A Perception Payload for Small-UAS Navigation in Structured Environments

A thesis presented to

the faculty of

the Russ College of Engineering of Ohio University

In partial fulfillment

of the requirements for the degree

Master of Science

Akshay S. Bharadwaj

August 2018

© 2018 Akshay S. Bharadwaj. All Rights Reserved.

This thesis titled

A Perception Payload for Small-UAS Navigation in Structured Environments

by

AKSHAY S. BHARADWAJ

has been approved for

the Department of Electrical Engineering and Computer Science

and the Russ College of Engineering by

Maarten Uijt de Haag

Professor of Electrical Engineering and Computer Science

Dennis Irwin

Dean, Engineering and Technology

Abstract

BHARADWAJ, AKSHAY S., M.S., August 2018, Electrical Engineering

A Perception Payload for Small-UAS Navigation in Structured Environments (125 pp.)

Director of Thesis: Maarten Uijt de Haag

Unmanned Aircraft System (UAS) are proving to be increasingly favorable in military and commercial applications. The range of applications include surveillance, aerial photography, environmental observations, search and rescue, mapping, forestry, agricultural survey, law enforcement among many. The small size unmanned multi-copters are highly capable and cost effective for low altitude operations and have extended access to dangerous and hazardous environments which were previously unavailable. Irrespective of the applications, a position and navigation solution are necessary to fly the UAS completely autonomous or even to manually control it easily. The Global Navigation Satellite System (GNSS) has become one of the most dependable solution for position and navigation outdoors but does not perform well in the indoor environment as the signal is obstructed by the roof and the walls. Hence, there is a need for non-Global Positioning System (GPS) position and navigation solution methods for indoors. Simultaneous Localization and Mapping (SLAM) and feature-based integrated navigation are two methods that can be used for this purpose, using various types of sensors like ranging sensors, cameras, and Inertial Measurement Unit (IMU). This thesis will focus on integrating depth imagery, Short Wave Infrared (SWIR) imagery and Long Wave Infrared (LWIR) imagery with an IMU to obtain and estimate of both the position and the map of the environment. In this discussion, the region of operation is restricted to structured environments and would be extended to unstructured environments in the future. This work will include preliminary flight test results from a small-size Blackout quadcopter operated in a structured indoor environment for maintenance purposes. The quadcopter has been equipped with a 3DR Pixhawk flight controller and an Odroid XU4 onboard computer running Ubuntu. The

Robotics Operating System (ROS) is used to interface with and integrate all the sensors and control the flight controller. The optical sensors include a forward-pointing Occipital providing both SWIR imagery and depth information, and a Lepton LWIR camera to pick up heat signature image of objects in the environment. The latter is connected to the Odroid via a Serial Peripheral Interface (SPI).

To,

My Mother, who has always supported me through every endeavor of my life...

ACKNOWLEDGMENTS

It has taken couple of years' worth of hard work for the completion of this work. Throughout my master's, few people have stood by me and given their love, support and encouragement, without which this work would have not been possible.

First and foremost, I want to thank my advisor Dr. Maarten Uijt de Haag without whom this work would have been impossible. I can say without any doubt that he is one of the best professors I have worked under. I thank him for all the support and knowledge he has given over the many years I have known him. I couldn't have asked for a better advisor.

I thank my lab mates Adam Shultz and Joel Huff, for helping me with data collection in the dusty maintenance tunnels. Avionics Engineering Center for partly funding my research. Matthew Riley and Joel Baetens from Ohio University utilities, for giving access to the maintenance tunnels and escorting us down there whenever we needed access to it. Scott Miller from Russ College of Engineering and Technology for his support.

Lastly, I would like to thank my friends and family for being there when I needed them.

Table of Contents

Page

Abstract 3

Dedication 5

Acknowledgments 6

List of Tables 9

List of Figures 10

List of Acronyms 12

1 Introduction 15

2 Background 19
2.1 Inertial Navigation Systems 19
2.2 Optical Navigation Systems 21
2.2.1 Vision-based Navigation Systems 21
2.3 3D Imagers 23
2.4 Sensor Integration Tools: Kalman Filtering 25
2.5 Simultaneous Localization and Mapping (SLAM) 27
2.6 SLAM Using 3D Imaging Sensors 29

3 Navigation Methodology 34

4 Sensors and UAS Hardware Setup 38
4.1 Sensors 38
4.1.1 Occipital Structure SWIR and depth sensor 38
4.1.2 FLIR Lepton LWIR camera 39
4.1.3 Xsens IMU 43
4.2 System Setup 44
4.2.1 Unmanned Aerial Vehicle (UAV) frame and flight controller 44
4.2.2 On-board system 45
4.3 Flight Environment 49

5 Sensor Calibration 51
5.1 Pinhole Camera Calibration (Structure sensor and Lepton Camera) 51
5.1.1 Procedure for camera calibration: Structure sensor 55
5.1.2 Procedure for camera calibration: FLIR Lepton camera 56

5.2 Image Registration and Colored Point Cloud Generation 57
5.3 Calculation of IMU and Point Cloud Delay 59

6 Sensor Pre-processing 63
6.1 Point Cloud Segmentation 63
6.2 LWIR Image Segmentation 66

7 Feature Extraction and Navigation Approach 69
7.1 Feature Extraction 69
7.1.1 Random Sample Consensus 70
7.1.2 Extraction of planar surface in x- and y-axis 72
7.1.3 Extraction of planar surface in z-axis 73
7.2 Feature Integration and Navigation 75

8 Results 78
8.1 Calibration Results 78
8.1.1 Estimation of lens distortion 78
8.1.2 Estimation of relative pose of cameras 84
8.1.3 Estimation of delay between IMU and point cloud messages 92
8.2 LWIR Image Segmentation 98
8.3 Point Cloud Segmentation and Feature Extraction 100

9 Summary and Conclusions 112

10 Future Work 114

References 116

List of Tables

Table Page

1.1 Availability of GNSS w.r.t the operational environment [1] 17

4.1 Hardware specification of Occipital camera 40
4.2 Hardware specification of Lepton camera 41

8.1 Camera intrinsic and extrinsic calibration 81
8.2 Rotation and translation of structure camera w.r.t thermal camera 87
8.3 Timing calibration results 92
8.4 Pre-processing parameters 101
8.5 Segmentation parameters 102

List of Figures

Figure Page

1.1 Maintenance tunnel environment (steam tunnels) . . . 16

2.1 Inertial Navigation System (INS) reference frames . . . 20
2.2 Computational sequence for INS [2] . . . 21
2.3 Block diagram of general steps in visual odometry . . . 22
2.4 Graph and constraints for GraphSLAM . . . 28

3.1 High-level functional block diagram of the hierarchical guidance, navigation and control function . . . 34
3.2 Functional block diagram of the proposed navigation and mapping method . . . 36

4.1 Occipital structure 3D depth sensor . . . 39
4.2 FLIR Lepton LWIR Camera . . . 40
4.3 Block diagram of Lepton camera . . . 41
4.4 Lepton thermal image processing . . . 42
4.5 Xsens MTi 1-s IMU development board [3] . . . 43
4.6 Frame and the flight controller system used on the platform . . . 44
4.7 On-board embedded system: Odroid XU4 . . . 46
4.8 Platform for data collection with sensor payload . . . 46
4.9 Camera mount with depth and thermal camera . . . 47
4.10 Power distribution block diagram . . . 48
4.11 Sensor interface diagram . . . 48
4.12 Flight test environment . . . 50

5.1 Pinhole camera geometry [4] . . . 52
5.2 Lens distortion [4] . . . 53
5.3 Distortion caused due to misalignment of sensor [5] . . . 54
5.4 Calibration patterns . . . 56
5.5 Timing diagram depicting the sequence of messages and associated timing delays . . . 60
5.6 Setup for extraction of planar surface to calculate the time delay between point clouds and IMU . . . 60

6.1 Block diagram describing the point cloud segmentation . . . 63
6.2 Example image from Lepton LWIR camera . . . 66
6.3 Functional block diagram of LWIR feature extraction method . . . 67

7.1 Example of point cloud at n^{th} time epoch . . . 69
7.3 Spatial division of xy segmented point cloud . . . 74

7.4 Planar surface-based navigation geometry 76
7.5 Point-based navigation geometry 77

8.1 Calibration pattern detected by Occipital 3D camera in different orientations . . 79
8.2 Calibration pattern detected by Lepton LWIR camera in different orientations . 80
8.3 Image from Occipital camera before and after rectification 82
8.4 Image from LWIR before and after rectification 83
8.5 Detection of same asymmetrical circle grid pattern by both cameras 85
8.6 Detection of same checker board pattern by both cameras 86
8.7 Scene 1: Range image before and after rotation and translation 88
8.8 Scene 1: Image scene of a lab with few thermal features 89
8.9 Scene 2: Range image before and after rotation and translation 90
8.10 Scene 2: Image scene with significant depth and thermal signature 91
8.11 Estimation of latency between IMU and point cloud messages: Run 1 93
8.12 Estimation of latency between IMU and point cloud messages: Run 2 94
8.13 Estimation of latency between IMU and point cloud messages: Run 3 95
8.14 Estimation of latency between IMU and point cloud messages: Run 4 96
8.15 Estimation of latency between IMU and point cloud messages: Run 5 97
8.16 Image segmentation and feature extraction: Time epoch 1 98
8.17 Image segmentation and feature extraction: Time epoch 2 99
8.18 Image segmentation and feature extraction: Time epoch 3 100
8.19 Legend for points and planes in point cloud segmentation and feature extraction 102
8.20 Point cloud segmentation and extraction: Stocker 2nd floor, time epoch 1 . . . 103
8.21 Point cloud segmentation and extraction: Stocker 2nd floor, time epoch 2 . . . 104
8.22 Point cloud segmentation and extraction: Stocker 2nd floor, time epoch 3 . . . 105
8.23 Point cloud segmentation: Maintenance tunnels, time epoch 1 106
8.24 Point cloud segmentation and extraction: Maintenance tunnels, time epoch 1 . . 106
8.25 Point cloud segmentation and extraction: Maintenance tunnels, time epoch 2 . . 107
8.26 Point cloud segmentation and extraction: Maintenance tunnels, time epoch 3 . . 108
8.27 Point cloud segmentation and extraction: Stocker 3rd floor, time epoch 1 . . . 109
8.28 Point cloud segmentation and extraction: Stocker 3rd floor, time epoch 2 110
8.29 Point cloud segmentation and extraction: Stocker 3rd floor, time epoch 3 111

List of Acronyms

3D	Three-dimensional
AHC	Agglomerative Hierarchical Clustering
AI	Artificial Intelligence
ARM	Advanced RISC Machines
BA	Bundle Adjustment
b-frame	Body Frame
CAN	Control Area Network
CKF	Complementary Kalman Filter
DoF	Degrees of Freedom
EKF	Extended Kalman Filter
ENU	East North Up
FAST	Features from Accelerated Segment Test
FFC	Flat Field Correction
FPV	First Person View
GNSS	Global Navigation Satellite System
GNC	Guidance Navigation and Control
GPIO	General Purpose Input Output
GPS	Global Positioning System
HMP	Heterogeneous Multi-Processor
I^2C	Inter-Integrated Circuit
ICP	Iterative Closest Point
IMU	Inertial Measurement Unit
INS	Inertial Navigation System
IR	Infrared
I/O	Input or Output

KF	Kalman Filter
LiDAR	Light Detection and Ranging
LWIR	Long Wave Infrared
MEMS	Micro-machined Electromechanical System
ML	Machine Learning
n-frame	Navigation Frame
ORB	Oriented FAST and Rotated BRIEF
OS	Operating System
OU	Ohio University
PC	Personal Computer
PCA	Principle Component Analysis
PTAM	Parallel Tracking and Mapping
PWM	Pulse Width Modulation
RAM	Random Access Memory
RANSAC	Random Sample Consensus
RGB	Red Green Blue
RGB-D	RGB-Depth
RMS	Root Mean Square
ROS	Robotics Operating System
RTOS	Real Time Operating System
SIFT	Scale-Invariant Feature Transform
SLAM	Simultaneous Localization and Mapping
SPI	Serial Peripheral Interface
SSH	Secure Shell
sUAS	Small-size Unmanned Aircraft System
SURF	Speeded-Up Robust Features

SVD	Singular Value Decomposition
SWIR	Short Wave Infrared
TOF	Time of Flight
UART	Universal Asynchronous Receiver/Transmitter
UAS	Unmanned Aircraft System
UAV	Unmanned Aerial Vehicle
UKF	Unscented Kalman Filter
VGA	Video Graphic Array
VO	Visual Odometry
VoSPI	Video over SPI

1 INTRODUCTION

Robotic vehicles have a vast number of applications in our everyday lives. Commercially, they are used, for example, to transport people (e.g., semi-autonomous cars and trains), or to move boxes in warehouses. In military, they can be used to operate in dangerous and hazardous conditions for applications such as bomb disposal instead of humans. UAS are an extension of robotic vehicles which can provide access to hazardous, dangerous and previously inaccessible areas by people or rovers with the added advantage that they can fly through these environments unhindered by the floor surface.

UAS are proving to be increasingly favorable in military and commercial applications with Small-size Unmanned Aircraft System (sUAS) being more capable and cost effective for low altitude operations. The range of applications include surveillance, aerial photography, search and rescue, mapping, maintenance, survey among many.

In this research, we focus on the application of mapping and maintenance tasks in the maintenance tunnels under Ohio University (OU) (Figure 1.1). These tunnels host the network of pipes that supply the cold and hot air for air conditioning and heating across the OU campus. The many miles of OU maintenance tunnels vary in size from being inaccessible to humans to multiple meters in height. They may be swampy and inaccessible due to water, steam leaks and infrequent maintenance. Currently, the tunnels are mapped manually by surveyors, a time-consuming (estimated from 6 months to 9 months) and highly expensive process. The maintenance inspection is also manual and usually at longer intervals.

The use of UAS for mapping and maintenance support in the tunnels forms a good alternative for the current approach. UAS can travel faster than a person, and hence can map the network of tunnels in far lesser time than manual surveying. Due to the small size of UAS, it provides access to the previously unavailable or hard to access parts of the tunnel. It would be easier and less expensive to setup regular maintenance schedules using



Figure 1.1: Maintenance tunnel environment (steam tunnels)

autonomous or assisted remotely controlled UAS. OU tunnels are only a limited case of this research. The use of UAS for mapping can be applied to any environment that must be explored such as buildings, mines or caves.

For every aforementioned application, a position and navigation solution are a necessity to enable autonomous UAS flight, assist manual control, or help in geo-referencing the mapping sensors. In addition to a position solution, an mapping sensor is also needed for these applications to map and inspect the environment and for obstacle detection. GNSS has become one of the most dependable solutions for position and navigation outdoors, but its performance depends on the environment it is used in as shown in table 1.1. Table 1.1 shows the availability of GNSS in various operational environments. While operating indoors, the received signals are either not available or severely degraded due to **NLOS!** (**NLOS!**) signals. Since the target environment in this research for the operation of UAS is indoors, there is a need for non-GNSS position and navigation solution

for indoor environment. Lot of research has been done to find alternative means of positioning using cameras, lasers scanners and signals-of-opportunity.

Table 1.1: Availability of GNSS w.r.t the operational environment [1]

Environment	**Ground Vehicles**	**Aerial Vehicles**	**GNSS**
National airspace		X	Available
Rural (open-sky)	X	X	Available
Rural (foliage)	X	X	Challenged
Suburban	X	X	Available
Urban	X	X	Challenged
Indoor	X	X	Not available

Although inertial navigation is a possible alternative for indoor navigation, inertial navigation involves the integration of its gyro and accelerometer measurements over time, resulting in a position drift due to noise and bias errors in the measurements. High-quality IMU sensors such as navigation grade inertial navigation systems (IMUs) are too expensive and typically too heavy for sUAS applications. Alternatively, visual- and laser-based have their own limitations. The limitations of inertial and visual navigation are explained in chapter 2.

SLAM and feature-based integrated navigation are two approaches that can be applied to enable navigation in our target indoor environments using various types of sensors like ranging sensors, cameras, and IMU. This research focuses on integrating depth imagery and low-resolution LWIR imagery with an IMU to obtain an estimate of both the position and the map of the environment.

The selection of the sensors is targeted to mapping and maintenance in the tunnels. A Three-dimensional (3D) camera is used to obtain depth information of the environment (i.e., a point cloud). An LWIR camera is used to pick up heat signatures from various features in the tunnels such as hot pipes, poles, cold pipes, and support beams. The LWIR thermal camera is essential in this application, as it allows the operator or computer to detect steam and water leaks. Based on the intensity of the object in the image, the temperature of the object can be measured. A low cost IMU is used on the platform to integrate inertial measurement into the SLAM algorithm.

This thesis describes (i) the construction of the UAS platform with the above-mentioned sensors, (ii) the algorithms used for navigating UAS in tunnels, (iii) and the software implementation of the algorithms. Chapter 2 discusses the general background of existing position capabilities and their drawbacks. Chapter 3 introduces the reader to the overall proposed navigation methodology. Chapter 4 discusses the construction of the research platform along with the on-board sensors and software setup. Chapter 5 explains the math and implementation of camera calibration, which is needed to use the cameras for navigation algorithm. Chapter 6 explains the pre-processing algorithms used on 3D point clouds and LWIR imagery. Chapter 7 explains the feature extraction and navigation algorithm used. The results of camera calibration, point cloud segmentation and feature extraction are presented in chapter 8.

2 Background

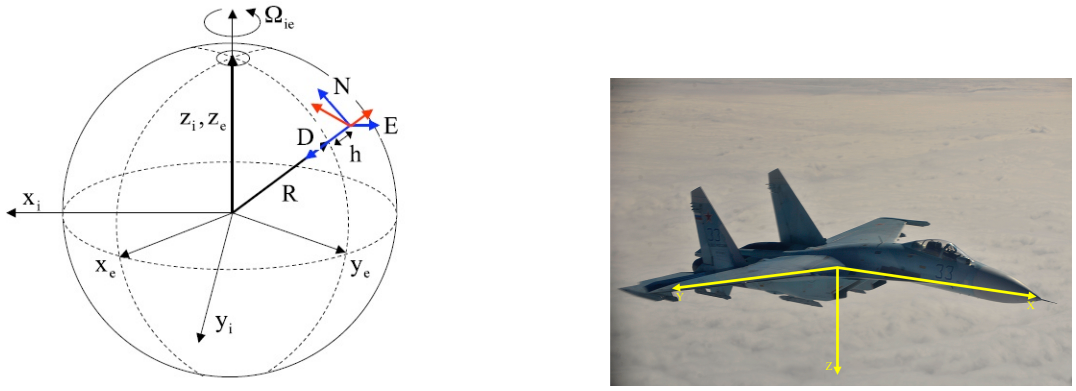
This chapter introduces the basics of the various subsystems and algorithms needed for the proposed sUAS navigation and sensing system.

2.1 Inertial Navigation Systems

The INS is a dead-reckoning navigation system that consists of an IMU and algorithms that take the IMU outputs and compute the aircraft's attitude (ϕ, θ, ψ), velocity ($\mathbf{v}^n$) and position ($\mathbf{r}^n$). The IMU is made up of two individual sensors: a triad of accelerometers that provides measurements of specific forces ($\mathbf{f}^b$) or delta-v's ($\mathbf{f}^b$ integrated over a short period of time), and a triad gyroscopes that outputs angular rates (ω^b) or delta-thetas (ω^b integrated over short period of time).

For INS, the choice of the reference frame is critical. The two most used in this document are the Navigation Frame (n-frame) and Body Frame (b-frame) . The n-frame is a locally leveled frame with the "+Z" axis pointing down in the direction of gravity, the "+X" axis pointing North and the "+Y" pointing East. The n-frame rotates along the earth frame to maintain its locally-leveled orientation. A b-frame originates at the center of mass of the platform. With regards to an aerial vehicle such as an aircraft, the "+X" is the longitudinal (roll) axis, "+Y" is the lateral (pitch) axis and "+Z" is the down (yaw) axis. For all the discussions in this document we consider the strap-down INS platform where the reference frame is the navigation frame.

All measurements from the IMU are expressed in the body frame, therefore, to use these measurements to determine position and velocity, the measurements must be transformed into the navigation frame. Transformation matrix $\mathbf{C}_b^n$ [2] in equation 2.1 can



(a) Navigation Frame(NED)

(b) Body frame

Figure 2.1: INS reference frames

be used.

$$\mathbf{C}_b^n = \begin{bmatrix} \cos\theta\cos\psi & -\cos\phi\sin\psi + \sin\phi\sin\theta\cos\psi & \sin\phi\sin\psi + \cos\phi\sin\theta\cos\psi \\ \cos\theta\sin\psi & \cos\phi\cos\psi + \sin\phi\sin\theta\sin\psi & -\sin\phi\cos\psi + \cos\phi\sin\theta\sin\psi \\ -\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta \end{bmatrix} \quad (2.1)$$

where ψ is the heading angle of the platform, θ is the pitch angle and ϕ is the roll angle.

Figure 2.2 shows the computational block diagram of INS. Although an INS seems to be an excellent replacement for GPS in GPS denied environments, the INS position will drift over time due to inertial's various error sources. In particular, the bias errors and sensor noise errors result in drift errors in the final estimates of attitude, velocity and position. The bias errors have two main properties: repeatability and stability [6]. Each time the IMU is turned on, the bias is different due to various sensor effects (the so-called turn-on bias). This bias can be estimated for each power cycle by keeping the sensor stationary for a given time, estimating the bias errors, and compensating the measurements for this bias error. After power up, the bias also changes as a function of time (i.e., the bias stability). Estimation of the change in bias over a period of time requires integration with additional sensors. In the short-term, the bias in acceleration results in a linear velocity error and

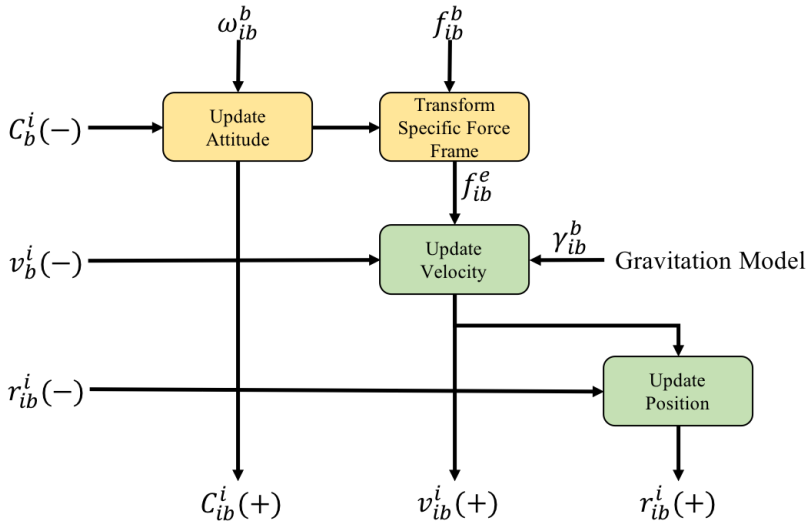


Figure 2.2: Computational sequence for INS [2]

a quadratic position error, whereas a gyro-bias results in a quadratic velocity error and a cubic position error [6]. The errors depends upon the quality of the IMU and can be reliably used without corrections for a short time duration only.

2.2 Optical Navigation Systems

Optical navigation methods mainly rely on optical sensors such as Light Detection and Ranging (LiDAR), cameras and 3D imaging sensors. This section gives a brief overview of existing optical navigation systems.

2.2.1 Vision-based Navigation Systems

Visual Odometry (VO) is the process of estimating the 3D motion of an object using input from a single (monocular) or multiple cameras (e.g., stereo) by exploiting the changes in the images introduced by the platform motion. In order for the estimation to be reliable, the consecutive sequence of images need to have sufficient overlap and each image should have enough texture [7]. Historically, VO has always been considered a limiting case of

structure from motion (SFM) where one is trying to estimate 3D motion from the input of sequential images.

Stereo VO is a type of VO that uses a stereo pair of images as the input. Nister et al. [8] was one of the first to propose a technique that was robust, real time and used outlier rejection. Motion estimation can be performed using images from a monocular camera. According to [9], the basic VO approaches consist of direct and indirect methods. Whereas the former uses the photometric relation from the image pixels , the indirect methods pre-process the image and use extracted features or optical flow vectors to determine the pose and construct a map. Additionally, [9] divides the methods into dense methods (use all of the image points), sparse methods (use only a subset of independent points), and semi-dense methods (use a large subset of the points).

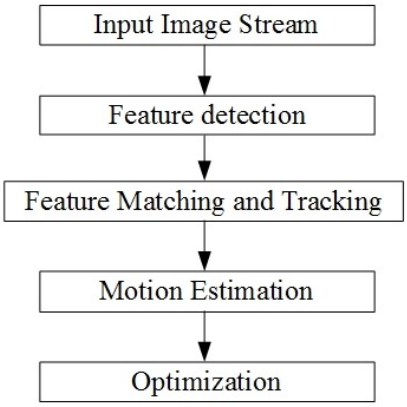


Figure 2.3: Block diagram of general steps in visual odometry

Figure 2.3 shows the general procedure followed in indirect VO. The preprocessed stream of images from a monocular / stereo pair camera is processed in real-time to estimate the full trajectory of the camera through overlapping images with unique descriptors. They need to be robust to noise, computationally efficient, repeatable and scale invariant

[10]. Harris corner detectors, Shi-Tomasi corner detectors [11], Features from Accelerated Segment Test (FAST) are some of the popular feature extractors from an image. Scale-Invariant Feature Transform (SIFT) [12], [13], Speeded-Up Robust Features (SURF), [14] and Oriented FAST and Rotated BRIEF (ORB) [15] are popular feature detector algorithms that use the feature extraction methods previously mentioned, additionally they also generate feature descriptors for the extracted features.

The extracted features then have to be matched across consecutive image frames and tracked. Feature matching techniques described in [10] could be used to efficiently match and track features. Finally, the pose is estimated from the matched and tracked features. The motion estimation problem can be classified into (i) 2D to 2D, (ii) 3D to 3D or (iii) 3D to 2D. 2D to 2D estimation is used when both the image coordinates are in 2D (Monocular camera). [16] describes each technique in detail. The poses that are estimated frame to frame are finally optimized using linear / non-linear optimization techniques to get the best trajectory.

Although VO seems like a replacement to INS, it has its own disadvantages. Since the motion estimate is frame to frame based, the error from each frame accumulates over a period of time and the solution starts to drift. To control the drift, a local optimization or a Bundle Adjustment (BA) might be required. BA is an optimization of camera poses in photogrammetry which uses key images and estimated poses to calculate an optimized trajectory. VO calculates only the path of the object but SLAM methods should be considered if information about the environment and map is needed.

2.3 3D Imagers

This thesis makes use of depth imagery where each image pixel is associated with an intensity value and a 3D point cloud value. Each pixel can, thus, be represented by a vector (x, y, z, η). One category of sensors that is capable of acquiring depth imagery is based on

the principle of Time of Flight (TOF), i.e., by measuring the time-of-flight for light emitted from the camera to reach a target and be reflected back. Examples of TOF sensors are active 3D camera systems that use the transmission and reception of pulses such as the 3D flash LiDAR products by Advanced Scientific Concepts, or sensors based on the transmission or reception of amplitude modulated Infrared (IR) such as the Swissranger and PMD sensors. The use of an active sensor to create 3D images overcomes the inability of a passive stereoscopic 3D image sensors to estimate the distance to an object in a scene when there is insufficient contrast between the object and the background. Due to the recent increase in research towards autonomous cars, 360° 3D LiDAR have become popular. These 360° 3D LiDAR incorporate an array of lasers and a rotating mirror to accurately measure the surroundings in 360°. Example of the 360° LiDARs are products from Velodyne, Quanergy among many. A more recent development is the solid state LiDARs which do not have any moving parts and offers excellent performance, highly power efficient and are designed towards autonomous vehicles. These TOF based 3D sensors generate a sparse point cloud and have high range usually between 100 - 150 meters.

More recent sensors that produce depth images are the Kinect and the Occipital. The principle of operation of both these sensors is based on structured light. The structured light based 3D camera was first developed by PrimeSense. The main principle behind the structured light 3D camera is the appearance of distortion in illumination when a narrow band of light is projected on a 3D environment when seen from a different perspective. A structured light 3D camera consists of an IR projector and an IR camera. The IR projector projects IR light through a diffraction grating on to the scene, which converts the projected light into dots. The IR camera observes these dots on the scene and matches it back to the dots in the IR projector. Since the geometry between the IR projector and the camera is known, the diffraction pattern is known, 3D scene is reconstructed using triangulation [17], [18]. These cameras also have an ability to generate a colored 3D point cloud which can

be represented by the vector (x, y, z, r, g, b) by including a color camera in the sensor. Both the cameras rely highly on camera (both intrinsic and extrinsic) calibration and generate a dense point cloud with max range of 3.5 - 6 meters.

2.4 Sensor Integration Tools: Kalman Filtering

The Kalman Filter (KF) is a very popular estimator for systems whose dynamics model is given by:

$$\mathbf{x}_t = \mathbf{\Phi}_t \mathbf{x}_{t-1} + \mathbf{B}_t \mathbf{u}_t + \mathbf{w}_t \tag{2.2}$$

where $\mathbf{x}_t$ is the state vector, $\mathbf{u}_t$ is the control input (if used), $\mathbf{\Phi}$ is the state transition matrix, and $\mathbf{w}_t$ is a zero-mean Gaussian noise component with covariance matrix $\mathbf{Q}_t$. And, whose measurement model is given by:

$$\mathbf{z}_t = \mathbf{H}_t \mathbf{x}_t + \mathbf{v}_t \tag{2.3}$$

where $\mathbf{z}_t$ is a vector with the measurements, $\mathbf{H}_t$ is the measurement matrix relating the measurements to the state vector elements and $\mathbf{v}_k$ is the measurement noise which is zero-mean Gaussian with covariance matrix $\mathbf{R}_t$. The KF finds the best estimate of the current state vector $\mathbf{x}_t$, given the state at the previous time-epoch $\mathbf{x}_{t-1}$, the new measurements $\mathbf{z}_t$ and, possibly, a control input $\mathbf{u}_t$. It does so by finding the expected value of the posterior distribution $p(\mathbf{x}_t|\mathbf{z}_{t-1}, \mathbf{u}_t)$ representing all the involved probability density functions by their mean and covariance matrix. This results in the following set of equations; the prediction step is given by:

$$\hat{\mathbf{x}}_t^- = \mathbf{\Phi}_t \mathbf{x}_{t-1} + \mathbf{B}_t \mathbf{u}_t \tag{2.4}$$

$$\mathbf{P}_t^- = \mathbf{\Phi}_t \mathbf{P}_{t-1} \mathbf{\Phi}_t^T + \mathbf{Q}_t \tag{2.5}$$

where $\hat{\mathbf{x}}_t^-$ is the predicted state and $\mathbf{P}_t^-$ is its covariance. The update step or calculation of the innovation and updating the prediction is given by

$$\mathbf{K}_t = \mathbf{P}_t^- \mathbf{H}_t^T (\mathbf{H}\mathbf{P}_t^- \mathbf{H}^T + \mathbf{R}_t)^{-1} \tag{2.6}$$

$$\hat{\mathbf{x}}_t = \hat{\mathbf{x}}_t^- + \mathbf{K}_t(\mathbf{z}_t - \mathbf{H}_t\hat{\mathbf{x}}_t^-) \tag{2.7}$$

$$\mathbf{P}_t = (\mathbf{I} - \mathbf{K}_t\mathbf{H}_t)\mathbf{P}_t^- \tag{2.8}$$

where $\hat{\mathbf{x}}_t$ is the state estimate and $\mathbf{P}_t$ is its covariance. The Kalman gain $\mathbf{K}_t$ is calculated in equation (2.6) and specifies the weight of the measurement in the state estimate.

Even though the KF works extremely well, the major limitation is that it expects a linear dynamics and measurement equation. This is rarely the case in the real world. To extend the KF to robotic problems with non-linear state transitions and measurements, we use non-linear functions $\mathbf{g}()$ and $\mathbf{h}()$. The modified KF is known as Extended Kalman Filter (EKF).

In addition to the KF and EKF, a couple of other filters are regularly used in robotics application including the Unscented Kalman Filter (UKF), the information filter and particle filter [19]. The information filter is a parametric type of filter that uses an information matrix $\mathbf{\Omega}$ and information vector ξ, rather than a covariance matrix and a mean vector. Particle filters are a non-parametric filters where the estimate is obtained from the sample mean of an approximate of the posterior density function rather than linearizing the dynamics and measurement equations.

If the error states (example, IMU bias) is estimated along with the filter position, the filter is often referred to as an error state filter or a complementary Kalman filter [20]. Furthermore, the state vector may include the positions of landmarks observed in the environment or even occupancy grid maps [19]. The latter is referred to as SLAM.

2.5 Simultaneous Localization and Mapping (SLAM)

This subsection provides background on SLAM and will end with a short literature review on solutions to existing SLAM problems using a 3D depth sensor and camera.

The SLAM problem of a robot is similar to a bootstrapping situation, where the robot is introduced to a new environment without a prior knowledge of the environment. The robot has to localize itself in the environment while generating a map of the environment incrementally, based on the information from sensors. Generating the map of the environment is important since it is used not only for navigation, but also for path planning purposes. Furthermore, the map can be used to reduce the localization error covariance by re-visiting older map sections, known as **loop closure**. The inertial sensors and the visual odometry could be combined together to obtain a visual inertial system which solves for SLAM [21].

There are several techniques to address the SLAM problem, and some of the most popular ones are discussed here. A SLAM problem can be solved using filtering methods or smoothing methods. Filtering methods solve the online SLAM problem and find the mean of the posterior over the most recent pose or $p(\mathbf{x}_t, \mathbf{m}|\mathbf{z}_{1:t}, \mathbf{u}_{i:t})$. On the other hand, smoothing methods solve the full SLAM problem and try to estimate entire path or $p(\mathbf{x}_{1:t}, \mathbf{m}|\mathbf{z}_{1:t}, \mathbf{u}_{i:t})$.

Examples of the former approach include EKF based SLAM (EKF-SLAM) [22] and Rao-Blackwellized particle filter SLAM (FastSLAM) [23]. Various approaches have been investigated to reduce this computational complexity including the use of submaps. FastSLAM is another approach that reduces computational complexity, by separating the estimation of the pose and the map. In FastSLAM, particles consist of a 2D pose with M associated 2D Kalman filters for each of the landmark locations.

GraphSLAM is a smoothing method to solve the full SLAM problem (i.e., find both the whole path and its associated map given all measurements and control inputs). In GraphSLAM, a factor graph is setup of nodes (i.e., the state consisting of poses, $\mathbf{x}_{1:t}$, and

landmark locations, $\mathbf{m}_{1:M}$) and edges or factors between the nodes that represent existing constraints imposed by measurements, $\mathbf{z}_{1:t}$ or control inputs, $\mathbf{u}_{1:t}$.

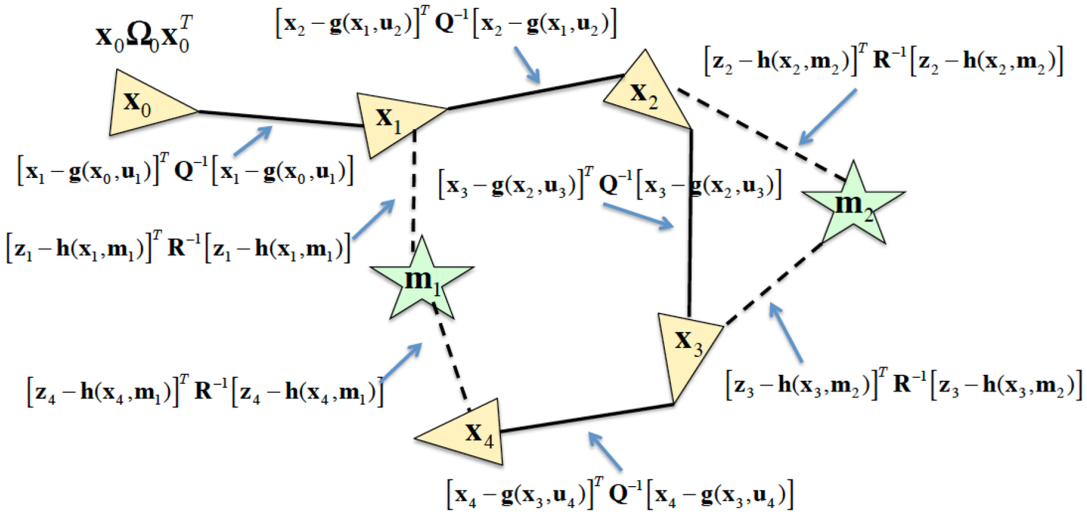


Figure 2.4: Graph and constraints for GraphSLAM

Next, a mathematical representation is derived for this factor graph and solved in same fashion. For example, for the graph shown in figure 2.4, the full SLAM could be solved by finding the poses and landmark, that minimize the following cost function:

$$J = \mathbf{x_0}\boldsymbol{\Omega_0}\mathbf{x_0^T} + \sum_k [\mathbf{x_k} - \mathbf{g}(\mathbf{x_{k-1}}, \mathbf{u_k})]^T \mathbf{Q}^{-1} [\mathbf{x_k} - \mathbf{g}(\mathbf{x_{k-1}}, \mathbf{u_k})] ... \\ + \sum_k [\mathbf{z_k} - \mathbf{h}(\mathbf{x_k}, \mathbf{m})]^T \mathbf{R}^{-1} [\mathbf{z_k} - \mathbf{h}(\mathbf{x_k}, \mathbf{m})] \tag{2.9}$$

Various approaches to solving the constraint equation have resulted in numerous Graph-based SLAM approaches such as iSAM, GTSAM, g2o, etc. Typically, GraphSLAM is divided into front-end and back-end. In the front-end, the sensor data is processed, and the factor graph is built. The back-end finds a solution for the factor graph in a manner similar to BA in vision processing.

The current trend in research is active SLAM, where machine learning is used to make decisions in selecting the future robot controls based on the acquired map in-order

to minimize the positional error associated. This also includes incorporating deep learning and other Machine Learning (ML) concepts into SLAM algorithms to estimate poses from the raw input data without processing it [24], [25]. Use of ML in SLAM algorithms have shown quite a few advantages over the traditional methods, but it still raises the question about the confidence of the solution obtained using ML methods. ML is still in its nascent stage and further research is needed to determine the integrity of the solution using ML.

2.6 SLAM Using 3D Imaging Sensors

In recent times, the structure light based mobile 3D sensors like Microsoft Kinect, PrimeSense and similar devices have become very popular in SLAM applications due to its low weight and cost. Newcombe et al. [26] recreated a 3D object by moving a hand-held Kinect around a 3D object. One of the earliest methods of registering 3D points is given by Besl and McKay [27]. Iterative Closest Point (ICP), also known as laser scan matching is one of the most popular and common methods used for SLAM with 2D LiDAR data. When extended to 3D, it is very tedious and computationally expensive. It needs sufficient features with depth variations and doesn't work well in the presence of planar surfaces [28]. To reduce the computation time of ICP for 3D, Su et al. [28] proposed a method to use features from Red Green Blue (RGB) camera and matched it with the depth frame to find homography using Random Sample Consensus (RANSAC). The corresponding pairs align with its depth measurement are back projected to 3D space and used as a starting point in ICP.

To reduce the computational complexity of 3D ICP and solve the unstable convergence issue, Wen et al. [29] proposed a laser scanner and RGB-Depth (RGB-D) particle filter based fusion technique, where 2D SLAM poses are used along with the features from RGB images as an initialization point for fusion ICP. Similar to the fusion technique, Singh et al. [30] proposed a fusion of RGB-D camera with an in-house designed LiDAR.

Principle Component Analysis (PCA) is used to transform the LiDAR data into fixed frame and ICP is used to align the data to the reference frame. Bose and Richards [31] proposed a technique to reduce the number of points used in the ICP for pose estimation, where only the edge points from the depth imagery were used in ICP to estimate the camera pose.

Wooden et al. [32] designed a pure visual-inertial based system for pose estimation for the Big Dog which uses SICK laser 291 LiDAR scanner, pointgrey bumblebee stereo vision cameras, NovAtel GPS receiver and proprioceptive IMU sensors to calculate ground contact state, ground plane orientation, body velocity etc. The LiDAR scans and camera images are processed to objects and obstacles which are individually segmented and tracked over time. The pose is estimated by fusion of measurements from IMU, visual odometry and kinematic information from legs. The disparity map from the stereo vision maintains a 3D terrain map.

Valenti et al. [33] proposed a visual-inertial odometry based navigation system for MAVs. Visual odometry at 30Hz is combined with inertial data using Kalman filter. An x86 architecture processor based onboard system with dual level system architecture is used [34]. Shi-Tomasi algorithm is used to detect the features, the corresponding 3D co-ordinates are used to align the frame with previously detected features, Mahalanobis distance is used for data association. KF uses the associated data for prediction as given by Dryanovski et al. [35] which is cascaded with an alpha beta filter that acts as a correction to KF. g2o is used for optimization. Brunetto et al. [36] proposed a visual-inertial fusion approach to improve the RGB-D SLAM algorithm. Camera tracking is done using VO, the resultant pose is used in the EKF where the inertial measurements are fused to obtain a more precise pose. A global optimization of key frames is done to get a globally consistent map. Tiefenbacher et al. [37] proposed a similar UKF based approach to fuse the inertial measurements with visual pose estimation based on Parallel Tracking and Mapping (PTAM) algorithm [38].

Henry et al. [39] and Dong et al. [40] proposed a RGB-D odometry based indoor reconstruction where the features from RGB-D images are extracted and matched from the previous frame. RANSAC is used to calculate the rigid body transform between the two frames, this transform is used as an initializer for ICP. Henry et al. [39] performs global optimization using TORO, proposed by Grisetti et al. [41], [42] and the points are represented as surfels [43]. Endres et al. [44] evaluated RGB-D SLAM for three feature descriptors i.e., SIFT, SURF and ORB. The features from the input RGB images are detected and matched against the previous image. The depth image was processed at these features to obtain the point wise 3D correspondence which was used to calculate relative pose using RANSAC. The poses are then optimized by g2o optimizer which is a Graph SLAM method. This is available as an open source project [45].

Fischer et al. [46] proposed a stereo based localization method for rough terrain walking robot that could handle arbitrary camera motion. PTAM proposed by Pire et al. [47] consisting of tracking part and mapping part simultaneously. The tracking section extracts the features from the stereo images and projects the map points on to the image plane based on current pose estimation. The features are matched based on the closest point between the map points and the extracted features. Simultaneously the mapping part continuously refines the map by Levenberg-Marquardt based BA. Meng et al. [48] and Belter et al. [49] proposed a VO and Graph SLAM based method which uses ORB feature extractors. Both of them use g20 for optimization. [48] uses ICP for pose estimation, whereas [49] uses VO to estimate the frame to frame pose. [49] proceeds to integrate the features detected from ORB into the map and model the spatial uncertainty.

The monocular SLAM and RGB-D SLAM techniques have certain shortcomings that are exclusive to each other. Hu et al. [50] proposed an heuristic based SLAM algorithm that uses monocular SLAM and RGB-D SLAM to get a more reliable solution. A two stage ICP is used to estimate the transformation of the camera from two frames. Sparse

bundle adjustments are used for optimizing the global map. Davison et al. [51] proposed a real-time method to perform SLAM using monocular camera that achieves drift free performance. The system relies on the probabilistic sparse map of naturally occurring features detected using Shi-Tomasi detectors. A simple EKF is used to update the map based on smooth motion model. To overcome the limitation of initializing with known features or assumptions, Silveira et al. [52] used direct visual odometry approach using image intensities instead of extracting features.

Silveira et al. [52] proposed visual EKF SLAM which uses a combination of Harris detector and SIFT for feature extraction. The object recognition step is responsible for data association, where it checks if the object is already seen before, else it adds a new entry to the database. These features are used in EKF SLAM as landmarks for localization and mapping. Ye et al. [53] uses a 3D time of flight camera to perform six Degrees of Freedom (DoF) pose estimation for a EKF based navigation aid. The visual features are extracted by the intensity image using SIFT, and RANSAC is used to match these features. Feature based EKF is used for more accurate pose estimation.

Feng et al. [54] proposed a fast planar extraction method for RGB-D cameras in structured environment. The structured point cloud is divided equally into n number of graphs and an Agglomerative Hierarchical Clustering (AHC) is performed on all the graphs which merges the graphs belonging to the same plane. If finer results are needed, the plane is refined using pixel-wise region growing. Taguchi et al. [55] and Ataer-Cansizoglu et al. [56] proposed points and planes based SLAM algorithm for localization and mapping using RGB-D cameras. [55] uses SURF feature descriptors to extract features from the RGB image and multi-stage RANSAC method to extract planes from the point cloud. Using these planes and points, RANSAC based registration is performed. Finally, the map is optimized using BA. Work proposed in [56] is based on [55], but the pose of the camera is predicted using camera motion model. Planes and points from previous frames are tracked.

In case of loss of tracking features, they initiate a re-localization. The registration and optimization are same as [55]. Ataer-Cansizoglu et al. [57] proposed a pinpoint SLAM that estimates both online SLAM and full SLAM which is based on [55].

For evaluation of SLAM algorithms, publicly available data-sets with the ground truth values play a very important role. Handa et al. [58], [59] and [60] present a real world data recorded from Kinect with truth values for VO, SLAM and surface reconstruction.

Summarizing, most of the techniques used with RGB-D or similar structure light based depth camera for SLAM use ICP or feature based VO to estimate the rigid body transform from one frame to the next. ICP techniques are time consuming and feature detection based VO are vulnerable to lighting conditions. Even the planar based methods use RANSAC based registration for pose estimation. In this research, the data from the 3D and thermal camera is preprocessed and segmented such that the features for navigation can be extracted in real time on a low powered Advanced RISC Machines (ARM) processor based onboard computer.

3 Navigation Methodology

The high-level functional block diagram of a typical sUAS guidance, navigation and control system is depicted in Figure 3.1. and consists of two main functions: the tactical Guidance Navigation and Control (GNC) function and the strategic GNC function [61].

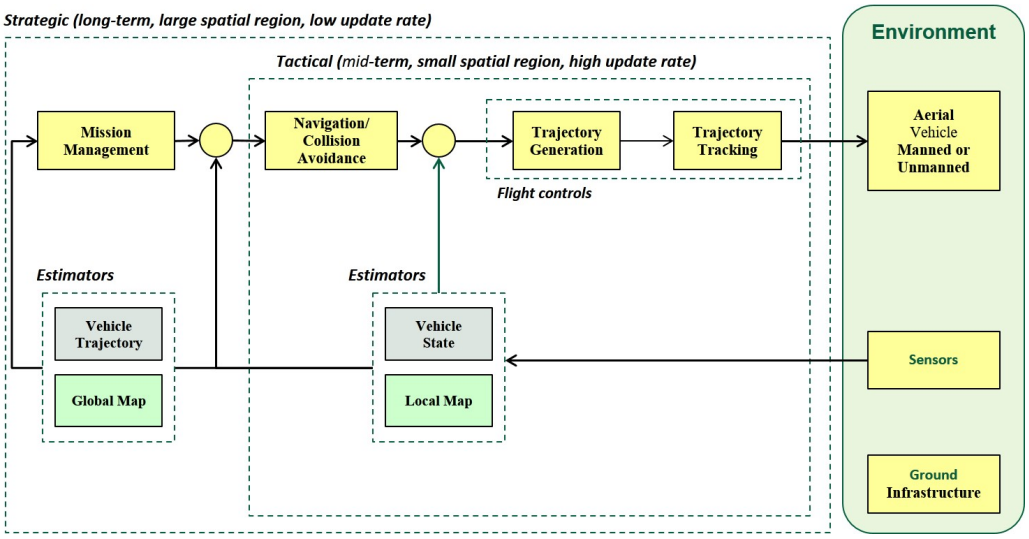


Figure 3.1: High-level functional block diagram of the hierarchical guidance, navigation and control function

The tactical GNC function makes optimal short-term trajectory planning decisions at a high update rate to accommodate a possible high speed of operation of the aerial vehicle in a potentially cluttered environment. These decisions are based on vehicle state (i.e., position, velocity, and attitude) updates from the navigation module and a local map of the environment. This map describes attributes of the local environment as perceived by the sensors that can be used in the path planning and collision avoidance processes. Example maps in the context of this thesis would be tunnel maps and time-to-contact maps.

The strategic GNC keeps track of the planned and actual flight trajectories and provides the sUAS with the next waypoint(s) that must be reached. In unknown

environments, the strategic GNC function keeps track of a rough estimate of the platform trajectory and associated global map of the environment. Based on this information and the goals defined by the specific mission (i.e., outdoor/indoor navigation and mapping mission), this function will provide the tactical GNC algorithms with constraints regarding its path planning decisions (i.e., preferred direction of travel, decision between multiple similar choices such as doors or windows) and associated control behaviors. The global map can be formed using various typical SLAM back-end methods like composition of the global map from multiple local maps using, for example, Hierarchical SLAM [62] or GraphSLAM methods (e.g. [63]).

A block diagram of the proposed tactical GNC navigator and mapping method is shown in Figure 3.2. Blocks (a) and (b) take accelerometer and gyro measurements from the onboard IMU and perform the attitude and position calculations, solely based on the IMU measurements using the method described in [64]. The point cloud provided by the 3D imager is expressed in the body frame and will be first converted to the navigation frame using a best estimate of the attitude. Then, features are extracted from the point cloud in (c) and associated from frame to frame in (d). As the environment is structured, typical 3D structures will be used such as planar surfaces, pipes, corners or edges. The displacement and/or relative orientation change from frame to frame can be used to estimate the delta-position, $\Delta\tilde{\mathbf{r}}^n_{DEP}$, and delta-orientation, $\Delta\tilde{\mathbf{C}}^n_b$, of the platform. The same can be obtained from the IMU mechanization ($\Delta\tilde{\mathbf{r}}^n_{IMU}, \Delta\tilde{\mathbf{C}}^n_b$), and both can be input to a Complementary Kalman Filter (CKF) with error state vector:

$$\mathbf{x_1} = \begin{bmatrix} \delta\mathbf{v}^n & \delta\psi_{nb} & \delta\omega^b_{ib} & \delta\mathbf{f}_b \end{bmatrix} \tag{3.1}$$

where $\delta\mathbf{v}^n$ is the velocity error in the NED frame, $\delta\psi_{nb}$ is the miss-orientation vector, $\delta\omega^b_{ib}$ is the gyro bias vector, and $\delta\mathbf{f}_b$ is the specific force bias error. After the CKF update step, the error states are fed back to the attitude and navigation computation blocks to reduce the

drift errors. The measurement vector is formed as follows:

$$\mathbf{z}_{INS} = \Delta\tilde{\mathbf{r}}^{n}_{INS} - \Delta\tilde{\mathbf{r}}^{n}_{DEP} \tag{3.2}$$

The error states in Equation 3.1 only concern the IMU error states, and current implementation assumes that the errors from the 3D imager based translation motion estimation are Gaussian in nature. In reality, this is, however, not the case and the effect of these non-Gaussian is proposed for future research.

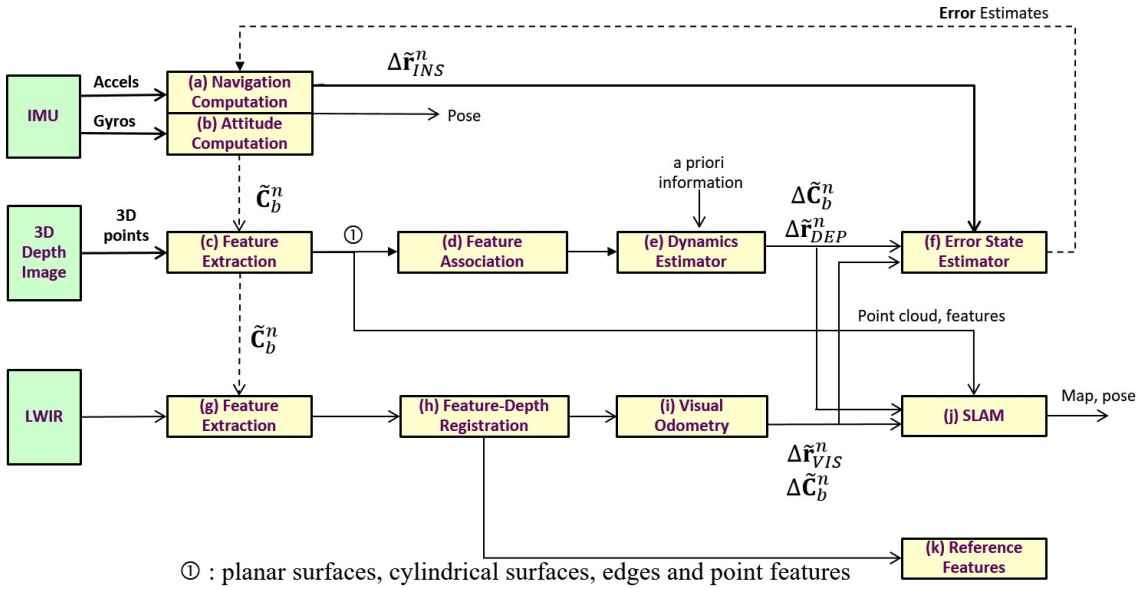


Figure 3.2: Functional block diagram of the proposed navigation and mapping method

Although the LWIR camera is a low-resolution sensor (80-by-60 pixels), features can still be extracted from the resulting imagery and is done so in block (g) of Figure 3.2. Feature examples are corners, lines and blobs. The lower resolution will result in a larger uncertainty of the feature location and, thus, larger estimation covariance when using these features for visual odometry [65]. Typical visual odometry results in an estimate of the change in orientation between the two image frames, and an estimate of the translational motion up to a scale factor due to unknown depth. The latter issue can be resolved by

associating the LWIR sensor pixels with pixels of the 3D imaging sensor and using the associated point cloud coordinate to obtain the unknown depth. The resulting translational motion can then input to the CKF of (f) by forming the following error measurement:

$$\mathbf{z}_{INS} = \Delta\tilde{\mathbf{r}}^{n}_{INS} - \Delta\tilde{\mathbf{r}}^{n}_{LWIR} \tag{3.3}$$

It is important to include good estimates of the error covariance matrices $\mathbf{P}_{LWIR}$ and $\mathbf{P}_{DEP}$ to reflect the difference in quality of both error measurements in the overall filter.

In addition to an error estimator to keep the inertial drift contained, the proposed method uses the odometry estimates from the LWIR camera and the 3D imager, the point cloud measurements and the extracted features are inputs to the 3D SLAM algorithm. Since the objective is to generate a position solution and map in real-time, this SLAM methods must be fast and robust. The method proposed in this thesis is based on the methods proposed in [66] and [67], where 3D features are extracted and subsequently used to perform odometry and generate a LiDAR map. The main difference between our method and the method proposed is the manner in which features are extracted and used to perform the odometry step. An alternate method is described in [68] and [69]. The SLAM method described in those papers consists of two stages; the first stage runs open loop and uses the pairwise ICP algorithm [66] to estimate the incremental pose. The second stage runs the loop closure, when loop closure is detected (earlier environment is encountered a second time), it uses an extension of Lu and Milios laser scan matching algorithm [70] to find the change in pose.

The final step in our methodology (k), takes the features extracted from the LWIR images and maps them on the 3D environment using a direct mapping from the LWIR image frame to the depth image frame and the map.

4 Sensors and UAS Hardware Setup

Before talking about the calibration, pre-processing and navigation algorithms, this chapter describes the various sensors and their interfaces with the on-board computer and the software setup of the research platform. Finally, this chapter also briefly discusses the target flight environment for this research.

4.1 Sensors

The main sensor suite of our platform consists of a 3D imaging sensor, a thermal camera and an IMU.

4.1.1 Occipital Structure SWIR and depth sensor

The Structure sensor is a 3D imaging sensor (a.k.a. depth imager) that started as a kick starter project from Occipital launched in September 2013 [71]. The structure sensor is a low cost, low power and light weight 3D camera designed specifically for mobile applications. The 3D depth imager uses OpenNI2 which is a framework developed by the OpenNI consortium to standardize the interaction between the onboard computer and 3D and 2D image sensors [72]. The 3D imager houses a laser projector which rotates and projects a laser plane on the environment [71]. A frequency matched IR camera observes the received laser reflections and uses the pattern of the received laser light to calculate the depth of objects in the scene. The sensor captures this depth information for each pixel based on the changes in the structured light seen by the IR camera [71]. The 3D imager generates a depth stream of size 640x480 pixels with each pixel representing a depth value in the real world at 30 frames per second (fps) [71]. The specifications of the sensor are listed in Table 4.1 [72]:



Figure 4.1: Occipital structure 3D depth sensor

4.1.2 FLIR Lepton LWIR camera

Lepton is a long wave infrared camera from FLIR, that captures infrared radiation and produces a thermal image. It is small in size, light weight and inexpensive compared to other LWIR cameras in market and designed specifically for mobile applications. Figure 4.2 shows an image of the camera. The camera connects to the system via a SPI and a two wire Inter-Integrated Circuit (I^2C) like interface [73]. The image data is transferred through SPI using the Video over SPI (VoSPI) protocol, where images are streamed over SPI, whenever the frames are available in the camera. The maximum clock rate for the data transmission over VoSPI is 20 MHz but is configured to 2 MHz which is sufficient to obtain the required frame rate. The camera transmits just under 27 fps to support streaming devices, but unique frames are transmitted at just below 9 fps [73]. It duplicates a frame twice and transmits it as a new image to attain 27 fps. The frame number is indicated in the packet and thus the driver ignores the duplicate packets [74]. Figure 4.3 show the block diagram of Lepton camera.

Table 4.1: Hardware specification of Occipital camera

Weight	95 grams
Operating Temperature	$0°C$ to $35°C$
Field of View(FoV)	• Horizontal: 58° • Vertical: 45°
Resolution	640 x 480
Frame Rate	30fps
Sensing Distance	• Minimum: 0.4m • Maximum: 3.5+m
Precision	1% of measured value

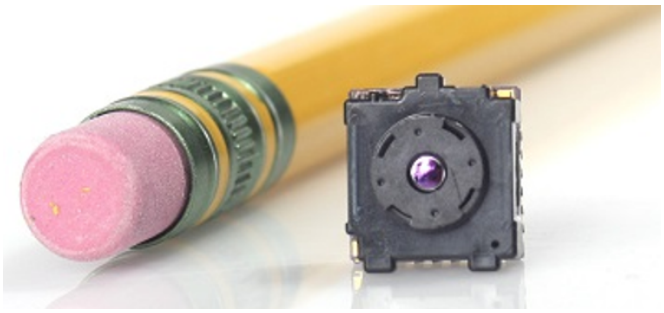


Figure 4.2: FLIR Lepton LWIR Camera

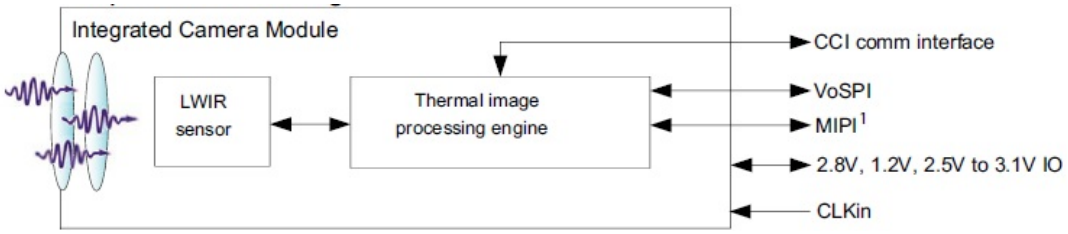


Figure 4.3: Block diagram of Lepton camera

The specifications of the camera are as presented in Table 4.2 [73].

Table 4.2: Hardware specification of Lepton camera

Weight	55 grams
Operating Temperature	$-10°C$ to $65°C$
Field of View (FoV)	• Horizontal: 51° • Diagonal: 63.5°
Power Consumption	150mW
Spectral Range	$8\mu m$ - $14\mu m$
Resolution	80 x 60
Frame Rate	8.6fps
Depth of Field	10 cm - inf
Depth of the Image	14 bits / pixel

The camera has a built-in image processing unit, which is capable of enhancing and correcting the images. The camera is capable of correcting grainy and blotchy images

through its Flat Field Correction (FFC) state. FFC is used before the camera is calibrated for internal and external camera calibration parameters. To initiate an image correction, the camera is exposed to a uniform thermal scene and the FFC mode is enabled. It uses a scene-based non-uniform correction algorithm to perform the correction. These features and modes of the camera can be controlled using the two-wire I^2C control interface [73][74].

The size of the image given by the lepton camera is small, i.e., 80x60 pixels, and this makes the image pixelated and hard to interpret visually. In order to preserve and extract as many details as possible, these images are scaled up to Video Graphic Array (VGA) resolution, i.e., 640x480 pixels. Although upscaling the image doesn't add any additional information, it is essential for camera calibration process, as detection of calibration pattern is extremely difficult at 80x60 image size. Bicubic interpolation is used to scale up the image as it does an excellent job in preserving the details of an image as compared to other techniques [75] . The interpolation function can be represented by the equation (4.1), where x, y are the pixel locations where the intensity has to be assigned, $v(x, y)$ is the intensity value calculated [75]. The bicubic interpolation uses 16 nearest neighbors to compute the intensity value of the pixel.

$$v(x, y) = \sum_{i=0}^{3} \sum_{j=0}^{3} a_{ij} x^i y^i \tag{4.1}$$

A color-map is applied to the interpolated image to convert the grayscale image to a pseudo-colored image that is used later on for generating colored point clouds. This process is shown in the figure 4.4

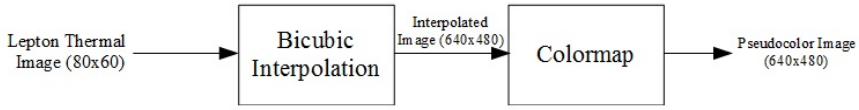


Figure 4.4: Lepton thermal image processing

4.1.3 Xsens IMU

Xsens MTi-1 is a Micro-machined Electromechanical System (MEMS) low cost, low power IMU ideally suitable for industrial applications including robotics and navigation. This IMU measures linear acceleration, angular velocity and the magnetic field along the 3 perpendicular axes [76]. The device supports I^2C, SPI and Universal Asynchronous Receiver/Transmitter (UART) interfaces with the host system. An image of the IMU is shown in figure 4.5.

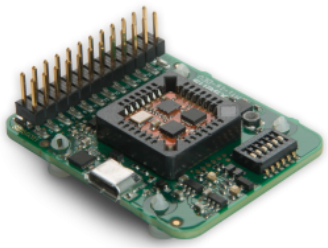


Figure 4.5: Xsens MTi 1-s IMU development board [3]

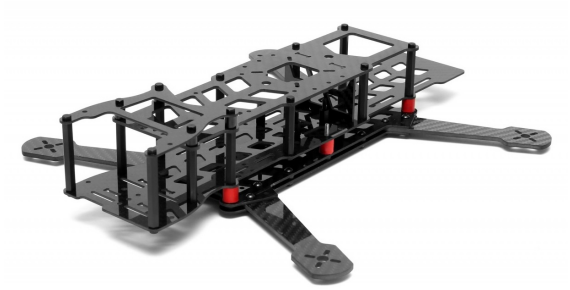
The MTi is a completely self-contained, factory calibrated IMU with an on-board signal processing unit that outputs raw measurements and estimates for attitude and heading. The accuracy is given as 2° Root Mean Square (RMS) for heading, 1° RMS for pitch and roll angles [76]. The MTi power consumption is around $45mW$ at $3.0v$ and the unit is capable of making measurements at a maximum rate of 100 Hz. A sensor fixed reference frame is used for angular velocity and linear acceleration measurements, but an East North Up (ENU) reference frame is used by default for reporting the orientation, linear acceleration and angular velocities. The device also reports a sample time, based on the internal oscillator in every message.

4.2 System Setup

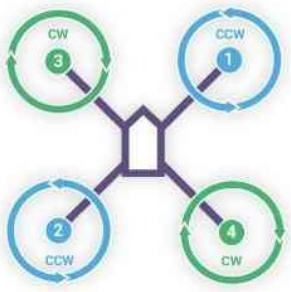
This section describes the hardware and software setup of the UAS platform as well as the UAS itself.

4.2.1 UAV frame and flight controller

The selected UAS airframe is a Blackout 330 mini H quadcopter frame. It is a small size UAS designed specifically for First Person View (FPV) flights. As the name suggests, the frame has four arms of length 330 mm from the center of the frame. Figure 4.6a shows the Blackout frame used. Four 2000Kv [77] motors are mounted on each arm. To each of these motors a propeller is attached, and their orientations are shown in figure 4.6b. Although the quadcopter is designed for racing with very minimal payload, it is altered to carry a significantly larger sensor research payload.



(a) Blackout 330 Quad frame[78]



(b) Direction of the propeller



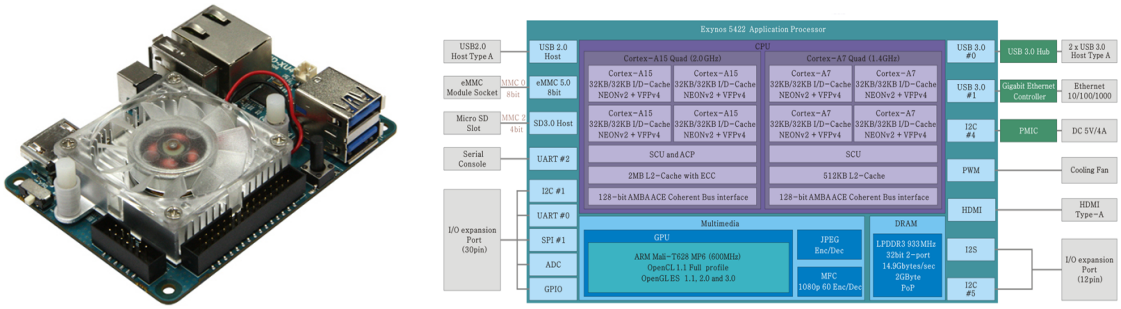
(c) Pixhawk flight controller and autopilot system

Figure 4.6: Frame and the flight controller system used on the platform

For UAS flight controls, the Pixhawk flight controller is selected due to its integration with ROS which is desirable for this research. The Pixhawk is a flight controller and an autopilot system for fixed wing UAS, multirotor UAS and other aerial and land robots. The module contains a 168 MHz 32-bit cortex M4F processor and a fail-safe co-processor with 256 KB of RAM and 2 MB flash running a Real Time Operating System (RTOS). The Pixhawk has on-board 3D accelerometer, 3D gyroscope, magnetometer and a barometer for navigation. It provides 14 Pulse Width Modulation (PWM) outputs to control the motors and several interfaces like I^2C, UART, Control Area Network (CAN) for external communication. The Pixhawk allows the integration of a companion Personal Computer (PC) for on-board computing and off-board autonomous control [79]. Figure 4.6c shows the image of the flight controller module.

4.2.2 On-board system

The Odroid XU4 is chosen as the on-board computer and the companion processor for the Pixhawk. The Odroid is a Heterogeneous Multi-Processor (HMP) based embedded computer on chip, powered by a Samsung Exynos octa-core processor made of quad cortex A-5 cores running at 2GHz and quad cortex A-7 cores running at 1.4GHz. XU4 has 2 GB of DDR3 Random Access Memory (RAM). The system houses two USB 3.0 ports, one USB 2.0 port, HDMI port for display and 30 pin expansion connectors for external embedded interface [80]. The Input or Output (I/O) voltage of the General Purpose Input Output (GPIO)s on the expansion header is 1.8V and a level converter board is used to interface with any other voltage standards. Figure 4.7a shows the image of Odroid XU4 and figure 4.7b shows the block diagram of the system. The XU4 is a relatively low cost, has good processing power and weighs 60 grams, which makes it an excellent choice as a companion PC for such applications.



(a) Odroid XU4 system on board (b) Block Diagram of Odroid [80]

Figure 4.7: On-board embedded system: Odroid XU4

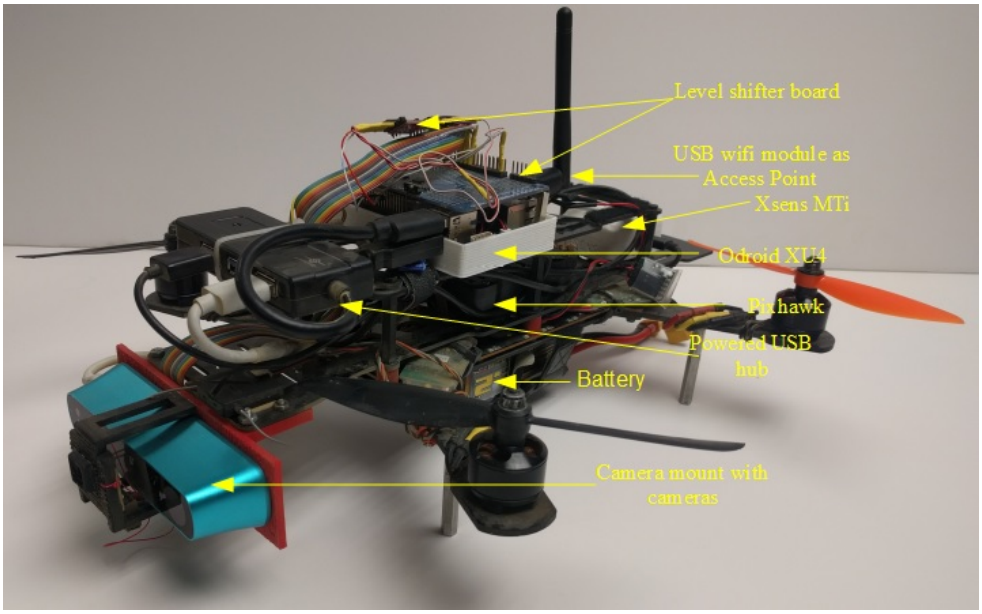


Figure 4.8: Platform for data collection with sensor payload

Although the system has a gigabit Ethernet port, USB WIFI module is used for communication with the system during flight operations. The WIFI module is configured as an access point and an external system is used to login through Secure Shell (SSH). The system runs the Ubuntu 14.04 LTS distribution of Linux with the indigo distribution

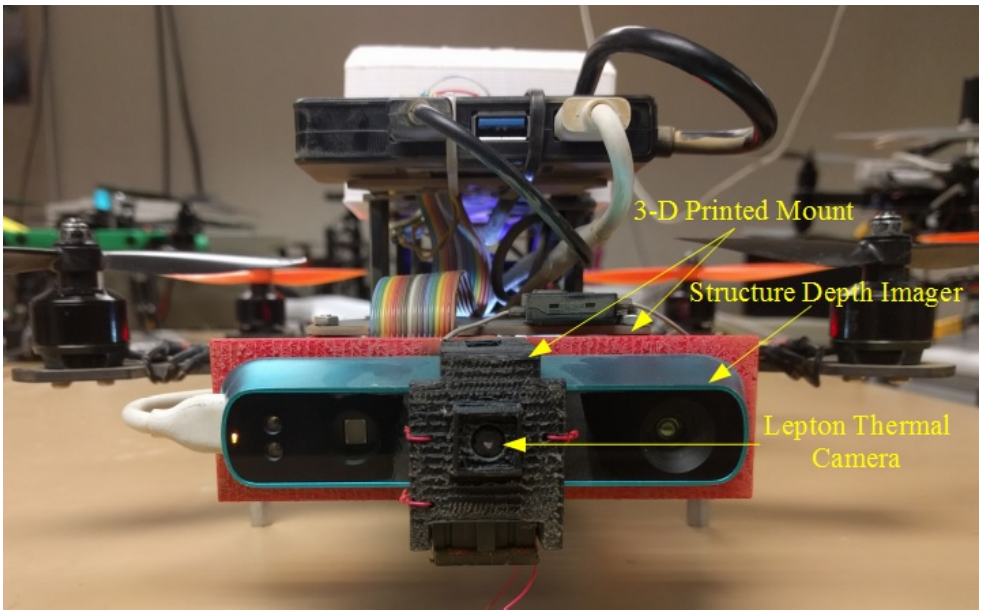


Figure 4.9: Camera mount with depth and thermal camera

of ROS on it. ROS is a pseudo Operating System (OS) that provides libraries, hardware abstraction, device drivers, a publisher-subscriber service and many more useful software packages specifically targeted towards the development of applications for robots and robot applications [81]. OpenNI2 drivers are installed on ROS for a Structure depth sensor and connected to the Odroid XU4 through USB. Custom camera drivers have been written by the author for the Lepton thermal camera in C++ using ROS libraries. These drivers get the images from the Lepton and publishes the thermal images on ROS environment. The Lepton camera is connected to Odroid through VoSPI for data and I^2C for control interface through level translator boards as shown in figure 4.8. The Xsens IMU is interfaced to the system through USB and custom drivers have been written for the device in C++ as well. Finally, the Pixhawk is interfaced with the system through an USB-to-UART cable to the UART terminal on the flight controller.

Figure 4.8 shows the platform with the complete sensor payload and figure 4.9 shows the placement of cameras with respect to each other through a 3-D printed camera mount. The platform weighs around 1,366 grams with the battery and has an average flight time just under 10 minutes. PID tuning was done on the flight controller by changing the roll rate and pitch rate parameters to make the platform more stable with the increased payload [82].

Figure 4.10 shows the power distribution of the platform and figure 4.11 shows the software and hardware interface of the sensors with the on-board PC.

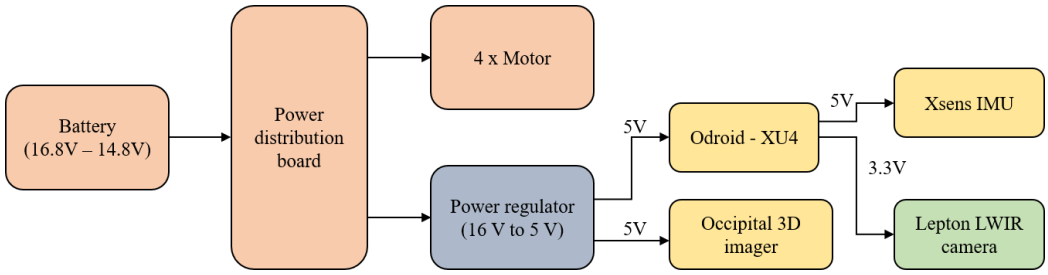


Figure 4.10: Power distribution block diagram

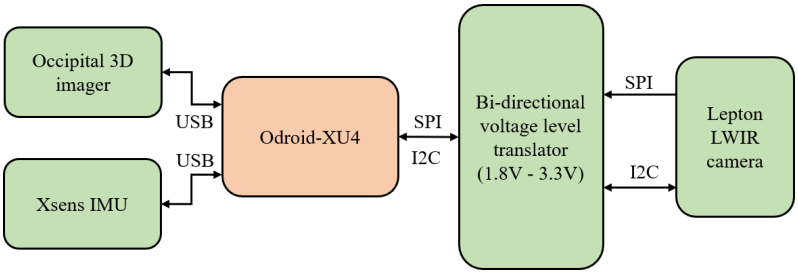


Figure 4.11: Sensor interface diagram

4.3 Flight Environment

This thesis proposes a perception payload for an alternate navigation method for small scale aerial robots in a GNSS denied environment. The indoor environment is considered as the target suitable flight environment as GNSS is only sparsely or not available. The underground steam tunnels at Ohio University and the hallways at Stocker Center are used as a test environment for this research. A picture of the steam tunnel and the hallway is shown in figure 4.12a and 4.12b, respectively.

The steam tunnel is a structured environment bound by walls on four sides. The space is rich in 3D features likes pipes, poles, outlets, holders, etc. Thermal heat signatures can be picked up from the pipes as steam and condensed water are flowing through them in separate pipes. The tunnels offer a tight space for the operation of the UAV with obstacles. The building hallways offer structured environment similar to the steam tunnels.



(a) Steam tunnels



(b) Structured hallways

Figure 4.12: Flight test environment

5 Sensor Calibration

As mentioned in the previous chapter, the platform payload consists of two cameras, namely the Occipital Structure camera and the Lepton LWIR camera. To use the cameras for navigation, the cameras have to be calibrated for their intrinsic parameters, their extrinsic parameters, and the latency/delay of the images w.r.t other sensors. This chapter discusses (i) the camera calibration procedure followed, (ii) image registration procedure between the 3D camera and the LWIR camera, (iii) and time delay between the 3D data / point clouds and IMU.

5.1 Pinhole Camera Calibration (Structure sensor and Lepton Camera)

Camera calibration in the context of computer vision is defined as the process of determining the camera's intrinsic (internal) and extrinsic (external) parameters. The intrinsic parameters are the internal geometrical and optical characteristics. The extrinsic parameters represent the 3-D position and rotation of the camera frame relative to the platform coordinate frame [83]. Furthermore, camera calibration includes the estimation of distortions due to the camera lens.

For our cameras, we use a basic pinhole camera model, which is shown in figure 5.1 and can be represented by equation (5.1). $\widetilde{x_i}$ and $\widetilde{y_i}$ are the coordinates of the projection of a point $\mathbf{p}$ given by $\begin{bmatrix} X & Y & Z \end{bmatrix}^T$ on the image plane [83].

$$\begin{bmatrix} \widetilde{x_i} \\ \widetilde{y_i} \end{bmatrix} = \frac{f}{z_i} \begin{bmatrix} x_i \\ y_i \end{bmatrix} \tag{5.1}$$

Representing the image points and 3D world coordinates as vectors with a 1 appended (i.e., the so-called homogeneous coordinates), the modified camera model is given by

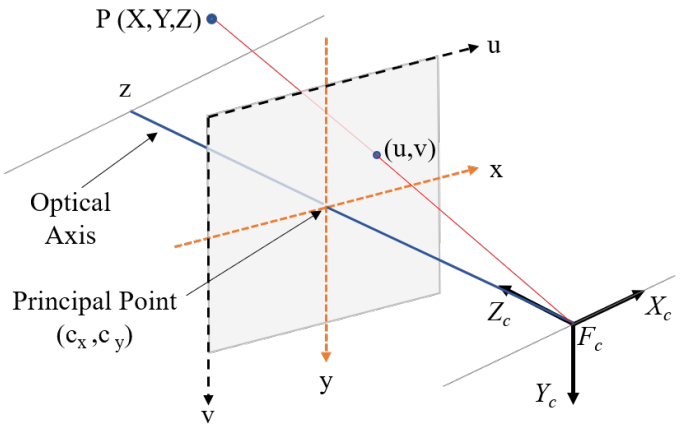


Figure 5.1: Pinhole camera geometry [4]

equation (5.2) [84][85]. (5.2).

$$s\begin{bmatrix}\widetilde{x}\\ \widetilde{y}\\ 1\end{bmatrix} = \begin{bmatrix}f_x & 0 & c_x\\ 0 & f_y & c_y\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}r_{11} & r_{12} & r_{13} & t_1\\ r_{21} & r_{22} & r_{23} & t_2\\ r_{31} & r_{32} & r_{33} & t_3\end{bmatrix}\begin{bmatrix}X\\ Y\\ Z\\ 1\end{bmatrix} \tag{5.2}$$

or, for short:

$$s\widetilde{\mathbf{m}} = \mathbf{K}[\mathbf{R}|\mathbf{t}]\mathbf{M} \tag{5.3}$$

The **K** matrix in equation (5.3) is known as the calibration matrix containing the intrinsic parameters, and **[R|t]** are referred to as the extrinsic parameters given by the rotation matrix and translation vector. The factor s is an arbitrary scale factor, $\widetilde{x}$ and $\widetilde{y}$ are the co-ordinates of projection points on the image plane, c_x and c_y are the principal points at the center of the image, f_x and fy are the focal lengths expressed in pixel units. Equation (5.3) explains the transformation of a point in 3D onto 2D image plane. Isolating the effect of the calibration matrix and the camera extrinsics, gives us equation (5.4) where **P** is the 4x3 camera matrix. The camera matrix, sometimes referred as projections matrix,

describes the projection of a point from world coordinate frame to image plane, whereas the camera extrinsics transforms the world coordinates into camera coordinates and the calibration matrix transforms the camera coordinates to image plane.

$$\mathbf{P} = \mathbf{K}[\mathbf{R}|\mathbf{t}] \tag{5.4}$$

The intrinsic parameters are specific to the camera and are constant given a selected lens and focal length setting. The extrinsic parameters, i.e., $[\mathbf{R}|\mathbf{t}]$, defines the motion of the camera around a static scene and include the lever arm and relative alignment to the motion platform [4].

The lens on the camera introduces distortion on the captured image. Distortion can be of several types, but the most significant ones are radial distortion and tangential distortion. An example effect of radial distortion is shown in figure 5.2 and the cause of tangential distortion is shown in figure 5.3. Radial distortion can be is approximated by equation



Figure 5.2: Lens distortion [4]

(5.5) and the tangential distortion is approximated by equation (5.6) where k_1, k_2 and k_3 are the coefficients for radial distortion, $r_i = \sqrt{\widetilde{x}_i^2 + \widetilde{y}_i^2}$ and p_1 p_2 are the coefficients for tangential distortion. The math behind the calculation of intrinsic and extrinsic parameter

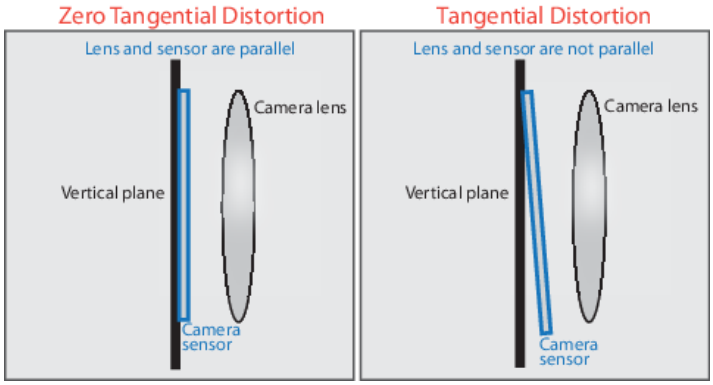


Figure 5.3: Distortion caused due to misalignment of sensor [5]

expressions are not shown here but can be found in [83][84][86].

$$\begin{bmatrix} \delta x_i^{(r)} \\ \delta y_i^{(r)} \end{bmatrix} = \begin{bmatrix} \widetilde{x_i}(k_1 r_1^2 + k_2 r_1^4 + k_3 r_3^6) \\ \widetilde{y_i}(k_1 r_1^2 + k_2 r_1^4 + k_3 r_3^6) \end{bmatrix} \tag{5.5}$$

$$\begin{bmatrix} \delta x_i^{(t)} \\ \delta y_i^{(t)} \end{bmatrix} = \begin{bmatrix} [2p_1 \widetilde{x_i}\widetilde{y_i} + p_2(r^2 + 2\widetilde{x_i}^2)] \\ [p_1(r^2 + 2\widetilde{y_i}^2) + 2p_2 \widetilde{x_i}\widetilde{y_i}] \end{bmatrix} \tag{5.6}$$

The camera calibration and 3D reconstruction tool box in OpenCV [4] is used to calculate the intrinsic and extrinsic parameters of the camera calibration. The lens distortion coefficients calculated are given in equation (5.7). The distortion model considered here is plumb_bob model. The camera calibration procedure calculates all the above-mentioned parameters and writes them into a .yaml file [87].

$$Distortion_{coefficients} = \begin{bmatrix} k_1 & k_2 & p_1 & p_2 & k_3 \end{bmatrix} \tag{5.7}$$

Image rectification for radial and tangential distortion is performed by applying equation (5.8). In the calculation of intrinsic and extrinsic parameters, functions D_x and D_y are used to convert the metric units in world coordinates to pixel units in image frame

[83]

$$\begin{bmatrix} \widehat{x_i} \\ \widehat{y_i} \end{bmatrix} = \begin{bmatrix} \widetilde{x_i}(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + 2p_1 \widetilde{x_i}\widetilde{y_i} + p2(r^2 + 2\widetilde{x_i^2}) \\ \widetilde{y_i}(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + p1(r^2 + 2\widetilde{y_i^2}) + 2p_2 \widetilde{x_i}\widetilde{y_i} \end{bmatrix} \tag{5.8}$$

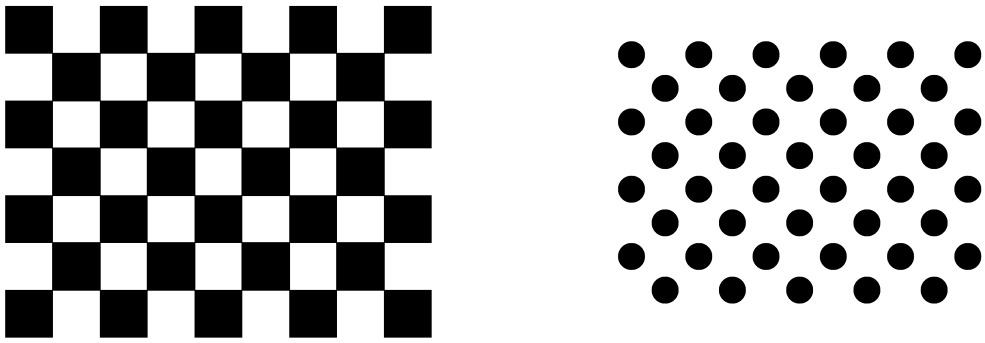
Two calibration patterns are widely used for calibration of pin hole cameras. These calibration patterns are the checkerboard pattern and the asymmetrical circle grid pattern and are shown in figure 5.4a and 5.4b, respectively. In this thesis, the checkerboard pattern is used for calibrating the intrinsic parameters of each camera and asymmetrical circle grid pattern for calculating relative position of the cameras w.r.t each other.

The reason asymmetrical circle grid pattern is used in computing the relative position of the cameras, is to limit the error that is induced due to the mis-detection of checker board corners by thermal camera. As discussed in 4.1.2, the thermal images are scaled up significantly, hence the thermal images are not crisp. Corner detectors are used to detect the checker board corners, which results in false corners in a smooth image. These false corners add errors into the computed relative pose. On the other hand, blob detectors are used in detecting the circle grids, which calculates the center and radius of the circle. The center of a circle detected by the blob detectors have smaller error in the scaled image compared to checker board pattern.

5.1.1 Procedure for camera calibration: Structure sensor

The Structure sensor is based on an IR camera. Hence, calibrating the IR camera also calibrates the depth images and, in turn, the point clouds. The calibration procedure is based on the procedure presented in [84] and detailed below.

- **Step 1:** For a clear and well exposed image, the image should be captured in an environment which is well lit with IR light (Ex: outdoors during the day) .
- **Step 2:** A checkerboard pattern with a known square size is printed out on paper and attached firmly on a hard planar surface that doesn't bend.



(a) Checkerboard pattern

(b) Asymmetrical circle grid pattern

Figure 5.4: Calibration patterns

- **Step 3:** Take several images of the checkerboard pattern in different orientations by moving the planar surface. Make sure that all the corners of the checkerboard pattern are seen in the image.

- **Step 4:** Perform checkerboard detection on all the pictures to detect the checker board corners and extract the feature points.

  Note: For estimating the intrinsic and extrinsic parameters, 52 images of extracted feature points are needed.

- **Step 5:** Estimate the camera intrinsic parameters, lens distortion coefficients and external parameters using extracted feature points and OpenCV libraries.

The calibration data is written to a .yaml file [87] which is read by the camera driver and published as the camera information message.

5.1.2 Procedure for camera calibration: FLIR Lepton camera

The FLIR Lepton camera is an LWIR camera that images thermal heat signature. Calibration of the Lepton LWIR camera is harder compared to other cameras because the

camera senses light in the LWIR spectrum, not the visible-light spectrum. The previously mentioned checker board is reused for the calibration procedure of the Lepton camera, but with one modification. The checkerboard is placed in the direct sunlight for 10-15 minutes and the board is allowed to heat up. Since the checker pattern has white and black squares, the black squares absorb more heat compared to the white squares, which results in the checkerboard pattern on the thermal images. Hence, the checkerboard exposed to direct sunlight is used for the calibration of LWIR camera. The resolution of the image is extremely low (80x60), for visibility and to ease the detection of the calibration patterns, the images are scaled up to VGA resolution.

5.2 Image Registration and Colored Point Cloud Generation

The rectified images from the camera can now be used in computer vision algorithms. Although the individual rectified images from both cameras provide a lot of information, extracting features from both the cameras, and referencing them to a common coordinate frame, i.e., simultaneous knowledge of the presence of a problem and its location would be more useful, (e.g., detecting a thermal signature on LWIR image and extracting the shape of the object on point clouds or the distance from the platform, etc.) . To find the epipolar line of a point in thermal camera on the depth camera, the relative pose of the cameras has to be calculated.

Image registration is the process of spatially aligning an image from one camera to an image of a second camera in a different coordinate frame such that the two images have pixel to pixel correspondence and are in the same coordinate frame [88]. For image registration, the rotation and the translation of the image frame to the reference image frame has to be calculated. This can also be considered as a limiting case of stereo camera calibration. Since the image registration is used as a tool, the registration algorithm and the epipolar geometry is not explained here but can be found in [4] and [89].

Since the geometry between the thermal camera and the depth camera is fixed, a point p in thermal camera can be related to p' in depth camera by equation (5.9). $\mathbf{F}$ in equation (5.9) is called the fundamental matrix and represents the epipolar geometry between the cameras algebraically [89]. The fundamental matrix can be calculated using the calibration matrices of the thermal ($\mathbf{K}$) and the depth camera ($\mathbf{K}'$) as shown in (5.10) [89], where $[\mathbf{K}\mathbf{R}^T\mathbf{t}]_x$ is a cross product matrix, $\mathbf{R}$ and $\mathbf{t}$ are the rotation and translation between the thermal and the depth camera.

$$p'^T\mathbf{F}p = 0 \tag{5.9}$$

$$\mathbf{F} = \mathbf{K}'^{-T}\mathbf{R}\mathbf{K}^T[\mathbf{K}\mathbf{R}^T\mathbf{t}]_x \tag{5.10}$$

To calculate $\mathbf{R}$ and $\mathbf{t}$, asymmetrical circle grid pattern is observed on both cameras, a pose of the object is calculated individually to each camera ($[\mathbf{R}_1|\mathbf{t}_1]$) and ($[\mathbf{R}_2|\mathbf{t}_2]$). Finally $[\mathbf{R}|\mathbf{t}]$ is calculated such that equation (5.11) is satisfied [4].

$$R_2 = R * R_1 t_2 = R * t_1 + t_2 \tag{5.11}$$

The essential matrix is given by 5.12 and the fundamental matrix is given by 5.13.

$$\mathbf{E} = \begin{bmatrix} 0 & -\mathbf{t}_2 & \mathbf{t}_1 \\ \mathbf{t}_2 & 0 & -\mathbf{t}_0 \\ -\mathbf{t}_1 & \mathbf{t}_0 & 0 \end{bmatrix} * \mathbf{R} \tag{5.12}$$

$$\mathbf{F} = \mathbf{K}'^{-T}\mathbf{E}\mathbf{K}^{-1}] \tag{5.13}$$

To perform the coordinate transform from one frame to the other, the Euler angles have to be calculated from the rotational matrix $\mathbf{R}$. The Euler angles can be calculated

using equation (5.14) where $\mathbf{R}_{31}$, $\mathbf{R}_{32}$, $\mathbf{R}_{33}$, $\mathbf{R}_{21}$, $\mathbf{R}_{11}$ are the elements of rotational matrix. The exceptions and the special cases are not discussed here but can be looked up in detail in [90].

$$\begin{aligned} \theta &= -\sin^{-1}(R_{31}) \\ \psi &= \arctan2\left(\frac{R_{32}}{\cos(\theta)}, \frac{R_{33}}{\cos(\theta)}\right) \\ \phi &= \arctan2\left(\frac{R_{21}}{\cos(\theta)}, \frac{R_{11}}{\cos(\theta)}\right) \end{aligned} \tag{5.14}$$

To perform the stereo calibration, the procedure followed is as given below.

- **Step 1:** Both structure camera and lepton camera have to observe the same calibration pattern (chess board or asymmetrical circles grid) on the image.
- **Step 2:** The calibration pattern points are extracted separately for each image. For the asymmetrical circle grid, the center of the circles is used as the observation points, whereas for the chessboard, the corners of the black squares are used as the observation points.
- **Step 3:** The observed points are used along with the known intrinsic and extrinsic parameters to calculate the transformation between the second camera to the reference camera.

The results of image registration are presented in the results section.

Note: For the camera calibration procedure, gray scale images from the Lepton LWIR camera are used. The gray scale images are pseudo-colored for visualization purposes, while generating colored point clouds.

5.3 Calculation of IMU and Point Cloud Delay

The platform uses an ARM based computer (Odroid XU4) as an on-board computer, which results in much processing delay while generating the colored point clouds from the

depth imagery. In order to align the IMU with the point clouds, calculation of the time delay between the IMU message and point clouds becomes vital. Figure 5.5 shows the timing sequence of the incoming messages and different processing times.

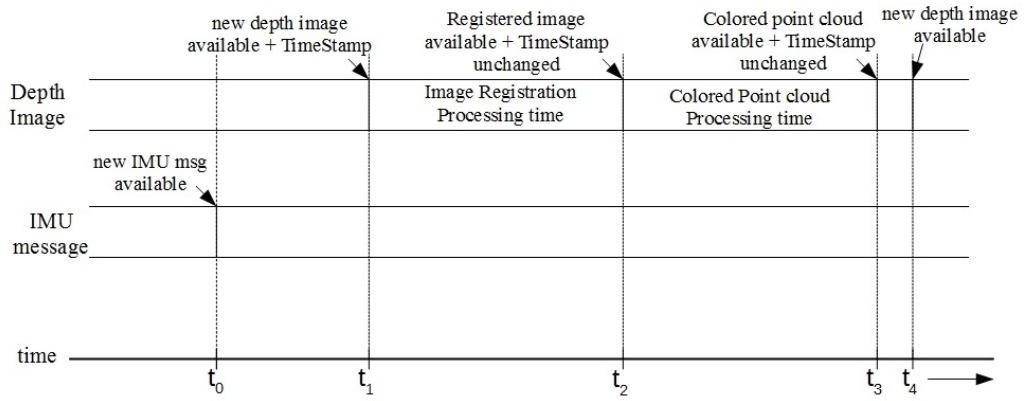


Figure 5.5: Timing diagram depicting the sequence of messages and associated timing delays

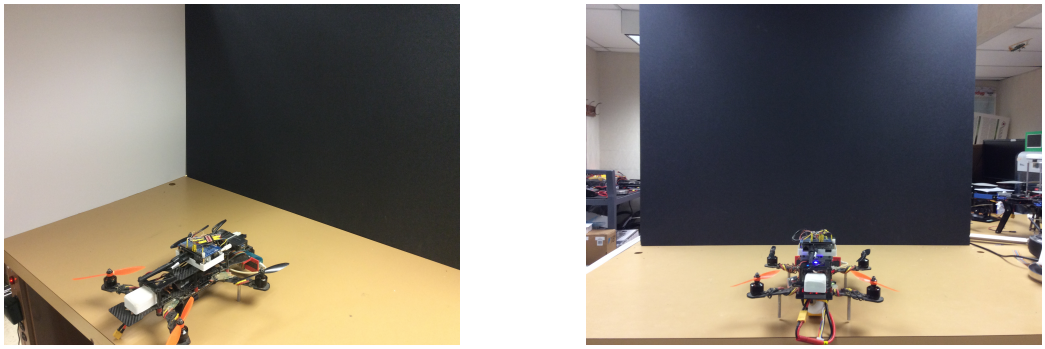


Figure 5.6: Setup for extraction of planar surface to calculate the time delay between point clouds and IMU

To calculate the delay between the IMU message and the point clouds, the platform is placed in front of a planar surface as shown in figure 5.6 and is given an oscillating

motion about the z axis (yaw). The change in yaw calculated from the IMU and the point cloud is plotted against time which would indicate a delay on the plot. The Euler angles are readily available from the IMU, but the change in yaw angle has to be estimated from the point cloud. To calculate the change in yaw from the point cloud, the planar surface parameters are obtained using RANSAC-based planar fitting method. The planar surface is mathematically represented by (5.15). The normal vector to the plane is given by $\begin{bmatrix} a & b & c \end{bmatrix}$, which are the coefficients of the planar equation. The details of the algorithm and the planar extraction is not discussed here and can be found in [91] and the implementation is found in [92].

$$ax + by + cz = d \tag{5.15}$$

As the platform rotates about the z axis, the angle of the plane changes and so does the normal vector to the plane. By tracing the change in the angle between the two vectors, the rotation of the platform can be determined. The angle between the two vectors is given by equation (5.16) where $\vec{n_0}$ is the normal vector at time epoch zero, $\vec{n_i}$ is the normal vector at the current time epoch, $||\vec{n_0}||$ is the length of the first normal vector and $||\vec{n_i}||$ is the length of the current normal vector.

$$\psi = \cos^{-1}\left(\frac{\vec{n_0} \cdot \vec{n_i}}{||\vec{n_0}|| \cdot ||\vec{n_i}||}\right) \tag{5.16}$$

Merely plotting the change in angles between the IMU and angles obtained from the observed point cloud is not sufficient to accurately calculate the delay between the two signs. To accurately calculate the delay mathematically, we calculate the cross-correlation between the two signals. Cross-correlation can be described as the similarity between two signals by time shifting one signal with respect to the other. This is represented by the equation (5.17) where x_k and y_k are periodic or stationary random discrete signals and y_k^* is the complex conjugate [93].

$$\mathbf{R}_{xy}(k) = \mathbb{E}[x_k y_{n-k}^*] \tag{5.17}$$

$$fs = \frac{\frac{length(imu\_time)}{imu\_time(n-1)-imu\_time(0)} + \frac{length(plane\_time)}{plane\_time(n-1)-plane\_time(0)}}{2} \tag{5.18}$$

To perform cross-correlation, the sequences have to have the same number of samples, hence the signal with lower sample rate are interpolated to match the signal with higher sample rate. The interpolation function is given by (5.19) where x_1, x_2, y_1 and y_2 are the known values and x is the query point.

$$y = y_1 + (x - x_1)\frac{y_2 - y_1}{x_2 - x_1} \tag{5.19}$$

Finally, the latency is calculated by the time of maximum cross-correlation value. Equations (5.20) and (5.21) describe the calculation of the latency between IMU and point cloud messages δ_t where R_{max} is the maximum correlation value, n_{max} is the sample number corresponding to R_{max} . The maximum correlation and value and its sample number is extracted from the correlation function, The sample number

$$[R_{max}, n_{max}] = max(f(n) \star g(n)) \tag{5.20}$$

$$\delta_t = n_{max} * fs \tag{5.21}$$

6 Sensor Pre-processing

The point cloud output by the 3D imager at each time epoch consists of approximately M= 307,200 3D points, $\mathbf{p}_i$, requiring significant computational resources to process all the points and extract the necessary features to perform navigation. To reduce the computational load and enable real-time processing on the onboard computer, the point clouds from the 3D imager are segmented to eliminate non-essential points and retain sufficient data to extract meaningful features from it. This section describes the segmentation algorithm used here.

6.1 Point Cloud Segmentation

The proposed data reduction procedure assumes that the UAS operates in a structured environment with some planar surface S (e.g., walls, floor, ceiling). Our proposed approach slices the point could at discrete distances along the 3D camera frames x-, y- and z-axes as described in 6.1.

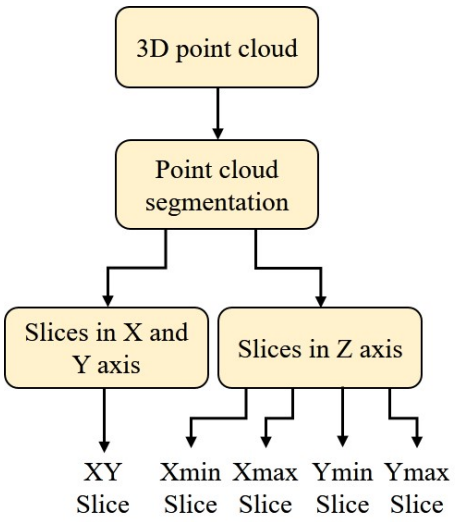


Figure 6.1: Block diagram describing the point cloud segmentation

N_z z slices are extracted by identifying the set of points whose z-coordinate fall within a z-value region:

$$S_{z,i} = \{\mathbf{p}_k | d_{z,i} - T_{zw} < z_k < d_{z,i} + T, k = 1, ..., M\} \tag{6.1}$$

where i is the slice index ($i = 1, ..., N_z$), $d_{z,i}$ is the center of the i^{th} region, and T_{zw} is the width of the window. In case more points are required for the estimate, the threshold value can be increased.

Each of the slices $S_{z,i}$ is further segmented into four regions i.e. $S_{z,xmin,i}$, $S_{z,ymin,i}$, $S_{z,xmax,i}$ and $S_{z,ymax,i}$. A given point is classified into the corresponding min max category if its closer than a threshold distance T_{zm} to the point. It can be represented by the following set of equations.

$$S_{z,xmin,i} = \{\mathbf{p}_k | x_k - x_{min} < T_{zm}, \mathbf{p}_k \in S z, i\} \tag{6.2}$$

$$S_{z,ymin,i} = \{\mathbf{p}_k | y_k - y_{min} < T_{zm}, \mathbf{p}_k \in S z, i\} \tag{6.3}$$

$$S_{z,xmax,i} = \{\mathbf{p}_k | x_{max} - x_k < T_{zm}, \mathbf{p}_k \in S z, i\} \tag{6.4}$$

$$S_{z,ymax,i} = \{\mathbf{p}_k | y_{max} - x_k < T_{zm}, \mathbf{p}_k \in S z, i\} \tag{6.5}$$

where x_{min} is the minimum x-coordinate of all point of $S_{z,i}$, etc.

Each of these min-max subsets of points are grouped together to give min and max points in each slice that now could represent a surface as shown in equations (6.6) through (6.9).

$$S_{z,xmin} = S_{z,xmin,i} + S_{z,xmin,i+1} + ... + S_{z,xmin,i+(n-1)} \tag{6.6}$$

$$S_{z,ymin} = S_{z,ymin,i} + S_{z,ymin,i+1} + ... + S_{z,ymin,i+(n-1)} \tag{6.7}$$

$$S_{z,xmax} = S_{z,xmax,i} + S_{z,xmax,i+1} + ... + S_{z,xmax,i+(n-1)} \tag{6.8}$$

$$S_{z,ymax} = S_{z,ymax,i} + S_{z,ymax,i+1} + ... + S_{z,ymax,i+(n-1)} \tag{6.9}$$

where $S_{z,xmin}$ is the collection of all $S_{z,xmin,i}$ points from all the slices and n is the total number of slices.

Slicing in the x- and y-axis, N_x, N_y slices are extracted by identifying the points whose x or y coordinates are closer than the threshold distance T_{dxy} to the slice window on the respective axis. The slice windows are defined as distinct x- or y-coordinate values equidistant from each other. The distance between each slice is evaluated by calculating the maximum range of the values possible in a particular axis and dividing the range by N_x or N_y slices. The values for slice window are the integral multiples of the distance between the slices summed with the minimum point. After all the points have been classified into x-slices or y-slices, both the clouds are then combined to give a point cloud with both x- and y-slices. This can be mathematically represented as follows:

$$d_{slice_x} = \frac{x_{max} - x_{min}}{n_x} \tag{6.10}$$

$$d_{slice_y} = \frac{y_{max} - y_{min}}{n_x} \tag{6.11}$$

$$[X_{val_pts}] = x_{min} + [1...n_x] \times d_{slice\_x} \tag{6.12}$$

$$[Y_{val_pts}] = y_{min} + [1...n_y] \times d_{slice\_y} \tag{6.13}$$

$$S_x = \{\mathbf{p}_k | ([X_{val\_pts}] - x_k) \exists < T_{dxy}\} \tag{6.14}$$

$$S_y = \{\mathbf{p}_k | ([Y_{val\_pts}] - y_k) \exists < T_{dxy}\} \tag{6.15}$$

$$S_{xy} = S_x + S_y \tag{6.16}$$

where d_{slice_x} and d_{slice_y} are the distances between the slices for x and y slices, respectively, n_x and n_y are the no of slices needed, S_x and S_y are the subset of points belonging to x- and y-slices respectively, S_{xy} are all the points that belong to xy slice.

This segmentation process yields a total of 5 segmented and sub-sampled point clouds $S_{z,xmin}$, $S_{z,ymin}$, $S_{z,xmax}$, $S_{z,ymax}$ and S_{xy}. These point clouds, except for S_{xy} usually represent a surface in a structured environment in the camera's b-frame. These categorized subsets of points significantly reduces the time taken to compute the planar surfaces using a method as explained in chapter 7. By varying the number of slices along x-, y-, z-axis, window

width T_{zw}, one can vary the extent of sub-sampling. Additional processing is required to extract planar surface from S_{xy} which is not parallel to x- and y-axis.

6.2 LWIR Image Segmentation

Since one of the main objectives of this thesis is develop a perception package that supports navigation in structured environments during maintenance and inspection tasks, one of the main functions of the LWIR camera is to find significant variations in the thermal signature of the environment, that would indicate air/gas leaks, bad insulation, or other phenomena with a thermal signature. This subsection explains the segmentation of a LWIR image and the extraction of thermal features like hot pipes and other hot-spots in the imagery. An example image from Lepton camera is shown in figure 6.2[94].



Figure 6.2: Example image from Lepton LWIR camera

The proposed feature extraction method is shown in figure 6.3. The first step in the feature extraction process is edge detection. The edges of the pipes, poles and other structures are obtained by applying Canny's edge detection filter on the image. The Canny edge detector consists of four steps. First, the given image is smoothed with a Gaussian

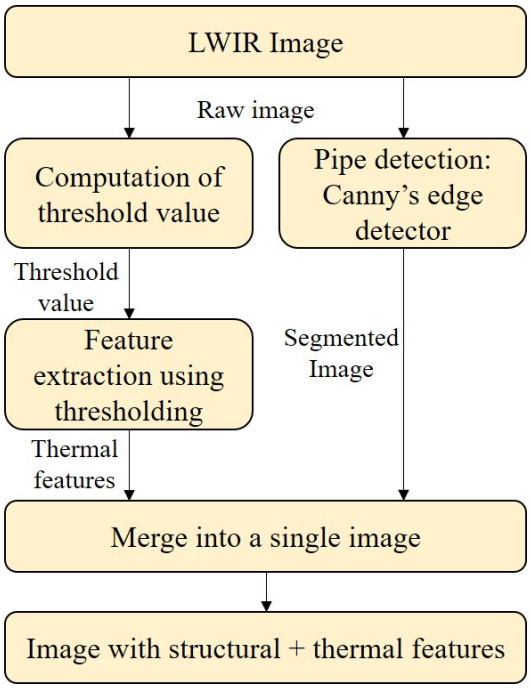


Figure 6.3: Functional block diagram of LWIR feature extraction method

function by convolution of the image with a 2D Gaussian function represented by (6.17)

$$f_s(x, y) = G(x, y) * f(x, y) \tag{6.17}$$

where $G(x, y)$ is the Gaussian function given in (6.18).

$$G(x, y) = e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{6.18}$$

The gradient magnitude of the image is given by equation (6.19) and the angle of the image is given by equation (6.20), where $g_x = \frac{\delta f_s}{\delta x}$ and $g_y = \frac{\delta f_s}{\delta y}$.

$$M(x, y) = \sqrt{g_x^2 + g_y^2} \tag{6.19}$$

$$\alpha(x, y) = tan_{-1}\left(\frac{g_y}{g_x}\right) \tag{6.20}$$

The gradient image contains the detected edge points along with the false edge points surrounding the true edge points. These false edges are removed by running a non-maxima suppression algorithm based on the angle obtained in equation (6.20). Finally, hysteresis thresholding is applied is applied to detect and link the edges [95].

The next important step is to extract the thermal features from the imagery to aid the detection irregularities in the tunnels. The thermal features are extracted by applying a adaptive thresholding technique presented in [96], where the pixel value of an integral image is compared with the pixels in an *nxn* neighboring window to check if the pixel of interest is *x%* greater than the rest of the pixels in the neighborhood. *x* here refers to the sensitivity of the algorithm.

Finally, the edge detected image is combined with thermal features extracted to provide spatial and thermal information on the features of interest.

7 Feature Extraction and Navigation Approach

This chapter discusses the feature extraction of pre-processed point clouds and the integration of these features into the navigation solution. An example of the point cloud output by the 3D imager in a relevant environment is shown in figure 7.1. It is clear from this figure that the observed environment can be considered structured (i.e., it consists of a mix of planar-like structures). The first step is to extract these relevant features from the point cloud. The first section discusses the feature extraction from the pre-processed point clouds and the following sections explains the use of these features in a navigation solution.

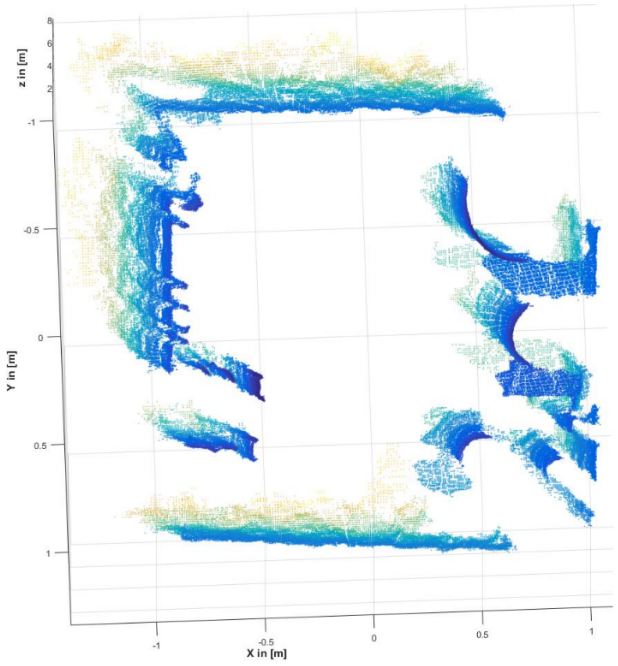


Figure 7.1: Example of point cloud at n^{th} time epoch

7.1 Feature Extraction

The point cloud pre-processing step was introduced to enable a real-time implementation of the feature extraction algorithm. The features of interest from the point clouds

are planes which are used for navigation. There are several methods to extract planes from point clouds namely least squares method, Singular Value Decomposition (SVD), etc, but one of the drawbacks of these methods are, the 3D fitted plane always include outlier points that makes the fitted plane erroneous. To calculate the best fitting plane and filter the outlier points, a robust feature estimator for planar surfaces is used along with RANSAC to smooth the solution and compute the best fitting model.

7.1.1 Random Sample Consensus

RANSAC is an iterative technique used to fit a model to a given data set [97] with outlier points. RANSAC uses a minimum data points (i.e., subset) required to fit a model and evaluates the obtained model against rest of the data set to find the best fit. This is contradictory to most other techniques that consider the maximum number of points possible.

RANSAC algorithm can be divided into 4 steps; each step is explained below. Consider a data set R with n data points such that $n > k$, where k is the absolute minimum number of data points needed to initialize a model that has to be fitted to R. A 2D example of this selection is shown in 7.2a, with a straight-line model.

Step 1: A subset of points R_1 containing k data points is randomly selected from R.

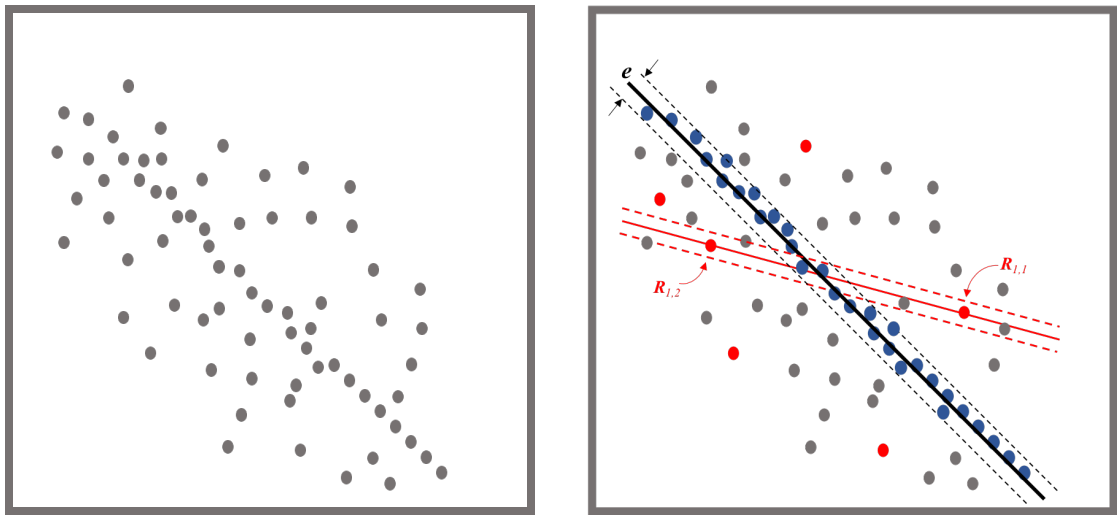
Step 2: A model X_1 is estimated from the subset R_1. Divergence between data points from R and the model X_1 is calculated to determine which data points from R conform with the model X_1 under the given error constraints e. This set of points that conforms with X_1l is referred to as the consensus set.

Step 3: If the number of points in the consensus set is greater than the threshold Δ, then a new model X_1' is calculated using a smoothing method from all the data points in the consensus set. The model X_1' is returned as the best fitting model to the data set R under the given constraints.

Step 4: If the number of points in the consensus set is less than the given threshold Δ, then a new subset X_2 containing k data points are randomly selected and the process is repeated all over again.

This process continues until a best-fit model is found or terminates when it has failed a predefined number of trails.

To better understand the algorithm, consider a sample set of points in 2D space as shown in figure 7.2a. A straight-line has to be fitted to the given data-set using RANSAC.



(a) Sample data-set of points in a 2D space

(b) Best fitting straight line to sample points

Recalling the steps, since the absolute minimum points needed to fit a straight line are two ($k = 2$), our first subset R_1 will contain 2 randomly selected points, say $R_{1,1}$ and $R_{1,2}$. Next, a line is calculated to with the points in R_1 and the consensus set is determined (shown as red solid line in the figure 7.2b) with the given error e. In this context, the error constraint would be the Euclidean distance of the point from the estimated line. Let the threshold points in the consensus set $\Delta = 22$. Recalling **step 4**, since number of points in the consensus is 9, the estimated model doesn't fit the criteria. A new subset R_2 is selected with 2 other randomly selected points and the process is repeated until the best fitting line

is determined. The black line in figure 7.2b represents the best fitted line model for the data set with all the inlier points colored blue.

One of the main parameters in RANSAC is the number of iterations / trials before returning failed. This is based on the expected value of trials i need to select j good data points from the data set. If w is the probability that the selected data points are within the error tolerance of the estimated model, expected value of k is given by equation (7.1) [97]

$$E(i) = \frac{1}{b} = w^{-j} \tag{7.1}$$

The parameter w in the equation (7.1) is important in this research. w decides the number of trials it takes to find the best fitting model, it also in-turn decides how long the algorithm takes to return the best fitting model. Higher the value of w lesser the number of trials and lesser is the time taken. A different criterion, given in Equation 7.2, can be used to improve the probability of fitting the model the points.

$$k = \frac{\log(1-p)}{\log(1-w^n)} \tag{7.2}$$

The point cloud pre-processing algorithm in the previous chapter, intends to do just that. Each min-max slice points mostly contains the points in the plane, which increases the probability of random points of a subset being a part of the estimated model.

A modified version RANSAC presented in [98] is implemented for extraction of planar features from the point clouds.

7.1.2 Extraction of planar surface in x- and y-axis

The planar extraction of surfaces perpendicular to x- and y-axis is pretty straight forward. The RANSAC algorithm explained in subsection 7.1.1 is applied on the segmented min-max point clouds i.e., x_{min}, y_{min}, x_{max} and y_{max} directly. Best fitting planar

surface is computed for each segmented point cloud and is expressed as a extracted feature. The only constraint that is exercised on the extraction is the error e. Section 8.3 presents the results of planar surface extraction of segmented min-max point clouds.

7.1.3 Extraction of planar surface in z-axis

The extraction of planar features is not as direct as extraction of planar surface from min-max point clouds. The extraction of planar surface perpendicular to z-axis is divided into two steps, (i) spatial subdivision of point clouds and (ii) aggregation of subdivided sections with similar planar surface.

Spatial subdivision of point cloud

Generally, point clouds can be classified into organized and unorganized point clouds. Points in organized point clouds have an order in a rows and columns similar to an image. Kinect and Occipital cameras generate an organized point cloud. This plays a key role in extraction of planar surfaces perpendicular to z-axis. Care is taken to maintain the organization of the point cloud during segmentation. The segmented point cloud with slices along x-axis and y-axis is further spatially subdivided into $m \times n$ rectangular regions as shown in the Figure 7.3. After the subdivision, planar extraction algorithm is applied to each subdivision separately, with the error constraint e and a maximum angular error constraint of e_a from being perpendicular to z-axis. This subdivision allows the algorithm to pick-up smaller planar features like columns, beam supports, sign boards, etc., which wouldn't be picked up when the planar extraction algorithm is applied to the point cloud as a whole.

Aggregation of Planar Surface

Running the planar extraction algorithm on $m \times n$ subdivisions, results in numerous planar surfaces, possibly up to $m \times n$ surfaces. But all these planes are not unique, and cannot

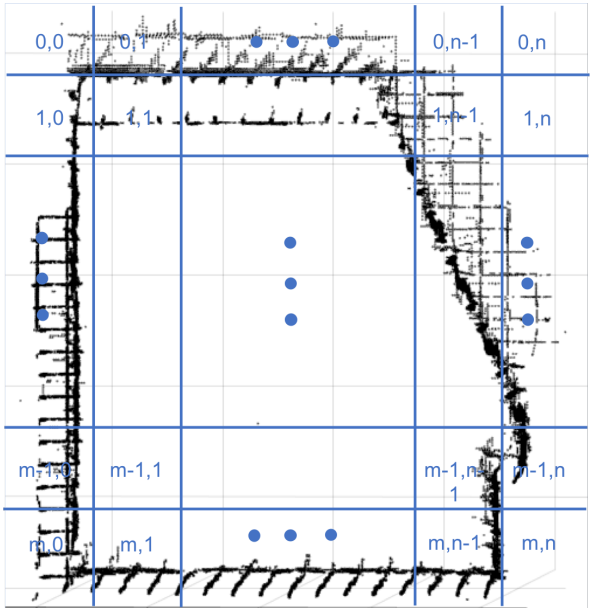


Figure 7.3: Spatial division of xy segmented point cloud

be considered as unique features. For example, consider a pillar in maintenance tunnels that runs from the ceiling to the floor. On the point cloud, the pillar runs across multiple subdivision, but all of them represent the same surface. In order to get unique features from the point clouds, similar planes from different subdivisions have to be aggregated.

$$ax + by + cz + d = 0 \tag{7.3}$$

Equation (7.3) represents a normalized planar equation, then (a, b, c) is the normal vector of unit length and d is the distance from origin. Since optical center of the camera is the origin, d is the distance of the plane from the 3D camera. After completing the planar extraction on each subdivision, plane extracted by each division is compared, and all the subdivisions with similar planes at the same distance from origin and similar angle are combined together. Inlier points from each plane is added and a more consistent plane

is computed which represents the entire planar surface. The extraction of planar surface perpendicular to z-axis is presented in section 8.3.

The number of subdivisions in a point cloud mainly depends on the environment. If the environment has big planar surfaces like huge sign board, partially wall, etc., then having fewer subdivision is helpful. On the other hand, if the environment has hallways, with minimal features, then larger number of subdivisions could be helpful.

7.2 Feature Integration and Navigation

The point cloud pre-processing procedure not only makes the plane estimation process faster and computationally less time consuming, but it also solves the feature association / plane association problem in the x- and y-axis direction.

For a reliable position using planar surfaces, three non-parallel planes are required. These three non-parallel planar surfaces are given by the xmax / xmin plane, ymax / ymin plane and the z planes. The relation between the incremental position, the closest distance to the planar surface and the planar surface normal vectors are given by equation (7.4) [99].

$$\Delta \mathbf{r} \cdot \mathbf{n}_j(t_i) = \rho_j(t_{i+1}) - \rho_j(t_i) = \Delta\rho_j \tag{7.4}$$

If three non-parallel planar surfaces are available, then this system of equations can be solved to obtain the translational motion via an ordinary least squares solution given by equation (7.5) where $\mathbf{H}$ is $\begin{bmatrix} n_i^{\mathbf{T}}(t_i) \\ \vdots \\ n_N^{\mathbf{T}}(t_i) \end{bmatrix}$ and $\Delta\rho$ is $\begin{bmatrix} \Delta\rho_1 \\ \vdots \\ \Delta\rho_N \end{bmatrix}$

$$\Delta\hat{\mathbf{r}} = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\Delta\rho \tag{7.5}$$

Given the planar surface normal vectors, estimates for the rotational motion can be obtained as well. This relation is given in expression (7.6) and (7.7).

$$\mathbf{C}_{t_i}^{t_{i+1}}\mathbf{n}_j(t_i) = \mathbf{n}_j(t_{i+1}) \tag{7.6}$$

$$\hat{\mathbf{C}}_{t_i}^{t_{i+1}} = argmin\left\{\sum_{\mathbf{j}=1}^{\mathbf{N}} \mathbf{C}_{t_i}^{t_{i+1}}\mathbf{n}_j(t_i\mathbf{n}_j(t_{i+1}))\right\} \tag{7.7}$$

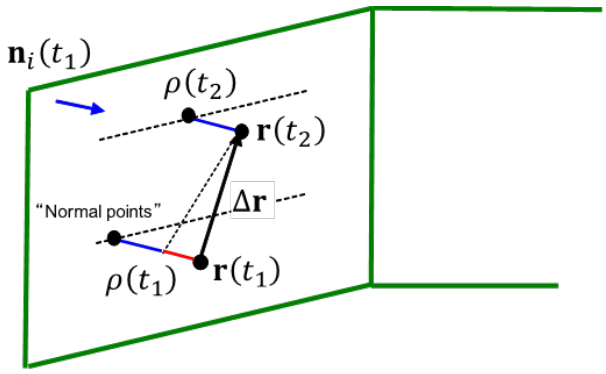


Figure 7.4: Planar surface-based navigation geometry

This estimate only needs two non-parallel surfaces and would work if one of the side walls and floor or ceiling is available. A method to solve equation (7.7) using quaternions is provided in [99].

When three non-parallel surfaces do not exist, the values for ρ_j and $\mathbf{n}_j$ and can still be used in the CKF mechanization of a tight laser-inertial integration approach by forming equation (7.8), where $\Delta\rho_{INS}$ is synthesized using equation (7.4) by substituting $\Delta\mathbf{r}$ by $\Delta\mathbf{r}_{INS}$.

$$z_{NCP} = \Delta\rho_{INS} - \Delta\rho_{DEPTH} \tag{7.8}$$

An alternate method to obtain estimates of the translational motion and rotation is the use of ICP [100]. An ICP-based method would require the full point clouds of two consecutive image frames and would, therefore, a significant amount of computational

resources. However, ICP-based methods could be investigated that use only the pre-processed points or a reduction via a voxel grid.

Alternatively, line and point features in 3D can be identified and associated from one frame to the next based on the difference between their projected location (using the translation and rotation estimates of (a) and (b) and associated uncertainty gates) and their observed location in the next frame. The odometry (translation) can be directly observed form the point features:

$$\textbf{Points}: \Delta\hat{\mathbf{r}} = \mathbf{p}_j(t_{i+1}) - \mathbf{C}_{t_i}^{t_{i+1}}\mathbf{p}_j(t_i) \tag{7.9}$$

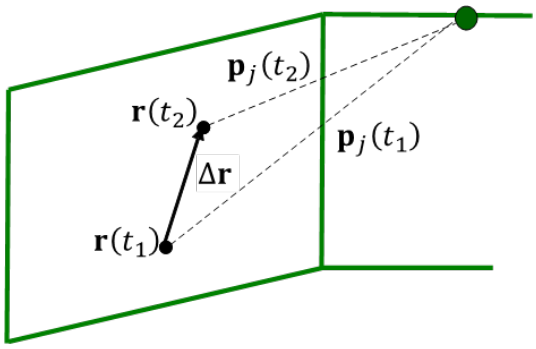


Figure 7.5: Point-based navigation geometry

In case of the line features, their orientation can be used to estimate the rotational motion between two consecutive frames in a manner similar to equation (7.5). An estimate of translation cannot be directly observed from one line segment as there is an ambiguity in the direction of the line, $\mathbf{l}$, and in a cylinder around the line. However, if two or more non-parallel lines are observed and rotated to align, an estimate of the translation can be obtained by finding the translation that maps the points on the lines in the next frame on the current frame.

8 RESULTS

This chapter presents the results achieved by implementing the proposed methods and algorithms in chapter 5, chapter 6 and chapter 7. Firstly, the calibration results are presented, followed by the segmentation results, and, finally, the feature extraction. The findings presented here are the result of applying the techniques proposed in previous chapters in real-time to real-time data collected from the platform described in chapter 4.

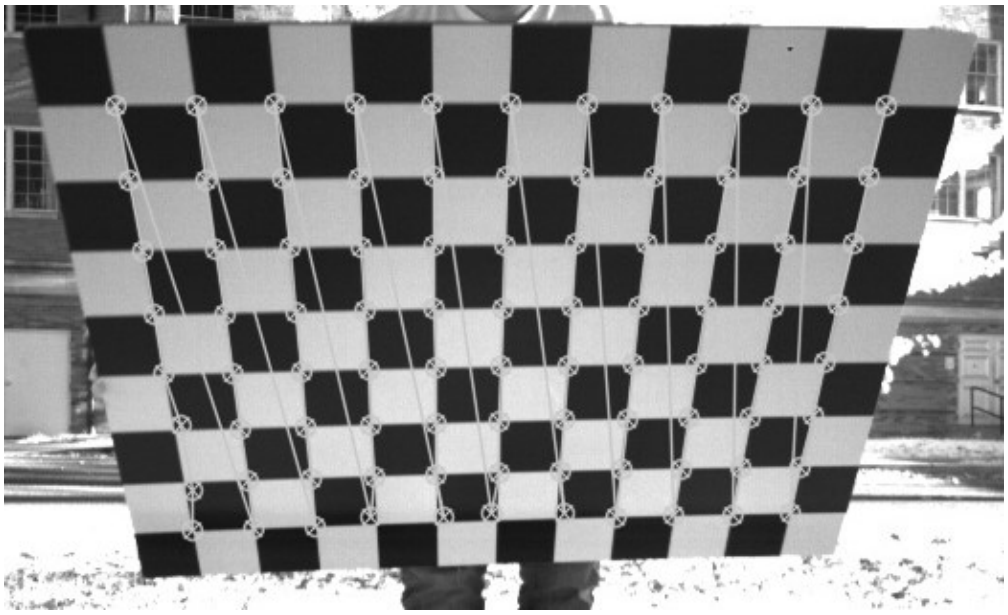
8.1 Calibration Results

The camera calibration procedures explained in the earlier section were implemented on the data collected from the UAS. The first test was the estimation of the lens distortion of Occipital 3D camera and Lepton LWIR camera. The second test is the estimation of point cloud latency from Occipital 3D camera. The final test under calibration is the estimation of delay between IMU messages and point cloud messages.

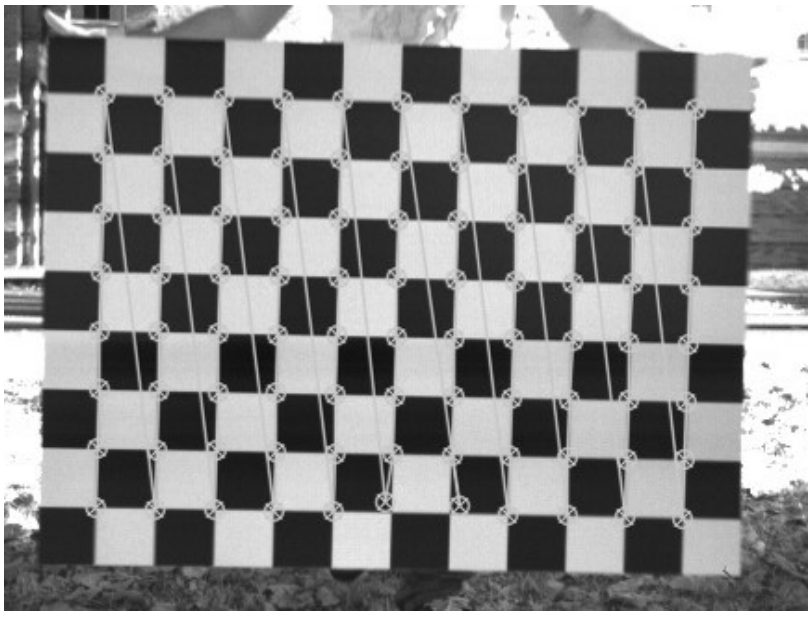
8.1.1 Estimation of lens distortion

This sub-sections presents the results of estimating the lens distortion. For the calibration, we used a checker board with a 109mm square size and an asymmetrical circle grid pattern with circle diameter of 20mm and 50mm were used.

Figure 8.1 shows the detected checker board pattern from the IR image of the Occipital camera and figure 8.2 shows the patterns detected by LWIR camera. Revisiting section 5.1, equation (5.3) defines Camera matrix and its relation with calibration matrix and rotation matrix, Where the calibration matrix defines the relationship of a point in camera coordinate frame to image plane and the rotation matrix / camera extrinsics defines the relationship of a point in 3D space to camera coordinate frame. For a monocular camera, the rotation matrix is always an identity matrix [101].

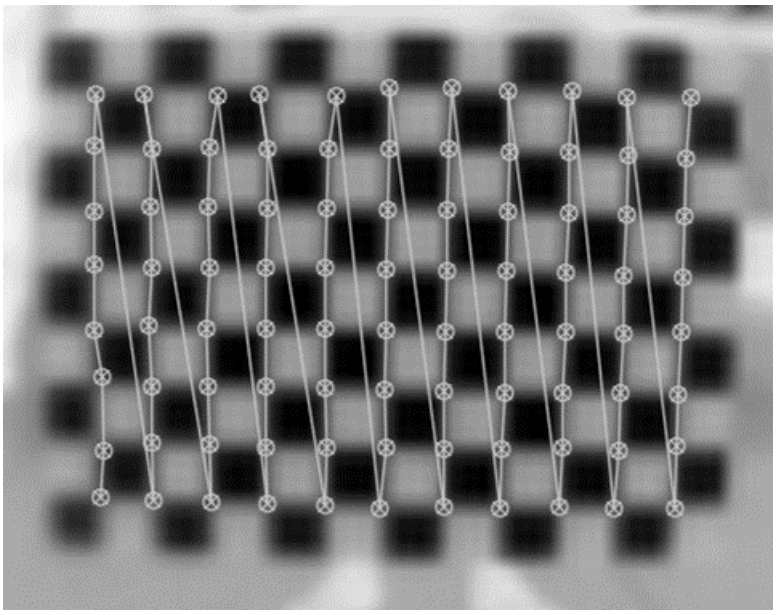


(a)

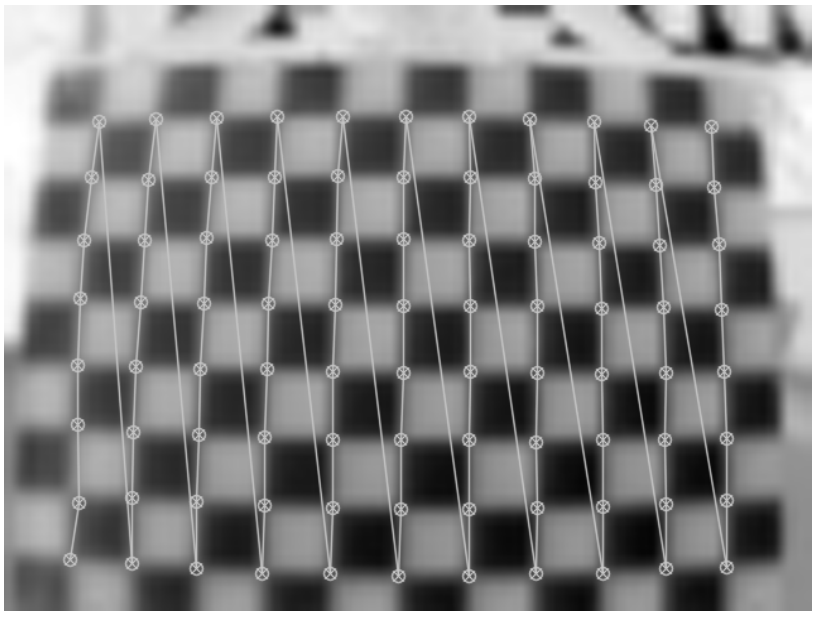


(b)

Figure 8.1: Calibration pattern detected by Occipital 3D camera in different orientations



(a)



(b)

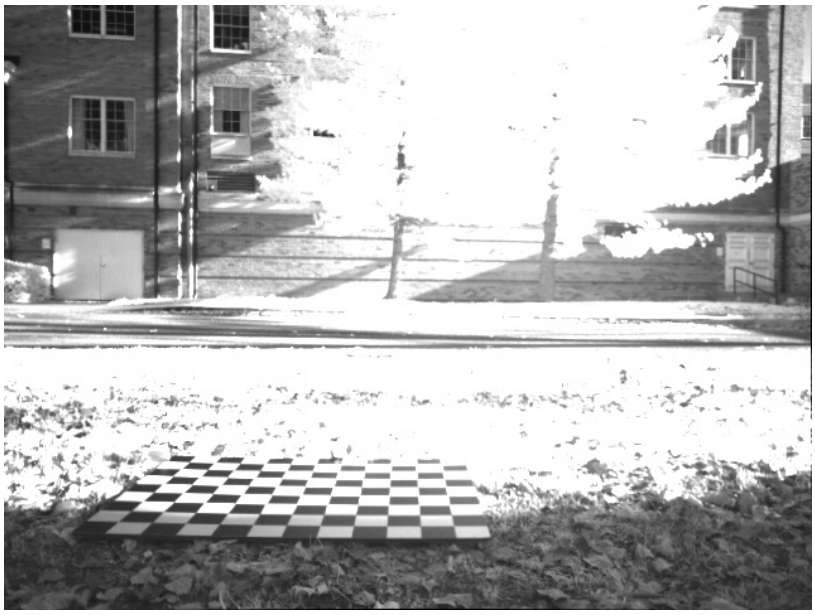
Figure 8.2: Calibration pattern detected by Lepton LWIR camera in different orientations

Properties	**Structure Camera**	**Lepton Camera**
Camera name	depth_PS1080_Primesense	flir_lepton
Image size	640×480	640×480
Calibration matrix	$\begin{bmatrix} 564.60 & 0 & 321.32 \\ 0 & 563.83 & 242.51 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 674.82 & 0 & 317.72 \\ & 673.85 & 235.39 \\ 0 & 0 & 1 \end{bmatrix}$
Distortion model	plumb_bob	plumb_bob
Distortion coefficients	$[\,-0.036\ \ 0.015\ \ -0.0012\ \ 0.0003\ \ 0\,]$	$[\,-0.371\ \ 0.297\ \ -0.004\ \ -0.001\ \ 0\,]$
Rotation matrix	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
Camera matrix	$\begin{bmatrix} 557.90 & 0 & 321.02 & 0 \\ 0 & 559.205 & 241.68 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$	$\begin{bmatrix} 621.60 & 0 & 316.23 & 0 \\ 0 & 641.33 & 233.37 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$

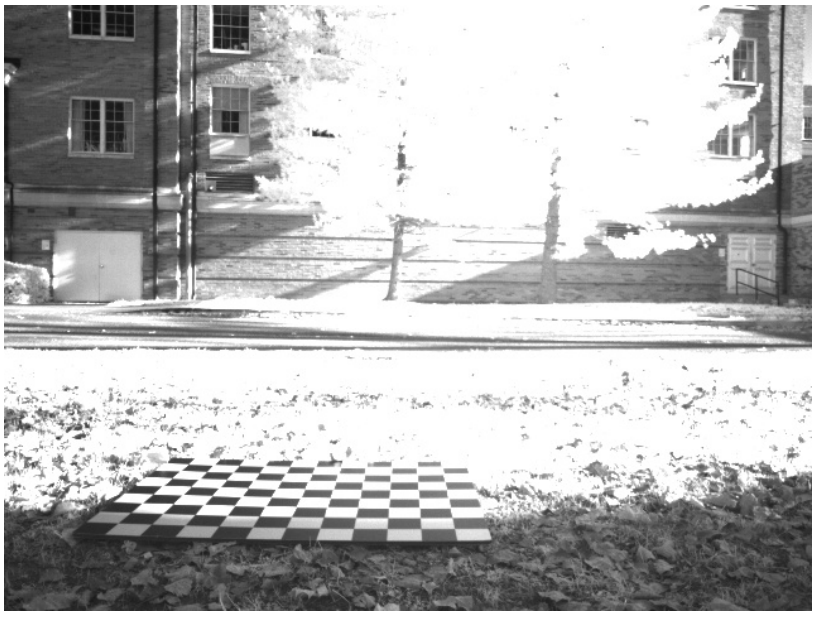
Table 8.1: Camera intrinsic and extrinsic calibration

Table 8.1 presents the results for lens distortion for both the Occipital 3D camera and the Lepton LWIR camera. The distortion coefficients presented in the table were estimated with more than 52 images where the patterns were detected at various distances from the image at different orientations.

Figure 8.3 and 8.4 shows the comparison between the un-rectified and rectified image of a scene using the estimated distortion coefficients given in Table 8.1. Upon close comparison between the un-rectified and rectified images, it is visible that the un-rectified images have barrel distortion in the image that was rectified using the distortion coefficients

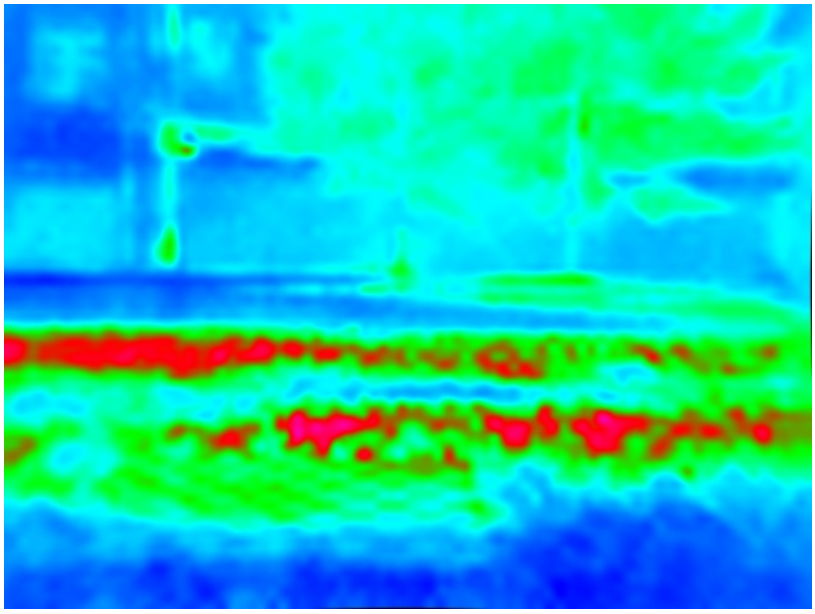


(a) Un-rectified IR image

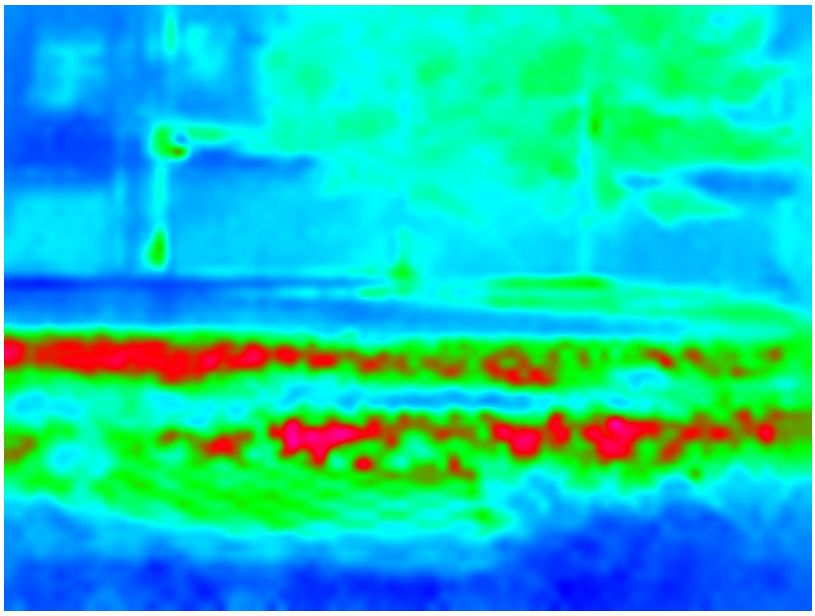


(b) Rectified IR image

Figure 8.3: Image from Occipital camera before and after rectification



(a) Un-rectified IR image



(b) Rectified IR image

Figure 8.4: Image from LWIR before and after rectification

in (5.8) in chapter 5. The lens on the Lepton camera seems to have higher barrel distortion compared to the lens on the Occipital camera.

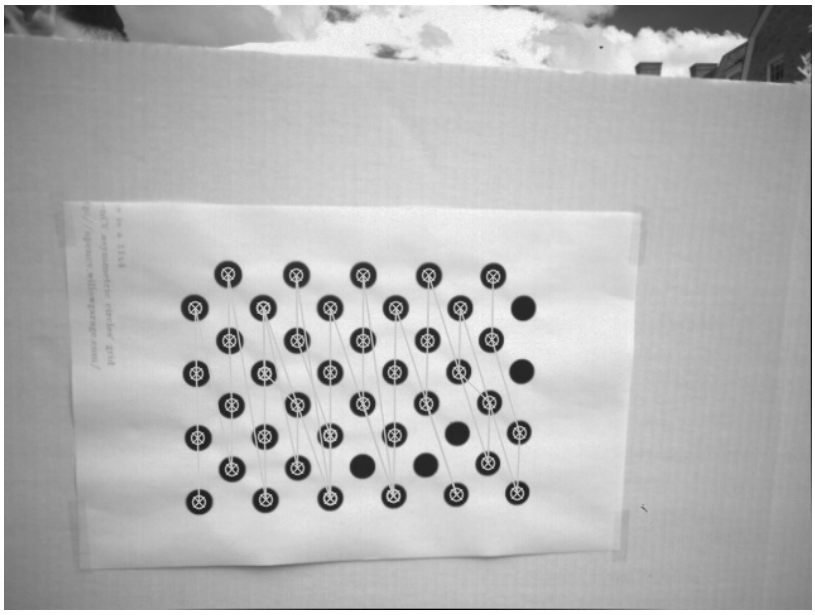
Errors

The estimated distortion coefficients seem to have some degree of error in their computation. One of the major sources of this error is the incorrect detection of the corner pixel checker pattern squares. Although this is less likely with the Occipital cameras, it is highly probable with LWIR camera, due to the image's blurry nature and the up-scaling operation (by a factor of 8) on the LWIR images, which makes the edges less prominent for edge and corner detection.

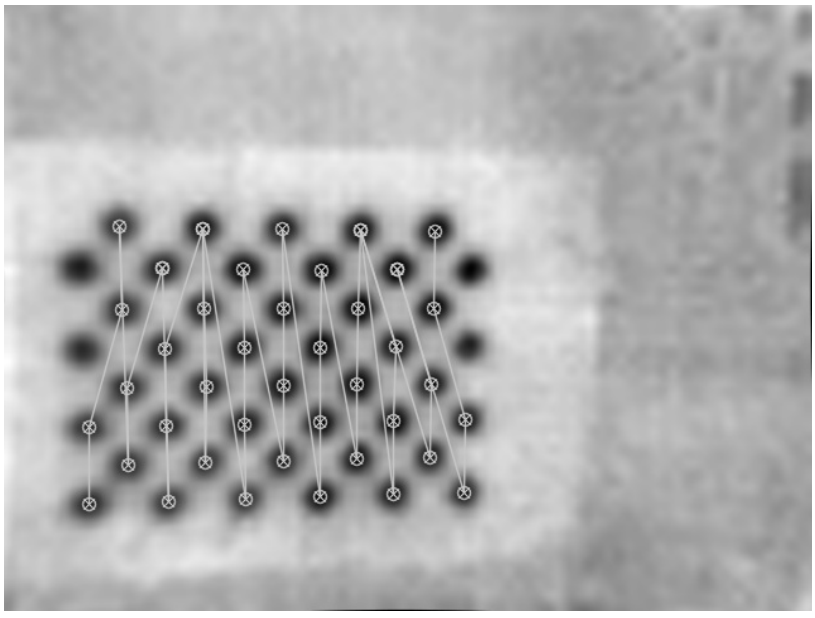
8.1.2 Estimation of relative pose of cameras

In this section, the results of the relative pose estimation between the Occipital camera and the Lepton camera are presented. Similar to the previous section, the patterns are detected by the camera and the relative position of the patterns on the image plane is used to calculate the projection of the image scene on the respective image planes and hence the relative poses of the cameras.

Figure 8.5 and 8.6 show the simultaneous detection of the asymmetrical circle grid pattern and detection of the checker board pattern by the Occipital and Lepton cameras. Similar to the estimation of the lens distortion, the detected points from more than 50 sample images along with the rotation and projection matrices from table 8.1 for both the cameras are used to estimate the rotation and translation of image plane of the Occipital with respect to the image plane of the Lepton camera. Revisiting section 5.2 discusses the theory behind the image registration using stereo camera setup. The fundamental and essential matrix describes the epipolar geometry between the lepton and Occipital camera . The estimation results of the rotation matrix and the fundamental matrix is given in table 8.2.

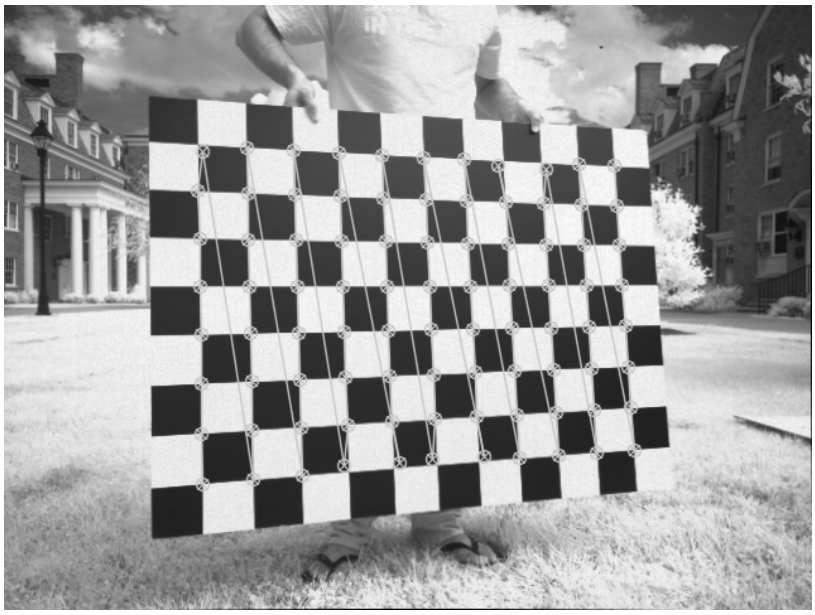


(a) Asymmetrical circle grid pattern detected in IR image

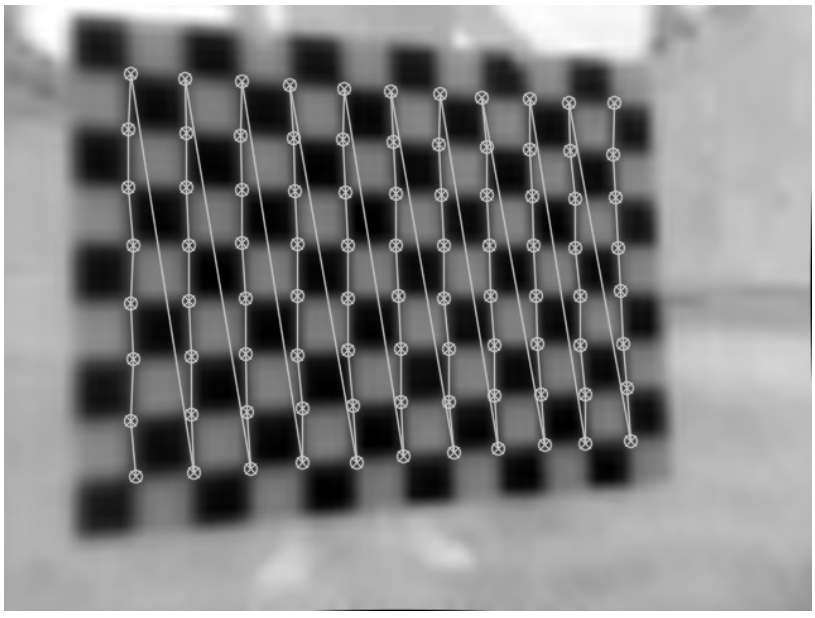


(b) Asymmetrical circle grid pattern detected in LWIR image

Figure 8.5: Detection of same asymmetrical circle grid pattern by both cameras



(a) Checker board pattern detected in IR image



(b) Checker board pattern detected in LWIR image

Figure 8.6: Detection of same checker board pattern by both cameras

Properties	**Estimations**
Rotation matrix	$\begin{bmatrix} 9.94e-01 & 3.38e-02 & 1.088e-01 \\ -3.57e-02 & 9.99e-01 & 1.73e-02 \\ -1.075e-01 & -2.10e-02 & 9.94e-01 \end{bmatrix}$
Translation vector	$\begin{bmatrix} -3.20e+01 & 2.66e+01 & 2.51e+02 \end{bmatrix}$
Essential matrix	$\begin{bmatrix} 6.11 & -2.51e+02 & 2.21e+01 \\ 2.46e+02 & 7.78 & 5.90e+01 \\ -2.53e+01 & -3.29e+01 & -3.43 \end{bmatrix}$
Fundamental matrix	$\begin{bmatrix} -1.39e-06 & 5.74e-05 & -1.65e-02 \\ -5.61e-05 & -1.78e-06 & 9.17e-03 \\ 1.76e-02 & -1.27e-02 & 1 \end{bmatrix}$

Table 8.2: Rotation and translation of structure camera w.r.t thermal camera

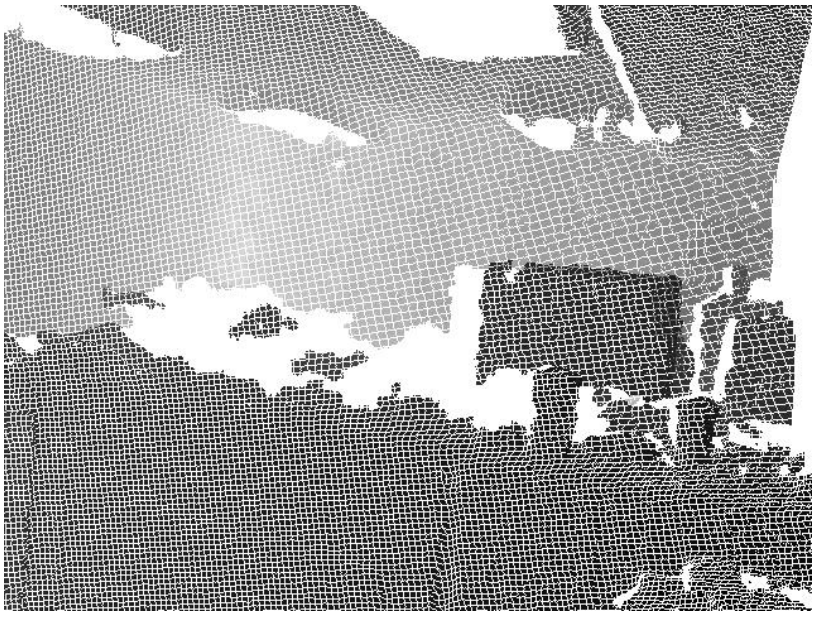
Note: The images with the detected patterns are essentially the same scene being recorded by both the cameras at the same time. The images from the two cameras with the same time stamps are paired for estimation.

Using the matrices and vectors from the table, the corrected range image from the Occipital camera and the rectified image from the LWIR Lepton camera can be registered resulting in pixel whose attributes consists of its position (x, y, z) and its temperature-based color $(reg, green, blue)$.

Figures 8.8 and 8.10 show the registered point clouds/thermal image based on the range images shown in figures 8.7 and 8.9, respectively. For comparison, the thermal images themselves are shown in figures 8.8a and 8.10a as well.

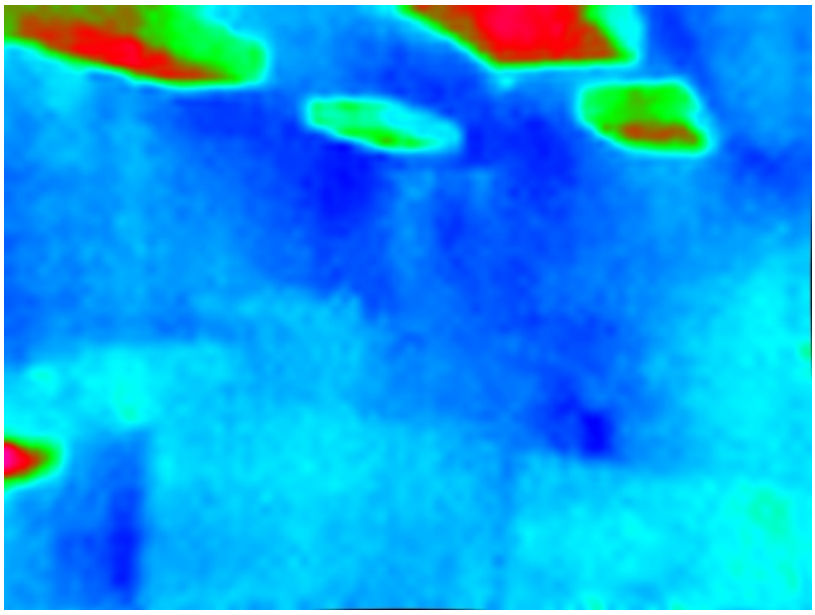


(a) Un-rectified range image

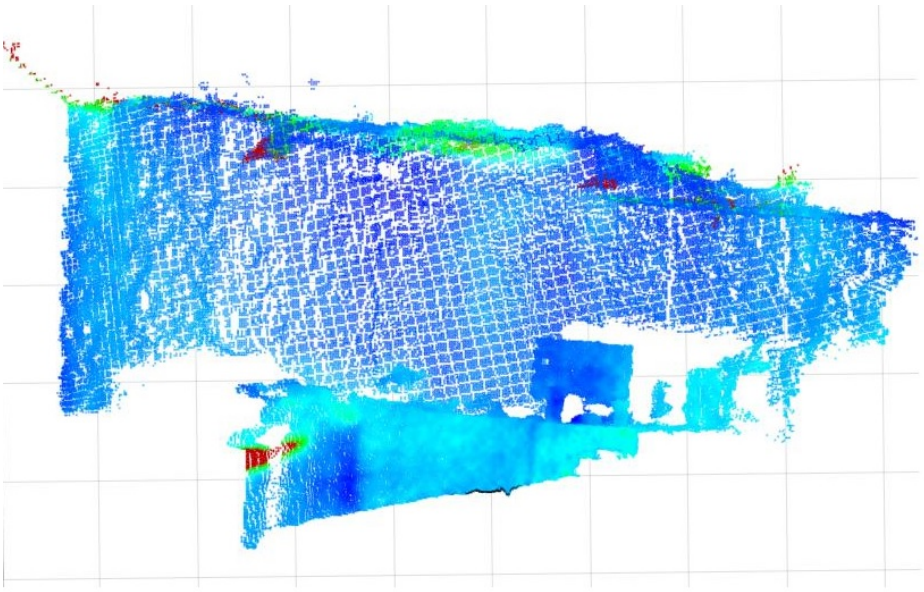


(b) Rotated and translated range image

Figure 8.7: Scene 1: Range image before and after rotation and translation



(a) Original thermal image



(b) Point cloud with corresponding thermal values

Figure 8.8: Scene 1: Image scene of a lab with few thermal features

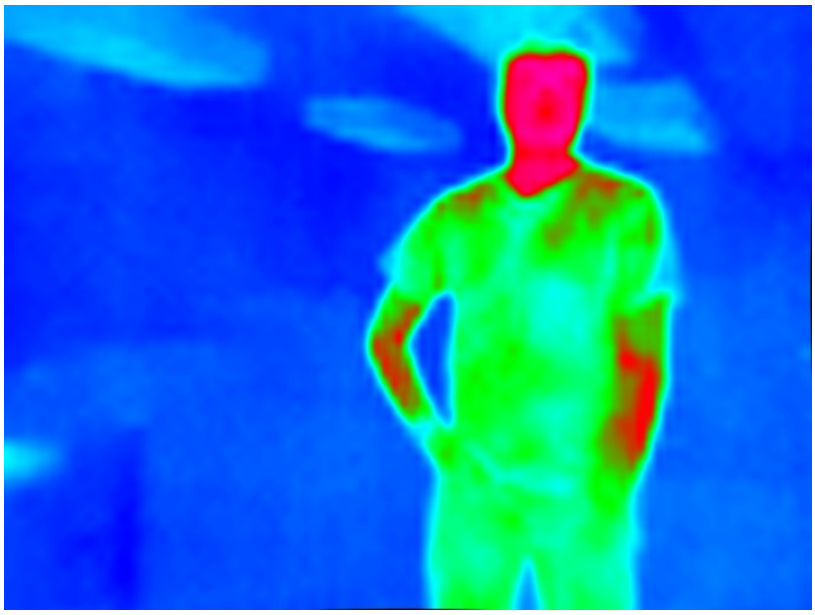


(a) Un-rectified range image

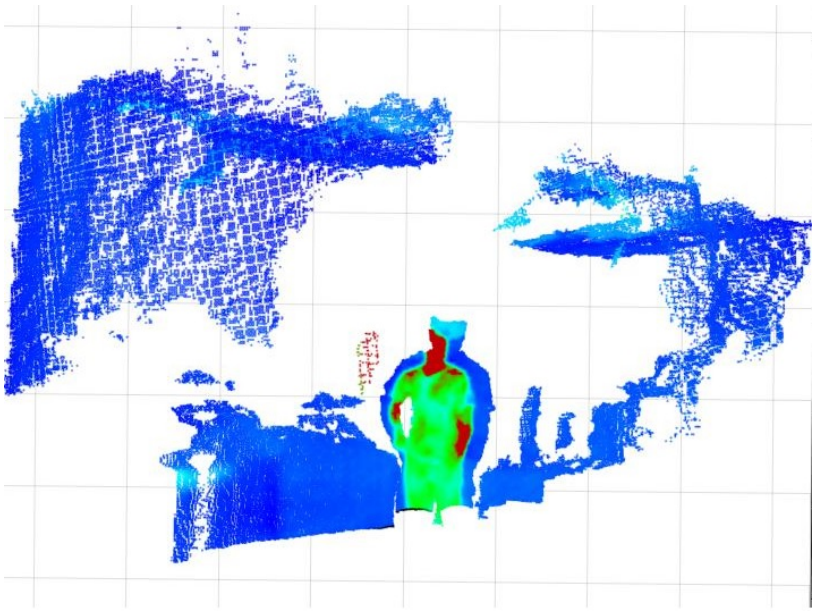


(b) Rotated and translated range image

Figure 8.9: Scene 2: Range image before and after rotation and translation



(a) Original thermal image



(b) Point cloud with corresponding thermal values

Figure 8.10: Scene 2: Image scene with significant depth and thermal signature

Errors

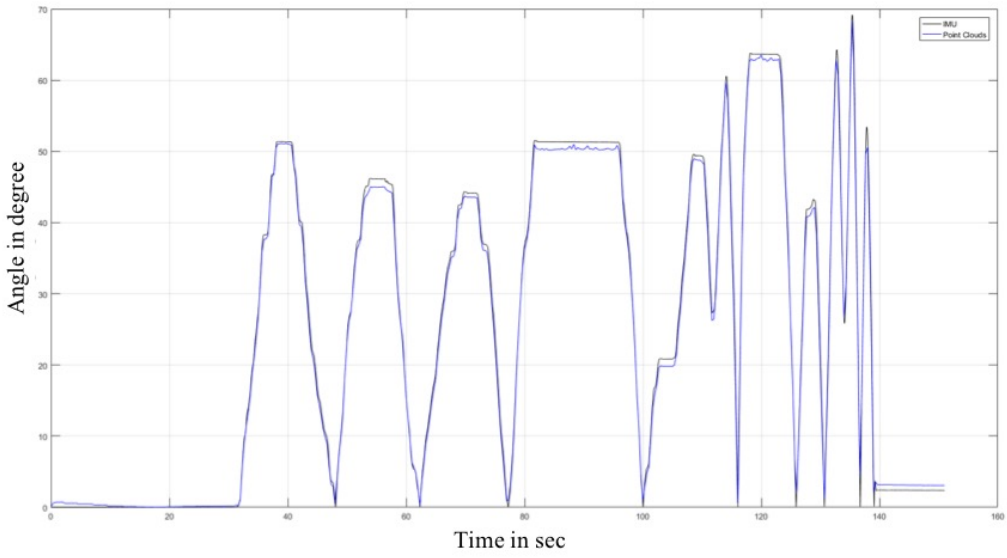
As observed in the figure 8.10b, there is a bit of scaling error in the images from LWIR camera and 3D camera. From the section 4.1, the viewing angle of the Occipital camera is 58 deg H and 45 deg V, which is larger than that of LWIR camera, this difference in the viewing angle caused the scaling error in the colored point clouds. Error in estimated camera parameters, lens distortion, Rotation and translation also contribute to the error observed. The error observed is small for the specified operating range of the 3D camera ($0.4m$ to $3.5m$), further adjustment is not made.

8.1.3 Estimation of delay between IMU and point cloud messages

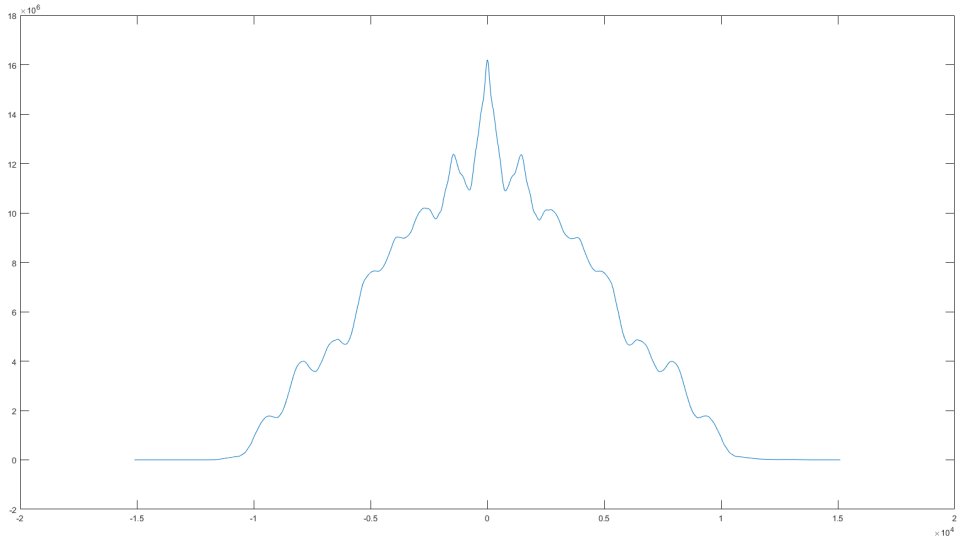
The delay estimates between the IMU messages and the corresponding point cloud messages are presented here. For every test run, the platform went through a power cycle to ensure independence of the runs. Table 8.3 shows the estimated delay for each run. The mean delay is calculated to be $23msec$ with a standard deviation of $4msec$. This delay is acceptable when the UAS is moving at $1m/sec$ or less. At higher dynamic rates, live delay estimation (for example, continuously comparing VO with IMU during UAS operation) have to be employed.

	Delay (in sec)
Run #1	0.020
Run #2	0.020
Run #3	0.024
Run #4	0.020
Run #5	0.030

Table 8.3: Timing calibration results

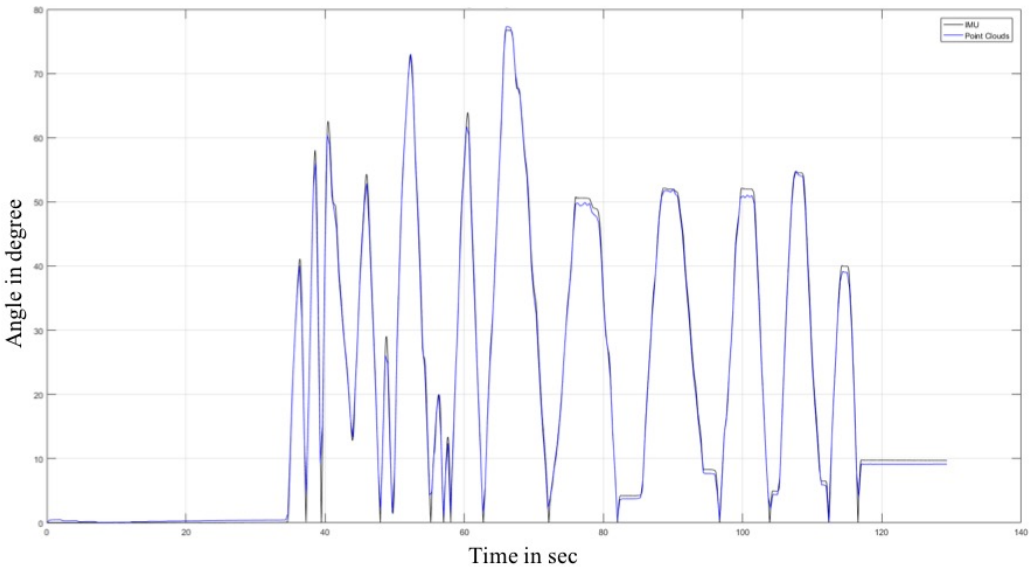


(a) Change in orientation of IMU w.r.t point cloud as a function of time

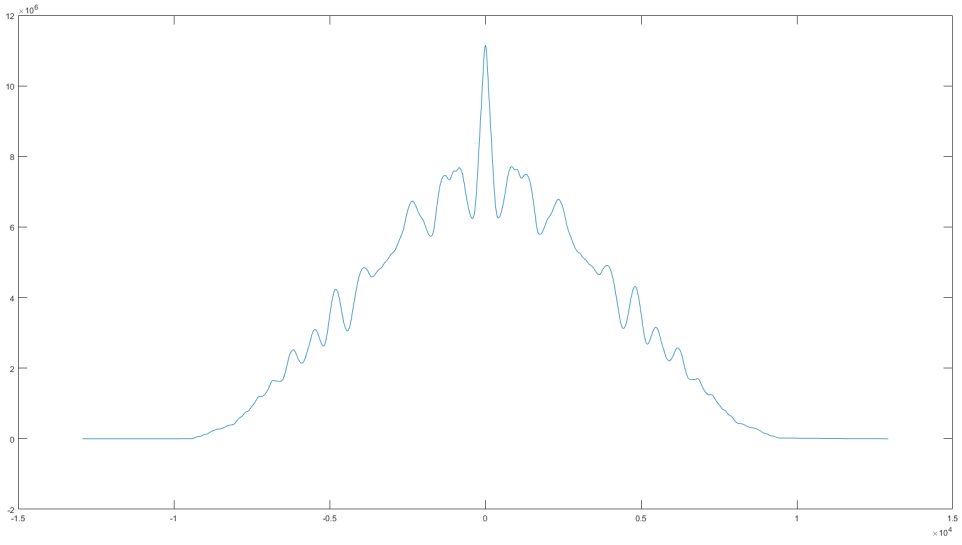


(b) Cross-correlation function

Figure 8.11: Estimation of latency between IMU and point cloud messages: Run 1

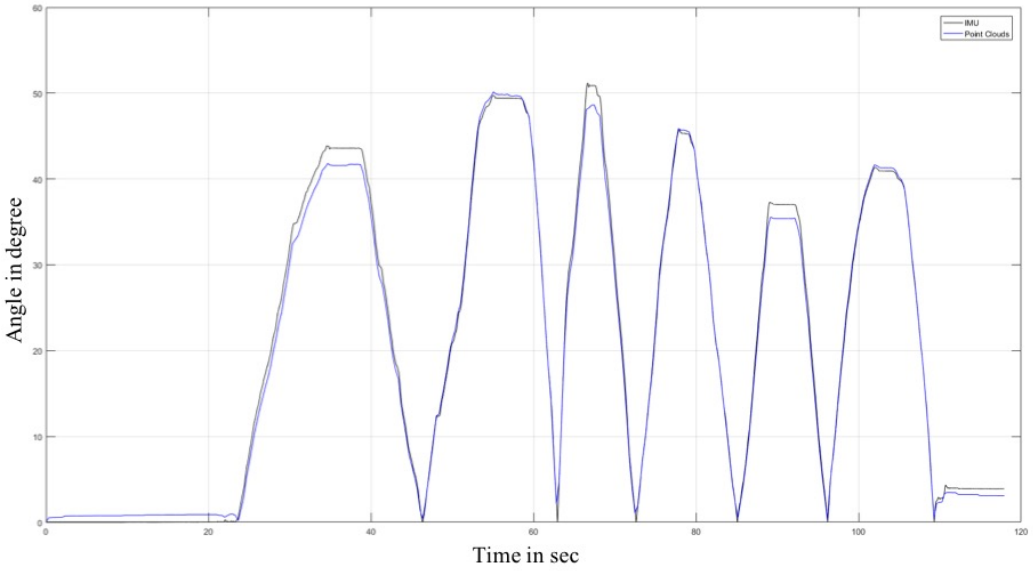


(a) Change in orientation of IMU w.r.t point cloud as a function of time

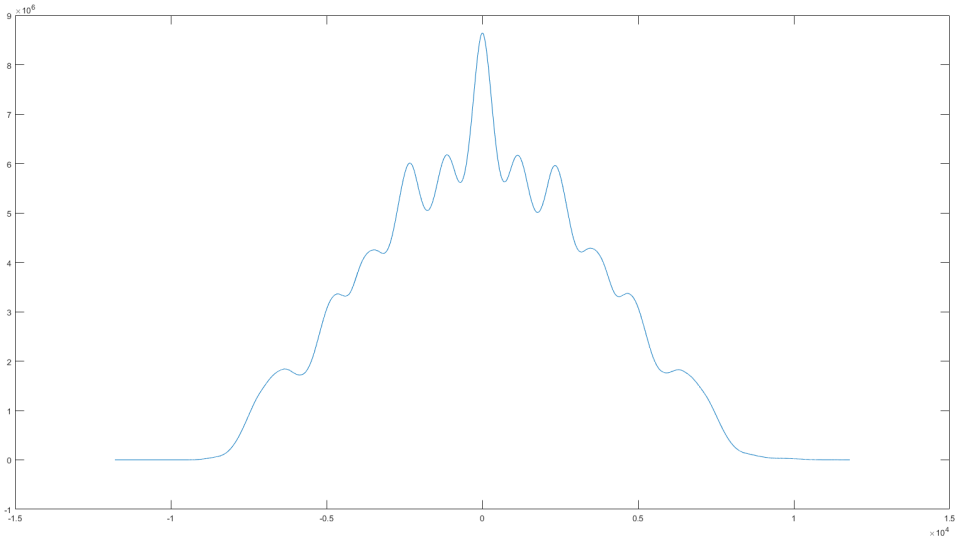


(b) Cross-correlation function

Figure 8.12: Estimation of latency between IMU and point cloud messages: Run 2

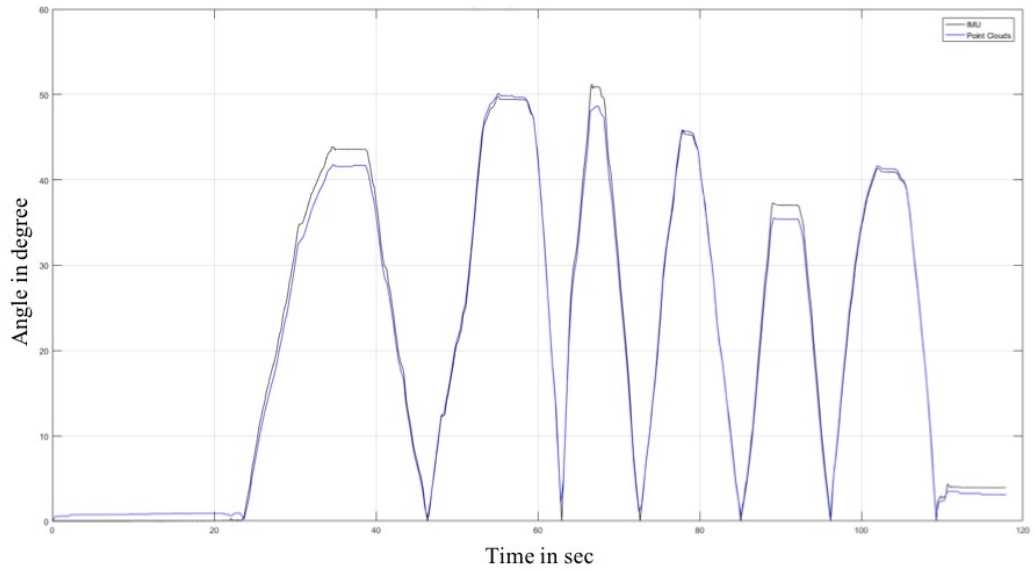


(a) Change in orientation of IMU w.r.t point cloud as a function of time

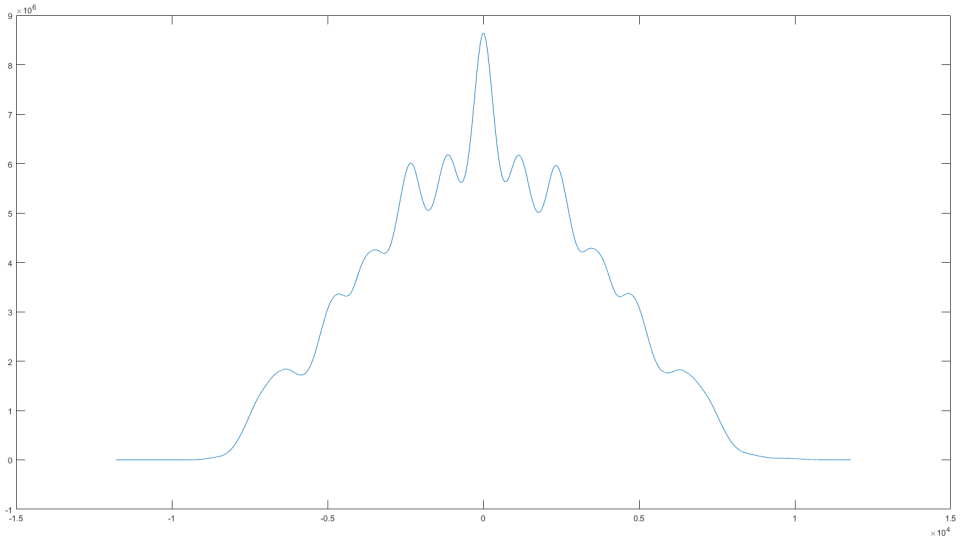


(b) Cross-correlation function

Figure 8.13: Estimation of latency between IMU and point cloud messages: Run 3

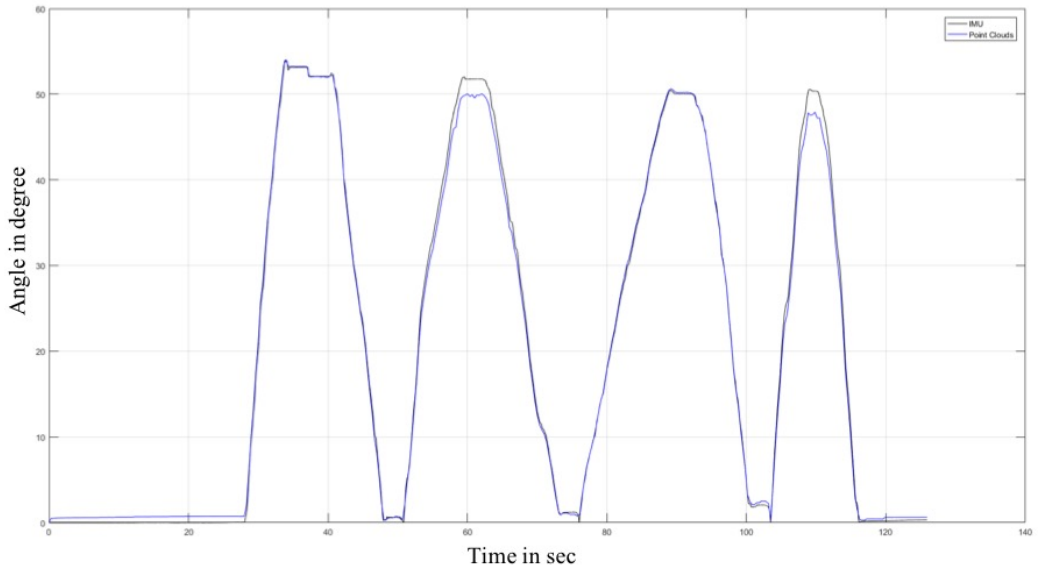


(a) Change in orientation of IMU w.r.t point cloud as a function of time

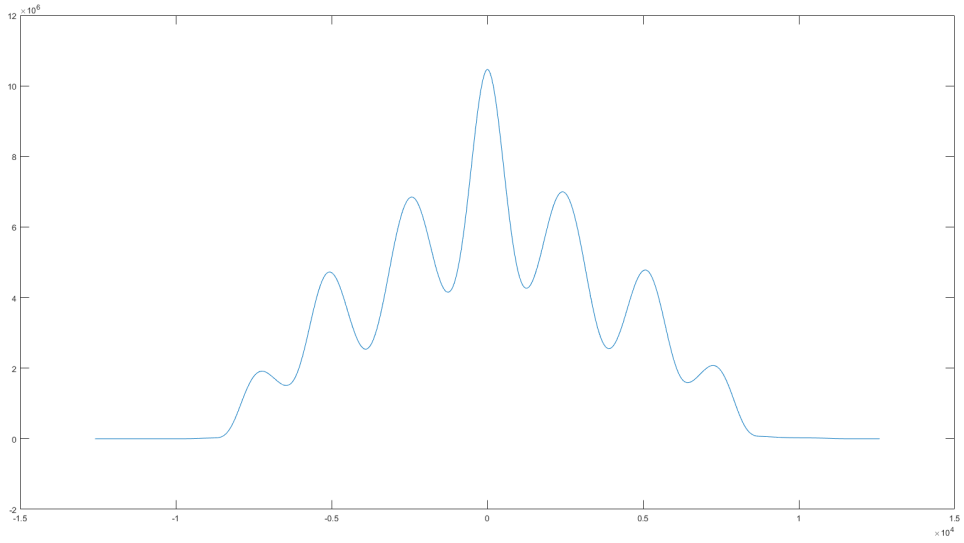


(b) Cross-correlation function

Figure 8.14: Estimation of latency between IMU and point cloud messages: Run 4



(a) Change in orientation of IMU w.r.t point cloud as a function of time



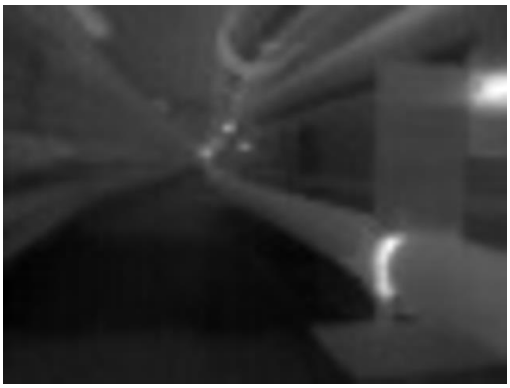
(b) Cross-correlation function

Figure 8.15: Estimation of latency between IMU and point cloud messages: Run 5

Figures 8.11a through 8.15a show plots for the change in orientation about the z-axis as a function of time based on the IMU and PC, respectively. Figures 8.11b through 8.15b show the cross-correlation function.

8.2 LWIR Image Segmentation

This section presents the results of applying the LWIR segmentation algorithm explained in chapter 6.



(a) Raw image

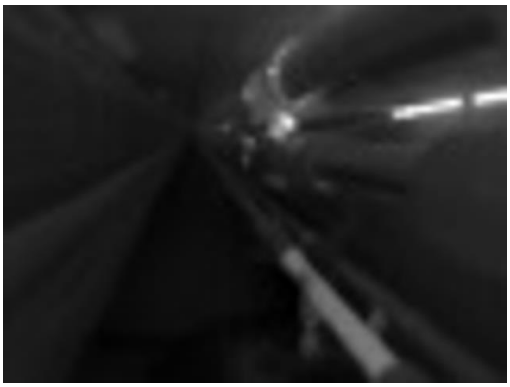


(b) Images with extracted physical features

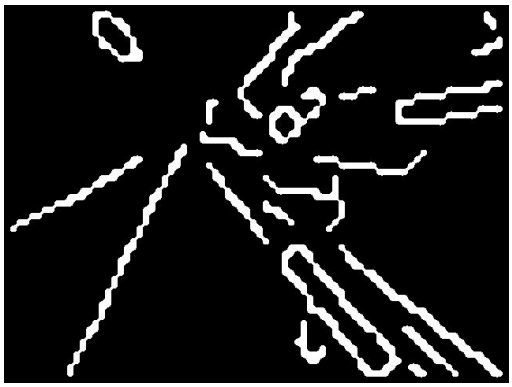


(c) Extracted thermal features with physical features

Figure 8.16: Image segmentation and feature extraction: Time epoch 1



(a) Raw image



(b) Images with extracted physical features



(c) Extracted thermal features with physical features

Figure 8.17: Image segmentation and feature extraction: Time epoch 2

Figure 8.16 through 8.18 show the results for detecting various physical features and thermal features in the maintenance tunnels. Whereas figures 8.16a, 8.17a and 8.18a show the unprocessed 16 bit image from the LWIR camera, figures 8.16b, 8.17b and 8.18b show the features extracted. Although the resolution is coarse (due to the limited resolution of the LWIR camera), feature like poles, pipes, and other support structures are easily identified. Finally, figures 8.16c, 8.17c and 8.18c show the features extracted with their thermal signatures superimposed.



(a) Raw image



(b) Images with extracted physical features



(c) Extracted thermal features with physical features

Figure 8.18: Image segmentation and feature extraction: Time epoch 3

Specifically, figure 8.16c shows a detected pole along with the pipes; figure 8.17c shows a hot steam pipe without insulation; and figure 8.17c shows a steam leak, with condensed water on the floor. Such features extracted from the LWIR imagery would be helpful to detect anomalies in the tunnels during inspection.

8.3 Point Cloud Segmentation and Feature Extraction

This section presents the combined results of the point cloud pre-processing algorithm and RANSAC planar feature extraction algorithm explained in chapter 6 and chapter 7

respectively. The pre-processing algorithm, although developed for the maintenance tunnels, has also been applied to other similar flight environments. This pre-processing algorithm does run in real time for the data collected by the platform at 3Hz (3 frames-per-second). On an Intel i5 computer, the execution time of the segmentation algorithm is about 40ms - 50ms, whereas on an Odroid XU4 it is about 70ms - 80ms when compiled with optimization level 2 or simply called as O2 optimization. The parameters used for the pre-processing are given in table 8.4.

The parameters used for plane extraction and estimation are provided in table 8.5 where t is the distance threshold for plane extraction as explained in chapter 7 and T_a is the angular threshold for plane parallel to z axis of the image plane. The planar extraction algorithm when implemented on an Intel i5 computer takes about 60 ms - 70 ms and about 100 ms - 110 ms on the Odroid XU4. Hence, to implement the proposed pre-processing and planar extraction algorithm is takes approximately 100 ms - 130 ms on an Intel i5 processor based system and 170 ms - 200 ms on an Odroid XU4 embedded pc.

Parameters	**Value**
T_{zw}	$\pm 2.5cm$
N_z	4
d_z	$[1.5, 2.0, 2.5, 3.0, 3.5, 4.0]m$
T_{zm}	$\pm 5cm$
$N_x = N_y$	25
T_{dxy}	$\pm 0.5cm$

Table 8.4: Pre-processing parameters

Figure 8.19 shows the legend for point cloud segmentation and feature extraction results. The first set of results are presented in figures 8.20 to 8.22 and show the application

Parameters	Value
e	$5cm$
e_a	20 deg
$m \times n$	8x8

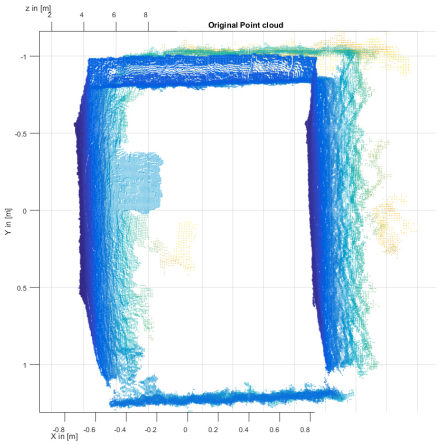
Table 8.5: Segmentation parameters

Color	Category
	Xmin points / clouds
	Xmax points / clouds
	Ymin points / clouds
	Ymax points / clouds
	Z points / clouds

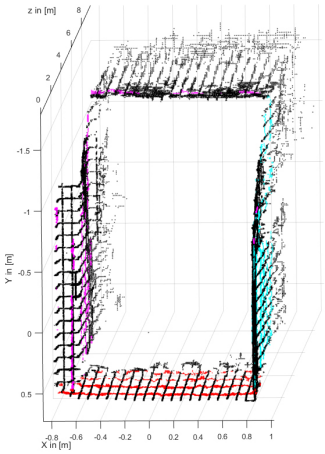
Figure 8.19: Legend for points and planes in point cloud segmentation and feature extraction

of segmentation and feature extraction to 3D data collected in the hallways of 2nd floor at Stocker center at three different time epochs. The second set of results (figure 8.24 through figure 8.26) show the application of the segmentation and planar feature extraction to the data from maintenance tunnels and figure 8.27 through figure 8.27 shows the results for data from 3rd floor Stocker hallways with inclined walls.

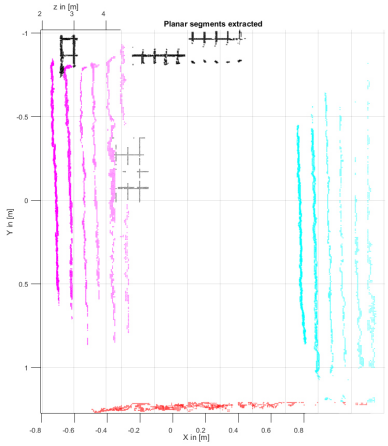
From the results presented, it can be observed that the proposed pre-processing algorithm, significantly down-samples a given point cloud without losing key details of the environment. The method, furthermore, categorizes subsets of points based on their



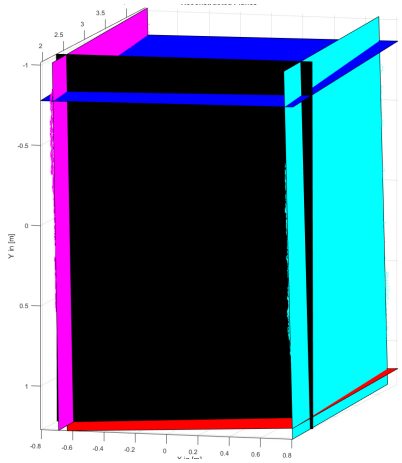
(a) Original point cloud



(b) Segmented point cloud



(c) Inlier points of a plane

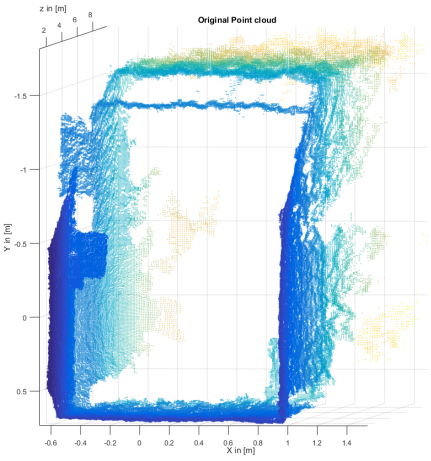


(d) Reconstructed planes from the inlier points

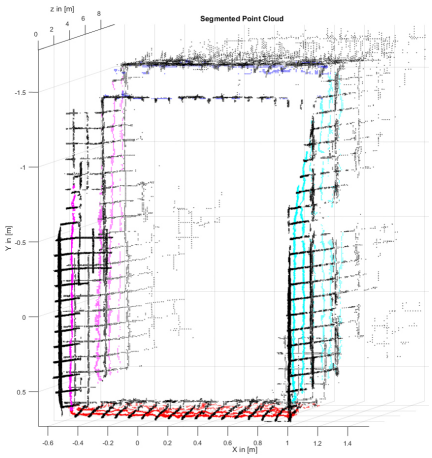
Figure 8.20: Point cloud segmentation and extraction: Stocker 2nd floor, time epoch 1

geometrical location (i.e., left-minimum, left-maximum, etc.) in the image plane, which makes it easier to extract the features from them.

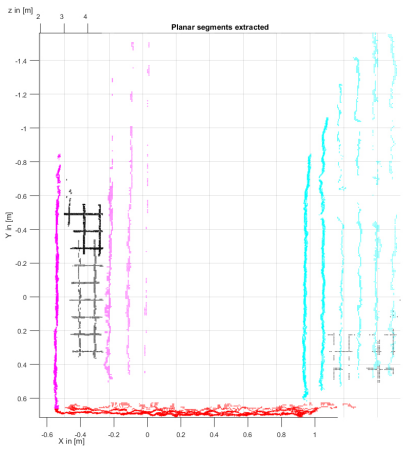
Figures 8.20d through 8.29d shows the planar surface reconstruction of the extracted planes from segmented pointed clouds. The surfaces that are black in color are the planar surfaces that are perpendicular to z-axis and represent the plane extracted for the points which are black in color. In figures 8.20c, 8.21c and 8.22c, It can be observed that the door



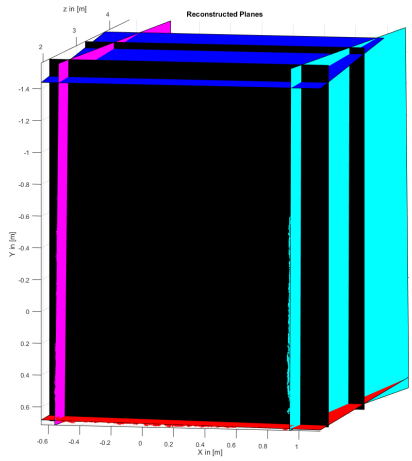
(a) Original point cloud



(b) Segmented point cloud



(c) Inlier points of a plane

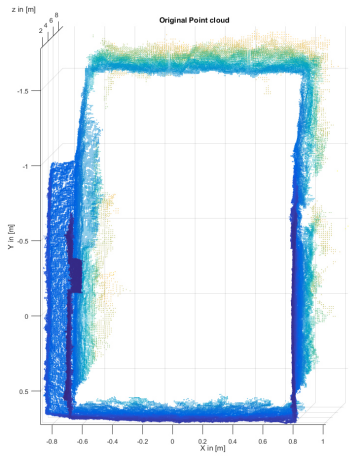


(d) Reconstructed planes from the inlier points

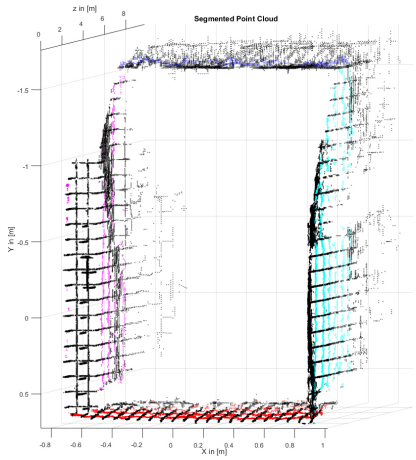
Figure 8.21: Point cloud segmentation and extraction: Stocker 2nd floor, time epoch 2

frame, artificially placed card board and ceiling bumps are picked up as plane. Since the surface area of these places are significantly smaller than the rest of the point clouds, these wouldn't have been extracted in the other algorithms.

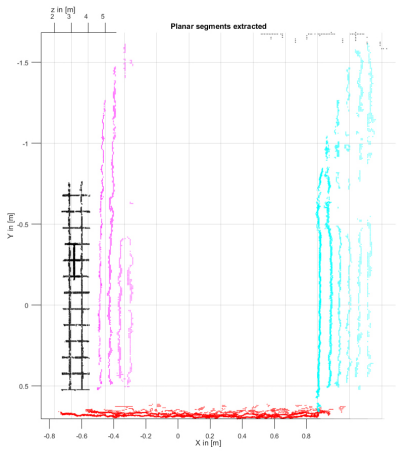
The Stocker hallways in the 3rd floor is an interesting flight environment, which has an unusual intermittent diagonal wall and window structure. In figure 8.27d and 8.28d, the planar extraction algorithm extracts planar surface parallel to y-axis, but planar surface



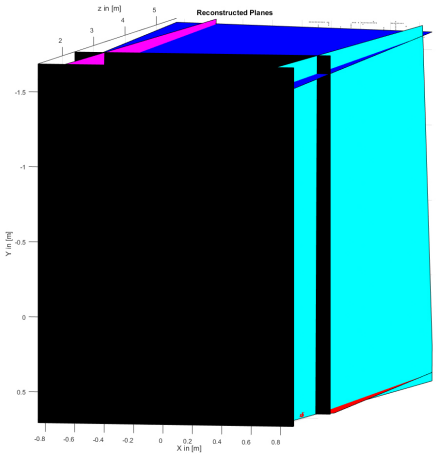
(a) Original point cloud



(b) Segmented point cloud



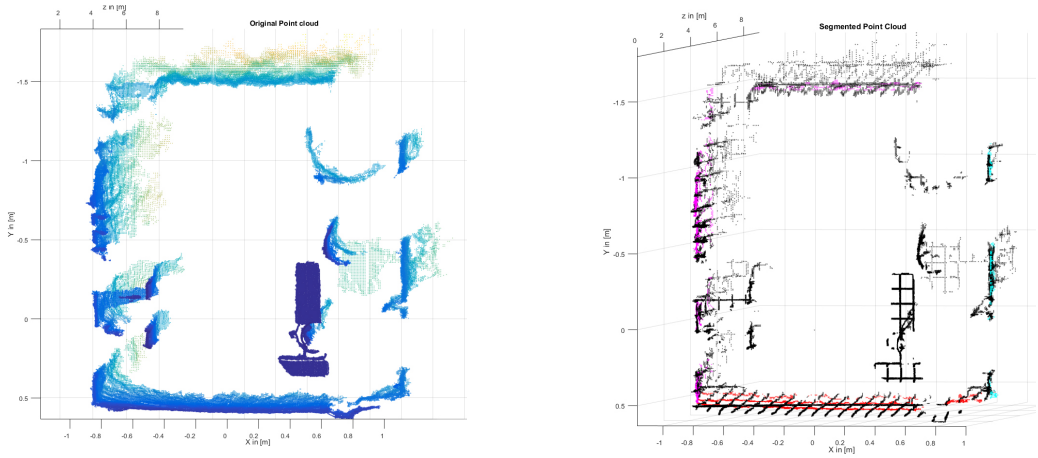
(c) Inlier points of a plane



(d) Reconstructed planes from the inlier points

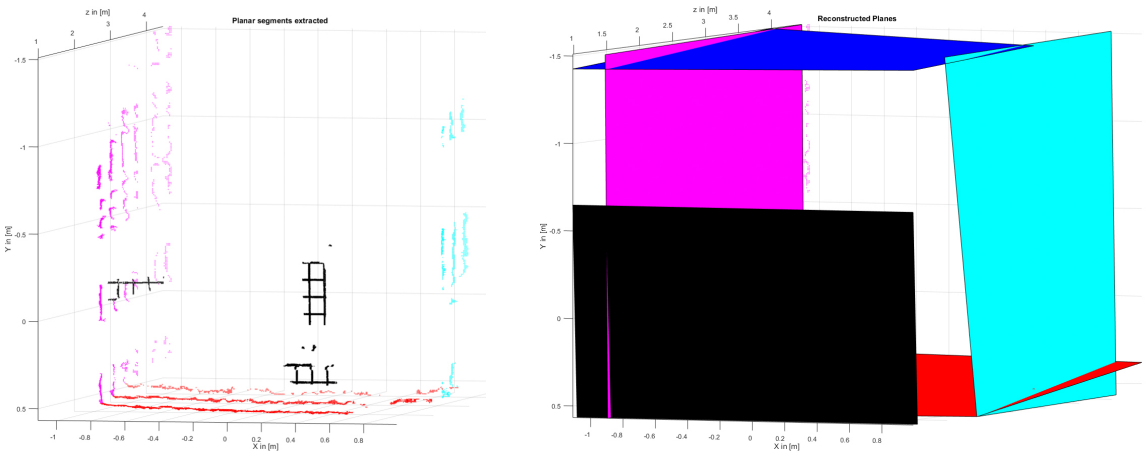
Figure 8.22: Point cloud segmentation and extraction: Stocker 2nd floor, time epoch 3

extracted in Figure 8.29d is a combination of the parallel wall and the diagonal wall. This difference is due the regions the z slices are calculated from, i.e., the point cloud is sliced at discrete z vales, at which the min-max segmented cloud is extracted, the majority of the z-slices falls in the area of windows, and hence the x_{max} point clouds have a very few points that are from the diagonal wall. When a plane is extracted from these points, the point that belong to diagonal plane are considered as outliers, planar surface is extracted using only



(a) Original point cloud (b) Segmented point cloud

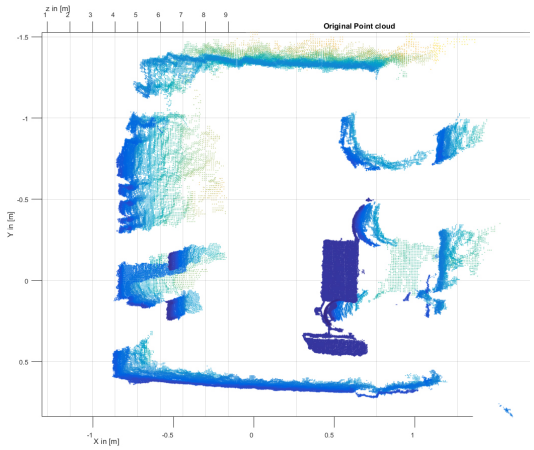
Figure 8.23: Point cloud segmentation: Maintenance tunnels, time epoch 1



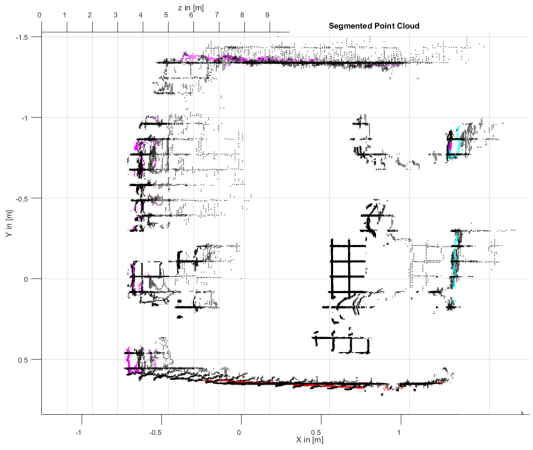
(a) Inlier points of a plane (b) Reconstructed planes from the inlier points

Figure 8.24: Point cloud segmentation and extraction: Maintenance tunnels, time epoch 1

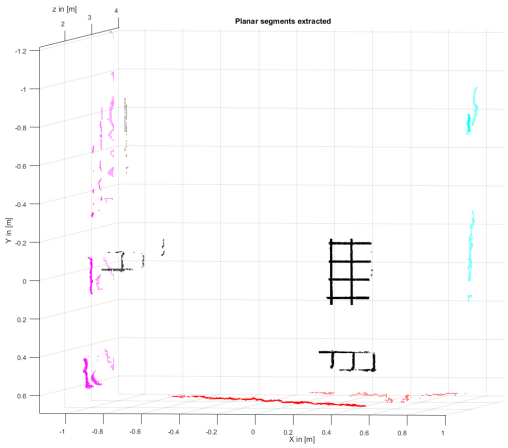
the parallel walls. On the other hand, the majority of the z-axis slices fall near the diagonal planes, which results in x_{max} segment having more points that belong to the diagonal wall. This results in the extracted plane which is the combination of parallel and diagonal planes.



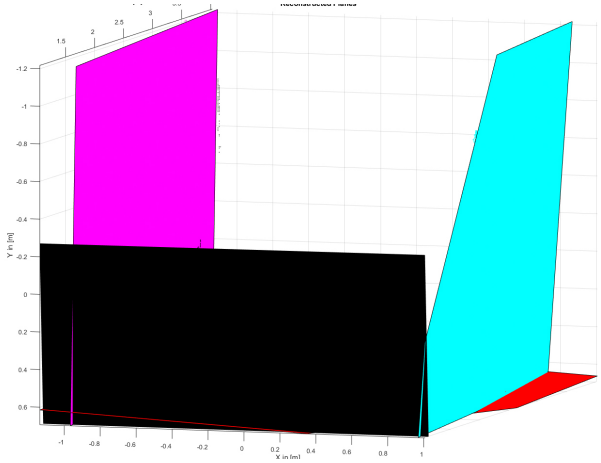
(a) Original point cloud



(b) Segmented point cloud

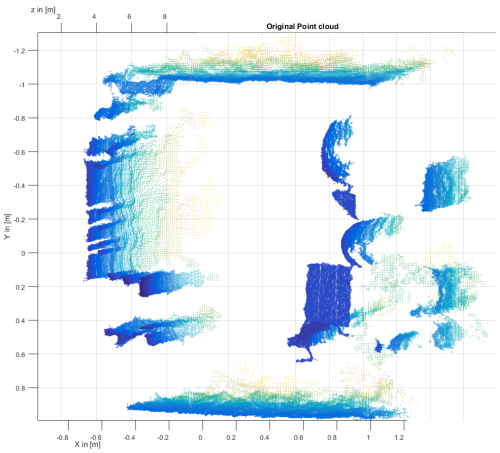


(c) Inlier points of a plane

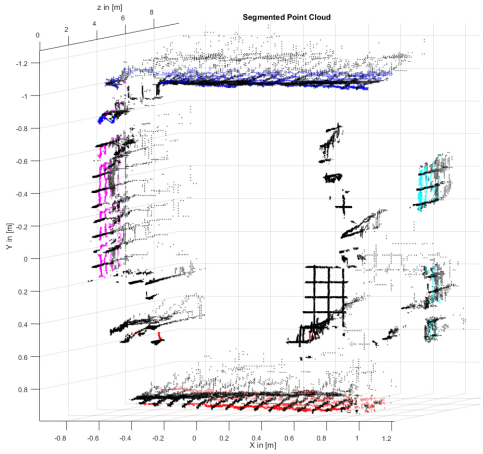


(d) Reconstructed planes from the inlier points

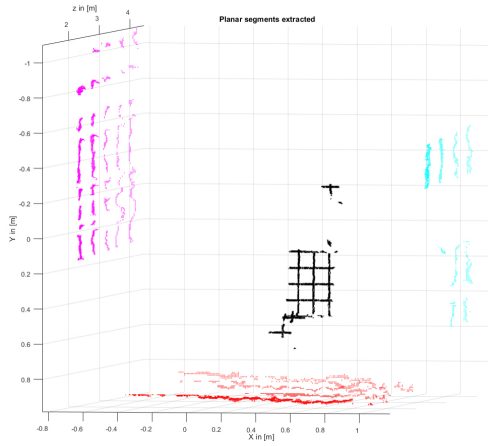
Figure 8.25: Point cloud segmentation and extraction: Maintenance tunnels, time epoch 2



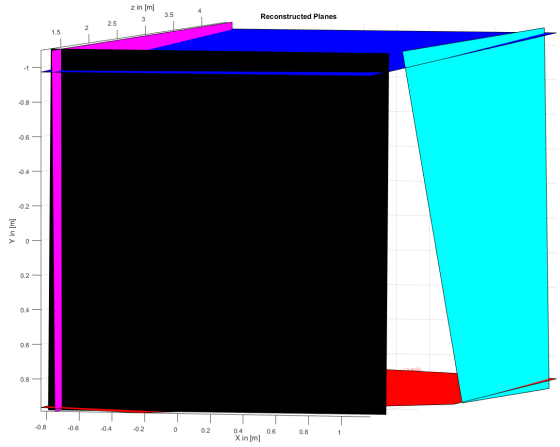
(a) Original point cloud



(b) Segmented point cloud

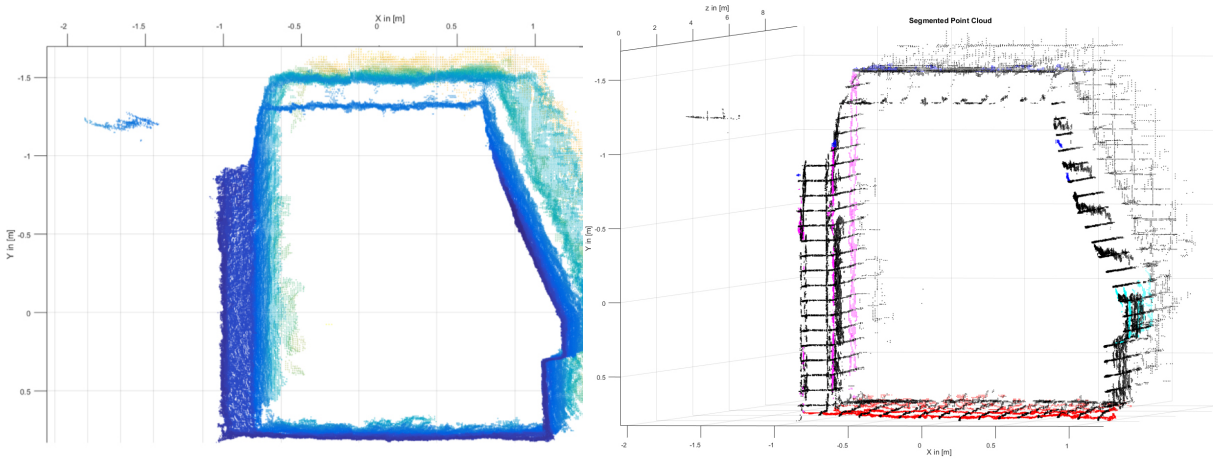


(c) Inlier points of a plane



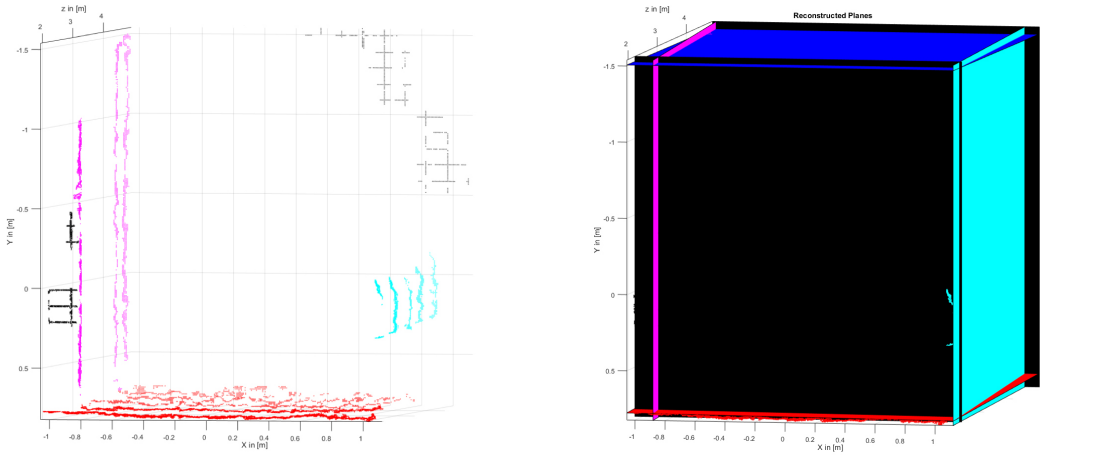
(d) Reconstructed planes from the inlier points

Figure 8.26: Point cloud segmentation and extraction: Maintenance tunnels, time epoch 3



(a) Original point cloud

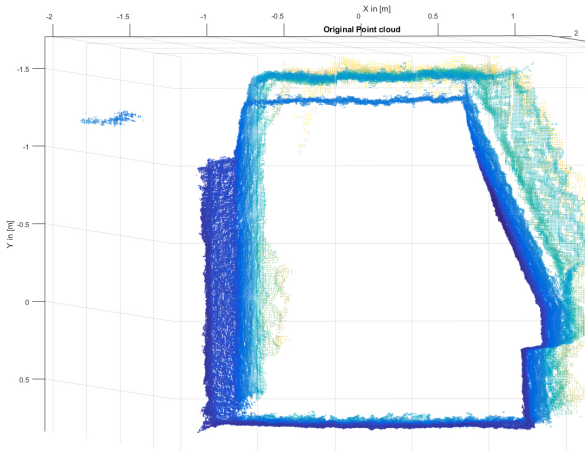
(b) Segmented point cloud



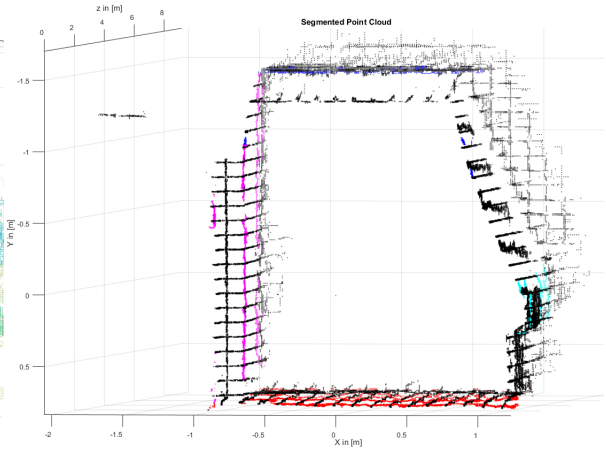
(c) Inlier points of a plane

(d) Reconstructed planes from the inlier points

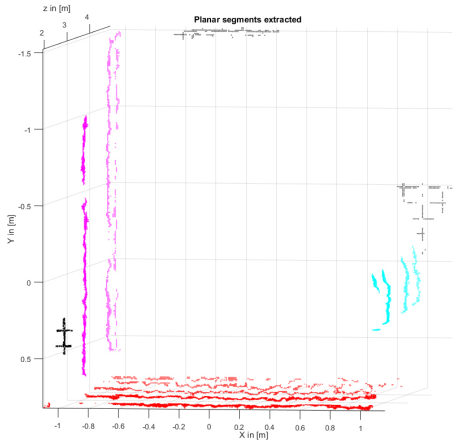
Figure 8.27: Point cloud segmentation and extraction: Stocker 3rd floor, time epoch 1



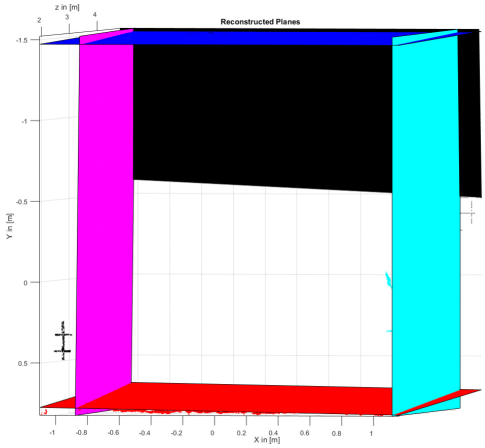
(a) Original point cloud



(b) Segmented point cloud

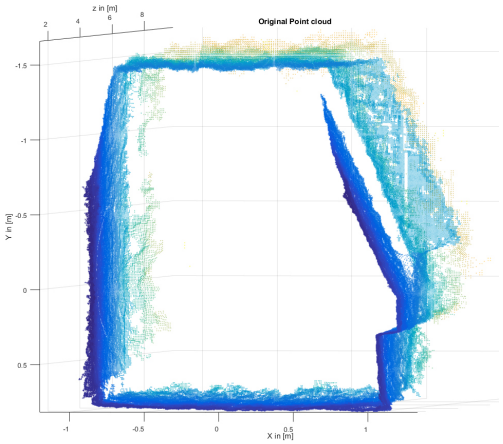


(c) Inlier points of a plane

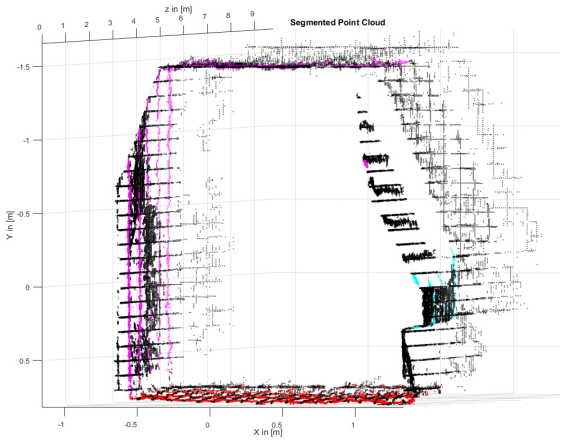


(d) Reconstructed planes from the inlier points

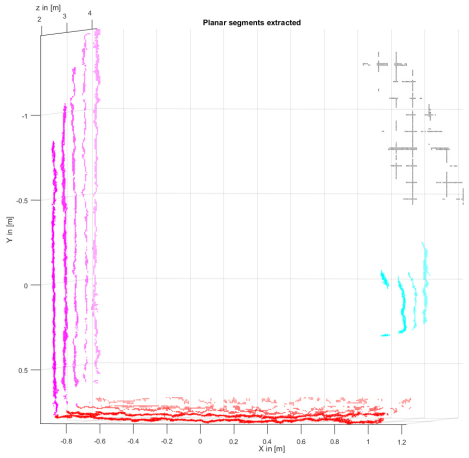
Figure 8.28: Point cloud segmentation and extraction: Stocker 3rd floor, time epoch 2



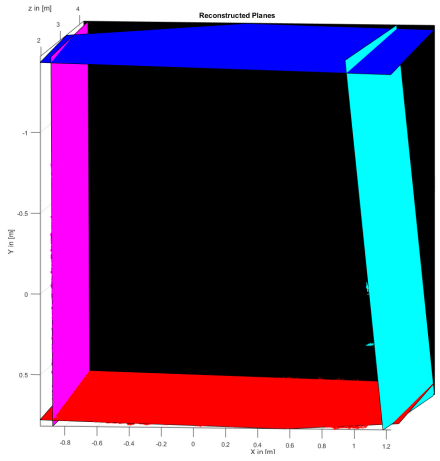
(a) Original point cloud



(b) Segmented point cloud



(c) Inlier points of a plane



(d) Reconstructed planes from the inlier points

Figure 8.29: Point cloud segmentation and extraction: Stocker 3rd floor, time epoch 3

9 Summary and Conclusions

This thesis discusses a perception payload for a sUAS with the associated data processing algorithm for navigation and mapping in a structured and GNSS-denied environment like maintenance tunnels and hallways. The sensor payload consists of a 3D imaging sensor, a LWIR sensor and an IMU. A multi-copter platform equipped with the aforementioned payload has been designed, built and flight-tested in an appropriate environment.

The on-board cameras have been calibrated for their lens distortion, to reliably extract 3D data from the 2D images. Moving further, relative poses of each onboard camera with respect to each other were estimated which resulted in pixel to pixel mapping of LWIR images with 3D depth images. This estimated is used to generate 3D points clouds with corresponding heat temperature.

The LWIR imagery was used to extract features that would help in detecting the heat-related defects in the tunnels. Physical features like pillars, pipes are other features were extracted successfully along with their thermal signatures supporting inspection and maintenance applications.

The proposed pre-processing algorithm was applied on the data collected by the multi-copter platform build. From the pre-processed point cloud presented in the previous chapter, it is evident that the pre-processed algorithm down-samples the PC significantly enough without losing out on key details in the PC. It also forms subsets of points from which planar surfaces can easily be extracted (i.e., the Xmin, Xmax, Ymin, Ymax and Z PC subsets).

A RANSAC based planar extraction algorithm was applied on the pre-processed and segmented PC and planar features are successfully extracted. The total time taken applying the proposed segmentation and feature extraction algorithm on PCs is approximately 110 ms - 130 ms on an i5 system and 170 ms - 200 ms on an odroid. This a significant reduction

in processing time as compared to most of the visual based navigation algorithms discussed in chapter 2 which capable of operating at a maximum of 3Hz.

The proposed algorithm is capable of running in real-time on the ARM based on-board pc supporting on-line SLAM. ‘

10 Future Work

The road map of this work is to operate the small size UAS platform in the maintenance tunnels completely autonomous on a maintenance mission. This work has discussed the construction of a small sized UAS platform along with its sensor stack, the calibration of thermal and 3D cameras individually and as a stereo pair, segmentation of point clouds and the real-time feature extraction, primarily planar features. What has been presented here is just one piece of a puzzle to achieve the end objective.

ROS camera calibration and rectification software pipeline is used for this thesis, data is presented based on the needs of software pipeline. A better software, specific for this application would enable better use of images in its original form. The RANSAC algorithm currently used doesn't support the calculation of trials needed to obtain a best fitting model. Use of modern RANSAC enables the determination of trials needed, which allows for a more efficient planar extraction.

With the features that have been extracted, the next step is to implement an online SLAM algorithm that utilizes the planar features and the inertial data to localize itself in the environment and build a map of it. Estimation of pose using 3D camera present non-Gaussian errors in the estimation. These non-Gaussian errors have to be addressed during the implementation of the SLAM algorithm. To implement the SLAM algorithm successfully, a key step is the feature association. Feature association is a very critical step in feature-based SLAM, where even a single mis-association of feature results in huge errors. Hence, a good feature association algorithm has to be implemented to develop and implement a good SLAM algorithm. Implementing the online SLAM algorithm isn't sufficient by itself to achieve the end objective. Online SLAM as described in chapter 2, is a coarse estimate of the position, and would probably become unreliable over a period of time. Hence a more reliable backend SLAM / Full SLAM or Graph SLAM has to be

implemented to smooth the estimation from online SLAM and give out the more reliable, smoothed and optimized trajectory and position with loop closure.

SLAM by itself is just a positioning solution, but the real challenge is the autonomous navigation for maintenance missions. For autonomous operation of the platform, much like the autonomous driving cars, need a fully-fledged path planning algorithm with intelligent Artificial Intelligence (AI) / ML based decision making capability along with collision avoidance feature.

For the detection of leaks in the steam tunnels, data from LWIR images has to be processed by and ML algorithms that can detect and classify leaks or any other abnormalities in the tunnels with high confidence. There is also a need to develop a robust scene understanding algorithm in 3D, which can combine the data thermal and the 3D imagery.

References

[1] Adam Schultz, Russell Gilabert, Akshay Bharadwaj, Maarten Uijy de Haag, and Zhen Zhu. A Navigation and Mapping Method for UAS During Under-the-Canopy Forest Operations. In *Proceedings of IEEE/ION PLANS 2016*, pages 739–746, 2016. URL https://www.ion.org/publications/abstract.cfm?articleID=13501.

[2] Paul D Groves. GNSS: Fundamentals, Signals and Satellites. In *Principles of GNSS, Inertial and Multisensor Integrated Navigation Systems*, chapter 8, pages 299–347. Artech House, second edi edition, 2005.

[3] Xsens. MTi 1-series, 2015.

[4] OpenCV 2.4.13.1. Camera Calibration and 3D Reconstruction OpenCV 2.4.13.1 documentation. URL http://docs.opencv.org/2.4/modules/calib3d/doc/camera{_}calibration{_}and{_}3d{_}reconstruction.html.

[5] MathWorks. What Is Camera Calibration? - MATLAB and Simulink - MathWorks Deutschland, 2016. URL https://www.mathworks.com/help/vision/ug/camera-calibration.html{#}buvr2qb-1https://de.mathworks.com/help/vision/ug/camera-calibration.html.

[6] Novotel. IMU Errors and Their Effects. Report APN-064 (Rev A). Technical report, 2014. URL http://www.novatel.com/assets/Documents/Bulletins/APN064.pdf.

[7] Davide Scaramuzza and Friedrich Fraundorfer. Visual Odometry : Part I: The First 30 Years and Fundamentals. *IEEE Robotics & Automation Magazine*, 18 (4):80–92, 2011. ISSN 10709932. doi: 10.1109/MRA.2011.943233. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6096039.

[8] D Nister, O Naroditsky, and J Bergen. Visual odometry. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 1, pages I–652–I–659 Vol.1, jun 2004. doi: 10.1109CVPR.2004.1315094.

[9] Jakob Engel, Vladlen Koltun, and Daniel Cremers. Direct Sparse Odometry. *CoRR*, abs/1607.0, 2016. URL http://arxiv.org/abs/1607.02565.

[10] Friedrich Fraundorfer and Davide Scaramuzza. Visual odometry: Part II: Matching, robustness, optimization, and applications. *IEEE Robotics and Automation Magazine*, 19(2):78–90, 2012. ISSN 10709932. doi: 10.1109/MRA.2012.2182810.

[11] Jianbo Shi and Carlo Tomasi. Good Features to Track. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94., 1994 IEEE Computer Society Conference on*, pages 593–600, 1994. ISBN 0-8186-5825-8. doi: 10.1109/cvpr.1994.323794. URL http://dx.doi.org/10.1109/cvpr.1994.323794.

[12] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004. ISSN 09205691. doi: 10.1023/B:VISI.0000029664.99615.94.

[13] David G. Lowe. Object recognition from local scale-invariant features. *Proceedings of the Seventh IEEE International Conference on Computer Vision*, 2([8):1150–1157, 1999. ISSN 0-7695-0164-8. doi: 10.1109/ICCV.1999.790410. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=790410.

[14] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. SURF: Speeded up robust features. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 3951 LNCS, pages 404–417, 2006. ISBN 3540338322. doi: 10.1007/11744023_32.

[15] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. ORB: An efficient alternative to SIFT or SURF. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2564–2571, 2011. ISBN 9781457711015. doi: 10.1109/ICCV.2011.6126544.

[16] R.˜I. Hartley and A Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[17] Zhengyou Zhang. Microsoft kinect sensor and its effect, 2012. ISSN 1070986X.

[18] Jason Geng. Structured-light 3D surface imaging: a tutorial. *Adv. Opt. Photon.*, 3(2): 128–160, jun 2011. doi: 10.1364/AOP.3.000128. URL http://aop.osa.org/abstract.cfm?URI=aop-3-2-128.

[19] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. Intelligent robotics and autonomous agents. the MIT Press, Cambridge (Mass.) (London), 2005. ISBN 0-262-20162-3. URL http://opac.inria.fr/record=b1102986.

[20] Robert Grover Brown and Patrick Y C Hwang. *Introduction to Random Signals and Applied Kalman Filtering with Matlab Exercises*. 2012. ISBN 9788578110796. doi: 10.1017/CBO9781107415324.004.

[21] C Cadena, L Carlone, H Carrillo, Y Latif, D Scaramuzza, J Neira, I Reid, and J J Leonard. Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age. *IEEE Transactions on Robotics*, 32(6):1309–1332, dec 2016. ISSN 1552-3098. doi: 10.1109/TRO.2016.2624754.

[22] Randall Smith, Matthew Self, and Peter Cheeseman. Estimating Uncertain Spatial Relationships in Robotics. *Machine Intelligence and Pattern Recognition*, 5(C): 435–461, 1988. ISSN 09230459. doi: 10.1016/B978-0-444-70396-5.50042-X.

[23] Giorgio Grisetti, Gian Diego Tipaldi, Cyrill Stachniss, Wolfram Burgard, and Daniele Nardi. Fast and accurate SLAM with Rao-Blackwellized particle filters. *Robotics and Autonomous Systems*, 2007. ISSN 09218890. doi: 10.1016/j.robot.2006.06.007.

[24] D Eigen and R Fergus. Predicting Depth, Surface Normals and Semantic Labels with a Common Multi-scale Convolutional Architecture. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 2650–2658, dec 2015. doi: 10.1109/ICCV.2015.304.

[25] F Liu, Chunhua Shen, and Guosheng Lin. Deep convolutional neural fields for depth estimation from a single image. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5162–5170, jun 2015. doi: 10.1109/CVPR.2015.7299152.

[26] R A Newcombe, S Izadi, O Hilliges, D Molyneaux, D Kim, A J Davison, P Kohi, J Shotton, S Hodges, and A Fitzgibbon. KinectFusion: Real-time dense surface mapping and tracking. In *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, pages 127–136, oct 2011. doi: 10.1109/ISMAR.2011.6092378.

[27] Paul Besl and Neil McKay. A Method for Registration of 3-D Shapes, 1992. ISSN 0162-8828.

[28] Po-Chang Su, Ju Shen, and Sen-ching S . Cheung. a Robust Rgb-D Slam System for 3D Environment With Planar Surfaces. In *Image Processing (ICIP), 2013 20th IEEE International Conference on*, pages 2–6, 2013. ISBN 9781479923410.

[29] Chenglu Wen, Ling Qin, Qingyuan Zhu, Cheng Wang, and Jonathan Li. Three-dimensional indoor mobile mapping with fusion of two-dimensional laser scanner and RGB-D camera data. *IEEE Geoscience and Remote Sensing Letters*, 11(4): 843–847, 2014. ISSN 1545598X. doi: 10.1109/LGRS.2013.2279872.

[30] Mahesh Kr Singh, K. S. Venkatesh, and Ashish Dutta. Accurate 3D terrain modeling by range data fusion from two heterogeneous range scanners. *11th IEEE India Conference: Emerging Trends and Innovation in Technology, INDICON 2014*, pages 0–5, 2015. doi: 10.1109/INDICON.2014.7030692.

[31] Laurie Bose and Arthur Richards. Fast depth edge detection and edge based RGB-D SLAM. *Proceedings - IEEE International Conference on Robotics and Automation*, 2016-June:1323–1330, 2016. ISSN 10504729. doi: 10.1109/ICRA.2016.7487265.

[32] David Wooden, Matthew Malchano, Kevin Blankespoor, Andrew Howardy, Alfred A. Rizzi, and Marc Raibert. Autonomous navigation for BigDog. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 4736–4741, 2010. ISSN 10504729. doi: 10.1109/ROBOT.2010.5509226.

[33] Roberto G. Valenti, Ivan Dryanovski, Carlos Jaramillo, Daniel Perea Strom, and Jizhong Xiao. Autonomous quadrotor flight using onboard RGB-D visual odometry. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 5233–5238, 2014. ISSN 10504729. doi: 10.1109/ICRA.2014.6907628.

[34] Ivan Dryanovski, Roberto G Valenti, and Jizhong Xiao. An open-source navigation system for micro aerial vehicles. *Autonomous Robots*, 34(3):177–188, 2013. ISSN 1573-7527. doi: 10.1007/s10514-012-9318-8. URL http://dx.doi.org/10.1007/s10514-012-9318-8.

[35] I Dryanovski, R G Valenti, and Jizhong Xiao. Fast visual odometry and mapping from RGB-D data. In *2013 IEEE International Conference on Robotics and Automation*, pages 2305–2310, may 2013. doi: 10.1109/ICRA.2013.6630889.

[36] N Brunetto, S Salti, N Fioraio, T Cavallari, and L D Stefano. Fusion of Inertial and Visual Measurements for RGB-D SLAM on Mobile Devices. In *2015 IEEE International Conference on Computer Vision Workshop (ICCVW)*, pages 148–156, dec 2015. doi: 10.1109/ICCVW.2015.29.

[37] Philipp Tiefenbacher, Timo Schulze, and Gerhard Rigoll. Off-the-shelf sensor integration for mono-SLAM on smart devices. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2015-Octob:15–20, 2015. ISSN 21607516. doi: 10.1109/CVPRW.2015.7301361.

[38] Georg Klein and David Murray. Improving the Agility of Keyframe-Based SLAM. In David Forsyth, Philip Torr, and Andrew Zisserman, editors, *Computer Vision – ECCV 2008: 10th European Conference on Computer Vision, Marseille, France, October 12-18, 2008, Proceedings, Part II*, pages 802–815. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. ISBN 978-3-540-88688-4. doi: 10.1007/978-3-540-88688-4_59. URL http://dx.doi.org/10.1007/978-3-540-88688-4{_}59.

[39] Peter Henry, Michael Krainin, Evan Herbst, Xiaofeng Ren, and Dieter Fox. RGB-D Mapping: Using Depth Cameras for Dense 3D Modeling of Indoor Environments. In Oussama Khatib, Vijay Kumar, and Gaurav Sukhatme, editors, *Experimental Robotics: The 12th International Symposium on Experimental Robotics*, pages 477–491. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014. ISBN 978-3-642-28572-1. doi: 10.1007/978-3-642-28572-1_33. URL http://dx.doi.org/10.1007/978-3-642-28572-1{_}33.

[40] Haiwei Dong, Nadia Figueroa, and Abdulmotaleb El Saddik. Towards consistent reconstructions of indoor spaces based on 6D RGB-D odometry and KinectFusion. *IEEE International Conference on Intelligent Robots and Systems*, (Iros):1796–1803, 2014. ISSN 21530866. doi: 10.1109/IROS.2014.6942798.

[41] G Grisetti, S Grzonka, C Stachniss, P Pfaff, and W Burgard. Efficient estimation of accurate maximum likelihood maps in 3D. In *2007 IEEE/RSJ International*

Conference on Intelligent Robots and Systems, pages 3472–3478, oct 2007. doi: 10.1109/IROS.2007.4399030.

[42] Giorgio Grisetti, Cyrill Stachniss, Slawomir Grzonka, and Wolfram Burgard. A tree parameterization for efficiently computing maximum likelihood maps using gradient descent. *Rss*, page 8, 2007. ISSN 2330765X. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.69.6288{&}rep=rep1{&}type=pdf.

[43] Hanspeter Pfister, Matthias Zwicker, Jeroen van Baar, and Markus Gross. Surfels: Surface Elements As Rendering Primitives. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '00, pages 335–342, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co. ISBN 1-58113-208-5. doi: 10.1145/344779.344936. URL http://dx.doi.org/10.1145/344779.344936.

[44] F Endres, J Hess, N Engelhard, and J Sturm .... An evaluation of the RGB-D SLAM system. *Icra*, 1(c):1691–1696, 2012. ISSN 978-1-4673-1403-9. doi: 10.1109/ICRA.2012.6225199. URL http://ieeexplore.ieee.org/xpls/abs{_}all.jsp?arnumber=6225199{%}5Cnpapers3://publication/doi/10.1109/ICRA.2012.6225199.

[45] Felix Endres, Juergen Hess, Nikolas Engelhard, Jrgen Sturm, and Wolfram Burgard. RGBDSLAM - 6DOF SLAM for Kinect-style cameras, 2012. URL https://openslam.org/rgbdslam.html.

[46] T Fischer, T Pire, P Cek, P De Cristforis, and J Faigl. Stereo vision-based localization for hexapod walking robots operating in rough terrains. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2492–2497, oct 2016. doi: 10.1109/IROS.2016.7759388.

[47] T Pire, T Fischer, J Civera, P De Cristóforis, and J J Berlles. Stereo parallel tracking and mapping for robot localization. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1373–1378, sep 2015. doi: 10.1109/IROS.2015.7353546.

[48] Lili Meng, Clarence W De Silva, and Jie Zhang. 3D Visual SLAM for an Assistive Robot in Indoor Environments Using RGB-D Cameras. *The 9th International Conference on Computer Science & Education ICCSE 2014*, (Iccse):32–37, 2014. doi: 10.1109/ICCSE.2014.6926426.

[49] Dominik Belter, Micha?? Nowicki, and Piotr Skrzypczy??ski. Improving accuracy of feature-based RGB-D SLAM by modeling spatial uncertainty of point features. *Proceedings - IEEE International Conference on Robotics and Automation*, 2016-June:1279–1284, 2016. ISSN 10504729. doi: 10.1109/ICRA.2016.7487259.

[50] Gibson Hu, Shoudong Huang, Liang Zhao, Alen Alempijevic, and Gamini Dissanayake. A robust RGB-D SLAM algorithm. *IEEE International Conference*

on Intelligent Robots and Systems, pages 1714–1719, 2012. ISSN 21530858. doi: 10.1109/IROS.2012.6386103.

[51] Andrew J. Davison, Ian D. Reid, Nicholas D. Molton, and Olivier Stasse. MonoSLAM: Real-time single camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1052–1067, 2007. ISSN 01628828. doi: 10.1109/TPAMI.2007.1049.

[52] G Silveira, E Malis, and P Rives. An Efficient Direct Method for Improving visual SLAM. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 4090–4095, apr 2007. doi: 10.1109/ROBOT.2007.364107.

[53] C Ye, S Hong, and A Tamjidi. 6-DOF Pose Estimation of a Robotic Navigation Aid by Tracking Visual and Geometric Features. *IEEE Transactions on Automation Science and Engineering*, 12(4):1169–1180, oct 2015. ISSN 1545-5955. doi: 10.1109/TASE.2015.2469726.

[54] Chen Feng, Yuichi Taguchi, and Vineet R. Kamat. Fast plane extraction in organized point clouds using agglomerative hierarchical clustering. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 6218–6225, 2014. ISSN 10504729. doi: 10.1109/ICRA.2014.6907776.

[55] Yuichi Taguchi, Yong Dian Jian, Srikumar Ramalingam, and Chen Feng. Point-plane SLAM for hand-held 3D sensors. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 5182–5189, 2013. ISSN 10504729. doi: 10.1109/ICRA.2013.6631318.

[56] Esra Ataer-Cansizoglu, Yuichi Taguchi, Srikumar Ramalingam, and Tyler Garaas. Tracking an RGB-D camera using points and planes. *Proceedings of the IEEE International Conference on Computer Vision*, pages 51–58, 2013. doi: 10.1109/ICCVW.2013.14.

[57] Esra Ataer-Cansizoglu, Yuichi Taguchi, and Srikumar Ramalingam. Pinpoint SLAM: A hybrid of 2D and 3D simultaneous localization and mapping for RGB-D sensors. *Proceedings - IEEE International Conference on Robotics and Automation*, 2016-June:1300–1307, 2016. ISSN 10504729. doi: 10.1109/ICRA.2016.7487262.

[58] Ankur Handa, Thomas Whelan, John McDonald, and Andrew J. Davison. A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 1524–1531, 2014. ISSN 10504729. doi: 10.1109/ICRA.2014.6907054.

[59] Jrgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of RGB-D SLAM systems. *IEEE International Conference on Intelligent Robots and Systems*, pages 573–580, 2012. ISSN 21530858. doi: 10.1109/IROS.2012.6385773.

[60] Oliver Wasenmuller, Marcel Meyer, and Didier Stricker. CoRBS: Comprehensive RGB-D benchmark for SLAM using Kinect v2. *2016 IEEE Winter Conference on Applications of Computer Vision, WACV 2016*, 2016. doi: 10.1109/WACV.2016.7477636.

[61] Adam Schultz, Russell Gilabert, and Maarten Uijt de Haag. Indoor Flight Demonstration Results of an Autonomous Multi-copter using Multiple Laser Inertial Navigation. In *Proceedings of the 2016 International Technical Meeting of The Institute of Navigation, Monterey*, pages 511–519, 2016.

[62] Carlos Estrada, José Neira, and Juan D. Tardós. Hierarchical SLAM: Real-time accurate mapping of large environments. *IEEE Transactions on Robotics*, 21(4): 588–596, 2005. ISSN 15523098. doi: 10.1109/TRO.2005.844673.

[63] Rainer Kümmerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, and Wolfram Burgard. G2o: A general framework for graph optimization. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 3607–3613, 2011. ISBN 9781612843865. doi: 10.1109/ICRA.2011.5979949.

[64] James L. Farrell. *Gnss Aided Navigation & Tracking: Inertially Augmented Or Autonomous*. American Literary Press, 1st edition, 2007.

[65] David Nister, Oleg Naroditsky, and James Bergen. Visual odometry for ground vehicle applications. *Journal of Field Robotics*, 23(1):3–20, 2006. ISSN 15564959. doi: 10.1002/rob.20103.

[66] Ji Zhang and Sanjiv Singh. LOAM : Lidar Odometry and Mapping in Real-time. *Robotics: Science and Systems*, 2014. ISSN 0929-5593. doi: 10.1007/s10514-016-9548-2.

[67] J Zhang and S Singh. Visual-lidar odometry and mapping: low-drift, robust, and fast. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2174–2181, 2015. ISBN 1050-4729 VO -. doi: 10.1109/ICRA.2015.7139486.

[68] Andreas Nüchter. *3D Robotic Mapping - The Simultaneous Localization and Mapping Problem with Six Degrees of Freedom*. Springer Berlin Heidelberg, 2009.

[69] Dorit Borrmann, Jan Elseberg, Kai Lingemann, Andreas Nüchter, and Joachim Hertzberg. Globally consistent 3D mapping with scan matching. *Robotics and Autonomous Systems*, 56(2):130–142, 2008. ISSN 09218890. doi: 10.1016/j.robot.2007.07.002.

[70] F Lu and E Milios. Globally Consistent Range Scan Alignment for Environment Mapping. *Autonomous Robots*, 4(4):333–349, 1997.

[71] USA Occipital, San Francisco, CA. Structure Sensor & Sdk :: Fact Sheet, 2015.

[72] USA Occipital, San Francisco, CA. Structure Sensor - Technical Specs, 2015.

[73] FLIR. FLIR LEPTON ® Long Wave Infrared (LWIR) Datasheet, 2014.

[74] FLIR. Lepton¢ Software Interface Description Document (IDD), 2014.

[75] Rafael C Gonzalez and Richard E Woods. Image Sampling and Quantization. In *Digital Image Processing*, chapter Chapter 2, pages 65–68. Pearson Prentice Hall, third edit edition, 2008. ISBN 978-81-317-2695-2.

[76] Xsens. Data sheet MTi 1-series, 2015.

[77] Sang-Hoon Kim. Brushless Direct Current Motors. In *Electric Motor Control: DC, AC, and BLDC Motors*, chapter 10, pages 389–415. Elsevier Inc., 1 edition, 2017. ISBN 978-0128121382.

[78] Minihquad.com. Blackout Mini H Quad. URL http://www.minihquad.com/frames/blackout-330-frame.

[79] Pixhawk.org. Pixhawk Autopilot, 2016. URL https://pixhawk.org/modules/pixhawk.

[80] Odroid. ODROID — Hardkernel. URL http://www.hardkernel.com/main/products/prdt{_}info.php?g{_}code=G143452239825{&}tab{_}idx=2.

[81] ROS.org. Documentation - ROS Wiki. URL http://wiki.ros.org/action/fullsearch/Documentation?action=fullsearch{&}context=180{&}value=linkto{%}3A{%}22Documentation{%}22.

[82] PX4 Dev Team. Multicopter PID Tuning Guide · PX4 User Guide, 2018. URL https://docs.px4.io/en/advanced{_}config/pid{_}tuning{_}guide{_}multicopter.html.

[83] J. Heikkila and O. Silven. A four-step camera calibration procedure with implicit image correction. *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1106–1112, 1997. ISSN 1063-6919. doi: 10.1109/CVPR.1997.609468. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=609468.

[84] Zhang Zhengyou. Flexible camera calibration by viewing a plane from unknown orientations. *Proceedings of the Seventh IEEE International Conference on Computer Vision*, 1(c):0–7, 1999. ISSN 01628828. doi: 10.1109/ICCV.1999.791289.

[85] Richard Hartley and Andrew Zisserman. Camera Models. In *Multiple View Geometry in Computer Vision*, chapter Chapter 6, page 672. Cambridge University Press, 2004.

[86] J.Y. Bouguet. Camera Calibration Toolbox for MATLAB, 2010. URL http://www.vision.caltech.edu/bouguetj/calib{_}doc/.

[87] O Ben-Kiki, C Evans, and B Ingerson. YAML Ain't Markup Language (YAML¢) Version 1.2, 2009. URL http://www.yaml.org/spec/1.2/spec.html.

[88] A Ardeshir Goshtasby. Introduction. In *Image Registration: Principles, Tools and Methods*, chapter 1, pages 1–6. Springer London, London, 2012. ISBN 978-1-4471-2458-0. doi: 10.1007/978-1-4471-2458-0_1. URL http://dx.doi.org/10.1007/978-1-4471-2458-0{_}1.

[89] Richard Hartley and Andrew Zisserman. Epipolar Geometry and the Fundamental Matrix. In *Multiple View Geometry in Computer Vision*, chapter 9, pages 239–261. Second edition, 2004. ISBN 0521540518.

[90] Gregory G Slabaugh. Computing Euler angles from a rotation matrix, 1999. URL http://gregslabaugh.name/publications/euler.pdf.

[91] Martin a Fischler and Robert C Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applicatlons to Image Analysis and Automated Cartography. *Communications of the ACM*, 24(6):381 – 395, 1981. ISSN 0001-0782. doi: 10.1145/358669.358692. URL http://dx.doi.org/10.1145/358669.358692.

[92] PointClouds.org. Point Cloud Library (PCL): Module sample_consensus, 2016. URL http://docs.pointclouds.org/trunk/group{_}{_}sample{_}{_}consensus.html.

[93] Bernard Widrow and Samuel D. Stearns. *Adaptive Signal Processing*. Pearson Education, 1985. ISBN 9788131705322. URL https://books.google.com/books?id=S{_}b4GBOlWfwC.

[94] Akshay Bharadwaj, Adam Schultz, Russell Gilabert, Joel Huff, and Maarten Uijt de Haag. Small-UAS Navigation Using 3D Imager and Infrared Camera in Structured Environments. In *Proceedings of IEEE/ION PLANS*, pages 599–606, Savannah, GA, 2016.

[95] Rafael C Gonzalez and Richard E Woods. Image Segmentation. In *Digital Image Processing*, chapter 10, pages 719–725. Pearson Prentice Hall, third edition, 2008.

[96] Derek Bradley and Gerhard Roth. Adaptive Thresholding using the Integral Image. *Journal of Graphics Tools*, 12(2):13–21, 2007. ISSN 1086-7651. doi: 10.1080/2151237X.2007.10129236. URL https://www.tandfonline.com/doi/full/10.1080/2151237X.2007.10129236.

[97] Martin a Fischler and Robert C Bolles. Random Sample Consensus: A Paradigm for Model Fitting with. *Communications of the ACM*, 24:381–395, 1981. ISSN 00010782. doi: 10.1145/358669.358692.

[98] P. H.S. Torr and A. Zisserman. MLESAC: A new robust estimator with application to estimating image geometry. *Computer Vision and Image Understanding*, 78(1): 138–156, 2000. ISSN 10773142. doi: 10.1006/cviu.1999.0832.

[99] M Uijt de Haag, D Venable, M Smearcheck, J L Campbell, and M M Miller. Flash-LADAR Inertial Navigator Aiding. In *2006 IEEE/ION Position, Location, And Navigation Symposium*, pages 677–683, apr 2006. doi: 10.1109/PLANS.2006.1650661.

[100] Donald T. Venable. *Implementation of a 3D Imaging Sensor Aided Inertial Measurement Unit Navigation System*. PhD thesis, Ohio University, 2008.

[101] Richard Szeliski. *Computer Vision : Algorithms and Applications*. Springer-Verlag London Limited, 2011. ISBN 1848829345. doi: 10.1007/978-1-84882-935-0.



Thesis and Dissertation Services