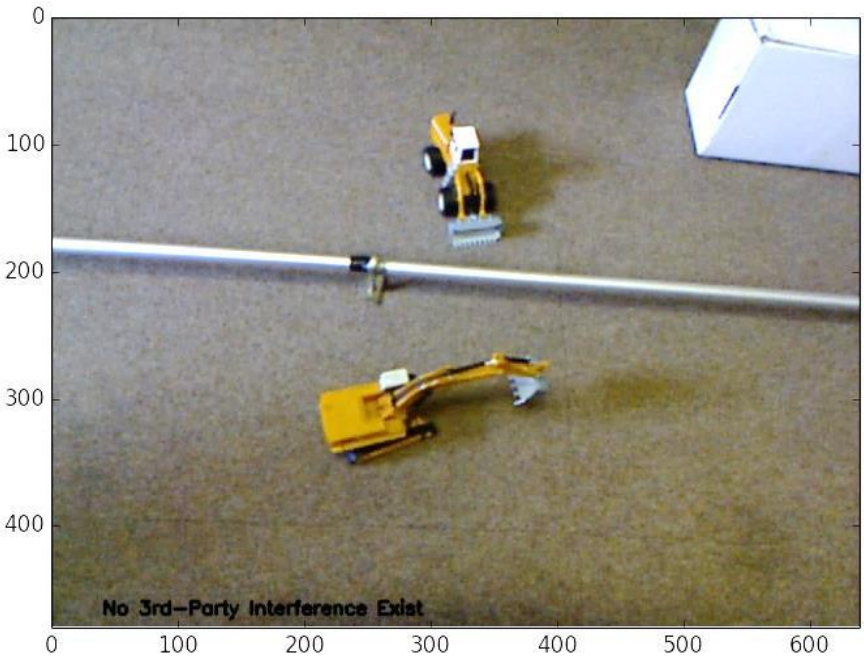
The false negative ratio of the third-party interference detection is the percentage of not detecting the third-party when present out of the total number (absence and presence) of the undetected third-party intervention. This ratio is to estimate the error rate of the detection when third-party interference are present. Flight tests 5 and 6, described in Table 7-1, are used to represent this ratio. Figure 7-10 illustrates the performance of the false negative rate of third-party interference detection for different altitudes. The performance results of the false negative ratio are shown in Table 7-4. The estimations results of the false negative ratio of the third-party interference detection, at 100 cm and 120 cm height, are 11.44% and 12.38%, respectively; which means, at those low rates, the algorithm is again capable of efficiently detecting the third-party interference correctly when present, with small error in case of the variation of pixel resolution.



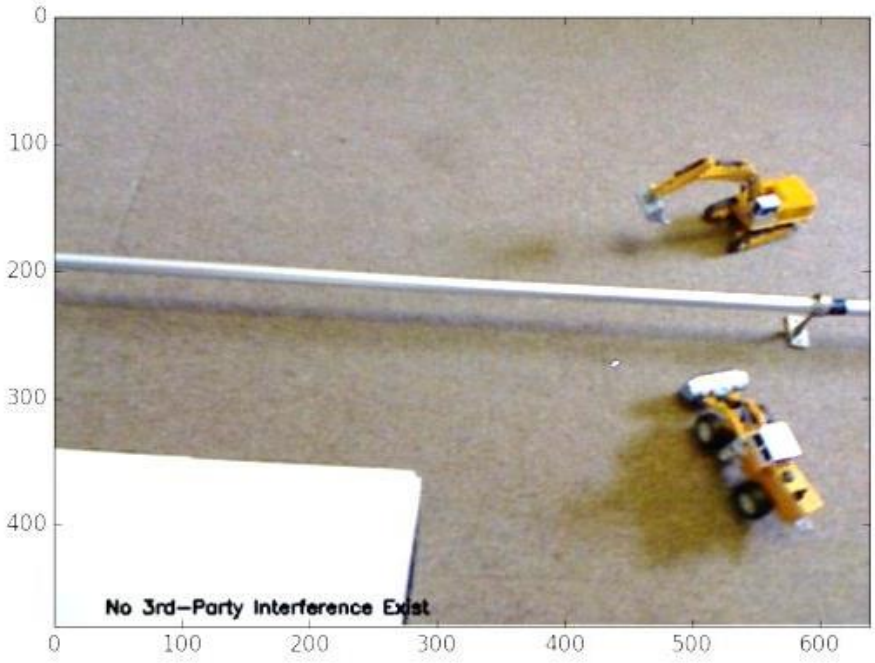
[a] Captured at 100 cm height

[b] Captured at 120 cm height

Figure 7-9: Demonstration of third-party interference detection capabilities at the presence of them (sensitivity rate), (a) captured at 100 cm altitude and at present of other objects (interruption effects) and (b) captured at different altitude 120 cm (resolution effects)



[a] Captured at 100 cm height



[b] Captured at 120 cm height

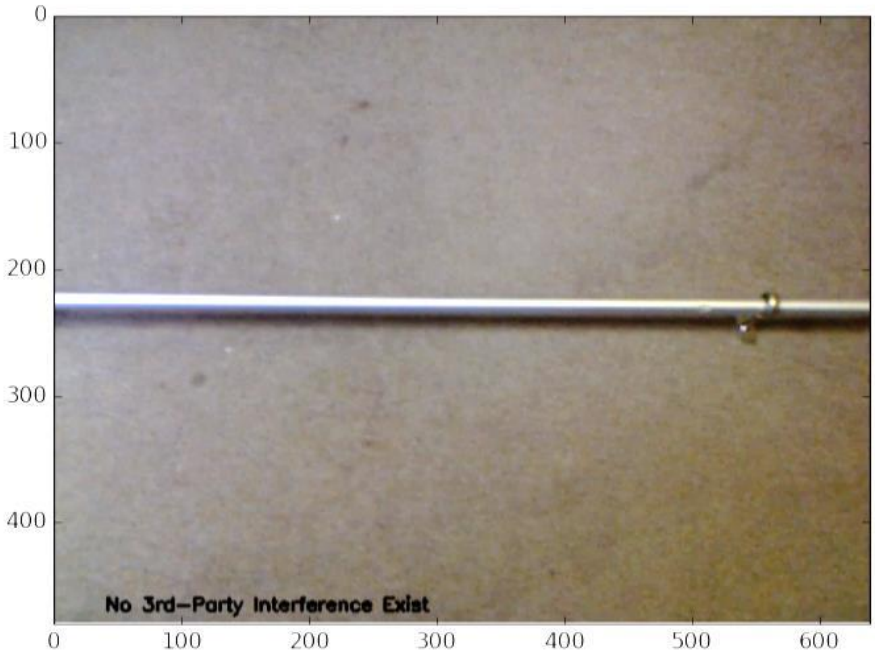
Figure 7-10: Demonstration of third-party interference detection error at the presence of them (false negative rate), (a) captured at 100 cm altitude and at presence of other objects (interruption effects) and (b) captured at different altitude 120 cm (resolution effects)

The specificity ratio of the third-party interference detection is the percentage of not detecting third-party interference at the absence of it into the total number (detected and undetected) of the absent third-party interference. This percentage is used to estimate the correct decision of the detection in the absence of the third-party interference objects.

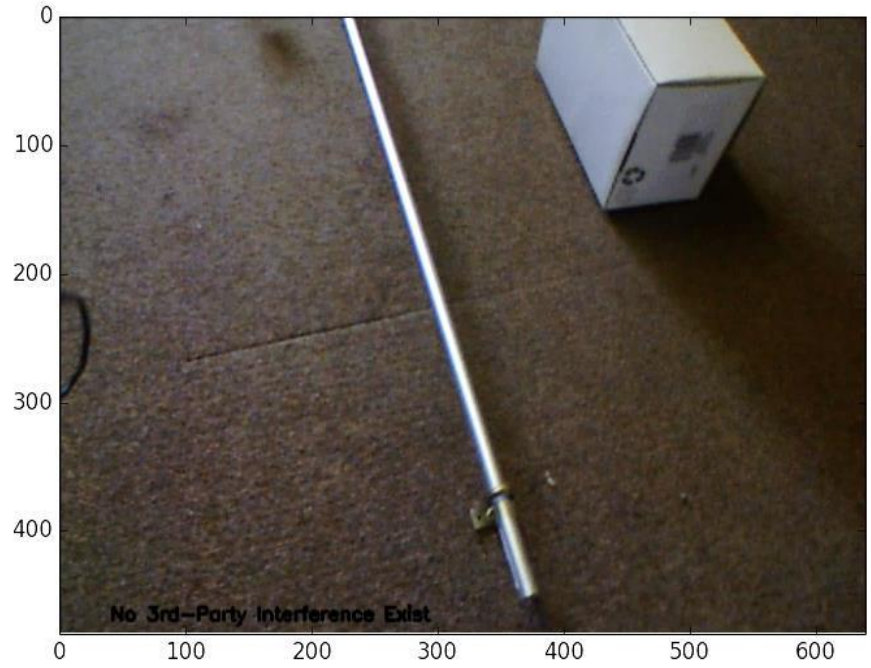
Two flight tests, that were performed to assess this proportion. Tests 1 and 3, are tabulated in Table 7-1. Figure 7-11 illustrates the performance of the specificity rate of the third-party interference detection at different altitudes. The performance results of the specificity ratio are shown in Table 7-4. The specificities rates of detecting the third-party interference, at 100 cm and 120 cm altitudes are 89% and 88.5%, respectively; which means, at those high rates, the algorithm is capable of efficiently detecting the third-party interference correctly in the absence of the third-party interference and in the case of variation in pixel resolution.

The false positive ratio of the third-party interference detection is the percentage of detecting the third-party interference at the absence of it into the total number (absence and presence) of the detected third-party intervention. This percentage is proposed to estimate the error of the detection in the absence of third-party interference. To represent this ratio, flight tests 1 and 3, which are tabulated in Table 7-1, are used.

Figure 7-12 shows the performance of the false positive rate of third-party interference detection at different altitudes. The performance results of the false positive ratio are summarized in Table 7-4. The estimations of the false negative ratio of the third-party interference detection, at 100 cm and 120 cm height are 11.06% and 11.62%, respectively; which means, at those low rates, the algorithm is capable of efficiently detecting the third-party interference correctly in the absence of the third-party interference and in case of variations in pixel resolution.



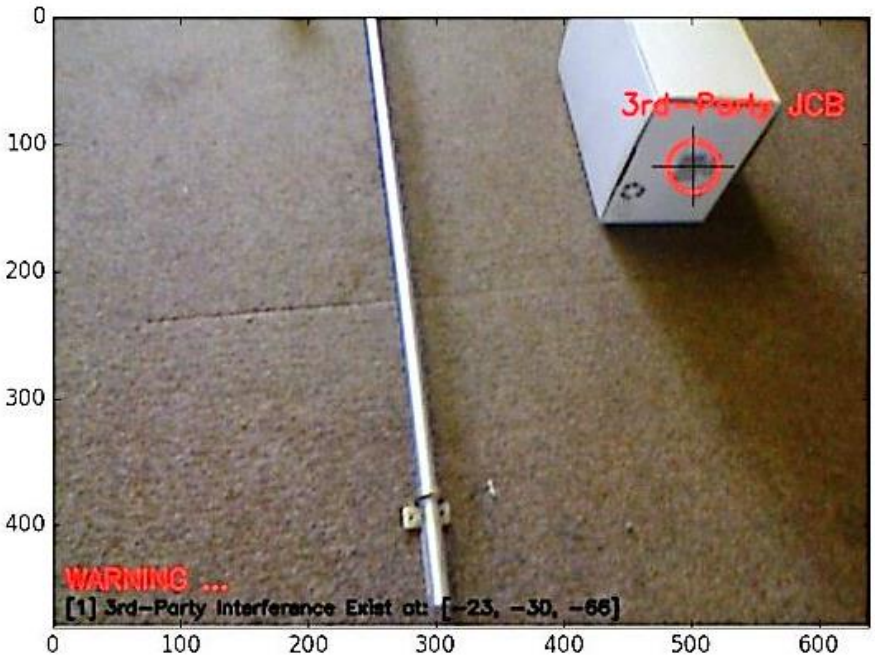
[a] Captured at 100 cm height



[b] Captured at 120 cm height

Figure 7-11: Demonstration of third-party interference detection capabilities at the absence of them (Specificity rate), (a) captured at 100 cm altitude and at presence of other objects (interruption effects) and (b) captured at different altitude 120 cm (resolution effects)

[a] Captured at 100 cm height



[b] Captured at 120 cm height

Figure 7-12: Demonstration of third-party interference detection error at the absence of them (false positive rate), (a) captured at 100 cm altitude and at presence of other objects (interruption effects) and (b) captured at different altitude 120 cm (resolution effects)

According to Table 7-4, the performance of the third-party interference detection algorithm shows that it has the capability to correctly detect the third-party interference at a high rate of 88% and also reject correctly at 89%. Simultaneously, it has an acceptable low detection error rate of 11% of detecting other objects and a low rate of 12% of missing the third-party interference. So, the overall performance and accuracy of the algorithm are sufficient in detecting the third-party interference objects, providing reliable performance. As can be seen in Table 7-4, the average processing speed to detect the third-party interference at each frame employing the on-board processor that was described in (section 3.4.3) is around 2 frames/second, which is sufficient to run the detection algorithm of the third party interference in near real-time.

Table 7-4: Performance results of the third-party interference detection algorithm

Third-party interference detection	**detected**	**Undetected**	**detected**	**Undetected**
Presence	**Test 5**		**Test 6**	
	177	23	175	25
Absence	**Test 3**		**Test 4**	
	22	178	23	177
Sensitivity	88.50%		87.50%	
Specificity	89.00%		88.50%	
False Positive	11.06%		11.62%	
False Negative	11.44%		12.38%	
Processing rate (fps)	2		2	

7.5 Pipeline Following/Tracking

This section presents and evaluates the performance of the pipeline following/tracking algorithm in real-time, as described in chapter 6. It is carried-out to confirm the capability and accuracy of generating the course waypoints (WP_s) online based on the identified endpoints (EP_s) of the pipeline. Validating the capability and accuracy of the waypoints navigation system, that was

developed to follow/track the pipeline structure by the air-vehicle, based on the generated waypoints (WP_s) of the air-vehicle's course, and finally proof the capability of processing this algorithm on-board. The performance of each was represented and evaluated offline based on the data logged in real-time, during the flight tests (1 and 3), that were described in Table 7-1, in the previous sub-section.

7.5.1 Course Waypoint Generation

This part represents and evaluates the performance of the real-time generation of the air-vehicle's course waypoints (WP_s), as shown in Figure 7-13, to confirm the accuracy of generating the air-vehicle's course waypoints (WP_s) online once the pipeline has been detected while following the pipeline. The current (WP_c) and target (WP_t) course waypoints are planned to construct the current-course segment of the air-vehicle based on the pipeline segment and the air-vehicle position, to keep the air-vehicle following the pipeline based on the desired cross-follow distance (d_{cf_d}). While the target (WP_t) and future (WP_f) course waypoints are proposed to make the cross-course segment, used to constrain the air-vehicle to acquire a turn around the forward-endpoint (EP_f) of the pipeline segment once the pipeline's relative point (P_r) starts passing the forward-endpoint (EP_f) of the pipeline segment or in case the air-vehicle is far away from the initial current waypoint (WP_c) by more than the required cross-follow distance (d_{cf_d}). However, the initial target waypoint (WP_t) will then be replaced by future waypoints (WP_f). The purpose of this evaluation is to represent the behaviour and quality assessments of generating the air-vehicle's course waypoints (WP_s) in real-time, based on the detected pipeline segment endpoints (EP_s), and the desired cross-follow distance. The pipeline segment's endpoints (EP_s) are visually detected online at 2 fps. The desired cross-follow distance is proposed to be 20 cm to preserve the pipeline integrity and avoid collisions with another monitoring system by taking into account, the sight coverage of the pipeline length and Right-of-Way and the air-vehicle turn radius.

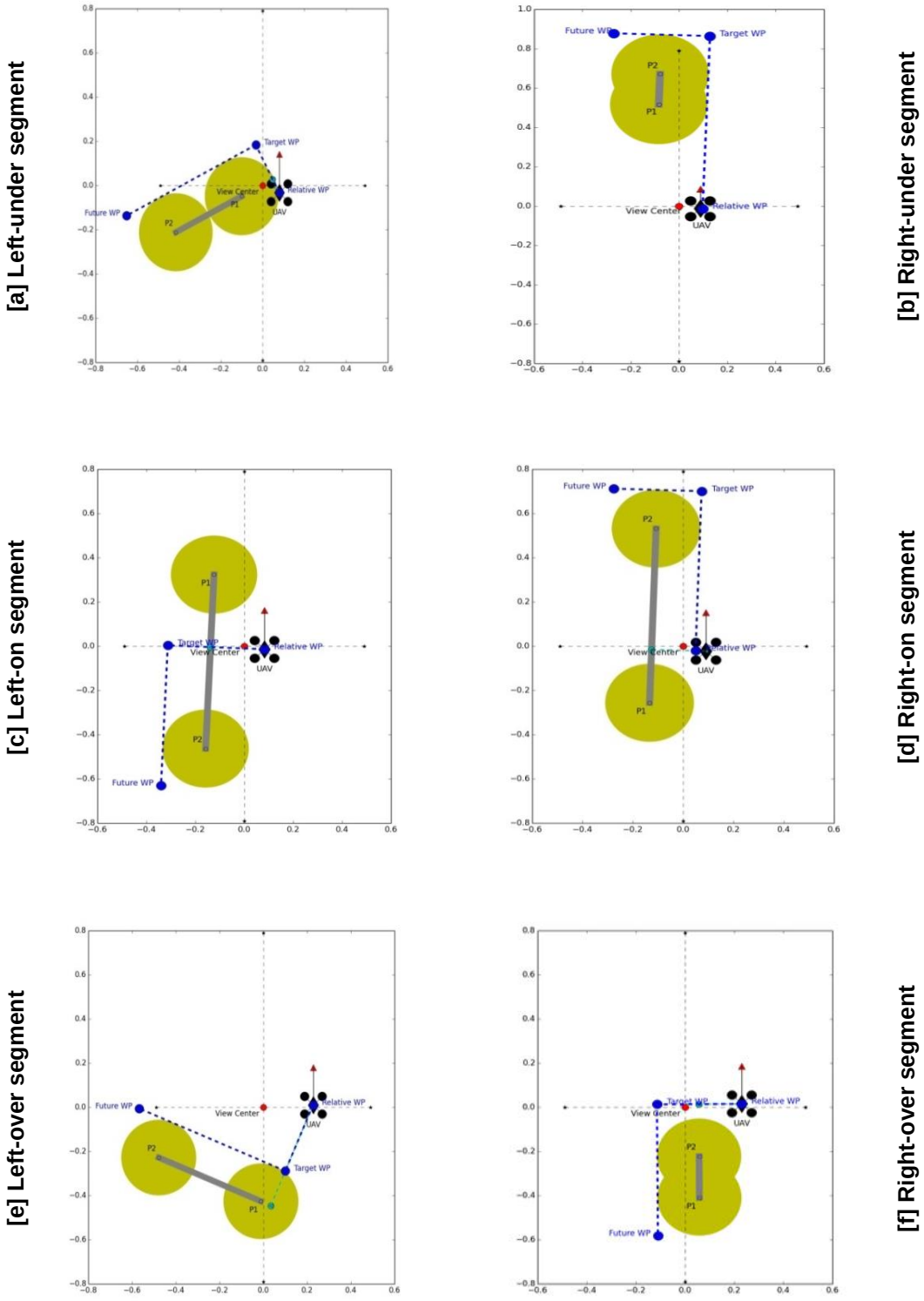


Figure 7-13: Cases results of the real-time configuration of the air-vehicle course waypoints (WP_s) based on the visual information of the pipeline segment endpoints (EP_s)

The position performance of the generated air-vehicle's course waypoints is represented in this chapter. Figure 7-14 and Figure 7-15, respectively, represent the 2D position generation in real-time of the course waypoints with respect to the camera frame in the frames sequence, where the air-vehicle flies at 100 and 120 cm altitude. As shown, the current, target, and future course waypoints ($WP_c, WP_t, \ and \ WP_f$) are represented in black, brown, and blue ,respectively. In addition, to the pipeline's relative point ($P_{relative}$) that is denoted in grey.

As shown in Figure 7-14, when the pipeline's relative point ($Y_{P_{relative}}$) is located within the pipeline segment, the east position of the current waypoint ($Y_{WP_{current}}$) assumed to be 20 cm away, to keep the disered cross-follow distance (d_{cf_S}), from that point. The curve should be constant to keep aligned the camera frame with the vector of the pipeline structure. However, once the pipeline's relative point ($Y_{P_{relative}}$) starts leaving the endpoint of the pipeline segment, this will lead the air-vehicle to turn around that endpoint and the current waypoint ($Y_{WP_{current}}$) will begin to converge to catch up the the pipeline's relative point ($Y_{P_{relative}}$) and reduces the cross-follow distance (d_{cf_S}) to zero once crossing the segment at the frames 500 and 1080 at 100 cm altitude as well as at the frames 500 and 1200 at 120 cm altitude, then return again to keep the desired 20 cm distance.

Simmilarly, the east position of the target waypoint ($Y_{WP_{target}}$) is assumed to have the same behavior of the current waypoint ($Y_{WP_{current}}$) when the pipeline's relative point ($Y_{P_{relative}}$) is locating within the pipeline segment. On the other hand, when the pipeline's relative point ($Y_{P_{relative}}$) start passing the endpoint of the pipeline segment (over segment), the air-vehicle will begin the turning phase and the vector of the pipeline segment will change with respect to the vector of the air-vehicle (camera frame) which leads to change the configuration of the waypoints. So in this case, the east position of the target waypoint ($Y_{WP_{target}}$) will change to have the same behaviour as the future waypoint ($Y_{WP_{future}}$) where it is far away along the length of the covered pipeline segment as seen at the

frame periods (400-600) and (1000-1300) at 100 cm altiude and at (450-650) and (1150-1400) at the 120 cm altitude.

Finally, the east position of the future course waypoint ($Y_{WP_{future}}$) should be opposite to the target waypoint ($Y_{WP_{target}}$), relative to the pipeline segment by 20 cm distance once the pipeline's relative point ($Y_{P_{relative}}$) is located within the pipeline segment. While the pipeline's relative point ($Y_{P_{relative}}$) start passing the endpoint of the pipeline segment (over segment), the future course waypoint ($Y_{WP_{future}}$) will change to reach the maximum long coverage of the pipeline segment that are 60 cm at 100 cm altitude and 75 cm at 120 cm altitude, then will reduce gradually to be 20 cm.

However, a small deviations occurred along the curves of the generated course waypoints (WP_s) at some of the frames that are due to the vector variation of the air-vehicle (camera frame) relative to pipeline segment, sensor calibration, vibrations of the sensor while flying, acceleration of the flight, and measurments errors.

Based on the comparison between the results of 100 cm and 120 cm altitudes, the north (X_{WP}) and east (Y_{WP}) position behaviours of the generated course waypoints were not affected when the pixels resolution are varied as shown at the top and bottom graphs in Figure 7-14.

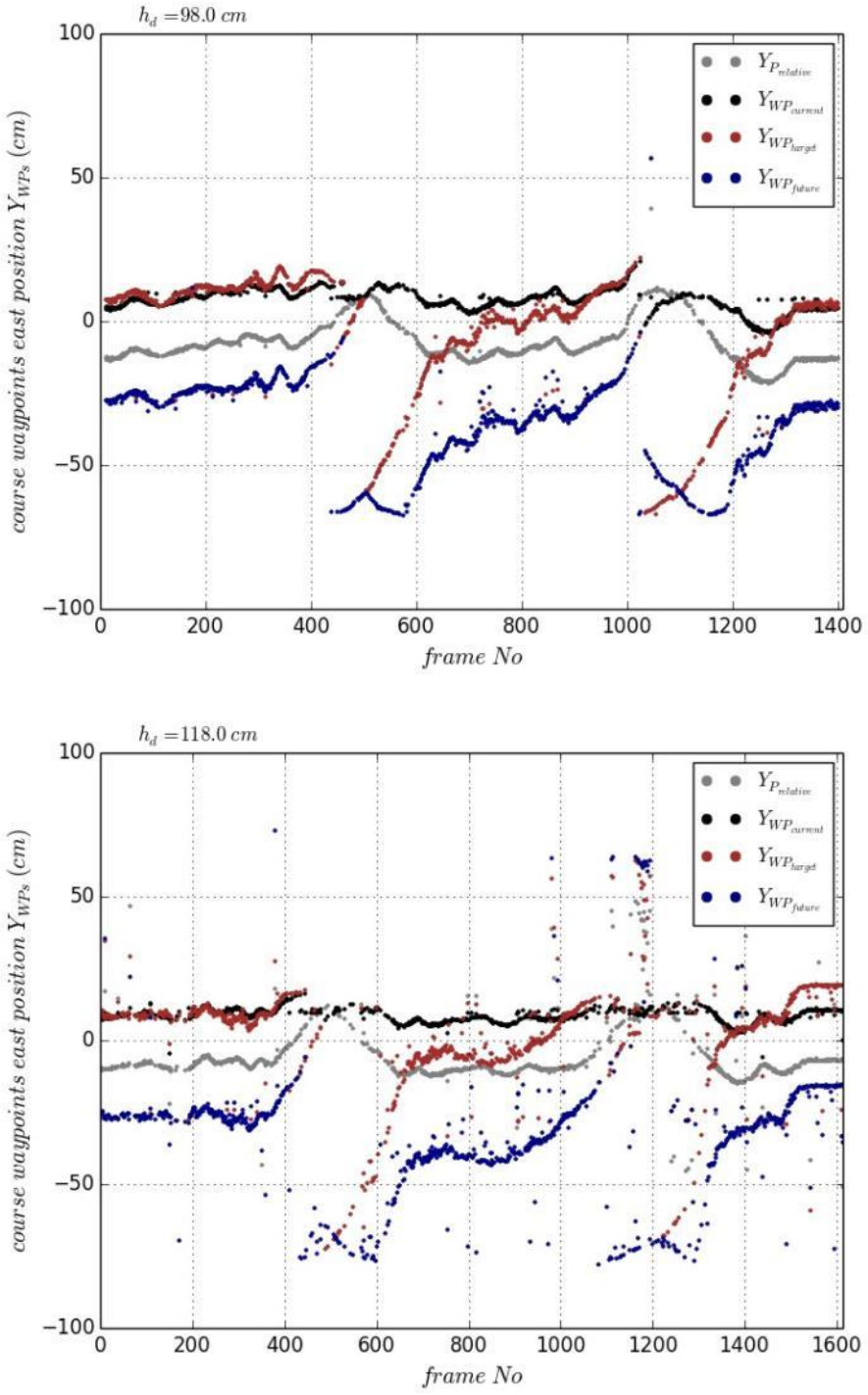


Figure 7-14: Behaviour representation of the east position of the generated course waypoints relative to the camera frame includes pipeline's relative point ($Y_{P_{relative}}$), current waypoint ($Y_{WP_{current}}$), target waypoint ($Y_{WP_{target}}$), and future waypoint ($Y_{WP_{future}}$), while, [Top]: captured at 100 cm altitude, [Bottom]: captured at 120 cm altitude

As shown Figure 7-15, when the north position of the pipeline's relative point ($X_{P_{relative}}$) is locating within the pipeline segment, the north position of the current waypoint ($X_{WP_{current}}$) aligns with the north position of the pipeline's relative point ($X_{P_{relative}}$) around zero. However, once the north position of the pipeline's relative point ($X_{P_{relative}}$) starts leaving the endpoint of the pipeline segment, the north position of the current waypoint ($X_{WP_{current}}$) will change, due to the vectors change between the reference (camera frame) and the pipeline segment, then return again to align with the north position of the pipeline's relative point ($X_{P_{relative}}$), as shown in Figure 7-15, around the frames 500 and 1080 at 100 cm altitude and 500 and 1200 at 120 cm altitude.The north position of the target waypoint ($X_{WP_{target}}$) should be located at the maximum north coverage of the pipeline segment that are 60 cm at 100 cm altitude and 75 cm at 120 cm altitude when the north position of the pipeline's relative point ($X_{P_{relative}}$) is located within the pipeline segment and keep constant while following the length of pipeline, as shown in Figure 7-15, at frame periods (0-300), (600-900), and (1200-1400) at 100 cm altitude; and (0-350), (650-950), and (1350-1600) at 120 cm altitude. On the other hand, when the north position of the pipeline's relative point ($X_{P_{relative}}$) is just immersing the maximum north coverage of the pipeline segment, the north position of the target waypoint ($X_{WP_{target}}$) will converge to the north position of the pipeline's relative point ($X_{P_{relative}}$) because the air-vehicle starts reaching the end of the pipeline structure. Once the air-vehicle moves closer to the forward-endpoint (EP_f) of the pipeline segment, the position is supposed to smoothly reduce down until the half of the turn phase is completed, then back again into the full coverage position, as shown in Figure 7-15, throughout the frame periods (300-600), and (900-1200) at 100 cm altitude and (300-600), and (1000-1300) at 120 cm altitude. Similarly, the north position of the future waypoint is following the same behaviour of the north position of the target waypoint, i.e. once the north position of the pipeline's relative point is locating within the pipeline segment or if it starts leaving the forward-endpoint (EP_f) of the pipeline segment.

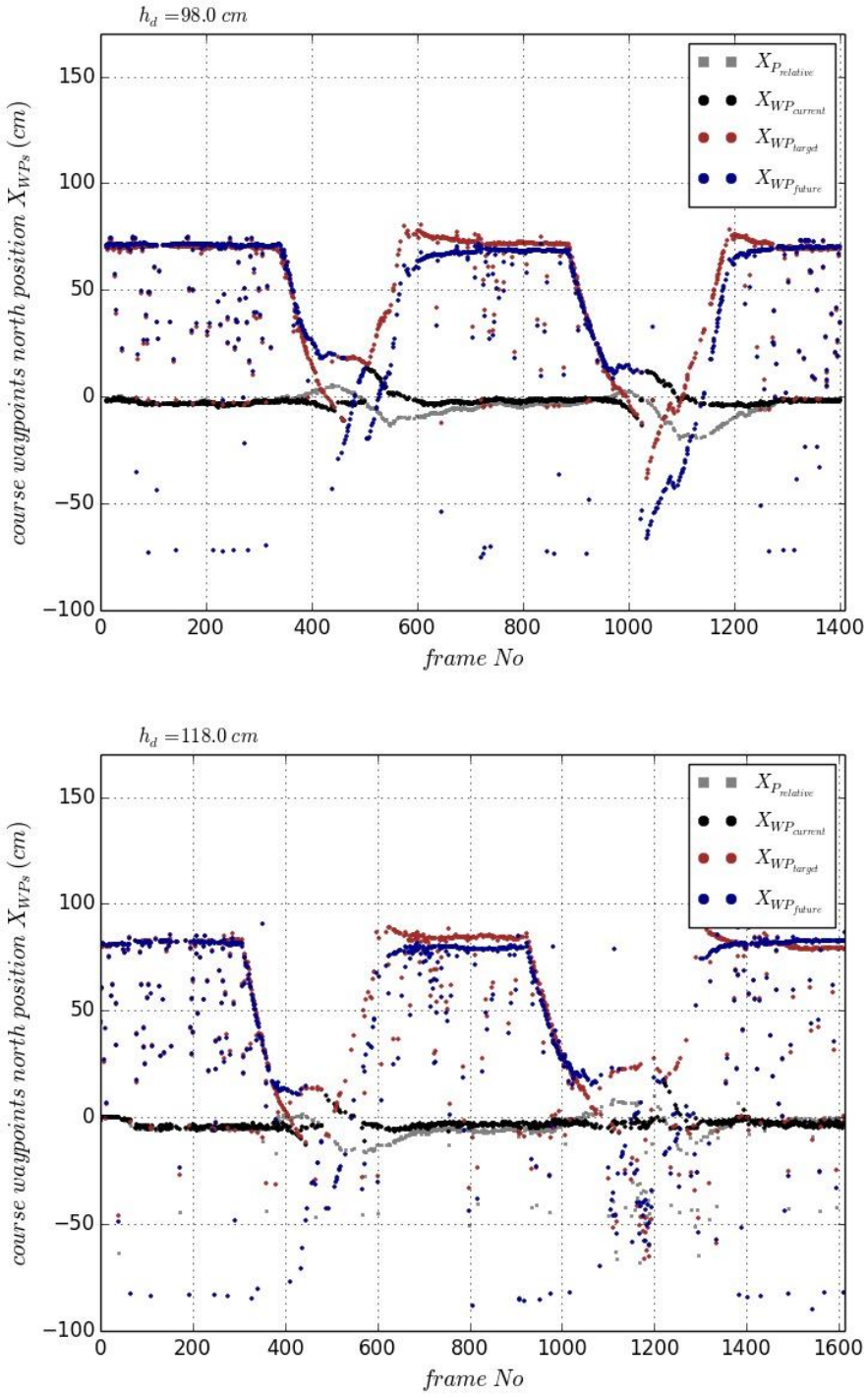


Figure 7-15: Behaviour representation of the north position of the generated course waypoints relative to the camera frame includes pipeline's relative point ($X_{P_{relative}}$), current waypoint ($X_{WP_{current}}$), target waypoint ($X_{WP_{target}}$), and future waypoint ($X_{WP_{future}}$), while, [Top]: captured at 100 cm altitude, [Bottom]: captured at 120 cm altitude

7.5.2 Waypoints Navigation

This section represents the performance of the waypoints' navigation system and evaluates its accuracy and capabilities. Hence, the following items are used to evaluate this performance, which are:

- Forward-follow distance (d_{ff}), between the pipeline's relative point and the forward endpoint of the pipeline segment. Note: the desired forward-follow distance is considered just throughout the cruise phase; whereas it neglects the comparison once the air-vehicle reaches the endpoint and turns around it.
- Cross-follow distance (d_{cf}) is the shortest distance between the air-vehicle and the pipeline's relative point.
- The 3D position of air-vehicle (x, y, h) relative to the camera reference frame.
- The 3D displacements of the air-vehicle position ($\Delta x, \Delta y, \Delta z$).
- Air-vehicle's orientations angles (θ, Φ, Ψ) relative to the camera frame.

7.5.2.1 Follow Distance

This section represents the performance of following/tracking the pipeline structure using 2D following distance (d_f) and verifies how the air-vehicle could accurately follow the generated course waypoints (WP_s) in real-time and keep track of the pipeline structure. Since the pipeline following is based on the local camera frame, the evaluation criteria, used to assess the performance, is observed through any unexpected behaviour's change to the estimated forward-follow (d_{ff}) and cross-follow (d_{cf}) distances, as shown in Figure 7-16, and compares them with their desired values, as shown in Figure 7-17. Based on the proposed tests scenario, there should be a straight-line phase to keep tracking the length of the pipeline and a turn-phase to return around the endpoints of the pipeline structure. The behaviour of the forward-follow (d_{ff}) and cross-follow (d_{cf}) distances is illustrated in Figure 7-16.

However, when the air-vehicle is performing the straight-line tracking, the distances of the forward-follow (d_{ff}) and cross-follow (d_{cf}) are approximately constant and steady at 50 cm and 20 cm, respectively, throughout the frame periods (0-350), (600-900), and (1200-1400) at 100 cm altitude. While, at 120 cm altitude, they are constant and steady at 65 cm and 20 cm, respectively, throughout the frame periods (0-300), (600-900), and (1300-1600).

In the proposed test scenario, there are two turn-phases, required to track the pipeline structure around each endpoint of the pipeline segment as shown at the rest of the frame periods. However, both the forward-follow (d_{ff}) and cross-follow (d_{cf}) distances are reduced smoothly once the air-vehicle starts turning around the endpoint, then increases again once the air-vehicle starts crossing the pipeline segment. As can be seen, the forward-follow distance (d_{ff}) started decreasing before the cross-follow distance (d_{cf}) from 50 cm and 65 cm of 100 cm and 120 cm altitudes, respectively, due to the proximity distance between the air-vehicle and the endpoint of the pipeline segment. While the cross-follow distance (d_{cf}) still keeps its straight-line phase distance at 20 cm at each altitude until the forward-follow distance (d_{ff}) becomes zero. Once the forward-follow curve (d_{ff}) is crossing zero, the turn-phase around the pipeline's endpoint is starting and the cross-follow curve (d_{cf}) is decreasing to zero (air-vehicle is converging from the course to the pipeline segment). If the cross-follow distance (d_{cf}) is increasing again from zero, where the air-vehicle is crossing the segment of the pipeline, to 20 cm, where the air-vehicle is diverged from the pipeline segment to follow the course of the air-vehicle (straight-line phase). The forward-follow curve (d_{ff}) has a gap at each turn-phase, these gaps are generated once the air-vehicle is crossing the pipeline segment where the configuration of the forward and backward endpoints are changed.

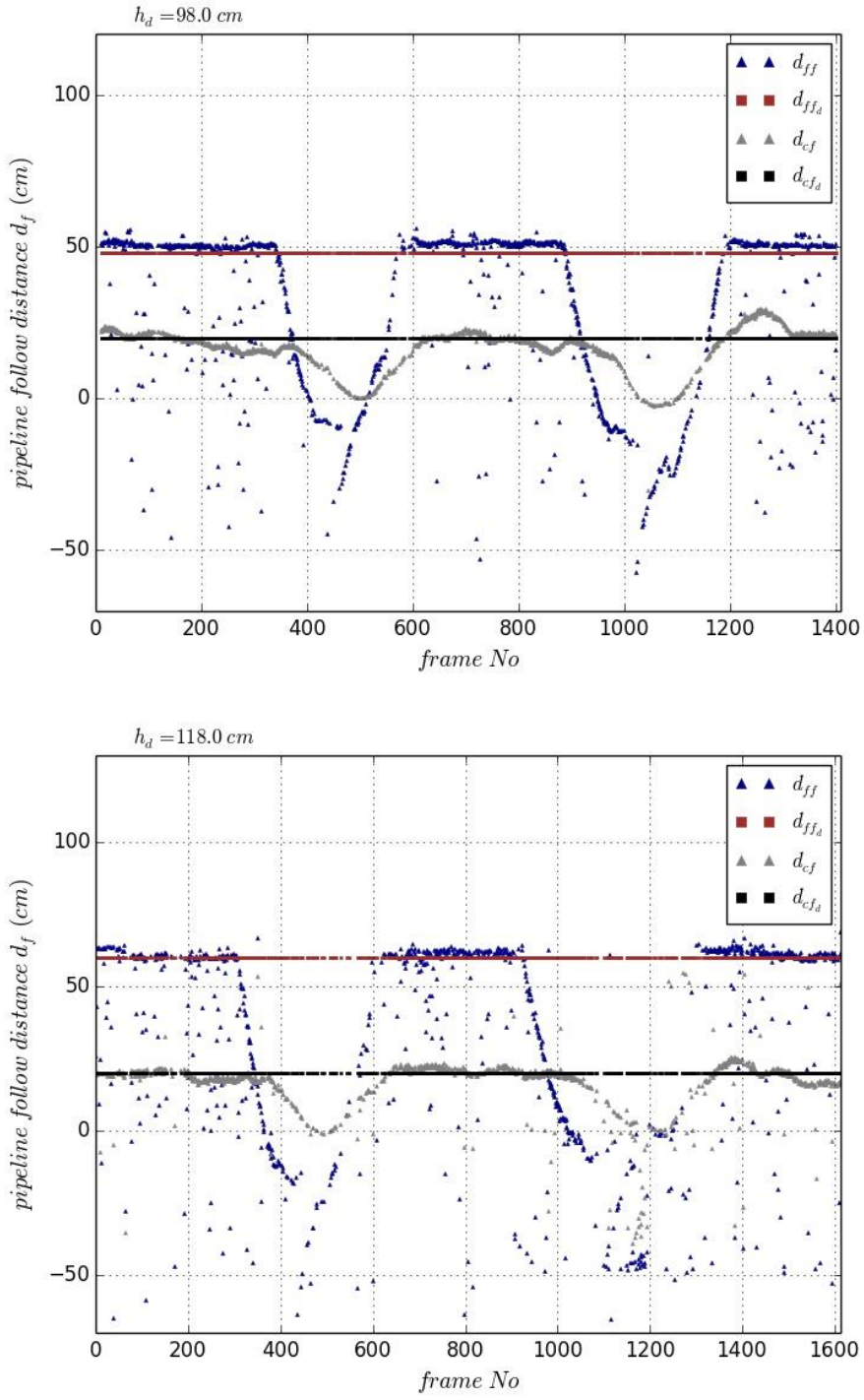


Figure 7-16: Behaviours of the forward-follow (d_{ff}) and cross-follow (d_{cf}) distances, where, (d_{ff_d}) and (d_{cf_d}) are the desired values of the forward and cross follow distances, respectively, while, [Top]: captured at 100 cm altitude, [Bottom]: captured at 120 cm altitude

The errors of the forward-follow (δ_{ff}) and cross-follow (δ_{cf}) were represented in Figure 7-17 to evaluate the north and east positions' accuracy of the air-vehicle when following the pipeline at different altitudes. The forward-follow error (δ_{ff}) represents the difference between the ground-truth (d_{ff_d}) and estimated (d_{ff}) forward-follow distances where those distances refer to the north coordinate's length between the air-vehicle and the forward-endpoint (EP_f) of the pipeline segment, relative to the camera frame. The proposed ground-truth forward-follow distances (d_{ff_d}), are equal to 50 cm and 65 cm at 100 cm and 120 cm altitude, respectively. As shown in red, the forward-follow error (δ_{ff}) shows how the pipeline is accurately forward-followed throughout the straight-line phase with a small error. At the turn-phases, the errors are neglected, because it is hard to know the ground-truth values of the forward-follow distance (d_{ff_d}) while the air-vehicle is performing this phase. However, the errors of the forward-follow (δ_{ff}) were evaluated at these phases based on the behaviour of forward-follow distance (d_{ff}) which is reasonable.

The cross-follow error ($\boldsymbol{\delta_{cf}}$) represents the difference between the cross-follow ground-truth ($\boldsymbol{d_{cf_d}}$) and estimated ($\boldsymbol{d_{cf}}$) distances where those distances refer to the east coordinate's length from the air-vehicle to the pipeline segment. The cross-follow ground-truth distance ($\boldsymbol{d_{cf_d}}$) was set to be around 20 cm at both 100 cm and 120 cm altitudes. Similarly, the cross-follows error ($\boldsymbol{\delta_{cf}}$) shows how the pipeline is accurately cross-followed throughout the sequential frames of the straight-phase with a small error as shown in blue in the figure.

The reasons of those errors come from several sources such as the calibrations of the sensor, vibrations, background detection, and external light effects. However, it is worth nothing that no effects are due to the altitude variation.

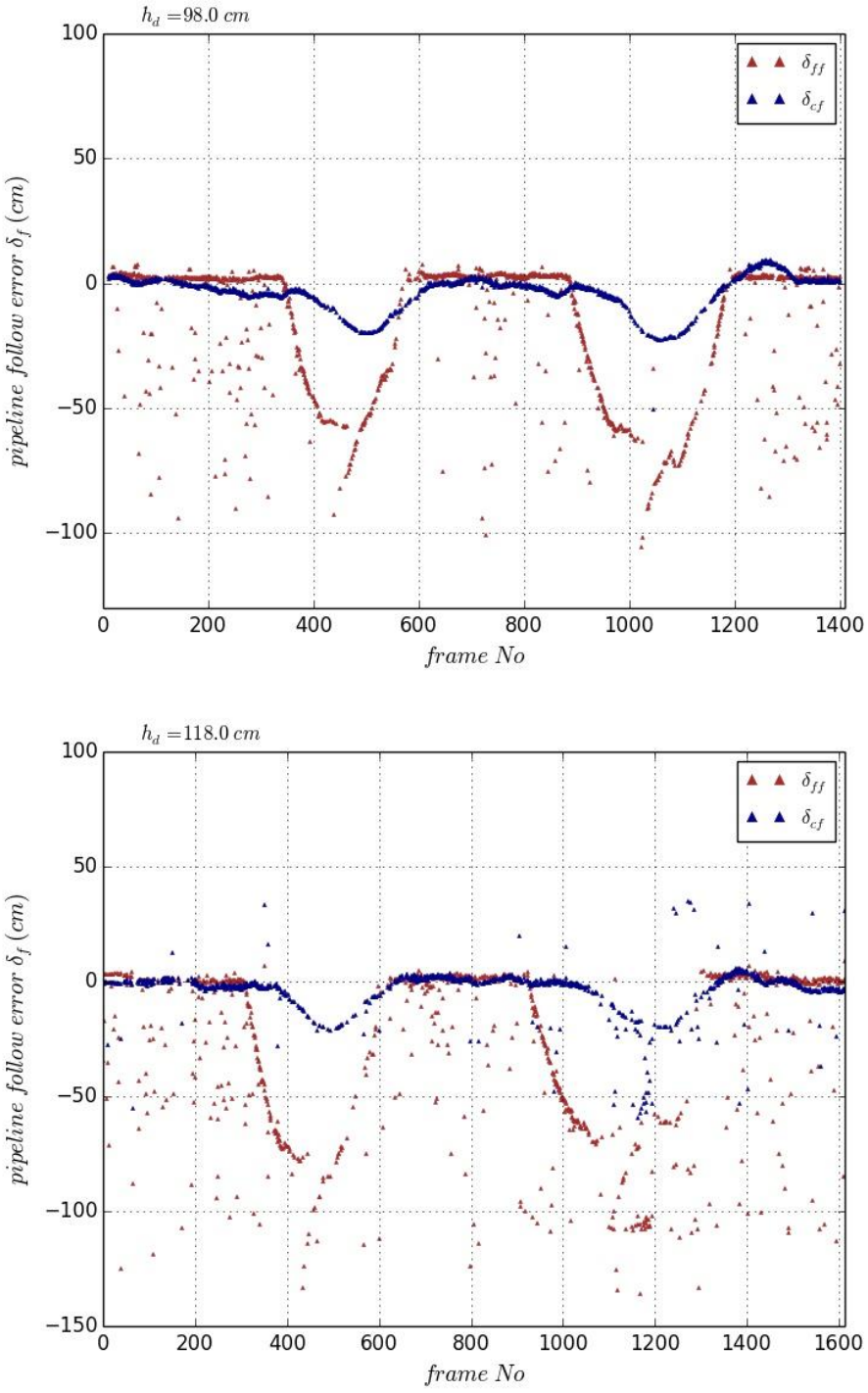


Figure 7-17: Behaviours of the errors of the pipeline forward-follow (δ_{ff}) and cross-follow (δ_{cf}), [Top]: captured at 100 cm altitude, [Bottom]: captured at 120 cm altitude

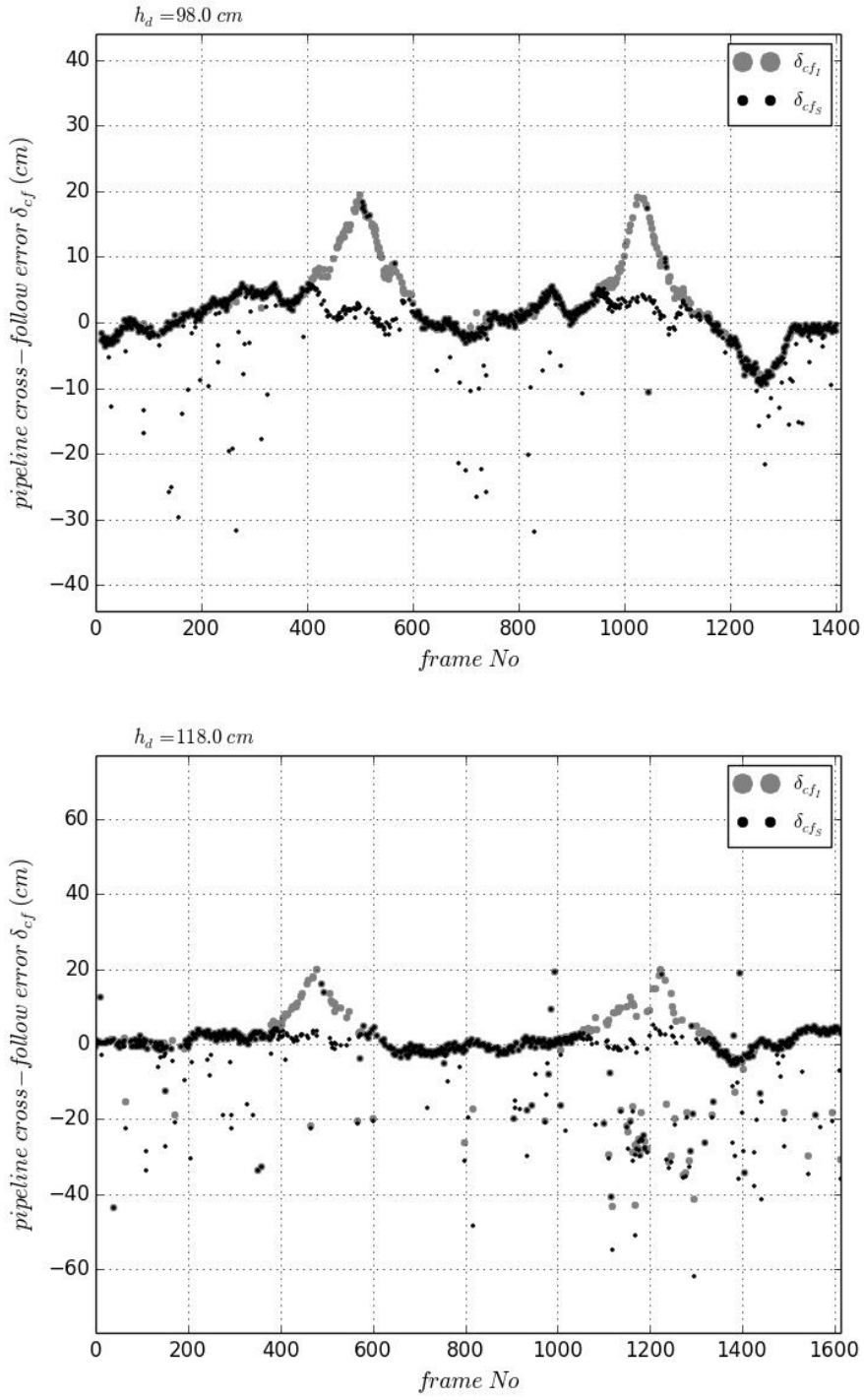


Figure 7-18: Behaviours of the pipeline cross-follow error based on the pipeline infinite (δ_{cf_l}) and pipeline segment (δ_{cf_s}), [Top]: captured at 100 cm altitude, [Bottom]: captured at 120 cm altitude

The errors of the cross-follow distances are represented in two ways as shown in Figure 7-18; the first one is represented with respect to the pipeline as a segment (δ_{cf_S}), and the other one is represented with respect to the pipeline as an Infinite line (δ_{cf_I}). The first approach is shown in grey, in which the air-vehicle approximately keeps maintaining the generated course based on the cross-follow error (δ_{cf_S}) at both straight-line and turn phases. At the same time, the second method (black colour) confirms that the air-vehicle is able of following the pipeline based on the generated course waypoints (WP_s), as shown in the figure in the zero cross-follow error (δ_{cf}) in straight-line follow phase. Then, the cross-follows error (δ_{cf}) starts increasing to reach 20 cm once the air-vehicle start passing the pipeline segment to keep following the pipeline by turning around the forward-endpoint (EP_f) then back again to zero to meet the straight-line following phase with around zero error.

7.5.2.2 3D Position of the Air-vehicle relative to Camera Frame

This section represents and evaluates the performance of the 3D position of the air-vehicle ($\boldsymbol{x, y, h}$) relative to the camera frame while following the pipeline. The 2D position $\boldsymbol{P_A(x,\ y)}$ of the air-vehicle relative to the camera frame reference, is supposed to be fixed throughout the frames' sequence at each test while following the pipeline. The 2D position $\boldsymbol{P_A(x,\ y)}$ of the air-vehicle was measured in (cm) at frame rate of 2 fps while the depth camera position is held at the centre of the gravity of the air-vehicle at 100 cm altitude ($\boldsymbol{x_d,\ y_d,\ h_d}$) = (9, -1.75, 100) and orientation ($\boldsymbol{\theta_d,\ \phi_d,\ \psi_d}$) = (1^o, -4.6^o, 0^o) and at 120 cm altitude ($\boldsymbol{x_d,\ y_d,\ h_d}$) = (9.5, -4.5, 120) and orientation ($\boldsymbol{\theta_d,\ \phi_d,\ \psi_d}$) = (2.3^o, -4.6^o, 0^o). The performance accuracy of the air-vehicle 2D position $\boldsymbol{P_A(x,\ y)}$ relative to the camera frame throughout the frame sequence at different altitudes is shown in Figure 7-19.

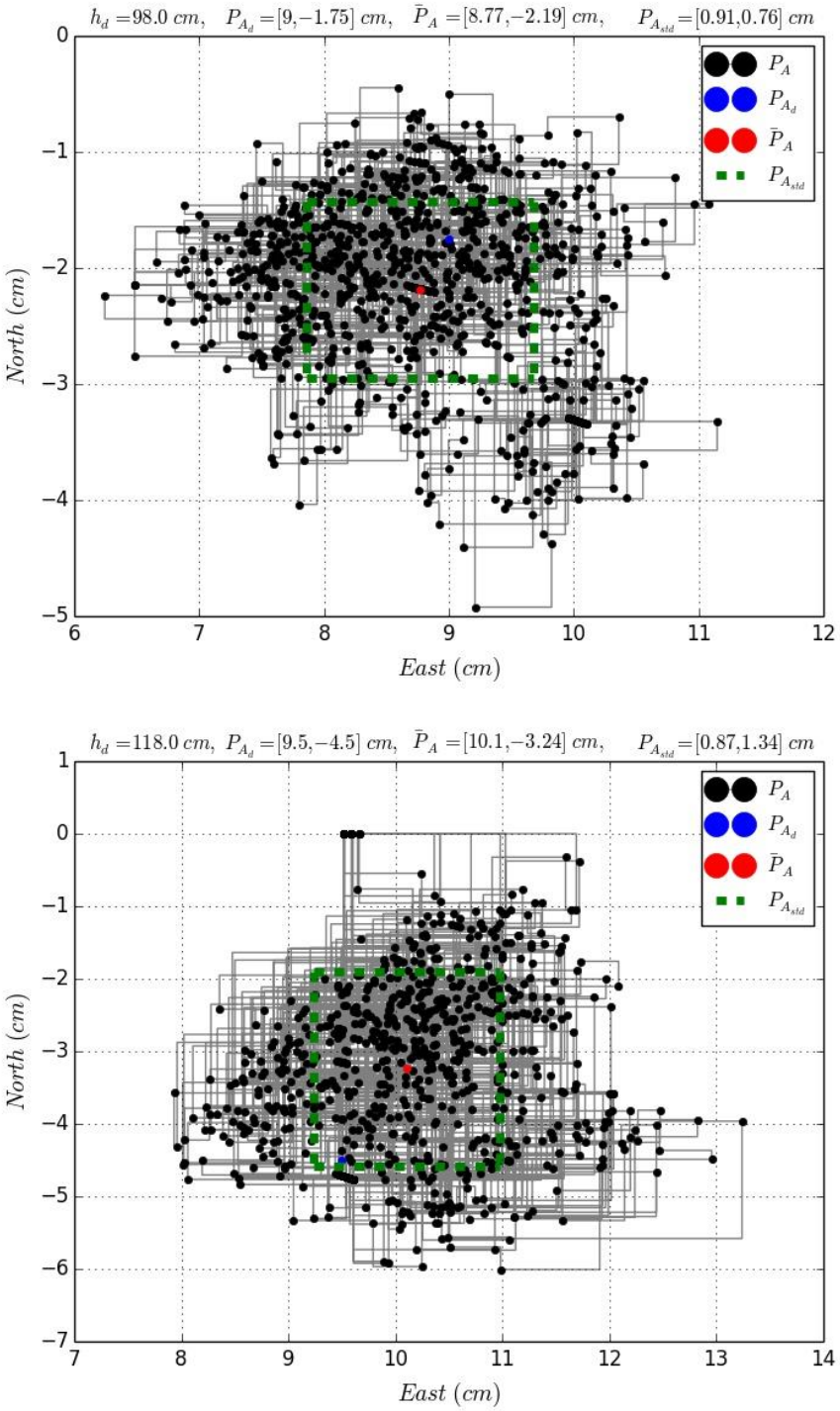


Figure 7-19: Behaviours of the 2D position of the air-vehicle (P_A) relative to the camera frame coordinates, involved the desired 2D position (P_{A_d}), the mean value ($\bar{P}_A$), the standard deviation ($P_{A_{std}}$), while, [Top]: captured at 100 cm altitude, [Bottom]: captured at 120 cm altitude

As shown in Figure 7-19, the estimated 2D position (P_A) of the air-vehicle, ground-truth (P_{A_d}), mean of measured ($\overline{P}_A$), and standard deviation ($P_{A_{std}}$) are presented in black, blue, red, and green, respectively, throughout the frames' sequence for the two altitudes. The resulted errors (δP_A) between the ground-truth 2D positions (P_{A_d}) and the mean of the estimated 2D positions ($\overline{P}_A$) is small for both of the altitudes, as tabulated in Table 7-5. These ground-truth 2D positions (P_{A_d}) are sited within the margin of the standard deviations ($\pm P_{A_{std}}$), hence confirming the performance capability and accuracy for maintaining ground-truth 2D positions (P_{A_d}) of the air-vehicle while following the pipeline.

To check the system ability of maintaining the ground-truth altitudes, while the air-vehicle following the pipeline, tests were carried-out and the results of which are shown in Figure 7-20. the results of the mean, standard deviation and absolute error of estimated altitude vs ground-truth are tabulated in Table 7-5. As can be seen the mean estimated error is small (less than 1cm), and with the standard deviation margin.

Finally, the results of the air-vehicle 3D position performance, which evaluate the air-vehicle's position capability and accuracy relative to the camera frame while following/tracking the pipeline at different altitudes are detailed in Table 7-5.

Table 7-5: Results of 3D position of the air-vehicle and their errors relative to the camera frame coordinates at different altitudes

h (cm)	***Ground-truth***			***Mean-estimated***			***std***			***error***		
	x_d (cm)	y_d (cm)	h_d (cm)	$\bar{x}$ (cm)	$\bar{y}$ (cm)	$\bar{h}$ (cm)	x_{std} (cm)	y_{std} (cm)	h_{std} (cm)	δx (cm)	δy (cm)	δh (cm)
100	9	-1.75	98	8.77	-2.19	98.02	0.91	0.76	0.45	0.23	0.44	0.02
120	9.5	-4.5	118	10.1	-3.24	117.99	0.87	1.34	0.3	0.6	1.26	0.01

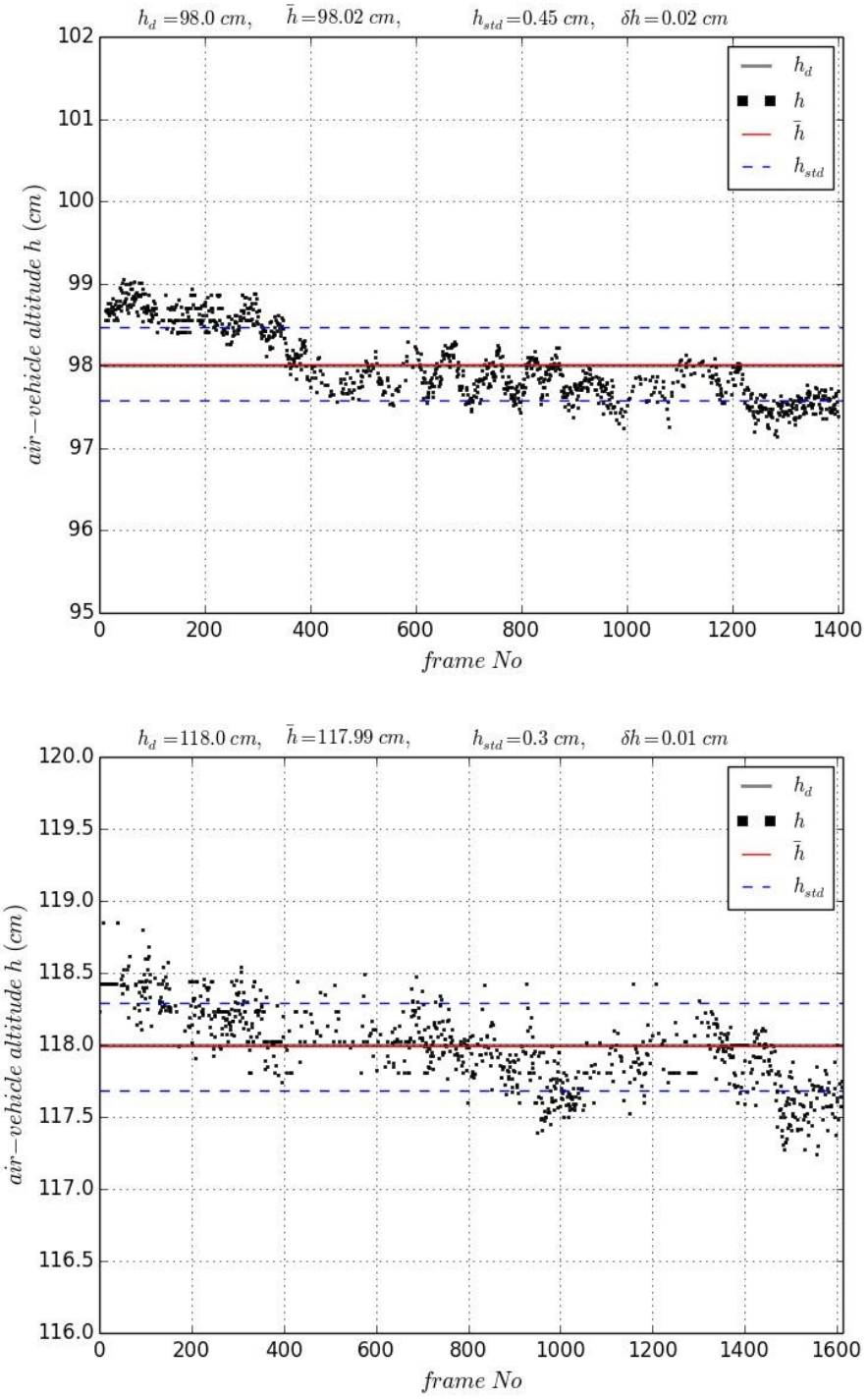


Figure 7-20: The estimated altitude of the air-vehicle (h) relative to the camera frame coordinates, the desired altitude of the air-vehicle (h_d), the mean value of the estimated altitude of the air-vehicle ($\bar{h}$), the standard deviation of the estimated altitude of the air-vehicle (h_{std}), [Top]: captured at 100 cm altitude, [Bottom]: captured at 120 cm altitude

7.5.2.3 3D Position Displacements of the Air-vehicle

This section represents the performance of the 3D position displacements of the air-vehicle that are measured among the frames sequence while following the pipeline to assess the potential effects of the following issues on the air-vehicle position:

- Sensor and air-vehicle vibrations.
- Plane detection accuracy.
- Air-vehicle acceleration.

Two tests were carried-out to represent and evaluate the effects of these issues. These tests were performed at two different altitudes to identify any effects due to change in pixel resolution. As shown in Figure 7-21, the dots represent the estimated 3D position displacements. The mean error values of the population are very close to zero i.e. with a small standard deviation for both altitudes. The results tabulated in Table 7-6 show that the effects of the three above mentioned issues; namely: sensor and air-vehicle vibrations, plane detection accuracy and air-vehicle acceleration are negligible.

Table 7-6: performance results of 3D position displacement and errors relative to the previous position at two altitudes scenarios

h (cm)	***desired***			***Mean-estimated***			***std***			***error***		
	Δx_d (cm)	Δy_d (cm)	Δz_d (cm)	$\bar{\Delta} x$ (cm)	$\bar{\Delta} y$ (cm)	$\bar{\Delta} z$ (cm)	Δx_{std} (cm)	Δy_{std} (cm)	Δz_{std} (cm)	$\delta \Delta x$ (cm)	$\delta \Delta y$ (cm)	$\delta \Delta z$ (cm)
100	0	0	0	-0.001	0	0.001	0.45	0.44	0.14	0.001	0	0.001
120	0	0	0	-0.001	-0.001	0.001	0.82	0.97	0.19	0.001	0.001	0.001

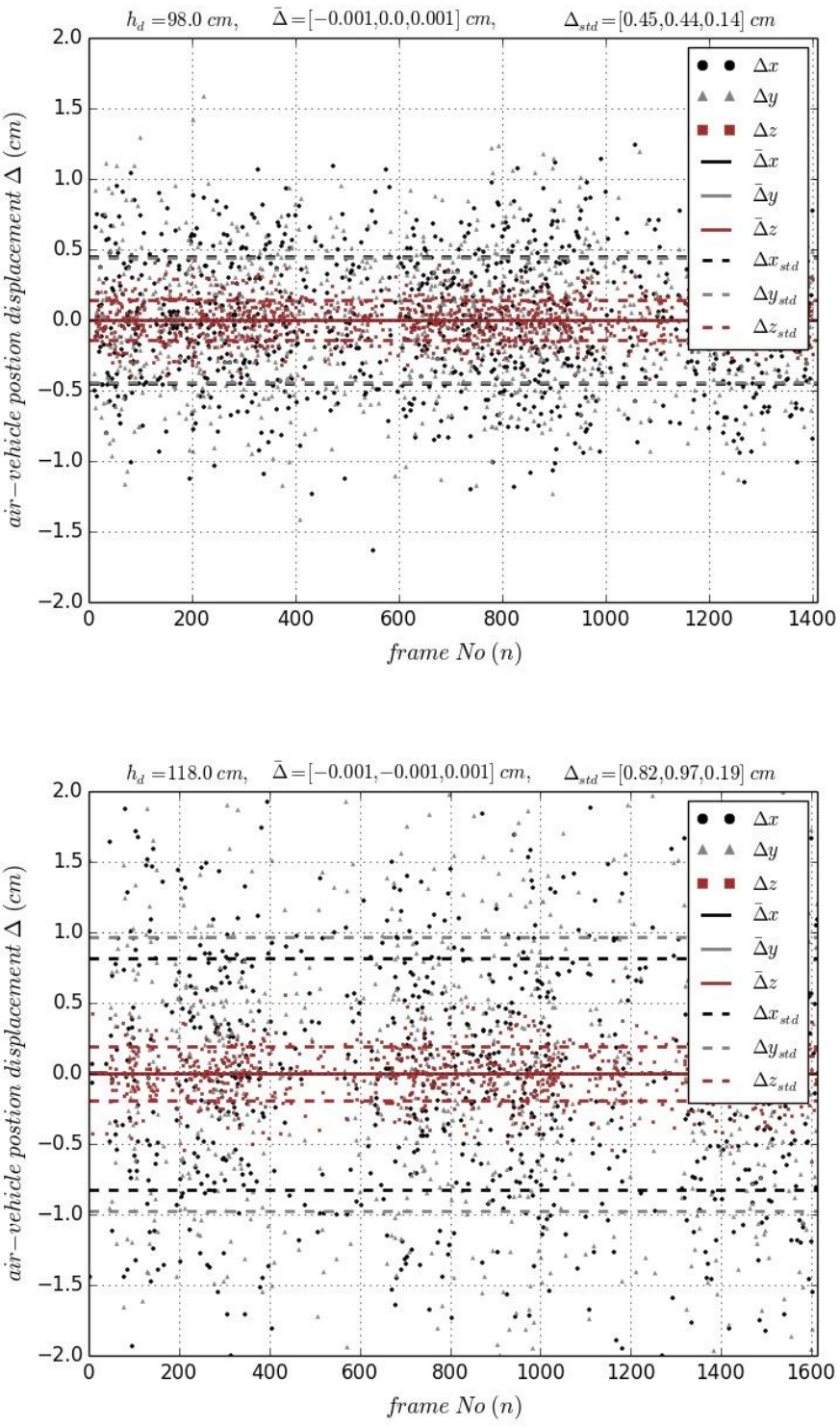


Figure 7-21: The estimated 3D position displacements of the air-vehicle (Δx, Δy, Δz), their main values, and standard deviations, [Top]: captured at 100 cm altitude, [Bottom]: captured at 120 cm altitude

7.5.2.4 Orientations of the Air-Vehicle

The orientation performance of the air-vehicle, relative to the camera frame while following the pipeline, is presented to test the reliability of the algorithm. This includes three-degrees of freedom for the air-vehicle's: pitch, roll, and yaw angles at two altitudes, 98 and 118 cm, respectively, to discover the effects of the change in pixel resolution.

As shown in Figure 7-22, the evolution of the estimated pitch angle, throughout the test, proves the ability of maintaining the desired pitch within 1 and 2.18 degrees at 98 and 118 cm altitudes, respectively, while the air-vehicle flyies around the pipeline in straight-line phase (to follow the pipeline segment) and turn phase (to turn around the endpoint of the pipeline segment). The minor errors of 0.28 and 0.61 degrees at 98 and 118 cm altitudes, respectively, are small deviations of 0.44 and 0.65 degrees. These results confirm the reliability of the pitch angle performance at both pipeline following phases.

Figure 7-23 shows that the evolution of the estimated roll angle throughout the test is capable of maintaining the desired roll of -5.25 and -4.6 degrees at 98 and 118 cm altitudes, respectively, while the air-vehicle flying around the pipeline in the straight-line phase (to follow the pipeline segment) and turn phase (to turn around the endpoint of the pipeline segment). The low errors of 0.14 and 0.29 degrees at 98 and 118 cm altitudes, respectively, are set within the low standard deviations of 0.51 and 0.44 cm. Hence again confirming the reliability of the roll angle performance in both pipeline following phases.

Figure 7-24 shows that the evolution of the estimated yaw angle, throughout the test, is capable of maintaining the desired yaw of zero degrees at both 98 and 118 cm altitudes, respectively, while the air-vehicle flying around the pipeline in a straight-line (to follow the pipeline segment) but during the turn phase (to turn around the endpoint of the pipeline segment), it was considered. The low errors of 0.14 and 0.29 degrees at 98 and 118 cm altitudes, respectively, are again sat within the low deviations of 0.51 and 0.44 degrees, confirming the reliability of the yaw angle performance in the straight-line phase.

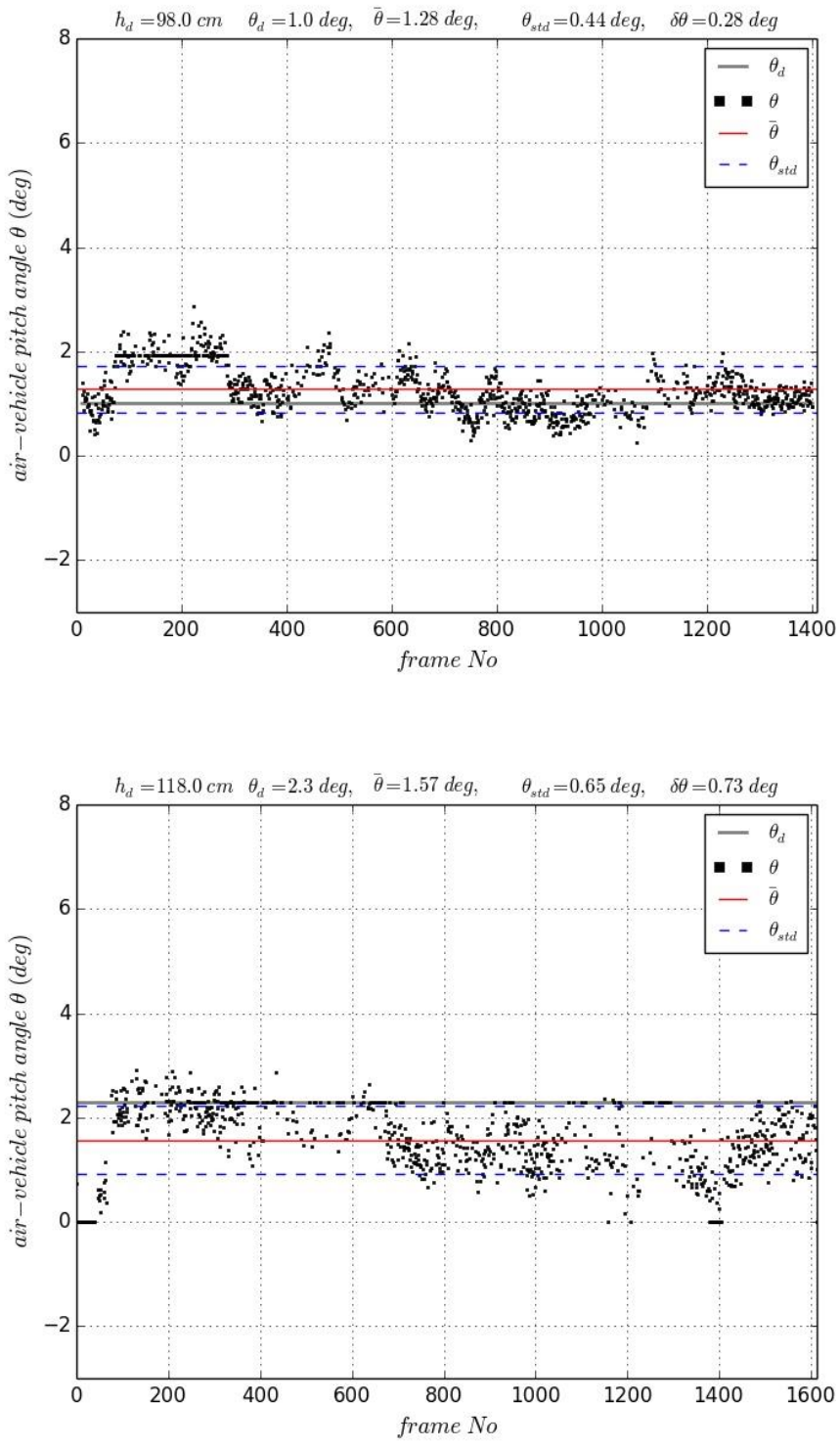


Figure 7-22: The estimated air-vehicle's pitch angle (θ), the desired pitch angle (θ_d), the mean value ($\bar{\theta}$), the standard deviation (θ_{std}), [Top]: captured at 100 cm altitude, [Bottom]: captured at 120 cm altitude

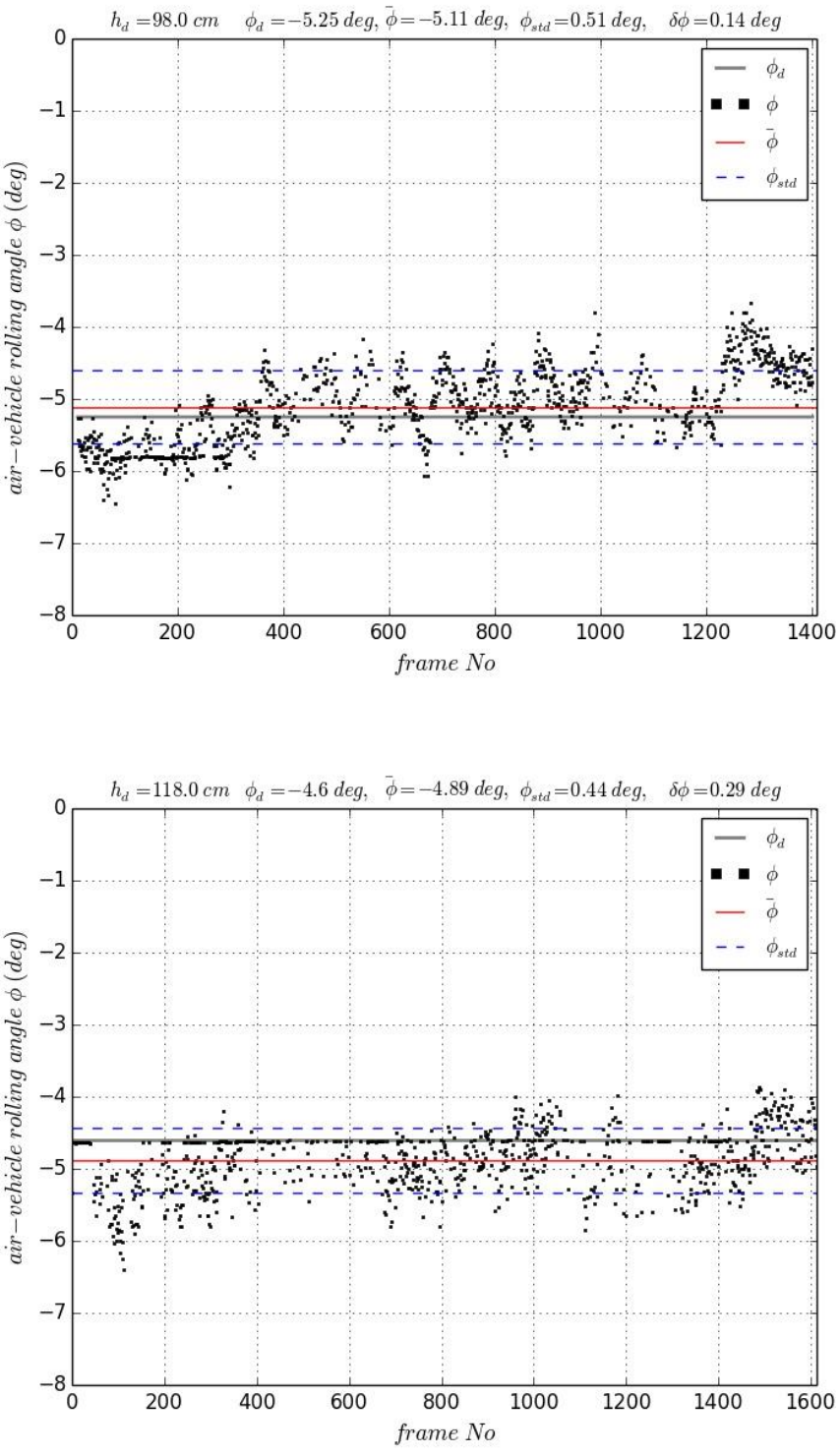


Figure 7-23: The estimated air-vehicle's roll angle (ϕ), the desired roll angle (ϕ_d), the mean value ($\bar{\phi}$), the standard deviation (ϕ_{std}), [Top]: captured at 100 cm altitude, [Bottom]: captured at 120 cm altitude

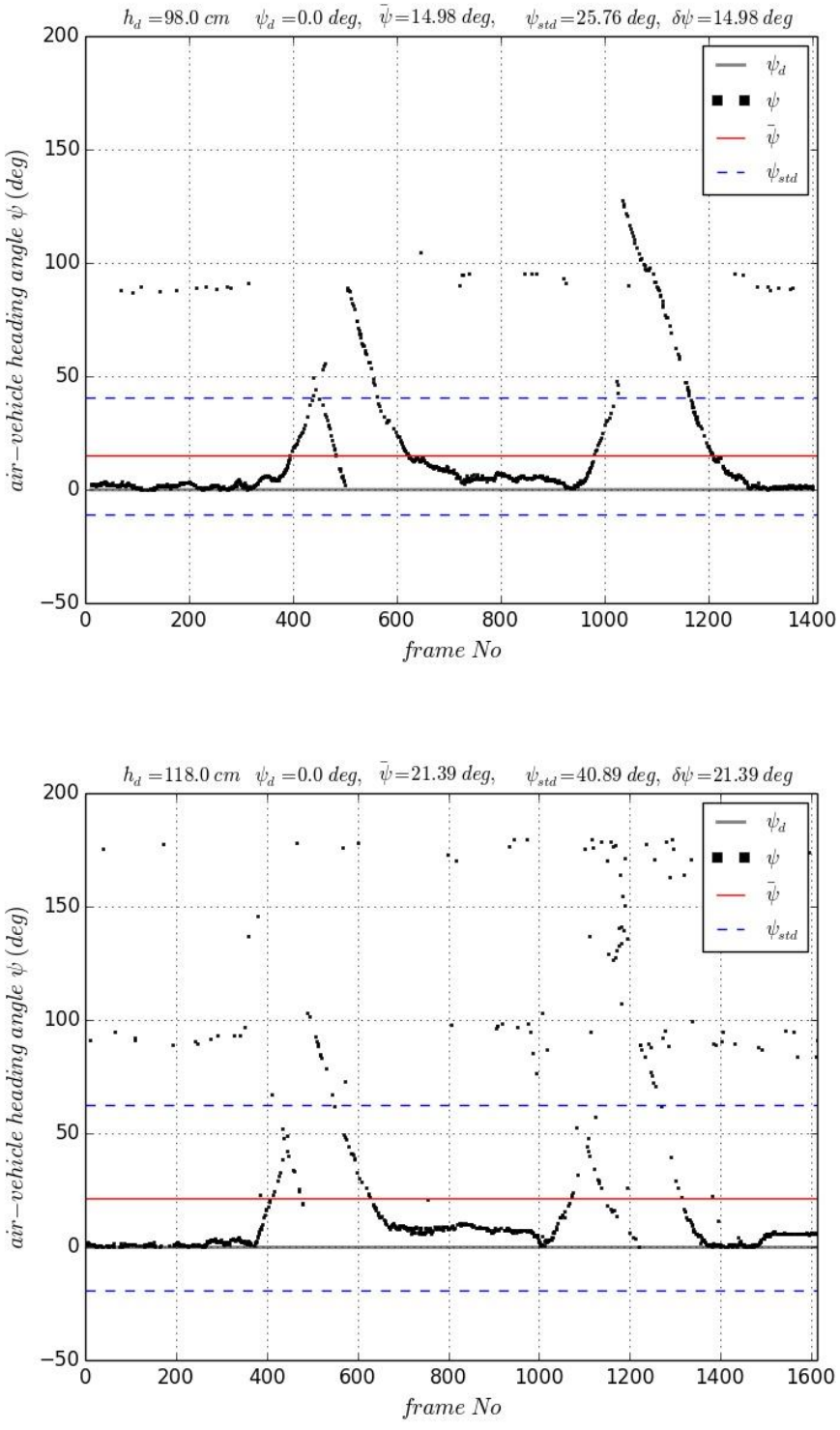


Figure 7-24: The estimated air-vehicle's yaw angle (ψ), the desired yaw angle (ψ_d), the mean value ($\bar{\psi}$), the standard deviation (ψ_{std}), [Top]: captured at 100 cm altitude, [Bottom]: captured at 120 cm altitude

The results of the air-vehicle orientation performance, which evaluate the air-vehicle's orientation accuracy, relative to the camera frame while following the pipeline at two different altitudes, are detailed in Table 7-7. These results confirm the air-vehicle ability of following the pipeline with low orientation errors.

Table 7-7: Results of air-vehicle orientations and their errors relative to the camera frame at different altitudes

h (cm)	***Ground-truth***			***Mean-estimated***			***std***			***error***		
	$\boldsymbol{\theta_d}$ (deg)	$\boldsymbol{\phi_d}$ (deg)	$\boldsymbol{\psi_d}$ (deg)	$\bar{\boldsymbol{\theta}}$ (deg)	$\bar{\boldsymbol{\phi}}$ (deg)	$\bar{\boldsymbol{\psi}}$ (deg)	$\boldsymbol{\theta}_{std}$ (deg)	$\boldsymbol{\phi}_{std}$ (deg)	$\boldsymbol{\psi}_{std}$ (deg)	$\delta\boldsymbol{\theta}$ (deg)	$\delta\boldsymbol{\phi}$ (deg)	$\delta\boldsymbol{\psi}$ (deg)
100	1	-5.25	0	1.28	-5.11	14.98	0.44	0.51	25.76	0.28	0.14	14.98
120	2.18	-4.6	0	1.57	-4.89	21.39	0.65	0.44	40.89	0.61	0.29	21.39

7.6 Chapter Summary

This chapter represented and evaluated the performance of the vision-based aerial pipeline ROW surveillance system. This performance involves the pipeline endpoints' identification, third-party interference detection, and the pipeline following algorithms. Due to the sensor limitations and the difficulty of finding above ground pipeline in the UK, indoor tests were performed to provide the desired data sets that are required to represent and evaluate the reliability of each algorithm in this system. Four tests were proposed to assess the performance of the pipeline endpoints identification algorithm, which include the attendance the pipeline structure and the altitudes difference. The presence of the pipeline in the image frame is used to assess the identification decision and the error when the pipeline exists, by estimating the sensitivity and false negative rates, respectively. It is also proposed to evaluate the quality and accuracy of the identification by analysing the performance of the position of the endpoints throughout the flight mission.

The test in which no pipeline is present in the image frame is used to assess the identification decision and its error, by estimating the specificity and false positive rates, respectively. Different altitudes tests were carried-out to identify and evaluate the effects on the performance of the identification, position, and processing speed when the pixel resolution changes. The result of these tests confirms that the system is capable of identifying the pipeline endpoints in near real-time efficiently with a high rate of identification and small errors. It can also estimate the endpoints position accurately with a low error at a different resolution of pixels.

Similarly, four tests were performed to demonstrate the performance of the third-party detection algorithm and the effect of changing altitudes. The presence of the third-party interference is used to assess the detection decision and its error, while tests in which the third-party interference exists in the image frame are checked by estimating the sensitivity and false negative rates, respectively. Analysing the performance of the position of the third-party interference throughout the flight mission is used to evaluate the quality and accuracy of the detection. The absence of the third-party interference in the image frame is used to assess the detection decision and its errors by estimating the specificity and false positive rates, respectively. The difference in altitudes is proposed to discover and evaluate the effects on the performance of the detection, position, and processing speed when the resolution of pixels changes. The result of the performance tests confirms that the system is capable of efficiently identifying the third-party interference on-board in near real-time with a high rate of detection and low errors. It can also estimate the third-party interference position accurately with a low error at a different pixel resolution.

Two tests were performed at various altitudes involving a pipeline structure, to evaluate the performance of the pipeline's following algorithm and confirm its capability. These tests involve the online course waypoints generation, waypoints navigation, and processing time. Based on the performance results, the system proved to be capable and accurate in autonomously following the

pipeline in real-time by using the vision-based identified endpoints of the pipeline to navigate the air-vehicle (quadrotor).

Chapter 8
Discussions and Conclusions

8.1 Introduction

This chapter discusses, concludes and remarks on the automatic pipeline surveillance air-vehicle research by focusing on the discussions of the results, satisfactions of the aim and objectives, limitations, and future recommendations.

8.2 Discussion of Research Results

To emulate the aerial pipeline surveillance mission, a depth sensor (Asus Xtion) mounted on-board a fully-functional quadrotor UAV platform was proposed to provide depth data (represented as a 16-bits format) and RGB data (as 8-bits) of the explored environment in real-time.

The endpoints of the pipeline segment were identified accurately in real-time based on two data types that are visibility and depth. Computer vision techniques were used to develop the visible-based pipeline endpoint identification algorithm. The first step involves the image processing algorithm that enhances the vision data to improve the edge detection performance. Then, the boundaries of the features are detected in the explored environment that are assessed, constructing a candidate straight boundary for the Hough transforms method, by using a canny edge detector. After that, based on the candidate's boundaries, several straight lines representing the straight

boundaries of the objects were constructed using Hough Transform method. Following this, those straight lines were filtered by comparing them with known pipeline segment endpoints in the explored area, allowing fast and accurate pipeline endpoints identification. This allows the system to reliably reject inaccurate measurements while retaining the correct pipeline detections and location. Based on the experimental results, the performance is capable of identifying the endpoints of the pipeline in real-time. As was shown in the analysis carried-out in the project, this vision-based pipeline endpoint identification has several limitations. These include the need to have pipeline end point position information, to fuse/compare with the vision based detection system, to improve the detection rate and increase the confidence level in the system.

Therefore, the depth-based pipeline endpoints identification technique was proposed, to complement the vision based approach. This technique includes 3D point cloud mapping, foreground and background extraction, boundary detection, boundaries height filtration, boundaries straight line detection, pipeline verification and pipeline endpoints estimation. First, the 16-bits depth data of the explored environment were transformed into 3D point clouds' world coordinates. Then, the foreground and background were extracted based on the transformed 3D point cloud to extract the plane that corresponds to the ground, using RANSAC approach. Simultaneously, the boundaries of the explored environment were detected based on the 16-bit depth data using Canny method. After that, these boundaries were filtered out, after being transformed into a 3D point cloud, based on the real height of the pipeline for fast and accurate measurements using a Euclidean distance of each boundary point relative to the plane of the ground extracted before. Then, those filtered boundaries were used to detect the straight lines of the object border (Hough lines), after being transformed into 16-bit depth data, using a Hough transform method. Following that, the pipeline was verified by estimating a centre line segment, in the 3D point cloud, between any two of those Hough line segments, after being transformed into 3D. That satisfy the following statements'

parallelism, distance between each other meets the constraints of the real width of the pipeline, and the foreground correspondence that the length of the centre line segment corresponds to the foreground region. The length and endpoints position of the detected centre line segments were enhanced to match the exact pipeline by extending them along their inlier foreground. Finally, the pipeline endpoints were recomputed if locally exist multi-segments intersections representing the pipeline structure in the same frame by estimating the local intersection points between the segments. This technique was tested indoors using a small scale representation of pipeline structures. The results obtained confirm the capabilities of the proposed method in identifying the pipeline endpoints at real-time.

Moreover, a computer vision technique was developed for real-time third-party interference detection based on four parameters; that are foreground 16-bit depth data, pipeline corresponds 16-bit depth data, pipeline endpoint location in the 3D point cloud and ROW proposed distance. This technique includes detection, classification, and localization algorithms. The detection was developed to detect any object, in general, within the pipeline's ROW region and consider them as a third-party interference objects. This detection is then processed by filtering the foreground depth data to concentrate on the area of interest that is expected to have a third-party interference objects present, hence reducing the processing load, and increasing the detection accuracy. This filtration includes the subtraction of the pipeline and the ROW outlier's areas. The pipeline area is filtered by subtracting the 16-bit depth matrix of the pipeline from the foreground directly. Then, the residual data, after being transformed into a 3D point cloud, is used to filter the ROW outliers area by using the Euclidean distance to reject any point outlier in the region of interest relative to the pipeline segment based on the ROW proposed distance. The detected objects are classified using Haar classifier, after data association and transformation is carried-out to sync with vision based data that are being recorded simultaneously. They are classified, for example into buildings, known vehicles, trees, and so on. The detected third-party interference objects were

then localized based on the camera frame using a centroid contour algorithm. The performance of this technique was experimentally validated using an indoor test, in which a small-scale of expected third-party interference objects are introduced. The results show that is the developed approach and algorithms are capable of efficiently detecting the third-party interference objects quickly at a high detection rate. The advantage of using an approach as the Haar classifier is that these types of can be trained off-line first, hence significantly improving the performance and speed of the algorithm. Another advantage is that new objects can be trained and detected by the algorithm.

Finally, a waypoints-based navigation system was developed to enable the air-vehicle to fly over an online generated course using heading demand to follow the pipeline structure autonomously in real-time. The waypoints are generated based on the online identification of the pipeline endpoints relative to camera frames. The proposed autopilot system, used to track the pipeline, consists of online waypoints generation for the air-vehicle's course, change of course angle, turn anticipation distance estimation, proximity distance estimation, and heading demand calculation. The system was tested indoors. The performance was satisfactory, confirming the ability to follow the pipeline based on the received information.

8.3 Fulfilment of Research Aim and Objectives

In this research project, the use of a low-cost aerial platform was investigated successfully for surveillance and routine inspections of lengthy pipelines and their Rights-of-Way. The payload system consists of a camera and a depth sensor mounted on-board a quadrotor platform. The data from the payload system were processed using, computer vision algorithms and associated for a real-time detection and visual following of the pipeline structure, detection of anomalies through appropriate image recognition algorithm, and video data relay to the ground station.

The developed algorithms were effective in estimating the position of the pipeline segments endpoints in near real-time. Also, a real-time vision-based waypoints navigation algorithm was developed to automatically track the pipeline structure without relying on GPS data but based on the endpoints information of the pipeline segment. Moreover, the developed vision based algorithms were capable of detecting the anomalies in the vicinity of the pipeline structure in near real-time and visualize them for the ground operator. A video data transmission datalink was proposed to transfer the data from the aerial platform to the Ground Control Station (GCS). The complete pipeline surveillance system was successfully tested by integrating both the hardware and the developed software together.

8.4 Research Limitations

The limitations of this research are outlined as follows:

- Although the processing speed of the on-board embedded system were adequate to perform some of the operations. As with all embedded system there are limit on the processing power. Therefore, the data processing was divided on-board (detection, tracking etc.) and off-board, on the GCS for the classification of 3rd party interference.
- The focus of the project was to test the system for the surveillance and inspection of the over-ground pipeline structure.
- The surveillance and inspection missions of the pipeline structure and its Right-of-Way covered only third-party activates.
- The remote sensor mounted on the system to provide the 3D visual data can only work indoor.
- A small-scale environment of pipeline structure and third-party activates were used in this research for the validation. Change in environmental factors e.g. sands, snow *etc.* were not considered.

8.5 Conclusion

In conclusion, the fulfilments of the research aim and objectives were successfully accomplished. The algorithms' developments and the system integration were presented and described. The performance results of the system were analysed and discussed. The limitations of the research were listed. Finally, the recommendations for the future work are covered in the section below (section 8.6).

8.6 Future Work

Below are some recommendation and improvements for future work:

- Comparing the implemented computer vision techniques in this project with other techniques in terms of; quantity/quality performance and processing time.
- Investigating for a hardware or software solutions to optimise and reduce the computational load of the developed algorithms that are processed on-board the aerial platform.
- Developing a method to detect underground pipeline structures by fusing other sensors, such as magnetometer, which are suitable to extract the information of the pipeline remotely in near real-time based on the material differentiations.
- Developing an algorithm to detect any oil or gas pipeline leak remotely in near real-time using suitable remote sensors such as thermal or hyperspectral.
- Integrating a LIDAR or a stereo vision sensor instead of the ASUS Xtion to provide visual depth and RGB data and preparing the system for outdoor tests.

- Carry-out outdoor flight tests over one of the existing over-ground pipeline system such as Trans Alaskan to validate the performance of the proposed system in real operational environment.

REFERENCES

Abkar, A., Sharifi, M. and Mulder, N. (2000), "Likelihood-Based Image Segmentation and Classification: a Framework for the Integration of Expert Knowledge in Image Classification Procedures", *International Journal of Applied Earth Observation and Geoinformation*, Vol. 2 No. 2, pp. 104–119.

Agrawal, R., Shevgaonkar, R. and Sahasrabudhe, S. (1996), "A Fresh Look at the Hough Transform", *Pattern Recognition Letters*, Vol. 17 No. 10, pp. 1065–1068.

Ahuja, N. (2004), "Vision Based Fire Detection", *Proceedings of the 17th International Conference on Pattern Recognition*, pp. 134–137.

Althouse, M. and Chang, C. (1995), "Chemical Vapor Detection and Mapping with a Multispectral Forward-Looking Infrared (FLIR)", *In Optical Sensing for Environmental and Process Monitoring, International Society for Optics and Photonics*, Vol. Vol. 2366, pp. 108–114.

Anthony, M. (2012), "100 Causes for 100 Spills", available at: http://homosapienssaveyourearth.blogspot.co.uk/2012/06/100-causes-for-100-pipeline-oil-spills.html, (accessed 1 July 2012).

Antolovic, D. (2008), "Review of the Hough Transform Method, With an Implementation of the Fast Hough Variant for Line Detection", *Department of Computer Science, Indiana University*.

Anupama, K.R., Kamdar, N., Kamalampet, S.K., Vyas, D., Sahu, S. and Shah, S. (2014), "A Wireless Sensor Network Based Pipeline Monitoring System", *International Conference on Signal Processing and Integrated Networks (SPIN)*, pp. 412–419.

ArduPilot. (2013), "Open Source Autopilot", available at: http://ardupilot.com, (accessed 4 March 2013).

ASUS. (2012), "Xtion PRO LIVE", available at: www.asus.com, (accessed 1 October 2013).

Baer, R. (2005), *Linear Algebra and Projective Geometry*, Courier Corporation.

Baker, J.M. (2009), "Mechanical Damage", *PHMSA-Office of Pipeline Safety*.

Behrens, T., Rohr, K., Stiehl, H.S. and Nistér, D. (2003), "Robust Segmentation of Tubular Structures in 3-D Medical Images by Parametric Object Detection and Tracking", *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, Vol. 33 No. 4, pp. 554–561.

Bennett, C., Carter, M. and Fields, D. (1996), "Hyperspectral Imaging in the

Infrared Using LIFTIRS", *In Optical Remote Sensing for Environmental and Process Monitoring*, Vol. 2883, p. 267.

Boljanovic, V. (2006), *Applied Mathematical and Physical Formulas Pocket Reference*, Industrial Press.

Borrmann, D., Elseberg, J., Lingemann, K. and Nüchter, A. (2011), "The 3D Hough Transform for Plane Detection In Point Clouds: a Review and a New Accumulator Design", *3D Research*, Vol. 2 No. 2, pp. 32:1–32:13.

Boundary Devices. (2012), "Nitrogen 6x Embedded Single Board", available at: https://boundarydevices.com, (accessed 15 October 2013).

Bradski, G. and Kaehler, A. (2008), *Learning OpenCV: Computer Vision with the OpenCV Library*, O'Reilly Media, Inc.

Breckon, T. and Barnes, S. (2009), "Autonomous Real-Time Vehicle Detection from a Medium-Level UAV", *Proc. 24th International Unmanned Air Vehicle Systems*, pp. 29.1–29.9.

Burkhardt, G.L. and Crouch, A.E. (2003), *Realtime Monitoring of Pipelines for Third-Party Contact*, *Sensor Systems and NDE Technology Department, Applied Physics Division, Southwest Research Institute*, Texas.

Canadian Energy Pipeline Association (CEPA). (2010), "Types of Pipelines", available at: http://www.cepa.com, (accessed 3 August 2012).

Canny, J. (1986), "A Computational Approach to Edge Detection", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 8, pp. 679–698.

Canon. (2011), "RGB Camera", available at: http://www.canon.co.uk, (accessed 10 November 2012).

Chen, W., Cao, L., Xiao, J. and Huang, S.G. (2011), "Surface Texture Detection for Composite Material Placement Based on Vision", *In Advanced Materials Research*, Vol. 186, pp. 16–20.

Collins, R., Tsin, Y., Miller, J. and Lipton, A. (1998), "Using a DEM to Determine Geospatial Object Trajectories", *DARPA Image Understanding Workshop*, pp. 115–122.

Cook, G.W. and Delp, E.J. (1998), "Gaussian Mixture Model for Edge-Enhanced Images with Application to Sequential Edge Detection and Linking", *In International Conference on Image Processing (ICIP98)*, Vol. 2, pp. 540–544.

Culshaw, B. and Dakin, J. (1996), *Optical Fiber Sensors: Components and Subsystems*, Vol 3., Artech House Publishers.

Derpanis, K. (2010), "Overview of the RANSAC Algorithm", *York University, Toronto, Canada*.

Devaud, J., Najko, S., Pierre Le, N., Maussire, C., Zante, E. and Marzat, J. (2012), “Full Design of a Low-Cost Quadrotor UAV by Student Team”, *International Conference on System Engineering and Technology (ICSET)*, pp. 1–6.

DIGI. (2014), “XBee Multipoint Wireless Networking RF Module”, available at: http://digi.com, (accessed 7 February 2014).

Dillman, D. (2002), “Method and System for Dynamic Surveillance of a Remote Object Using GPS”, *U.S. Patent No. 6,388,611*.

Dobrokhodov, V. and Kaminer, I. (2006), “Vision-Based Tracking and Motion Estimation for Moving Targets Using Small UAVs”, *In American Control Conference*.

Du, J.C. and Teng, H.C. (2007), “3D Laser Scanning and GPS Technology for Landslide Earthwork Volume Estimation”, *Automation in Construction*, Vol. 16 No. 5, pp. 657–663.

Eckbreth, A. (1977), “Laser Raman and Fluorescence Techniques for Practical Combustion Diagnostics”, *Applied Spectroscopy Reviews*, Vol. 13 No. 1, pp. 15–164.

Fiedler, M., Meier, W. and Hoppe, U. (1995), “Texture Analysis of the Surface of the Human Skin”, *Skin Pharmacology and Physiology*, Vol. 8 No. 5, pp. 252–265.

Fischler, M. and Bolles, R. (1981), “Random Sample Consensus: a Paradigm for Model Fitting With Applications to Image Analysis and Automated Cartography”, *Communications of the ACM*, Vol. 24 No. 6, pp. 381–395.

Frew, E., McGee, T. and Kim, Z. (2004), “Vision-Based Road-Following Using a Small Autonomous Aircraft”, *Aerospace Conference*, Vol. 5, pp. 3006–3015.

Fung, K.B., Fraser, D.W. and Gauthier, R.P. (1998), “Application of Remote Sensing Data for Monitoring of Gas Pipeline Right-of-Way”, *In Proceedings of GIS*, pp. 6–9.

Gaszczak, A., Breckon, T. and Han, J. (2011), “Real-Time People and Vehicle Detection from UAV Imagery”, *IS&T/SPIE Electronic Imaging, International Society for Optics and Photonics*.

Gonzalez, R.C. and Woods, R.E. (2003), *Digital Image Processing*, 2nd ed., Addison-Wesley, NJ.

Gopalsami, N. and Raptis, A.C. (2001), “Millimeter-Wave Radar Sensing of Airborne Chemicals”, *IEEE Transactions on Microwave Theory and Techniques*, Vol. 49 No. 4, pp. 646–653.

Guadalupe-Blanco River Authority (GBRA). (2010), *Oil and Gas Pipeline*

Failure, Texas.

Gupte, S., Mohandas, P.I.T. and Conrad, J.M. (2012), “A Survey of Quadrotor Unmanned Aerial Vehicles”, *Southeastcon, 2012 Proceeding of IEEE*, pp. 1–6.

Haala, N. and Brenner, C. (1999), “Extraction of Buildings and Trees in Urban Environments”, *ISPRS Journal of Photogrammetry and Remote Sensing*, Vol. 54 No. 2, pp. 130–137.

Hasan, K., Rahman, M. and Haque, A. (2009), “Cost Effective GPS-GPRS Based Object Tracking System”, *Proceedings of the International Multiconference of Engineers and Computer Scientists*, Vol. 1, pp. 18–20.

He, Y., Wang, H. and Zhang, B. (2004), “Color-Based Road Detection in Urban Traffic Scenes”, *Intelligent Transportation Systems, , IEEE Transactions on*, Vol. 5 No. 4, pp. 309–318.

Herrera, C., Kannala, J. and Heikkilä, J. (2012), “Joint Depth and Color Camera Calibration with Distortion Correction”, *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, Vol. 34 No. 10, pp. 2058–2064.

Hjelmås, E. and Low, B. (2001), “Face Detection: A Survey”, *Computer Vision and Image Understanding*, Vol. 83 No. 3, pp. 236–274.

Hopkins, P. (2002), “The Structural Integrity of Oil and Gas Transmission Pipelines”, *Comprehensive Structural Integrity, Elsevier Publishers Penspen Ltd., Berlin*.

Hopkins, P., Fletcher, R. and Palmer-Jones, R. (1999), *A Method for the Monitoring and Management of Pipeline Risk – A Simple Pipeline Risk Audit (SPRA)*, *Andrew Palmer and Associates*.

Hough, P.V.C. (1962), “Method and Means for Recognizing Complex Patterns.”

Huang, A.S.A., Bachrach, A., Henry, P., Krainin, M., Fox, D., Roy, N. and Maturana, D. (2011), “Visual Odometry and Mapping for Autonomous Flight Using an RGB-D Camera”, *International Symposium on Robotics Research (ISRR)*, pp. 1–16.

Huang, S., Lin, W., Tsai, M. and Chen, M. (2007), “Fiber Optic In-Line Distributed Sensor for Detection and Localization of the Pipeline Leaks”, *Sensors and Actuators A: Physical*, Vol. 135 No. 2, pp. 570–579.

Illingworth, J. and Kittler, J. (1988), “A Survey of the Hough Transform”, *Computer Vision, Graphics, and Image Processing*, Vol. 44 No. 1, pp. 87–116.

Jarząbek-Rychard, M., Revol-Muller, C., Peyrin, F., Carrillon, Y. and Odet, C. (2010), “Implementation of 3D Point Clouds Segmentation Based on Plane Growing Method”, *Practical Aspects of Geodesy and Cartography*, pp. 1–7.

Jin, Y. and Eydgahi, A. (2008), “Monitoring of Distributed Pipeline Systems by Wireless Sensor Networks”, *Proceedings of The 2008 IAJC-IJME International Conference.*, pp. 1–10.

Joseph, D. and Winslow, J. (1982), “Cathodic Protection Monitoring”, *U.S. Patent No. 4,351,703*.

Kanade, T., Amidi, O. and Ke, Q. (2004), “Real-Time and 3D Vision for Autonomous Small and Micro Air Vehicles”, *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, Vol. 2, pp. 1655–1662.

Kenet, B., Kenet, R. and Tearney, G. (1998), “Digital Optical Visualization, Enhancement, Quantification, and Classification of Surface and Subsurface Features of Body Surfaces”, *U.S. Patent No. 5,836,872*.

Koo, V., Chan, Y. and Gobi, V. (2012), “A New Unmanned Aerial Vehicle Synthetic Aperture Radar for Environmental Monitoring”, *Progress In Electromagnetics Research*, Vol. 122, pp. 245–268.

Kovac, J., Peer, P. and Solina, F. (2003), “Human Skin Color Clustering for Face Detection”, *IEEE*, Vol. 2, pp. 144–148.

Kulp, T. (1997), “Remote Imaging of Controlled Gas Releases Using Active and Passive Infrared Imaging Systems”, *AeroSense’97, International Society for Optics and Photonics*, pp. 269–278.

Kumar, A. (2008), “Computer-Vision-Based Fabric Defect Detection: A Survey”, *Industrial Electronics, IEEE Transactions on*, Vol. 55 No. 1, pp. 348–363.

Low, C. and Wang, D. (2008), “GPS-Based Path Following Control for a Car-Like Wheeled Mobile Robot with Skidding and Slipping”, *Control Systems Technology, IEEE Transactions on*, Vol. 16 No. 2, pp. 340–347.

Mahamuni, N.N. and Adewuyi, Y.G. (2009), “Fourier Transform Infrared Spectroscopy (FTIR) Method to Monitor Soy Biodiesel and Soybean Oil in Transesterification Reactions, Petrodiesel−Biodiesel Blends, and Blend Adulteration with Soy Oil”, *Energy & Fuels*, Vol. 23 No. 7, pp. 3773–3782.

Marinelli, W. and Green, B. (1996), “Infrared Imaging Volatile Organic Carbon Field Sensor”, *Optical Remote Sensing for Environmental and Process Monitoring*, Vol. Vol. 2883, p. 245.

McRae, T. and Kulp, T. (1993), “Backscatter Absorption Gas Imaging: A New Technique for Gas Visualization”, *Applied Optics*, Vol. 32 No. 21, pp. 4037–4050.

McStay, D., Kerlin, J. and Acheson, R. (2007), “An Optical Sensor for the Detection of Leaks from Subsea Pipelines and Risers”, *Journal of Physics: Conference Series*, Vol. 76 No. 1, p. 012009.

Moghimi, S. (2009), *Waypoint Naviation System Design for UAV*, Cranfield

University, MSc Thesis.

Mondragón, I. and Campoy, P. (2010), “3D Pose Estimation Based on Planar Object Tracking for UAVs Control”, *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pp. 35–41.

Mondragon, I.I.F., Campoy, P., Correa, J.F. and Mejias, L. (2007), “Visual Model Feature Tracking for UAV Control”, *Intelligent Signal Processing, 2007. WISP 2007. IEEE International Symposium on*, pp. 1–6.

Monteiro, G., Peixoto, P. and Nunes, U. (2006), “Vision-Based Pedestrian Detection Using Haar-Like Features”, *Robotica 24 (2006)*, pp. 46–50.

Multi-Rotor Technology (MRT). (2012), “Gaui 500x Plus Crane (II) Aerial Platform”, available at: http://www.gauimrt.com, (accessed 15 October 2013).

Murvay, P.-S. and Silea, I. (2012), “A Survey on Gas Leak Detection and Localization Techniques”, *Journal of Loss Prevention in the Process Industries*, Vol. 25 No. 6, pp. 966–973.

Nagai, M., Chen, T., Kanade, T., Amidi, O., Ke, Q., Navarro-Serment, L., Collins, R., et al. (2009), “UAV-Borne 3-D Mapping System by Multisensor Integration”, *Geoscience and Remote Sensing, IEEE Transactions on*, Vol. 47 No. 3, pp. 701–708.

NASA tutorial on remote sensing. (2011), “Platforms used by Remote Sensors”, available at: http://rst.gsfc.nasa.gov/Intro/Part2_1x.html, (accessed 1 January 2012).

Navarro-Serment, L. (2010), “Pedestrian Detection and Tracking Using Three-Dimensional Ladar Data”, *The International Journal of Robotics Research*.

Nevatia, R. and Ramesh Babu, K. (1980), “Linear Feature Extraction and Description”, *Computer Graphics and Image Processing*, Vol. 13 No. 3, pp. 257–269.

Nistér, D. (2005), “Preemptive RANSAC for Live Structure and Motion Estimation”, *Machine Vision and Applications*, Vol. 16 No. 5, pp. 321–329.

Oil & Gas Pipeline. (2011), “Images were produced from the following sources”, available at: https://upload.wikimedia.org/wikipedia/commons/8/8f/Trans-Alaska_Pipeline_System_Luca_Galuzzi_2005.jpg; https://upload.wikimedia.org/wikipedia/commons/b/b0/Alaska_Pipeline_and_caribou.jpg; https://upload.wikimedia.org/wikipedia/commons/6/6a/Trans_Alaska_oil_pipeline_crossing_South_fork_Koyukuk_River.jpg;http://3.bp.blogspot.com/-3uXqYvz1-00/T966FMEWpcI/AAAAAAAAE-o/EcL22Ffp6nI/s1600/Pipeline.jpg; http://www.hickerphoto.com/images/1024/pipeline_snow_t2192.jpg; http://www.hickerphoto.com/images/500/alaska_oil_pipeline_t1851.jpg;

http://www.spectraenergy.com/content/inline-images/operations/New_projects_pipeline.jpg;http://petesortwell.co.uk/wp-content/uploads/2014/01/22128.jpg; http://www.calumetpipeline.net/d/img/bg/01.jpg; http://www.gregsadventure.com/wp-content/uploads/2011/08/Alaska-pipeline.jpg; https://upload.wikimedia.org/wikipedia/commons/b/ba/CSIRO_ScienceImage_6380_Section_of_the_Perth_Kalgoorlie_water_supply_pipeline_near_Merredin_WA_1976.jpg; http://www.energy-solutions.com/wp-content/uploads/2014/07/blue-sky-white-pipeline-300x200.jpg; http://yellowairplane.com/Global_Warming/images/50-Beautiful_Fall_Colors.jpg; http://albawabacdn.albawabamiddleea.netdna-cdn.com/sites/default/files/imagecache/article_headline_node_big//sites/default/files/im/misc/pipeline_yemen1.jpg; http://www.betterworldclub.com/static/media/uploads/keystone_pipeline.jpg ; https://www.cs.uaf.edu/2011/spring/cs641/proj1/rltorgerson/pipeline.jpg; http://gs-press.com.au/utils/zimage.php?width=1200&height=800&image=images/news_articles/TAPS_pipeline_1.jpg, (accessed 1 January 2011).

Opencv dev team. (2012), “Open Source Computer Vision Library”, available at: http://docs.opencv.org, (accessed 1 January 2012).

OpenNI. (2012), “The Standard framework for 3D Sensing”, available at: http://openni.ru, (accessed 10 July 2012).

Owechko, Y., Medasani, S. and Korah, T. (2010), “Automatic Recognition of Diverse 3-D Objects and Analysis of Large Urban Scenes Using Ground and Aerial LIDAR Sensors”, *Conference on Lasers and Electro-Optics*, pp. 1–2.

Pal, M. and Mather, P. (2001), “Decision Tree Based Classification of Remotely Sensed Data”, *Paper Presented at the 22nd Asian Conference on Remote Sensing*, Vol. 5, p. 9.

Palmer A. (2002), *Appraisal of Pipeline Surveillance by High Resolution Satellite*, Vol. I.

Palmer, R., Hopkins, P., Fraser, A., Dezobry, J. and Merrienboer, H. (2004), “Evaluation of Satellite Technology for Pipeline Route Surveillance and the Prevention of Third Party Interference Damage”, *Journal of Pipeline Integrity*, Vol. 3 No. 1, pp. 23–54.

Paquis, S., Legeay, V. and Konik, H. (2000), “Road Surface Textures Classification Using Opening-Based Image Processing”, *International Archives of Photogrammetry and Remote Sensing*, Vol. 33 No. B3/2, PART 3, pp. 685–691.

Park, H.W., Sohn, H., Law, K.H. and Farrar, C.R. (2007), “Time Reversal Active Sensing for Health Monitoring of a Composite Plate”, *Journal of Sound and*

Vibration, Vol. 302 No. 1, pp. 50–66.

Piciarelli, C., Micheloni, C. and Martinel, N. (2013), "Outdoor Environment Monitoring with Unmanned Aerial Vehicles", *Image Analysis and Processing–ICIAP 2013*, pp. 279–287.

Pipeline Surveillance Company - Not for Public Release. (2012), *Aerail Footage of a Lengthy Pipeline Structure*.

Pipeline's Third-Party Interference. (2011), "Images were produced from the following sources", available at: http://www.cwplant.co.uk/wp-content/uploads/2013/01/JS220_Studio_image_front211.jpg; http://www.jcb.co.uk/~/asset/4/8889.ashx; http://senoksl.com/wp-content/uploads/JS205LC-Low2.jpg; http://modelbarn.co.nz/images/JCB-435-Joal243.jpg, (accessed 1 January 2012).

Prasad, N. and Geiger, A. (1996), "Remote Sensing of Propane and Methane by Means of a Differential Absorption Lidar by Topographic Reflection", *Optical Engineering*, Vol. 35 No. 4, pp. 1105–1111.

Rahman, R.M. and Afroz, F. (2013), *Comparison of Various Classification Techniques Using Different Data Mining Tools for Diabetes Diagnosis*, Vol. 2013.

Randell, C. (2010), *Operational Pipeline Integrity Monitoring Demonstration Using RADARSAT*.

Rathinam, S., Kim, Z. and Sengupta, R. (2008), "Vision-Based Monitoring of Locally Linear Structures Using an Unmanned Aerial Vehicle 1", *Journal of Infrastructure Systems*, Vol. 14 No. 1, pp. 52–63.

Rathinam, S., Kim, Z.K.Z., Soghikian, A. and Sengupta, R. (2005), "Vision Based Following of Locally Linear Structures Using an Unmanned Aerial Vehicle", *Journal of Infrastructure Systems*, Vol. 14 No. 1, pp. 52–63.

Revol-Muller, C. and Peyrin, F. (2000), "Automated 3D Region Growing Algorithm Governed by an Evaluation Function", *Image Processing, 2000. Proceedings. 2000 International Conference on*, Vol. 3, pp. 440–443.

Roper, W. and Dutta, S. (2005), "Remote Sensing and GIS Applications for Pipeline Security Assessment", *2005 ESRI User Conference Proceedings*, Vol. 15.

Roper, E. W., and Dutta, S. (2006), "Oil Spill and Pipeline Condition Assessment Using Remote Sensing and Data Visualization Management Systems", *Freshwater Spills Symposium, Natural Disasters, Human Error, and Equipment Failure-Causes for Major Inland Oil Spills and the Resulting Multifaceted Response*.

Saban, D. (2006), *Flight Control System Design for the Demon Vehicle*, Cranfield University, PhD Thesis.

San Martin Texmelucan. (2010), “Mexico Oil Pipeline Explosion, Fire”, *Dallas News*, available at: http://www.dallasnews.com/sharedcontent/dws/news/world/mexico/stories/DN-mexico_20int.ART.State.Edition1.7ce5c2.html, (accessed 1 January 2011).

Schindler, G., Krishnamurthy, P. and Dellaert, F. (2006), “Line-Based Structure from Motion for Urban Environments”, *3D Data Processing, Visualization, and Transmission, Third International Symposium on*, pp. 846–853.

Schowengerdt, R.A. (2006), *Remote Sensing: Models and Methods for Image Processing*, *Academic Press*, Academic Press.

Shaochuang, L., Hui, S., Maosheng, X., Hongjian, Y., Tong, L. and Shunkai, L. (1999), “Positioning Accuracy of Airborne Laser Ranging and Multispectral Imaging Mapping System”, *Journal of Wuhan Technical University of Surveying and Mapping (WTUSM)*, Vol. 2, pp. 341–344.

Sivathanu, Y. and Gore, J. (1991), “Simultaneous Multiline Emission Absorption Measurements in Optically Thick Turbulent Flames”, *Combustion Science and Technology*, Vol. 80 No. 1-3, pp. 1–21.

Sivathanu, Y., Gore, J. and Dolinar, J. (1991), “Transient Scalar Properties of Strongly Radiating Jet Flames”, *Combustion Science and Technology*, Vol. 76 No. 1-3, pp. 45–66.

Smith, B. and Laubscher, B. (1999), “IRISHS, the Infrared Imaging Spatial Heterodyne Spectrometer: a New Pushbroom Fourier Transform Ultraspectral Imager with no Moving Parts”, *AeroSense’99, International Society for Optics and Photonics*, pp. 501–509.

Sohn, S., Lee, B., Kim, J. and Kee, C. (2008), “Vision-Based Real-Time Target Localization for Single-Antenna GPS-Guided UAV”, *Aerospace and Electronic Systems, IEEE Transactions on*, Vol. 44 No. 4, pp. 1391–1401.

Spoonhower, J.P., Paz-Pujalt, G.R., Sherin, J.M., Graen, M.G. and Stearns, S. V. (2006), “Detecting Natural Gas Pipeline Failures”, *U.S. Patent No. 7,027,924*.

Stevens, B.L. and Lewis, F.L. (1992), *Aircraft Control and Simulation*, Wiley.

Tao, C. and Hu, Y. (2002), “Assessment of Airborne Lidar and Imaging Technology for Pipeline Mapping and Safety Applications”, *Integrated Remote Sensing at the Global, Regional and Local Scale, ISPRS Commission I Mid-Term Symposium in Conjunction with Pecora 15/land Satellite Information IV Conference Proceedings*.

Tarsha-Kurdi, F. (2007), “Hough-Transform and Extended RANSAC Algorithms for Automatic Detection of 3d Building Roof Planes from Lidar Data”, *ISPRS Workshop on Laser Scanning 2007 and SilviLaser 2007*, Vol. 36, pp. 407–412.

The Business of Photonics. (2010), “Hyperspectral Touted for Pipeline Inspection”, available at: http://optics.org/news/1/4/24, (accessed 1 November 2012).

Theoharis, T., Papaioannou, G., Platis, N. and Patrikalakis, N.M. (2008), *Graphics and Visualization: Principles & Algorithms*, CRC Press.

Toshihiro, M. (2010), “Trends in Wireless Access Technologies toward Expansion of Broadband and Ubiquitous Services”, *NTT Tsukuba Forum 2010 Workshop Lectures*, Vol. 9 No. 5.

Tsai, Y., Chang, Y. and Chen, L. (2006), “Block-Based Vanishing Line and Vanishing Point Detection for 3D Scene Reconstruction”, *Intelligent Signal Processing and Communications, 2006. ISPACS’06. International Symposium on*, pp. 586–589.

U.S. Energy Information Administration (eia). (2012), *The Natural Gas Pipeline Network*, available at: http://www.eia.gov/countries/cab.cfm?fips=SA, accessed 1 November 2012).

Viola, P. and Jones, M. (2004), “Robust Real-Time Face Detection”, *International Journal of Computer Vision*.

Vosselman, G. and Gorte, B. (2004), “Recognising Structure in Laser Scanner Point Clouds”, *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, Vol. 46, pp. 33–38.

Wacker, A. and Landgrebe, D. (1972), *Minimum Distance Classification in Remote Sensing*, *LARS Technical Reports*, Vol. 25.

Wang, Y., Shen, D. and Teoh, E.K. (2000), “Lane Detection Using Spline Model”, *Pattern Recognition Letters*, Vol. 21 No. 8, pp. 677–689.

Wang, Y., Teoh, E. and Shen, D. (2004), “Lane Detection and Tracking Using B-Snake”, *Image and Vision Computing*, Vol. 22 No. 4, pp. 269–280.

Weil, G. (1993), “Non Contact, Remote Sensing of Buried Water Pipeline Leaks Using Infrared Thermography”, *Water Management in the’90s@ sA Time for Innovation*, pp. 404–407.

Welch, C. (2010), “Trans-Alaska Pipeline”, *Alyeska Pipeline Service Company, State of Alaska, U.S. Department of the Interior Bureau of Land Management*, available at: http://www.earthlyissues.com/transalaska.htm, (accessed 1 September 2012).

Witayangkurn, A., Nagai, M., Honda, K., Dailey, M. and Shibasaki, R. (2011), “Real-Time Monitoring System Using Unmanned Aerial Vehicle Integrated With Sensor Observation Service”, *Conference on Unmanned Aerial Vehicle in Geomatics*.

Young, W.R. (1982), *The Helicopters*, Time Life Education, Chicago.

Zhang, Q., Wang, J., Liu, B., Cai, T., Qiao, L. and Zhang, Y. (2009), “Gas Pipeline Leak Detection Based on Tunable Diode Laser Absorption Spectroscopy”, *Spectroscopy and Spectral Analysis*, Vol. 29 No. 8, pp. 2017–2020.

Zongjian, L. (2008), “UAV for Mapping-Low Altitude Photogrammetric Survey”, *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, Vol. XXXVII No. Part B, 1, pp. 1183–1186.

.

APPENDICES

A. Samples of methods coded in Python

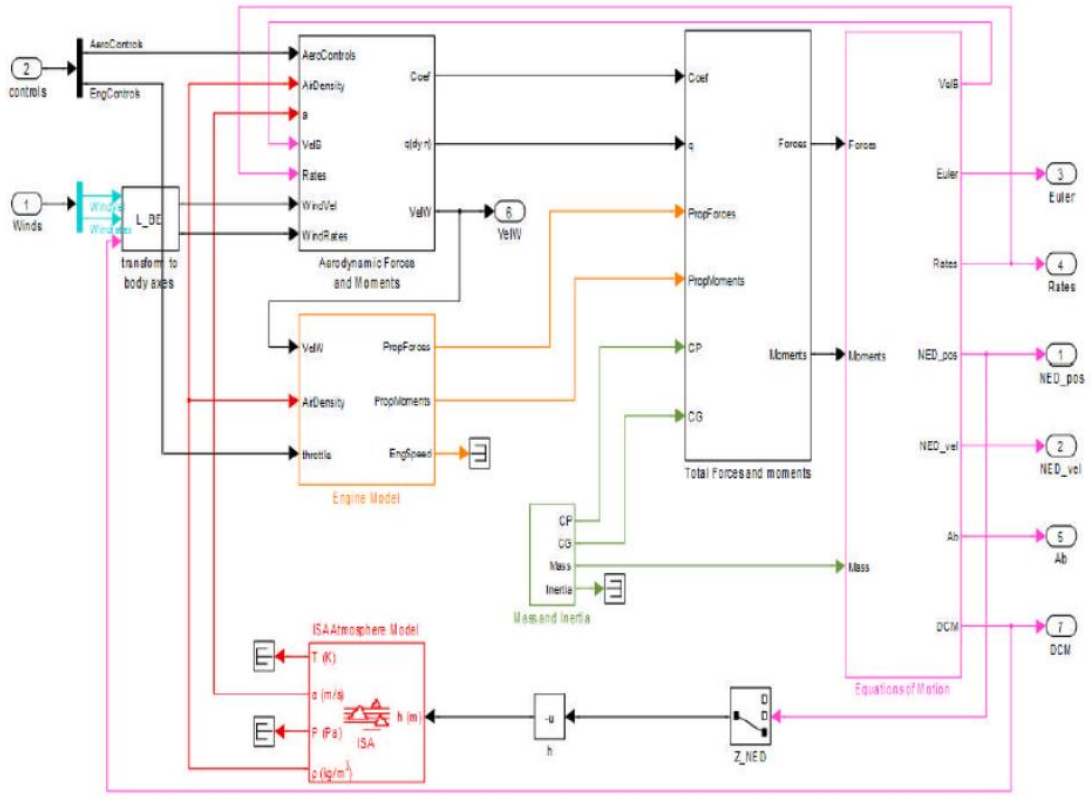
Code of Canny Edge Detector Method

```
def edge_depth(IR_FG, depth_image, thrs1,thrs2):
    q = depth_image
    u, v = np.mgrid[:q.shape[0], :q.shape[1]]
    IR_FG_uint8 = IR_FG.astype('uint8')
    IR_edges = cv2.Canny(IR_FG_uint8,thrs1,thrs2 )
    IR_edges=cv2.bitwise_and(depth_image,depth_image,mask=IR_edges)
    IR_edges = IR_edges.astype(np.uint16)
    dd = 1
    xyz_edges,uv_edges,uvd_edges=calibkinect.depth2xyzuv(IR_edges[::dd, ::dd], u,v)
    return xyz_edges, uv_edges, uvd_edges
```

Code of Hough Transform Method

```
def Line(depth_image, edges,PointsInLine, minLineLength,maxLineGap, linesNo):
    plinesd = cv2.HoughLinesP(edges.astype('uint8'), 1, np.pi/180, PointsInLine, np.array([]), minLineLength,maxLineGap)[0]
    print plinesd
    q = edges
    dd = 1
    u, v = np.mgrid[:q.shape[0]:dd, :q.shape[1]:dd]
    HIGHT, WIDTH = edges.shape
    imgHL1 = np.zeros((HIGHT, WIDTH))
    imgHL2 = np.zeros((HIGHT, WIDTH))
    HT = []
    for ee in range(len(plinesd)):#[:linesNo]:
        HTp = plinesd
        P1Y = HTp[:,0]
        P1X = HTp[:,1]
        P2Y = HTp[:,2]
        P2X = HTp[:,3]
        imgHL1[P1X,P1Y]= depth_image[P1X,P1Y]
        imgHL2[P1X,P1Y]= depth_image[P2X,P2Y]
        xyz_HT_P1, uv_HT_P1, uvd_HT_P1= calibkinect.depth2xyzuv(imgHL1[::1, ::1], u, v)
        xyz_HT_P2, uv_HT_P2 , uvd_HT_P2= calibkinect.depth2xyzuv(imgHL2[::1, ::1], u, v)
        HT_data= np.hstack((xyz_HT_P1,uv_HT_P1, uvd_HT_P1,xyz_HT_P2, uv_HT_P2, uvd_HT_P2))
    HT = np.array(HT_data)
    return HT, plinesd
```

B. Simulink Model of 6DOF dynamics of Piper Cub



C. PID controller's gains and saturation limits of SAS's of the Piper Cub UAV.

Type	PID Gains			Saturation Limit (radian)
	P	I	D	
Pitch	0.1	0	0.05	-0.13 to +0.15
Roll	1.5	0	0	-0.157 to +0.157
	1.2	0	0	
Yaw	2	0	0	-0.35 to +0.35