Real Time Visual SLAM for Mobile Robot Navigation and Object Detection

A thesis presented in partial fulfilment of the requirement for the degree of

Doctor of Philosophy

In

Engineering

at Massey University, Albany,

New Zealand

Changjuan Jing

2017

Acknowledgement

I would like to express my special appreciation and thanks to my supervisors, Associate Professor Johan Potgieter, Lecture Frazer Noble and Associate Professor Ruili Wang: their encouragement, guidance, and support help me grow as a research scientist.

I would also like to thank my committee members for letting my defence be an enjoyable moment, and for your brilliant comments and suggestions, thanks to you.

A special thanks to my family. Words cannot express how grateful I am to parents for all of the sacrifices that you've made on my behalf. Your prayer for me was what sustained me thus far. I would also like to thank all of my friends who supported me to strive towards my goal.

Abstract

This study developed a real-time Visual Simultaneous Localization and Mapping (SLAM) method for mobile robot navigation and object detection (SLAM-O), in order to establish the position of a mobile robot and interesting objects in an unknown indoor environment.

The VEX Robotics Competition (VRC) is one of the largest, fastest growing educational programmes in the world, and it is designed to increase student interest and involvement in Science, Technology, Engineering, and Mathematics (STEM). This study aims to enable an autonomous robot to compete with human participants in the VRC match, where robots are programmed to respond to a user's remote control. To win the competition, the robot needs to optimise between detection of goal objects, navigation and scoring. This thesis presents a Visual SLAM technique for robot localization and field objects' mapping, and aims to provide an innovative and practical approach to control robot navigation and maximise its scoring.

For visual observation, this study consists of an evaluation and comparison of widely used RGB-D cameras. Also, this thesis describes the integration of an iterated video frames module with the Extended Kalman Filter (EKF) for accurate SLAM estimation; where, a new frame selection method is employed. A novel SLAM-O method is developed for detecting objects in a robot's navigation process. The SLAM-O method uses a new K-Means-based colour identification method for semi-transparent object detection and a new concave-based object separation method for multi-connected objects, which outperform traditional methods.

Through conducting an investigation into RGB-D cameras' performance, in terms of repeatability and accuracy, colour images and depth point clouds accuracy from different cameras are evaluated and compared. These comparison's results provide a reference for choosing a camera for robot localisation. Depth errors and covariance are obtained from the investigation. The obtained results provide important parameters for a RGB-D camera related computation, such as a SLAM problem, etc.

An Extended Kalman Filter (EKF) based Visual SLAM method and an iterated video frames module integrated in the EKF are developed. The Visual SLAM method can handle feature detection and corresponding depth measurements in an efficient way, and the iterated video frames module is capable of maximise robot self-localization accuracy. Experimental results demonstrate the accuracy of states estimate. These two method enables a mobile robot navigates accurately in an indoor environment with low computation cost.

In addition, this study also presents a SLAM-O method by integrating object detection into Visual SLAM. SLAM-O enables robots to locate objects of interest, which can be used in robotic applications, such as navigation, object grasp, etc. To locate objects which are semi-transparent or closely connected, a K-Means method to clustering pixels on a semi-transparent object's surface and a concave based method for object separation are developed. Experimental results prove the methods efficiency. These two methods are efficient and useful tools for object detection in SLAM-O framework.

Contents

Acknowledgement i
Abstract iii
List of Figures ix
List of Tables xii
Chapter 1 Introduction 1
1.1 Motivation 1
1.2 Statement of Research Problem 2
1.2.1 RGB-D Cameras' Performance Investigation 3
1.2.2 Self-localization for a Mobile Robot 4
1.2.3 EKF Model based SLAM for Environment Observation 5
1.2.4 SLAM and Object Detection 6
1.3 Original Contribution 7
1.4 Publications 9
1.5 Thesis Structure 10
Chapter 2 Background and Related Work 12
2.1 VEX Robotics 12
2.1.1 Introduction of VEX Robotics 12
2.1.2 VEX Robot Competition 15
2.1.3 Problems Need to Solve in VEX 16
2.2 RGB-D Cameras 18
2.2.1 Overview 18
2.2.2 Application of RGB-D Cameras 20
2.2.3 RGB-D Cameras in 3D Mapping 21
2.3 SLAM and Navigation 22
2.3.1 Introduction of SLAM Method 22
2.3.2 Visual SLAM Methods 26
2.3.3 Robot Navigation 29
2.3.4 SLAM in a Navigation Process 32

2.4 Image Processing 36
2.4.1 Feature Detection 37
2.4.2 Object Detection 43
2.4.3 Machine Vision Techniques 45
2.5 Conclusion and Discussion 48
Chapter 3 RGB-D Cameras Comparison 50
3.1 Introduction 50
3.2 RGB-D Camera Development Framework 52
3.2.1 Hardware 52
3.2.2 Software 56
3.2.3 Hardware and Software Implementation 57
3.3 Experimental Approach 57
3.3.1 Experiment Design 58
3.3.2 Experiment Setup 58
3.4 Results 60
3.4.1 Depth Quality Comparison 60
3.4.2 RGB Images Quality and Chessboard Corners Detection Comparison 62
3.4.3 Experimental Method for Depth Measurements 68
3.4.4 Point to Plane Distance by Point Cloud Library 74
3.4.5 Processing Time 80
3.5 Discussion 81
3.6 Conclusion 82
3.7 Contribution 83
Chapter 4 A Real-time Visual SLAM Method for Robot Navigation 84
4.1 Introduction 84
4.2 Preliminaries 87
4.2.1 Robot and Landmarks Description 87
4.2.2 EKF Prediction and Update 90
4.2.3 Covariance Analysis 92
4.3 Approach 93

4.4 Simulation Results 96
4.5 Experimental Results 98
4.5.1 One Landmark 98
4.5.2 Two Landmarks Analysis 102
4.5.3 Three Landmarks Analysis 106
4.5.4 Selected Frames Processing 109
4.6 Discussion 114
4.7 Conclusion 116
4.8 Contribution 117
Chapter 5 A SLAM-O Method for Robot Navigation and Object Detection 118
5.1 Introduction 118
5.2 SLAM-O Method Framework 120
5.2.1 Object Localization Method in EKF-SLAM 120
5.2.2 K-Means Based Objects Detection 121
5.2.3 Connected-Object Separation 124
5.3 SLAM-O Method Implementation 126
5.3.1 Play-ground Description 126
5.3.2 SLAM-O Method Implementation 128
5.4 Results 128
5.4.1 Mapping of SLAM-O Method with 20 Iterated Frames 128
5.4.2 Mapping using Covariance based Frame Selection Method 131
5.4.3 Mapping using GMM Frame Selection Method 134
5.4.4 Colour Identification and Connected-Object Separation methods 140
5.4.5 Object Tracking in SLAM-O 141
5.4.6 Object Separation in SLAM-O 141
5.5 Discussion 144
5.6 Conclusion 146
5.7 Contribution 146
Chapter 6 Case Study 147
6.1 Introduction 147

6.2 Case Study One 147
6.3 Case Study Two 151
6.4 Conclusion 155
Chapter 7 Discussion 156
7.1 Future Work 156
7.2 Concluding Remark 158
Appendix A 159
Appendix B 168
References 171

List of Figures

Figure 2- 1: Mechanical structure of a VEX robot 14
Figure 2- 2: Game of 2013-2014 (Robotics) 16
Figure 2- 3: Game of 2014-2015 (VEX Robotics) 16
Figure 2- 4: Game of 2015-2016 (Robotics) 16
Figure 2- 5: Game of 2016-2017 (Robotics) 16
Figure 2- 6: The essential SLAM problem 24
Figure 2- 7: Illustrate of Non-Maximum suppression 39

Figure 3- 1: A PrimeSense sensor 54
Figure 3- 2: A Kinect V1 sensor 54
Figure 3- 3: An Xbox Kinect sensor 54
Figure 3- 4: A Kinect adapter for Windows 55
Figure 3- 5: Experiment design 59
Figure 3- 6: Experiment setup 59
Figure 3- 7: Depth points distribution on a chessboard 0.5m away 66
Figure 3- 8: Depth points distribution on a chessboard 1m away 66
Figure 3- 9: Depth points distribution on a chessboard 1.5m away 67
Figure 3- 10: Kinect V2 depth points in a chessboard at 2m, 2.5m and 3m away 67
Figure 3- 11: Camera and chessboard position description 69
Figure 3- 12: Point-to-plane distance 69
Figure 3- 13: Point-to-plane distance errors 70
Figure 3- 14: Point-to-plane covariance 70
Figure 3- 15: Primesense measurements fit equation 73
Figure 3- 16: Kinect V1 measurements fit equation 73
Figure 3- 17: Kinect V2 measurements fit equation 74
Figure 3- 18: A flow chart for obtaining a perpendicular plane with PCL 77
Figure 3- 19: Point-to-plane distance with PCL method 77
Figure 3- 20: Point to plane distance errors with PCL method 78
Figure 3- 21: Primesense measurements fit equation with PCL 78
Figure 3- 22: Kinect V1 measurements fit equation with PCL 79
Figure 3- 23: Kinect V2 measurements fit equation with PCL 79
Figure 3- 24: Data processing time for three cameras 80

Figure 4- 1: A Visual SLAM framework 86
Figure 4- 2: The designed mobile robot in this study 89

Figure 4- 3: Data flow in the robot 89
Figure 4- 4 NEES comparison 97
Figure 4- 5: Time costs comparison 97
Figure 4- 6: Experiment design 99
Figure 4- 7: Distance comparison from 0 to 0.3m 99
Figure 4- 8: Distance comparison from 0.3m to 0.6m 100
Figure 4- 9: Distance comparison from 0.6m to 0.9m 100
Figure 4- 10: Distance comparison from 0.9m to 1.2m 101
Figure 4- 11: Time comparison for each stop 101
Figure 4- 12: Distance comparison from 0 to 0.3m with two landmarks 104
Figure 4- 13: Distance comparison from 0.3m to 0.6m with two landmarks 104
Figure 4- 14: Distance comparison from 0.6m to 0.9m with two landmarks 105
Figure 4- 15: Distance comparison from 0.9m to 1.2m with two landmarks 105
Figure 4- 16: Time comparison for each stop when observing two landmarks 106
Figure 4- 17: Distance comparison from 0 to 0.3m with three landmarks 107
Figure 4- 18: Distance comparison from 0.3m to 0.6m with three landmarks 107
Figure 4- 19: Distance comparison from 0.6m to 0.9m with three landmarks 108
Figure 4- 20: Distance comparison from 0.9m to 1.2m with three landmarks 108
Figure 4- 21: Time comparison for each stop when observing three landmarks 109
Figure 4- 22: State estimates and covariance (every 30 frames) 110
Figure 4- 23: State estimates and covariance (every 15 frames) 111
Figure 4- 24: Two landmarks select frame method moving 1.2m 113
Figure 4- 25: Error comparison 113
Figure 4- 26: Time costs 114

Figure 5- 1: SLAM-O framework 121
Figure 5- 2: VRC Toss Up field elements. 123
Figure 5- 3: Flowchart of K-Means procedures 123
Figure 5- 4: Watershed separations transform 125
Figure 5- 5: Flowchart of object separation procedure 125
Figure 5- 6: Multiple connected-objects separation 125
Figure 5- 7: Play-ground designed 127
Figure 5- 8: Play-ground example of SLAM-O method 127
Figure 5- 9: Mapping with iterated 20 frames 129
Figure 5- 10: Object variance with 20 frames iteration 129
Figure 5- 11: K-Means based mapping with iterated 20 frames 131
Figure 5- 12: Object variance using K-Means method and with iterated 20 frames 131
Figure 5- 13: Mapping with covariance frame selection 133
Figure 5- 14: Target1 and target2's position variance 133
Figure 5- 15: Mapping with covariance frame selection and K-Means object detection 134

Figure 5- 16: Object1 and object2's position variance with K-Means object detection..... 134
Figure 5- 17: GMM based frame selection method..... 136
Figure 5- 18: Mapping with GMM frame selection..... 137
Figure 5- 19: Objects variance with GMM frame selection..... 137
Figure 5- 20: K-Means based mapping with GMM frame selection..... 138
Figure 5- 21: Object variance using K-Means method and with GMM frame selection..... 138
Figure 5- 22: Time costs..... 139
Figure 5- 23: Solid and semi-transparent objects detection..... 142
Figure 5- 24: Connected-object separation..... 142
Figure 5- 25: Multiple connected-object separation..... 142
Figure 5- 26: Five balls and semi-transparent balls separation results..... 142
Figure 5- 27: Object detection in noisy environments..... 142
Figure 5- 28: Objects tracking..... 143
Figure 5- 29: Object detection when two objects connected..... 144
Figure 5- 30: Object detection with the proposed separation method..... 144

Figure 6- 1: Case study1 field description..... 148
Figure 6- 2: VEX field when moving diagonal towards a landmark..... 148
Figure 6- 3: Mapping results of case study one..... 150
Figure 6- 4: Object variance in case study one..... 150
Figure 6- 5: Case study2 field description..... 152
Figure 6- 6: Mapping results of case study two..... 153
Figure 6- 7: Object variance in case study two..... 154

Figure A- 1: Flowchart of development process..... 161
Figure A- 2: 12 Volts rechargeable batteries..... 163
Figure A- 3: The mobile robot used in experiments of this thesis..... 165
Figure A- 4: Robot straight moving testing..... 166
Figure A- 5: Robot straight moving and turning..... 167

Figure B- 1: Electronics parts of the robot..... 168
Figure B- 2: Sizes of the electronics parts..... 169

List of Tables

Table 2- 1: Features of the goal objects in VEX fields 16
Table 2- 2: Stereo camera method to derive depth information 47

Table 3- 1: Hardware features for RGB-D cameras 55
Table 3- 2: Software compatibility for each RGB-D camera 55
Table 3- 3: Software libraries used for each RGB-D camera 55
Table 3- 4: Apparatus in the experiment setup 59
Table 3- 5: Depth distribution comparison 61
Table 3- 6: RGB images comparison 64
Table 3- 7: Covariance comparison 71

Table 4- 1: Comparison among selected frames, 10 frames and 20 frames 114

Table A- 1: Frame structure and driving system costs from Kiwibots (KIWIBOTS, n.d.) 161

Table B- 1: Hardware parts for the mobile robot structure 170
Table B- 2: Electronics parts 170

Chapter 1 Introduction

This chapter reviews the existing literature on robot navigation, introduces the Simultaneous Localization and Mapping (SLAM) problem. It provides a conceptual explanation of why localization and object detection is critical for successful robot navigation, and describes the contribution and organization of the thesis.

1.1 Motivation

The overall aim of this thesis was to contribute to the navigation and object detection of autonomous robots competing in the VEX Robotics Competition (VRC), in order to compete with human participants. The VRC is one of the largest, fastest growing educational programmes in the world. It is designed to "increase student interest and involvement in Science, Technology, Engineering, and Mathematics (STEM)" (Robotics, 2017a). Each VRC competition consists of a number of matches played out on a 12' x 12' field; where, each match consists of two teams (the RED and BLUE alliances) who compete against each other for two minutes. For an autonomous robot to win the competition, it needs to optimise between detection of goal objects, navigation, and scoring, in order to maximise its score within the match time.

To develop a fully autonomous robot that can win a VEX competition match, a reliable ability to perceive and handle the complexity of unknown and unconstrained environments is needed. If the robot has a map, and precisely know its current position, it can determine optimal pathway to move from one place to another. This field of research, known as SLAM, has been a

core area of research for several decades (G. Dissanayake et al., 2011). While research communities continue to make progress on algorithms for solving SLAM problems, a practical solution for a mobile robot at a critical level of effectiveness and robustness is still lacking. Some methods based on visual features, such as Scale-Invariant Feature Transform (SIFT) or Speeded-Up Robust Features (SURF) are too slow for real-time navigation (Endres et al., 2012; Henry et al., 2010). Some patch-based methods are fast, e.g. Reliable Patch Trackers (RPT), but do not optimally observe the environments and neglect major parts of the data (Klein & Murray, 2007; Y. Li et al., 2015). Most robots rely on external infrastructures, which do not assist navigation in some cases. For example, Global Positioning System (GPS) is often used (Konolige et al., 2008), but in many cases, GPS signals are unavailable, e.g. indoors, near tall buildings, under foliage, underground, and underwater (Kümmerle et al., 2015). To solve this problem, this study aims to increase the effectiveness and robustness of navigation systems by formulating a new solution to SLAM based on a strategy of repeating observations. This approach allows the robot to optimise between localization accuracy and simultaneous environment observation, aiming to maximally score interest objects in competition environments.

1.2 Statement of Research Problem

The problem that will be addressed in this thesis is the following: in a VEX competition, how can we enable a robot to logically balance the time spent between object detection, navigation, and scoring, how to maximise its scoring during a match and beat its human opponent?

This is a hard problem to solve for a number of reasons. Firstly, a robot needs to observe an environment before localizing itself and goal objects; however, observations from sensors are unpredictable. Take visual sensors for example: two frames of the same scene may be differ-

ent due to changes in lighting or blurring caused by arrival or departure of the robot. Secondly, how to determine the relationship between environment observation and pose estimation? This may relate to the state estimate and observation models in SLAM methods. Thirdly, and more challenging, the current environment may continually change and little information about the environment may be known at any given time. This means their navigation strategy must be capable of reacting on to these minimal and dynamic inputs. These are challenges for developing a navigation strategy in unknown environments.

Furthermore, how can we obtain an accurate and robust self-localization, and locate objects of interest with optimal observation of environments? Is it possible to obtain a more accurate mapping result if a robot keeps on observing the landmarks? What is the optimal balance between observing and mapping an environment? To answer these questions, this research focuses on three areas: visual sensors' accuracy evaluation and comparison, relationship between robot self-localization and environment observing, and object detection and mapping.

1.2.1 RGB-D Cameras' Performance Investigation

Light weight commodity RGB-D cameras allow for indoor robotic navigation requiring limited weight and power consumption. Thus, RGB-D cameras are prominent within indoor applications. Before usage, an RGB-D camera's performance needs to be measured. PrimeSense and Microsoft cameras are investigated in this study.

Light Coding technology owner, PrimeSense, provided their technology for the development of first generation of Microsoft Kinect, while the second generation of Kinect uses Time of Flight (ToF). Most research in RGB-D cameras focuses on building 3D maps of environment and spatial alignment of consecutive data frames. However, the ability to locate a single object, and

the performance in terms of repeatability and accuracy were neglected. These capabilities are essential for robotic applications, e.g. interesting object detection and object grasping. This thesis conducts a study of RGB-D cameras from PrimeSense and Microsoft, and further presents combined hardware and software solutions of RGB cameras for robotic application. Study of the cameras are investigated in terms of colour image quality, depth points' distribution, and depth accuracy when measuring point to plane distance.

1.2.2 Self-localization for a Mobile Robot

Visual systems for mapping and localising in an environment are a key technology for many real-time applications, such as automatic mobile robots and object detection (Steder, 2013; Vandewouw et al., 2016). To localize a mobile robot, odometry and SLAM methods can be used. However, both of these methods suffer from drift due to the accumulation of errors, e.g. dead-reckoning (Chung et al., 2001), and inaccuracy in SLAM estimation (Lee & Song, 2007; Y.-T. Wang et al., 2010). Some SLAM estimation results are not promising, either the output errors are relatively high (up to 0.3m), or the robot only moves a short distance (within 1m) which is, arguably, not convincing to prove the accuracy of a SLAM algorithm. However, SLAM methods have the possibility of eliminating errors through algorithm improvements, such as constrained individual image poses and large-scale reconstruction with infrequent loop closures, direct tracking by image-to-image alignment (Engel et al., 2014; Kerl et al., 2015). These methods require off-line working or assistance of a GPU to work in real-time, as such the new light weight RGB-D cameras normally asks for computationally simpler methods for efficiency (Moosmann, 2013). Due to this, an Extended Kalman Filter (EKF) based Visual SLAM method was developed. This method is capable of handling RGB feature detection and their corresponding depth processing in an efficient way. This study makes use of Visual SLAM techniques

for robot precise position estimates, and aims to provide a practical approach for efficient and accurate robot navigation in an indoor environment.

1.2.3 EKF Model based SLAM for Environment Observation

EKF is a classic module used in SLAM method, which was originally proposed by Davison (2003). The EKF is used to compute an estimate of a state vector representing a robot (which is also the camera's position) and environment features (e.g. landmarks), together with covariance P representing the correlated errors in the estimation. Practically, most SLAM implementations are nonlinear processes; however, EKF linearizes the states by implementing a Gaussian distribution over the state vectors. When the robot observes a feature and updates its location, other features and robot's position can also be accessed, because their errors are correlated. As the EKF is a dynamic and recursive implementation, the EKF solution to SLAM is efficient and has been used successfully in indoor environments (Pollefeys et al., 2008).

One disadvantage of the EFK algorithm is that states estimates are approximate due to linearization and may not accurately match the ground truth, i.e. the EKF output maybe inconsistent. In Bailey et al.'s study, these approximation may not accurately match the true states, as EKF SLAM produces inconsistent estimates (Bailey, Nieto, et al., 2006b). Also, another research found if a stationary robot measuring the relative position of a new landmark multiple times, the estimated variance of the robot will decrease (Julier & Uhlmann, 2001). However, the research did not demonstrate if the decreased variance affected state estimates. As variance is a parameter in the EKF model, the theory that changing the EKF model will yield more consistent estimates is supported in Tamjidi et al.'s study (Tamjidi et al., 2009). Therefore, the relationship between variance and state estimates will be a topic to discuss. By looking into the pa-

rameters of the EKF model, a method of iterating observation-frames is proposed to keep EKF consistent and maintain position estimates.

How to determine the observation times and how to select a subset number of observation are also discussed. A trade-off method between estimates accuracy and observation times is presented.

1.2.4 SLAM and Object Detection

The introduction of RGB-D cameras brought image processing techniques, such as feature detection, patch detection, etc. into SLAM methods (Henry et al., 2012; Song et al., 2015). This study integrates object detection into the SLAM method (SLAM-O), where objects can be detected and tracked from multiple view-points as a robot moves around. With SLAM estimates, objects can be located and tracked based on view-points observations.

The SLAM-O method was carried out on VEX robotics competition field. The identification of semi-transparent and separation of connected buckballs are two issues to solve. A new object detection method for semi-transparent objects in VEX field was proposed. Due to the nature of semi-transparent object is made of thin material and revealed the background through reflections. Overlapped buckyballs are a normal scene on VEX field due to variation of viewpoints, but could not be separated by traditional Watershed transformation significantly (Sezgin, 2004), which does not work well when two objects overlap. A K-Means clustering method was developed to classify the semi-transparent ball by accumulating the various colour tints, and a concave based method is developed for overlapped objects separation.

1.3 Original Contribution

A robot should have certain information processing abilities; including, how to use information under conditions of uncertainty, and how to extract features from visual information. Equipped with these abilities, a mobile robot can localize its own position, determine where the goal is, and navigate to the desired position. The best estimate of the robot's poses and feature positions for indoor environment is important for building maps successfully. Precise, real-time visual odometry is a key skill for a mobile robot. This study aims at generating best estimate of the robot poses, and optimise time consuming and estimation accuracy. A range of contributions have been made:

1. A novel EKF based Visual SLAM method using a RGB-D camera. Compared to a typical laser sensor based EKF, the presented system is equipped with higher environmental recognition ability due to an assisted RGB-D camera. It is found that the proposed method offers dramatic improvement over techniques, such as frame-by-frame matching, in that it handles feature detection and corresponding depth measurements in an efficient way. Furthermore, this method can be used in a wide range of scenarios, such as: robot competition, object grasping, etc. for robot navigation and interesting object detection.
2. An improved solution for state estimates in EKF SLAM: This study discusses if iterating observation frames can improve the state estimates. Each time a new observation is made and the EKF parameters are accordingly updated. The new EKF gives an accurate estimate which is closer to ground truth. By iterating observation frames, the output estimates converge faster. A relationship between observation frames and estimate

accuracy is discussed. This method gives more accurate state estimates than a normal EKF SLAM method.

3. How to select a subset of observation frames is discussed. A method that can determine the number of observation frames for iteration is proposed. In an EKF SLAM framework, this method achieves accuracy in a certain level and avoiding iterating a large number of frames, and thus balances computation time and estimate accuracy. The proposed method allows an EKF SLAM to obtain accurate estimates with relatively low time costs.

4. Functions of widely used RGB-D cameras are compared. Performance of colour image quality, depth data distribution and depth sensor accuracy is demonstrated by experiments. Software configuration methods for each camera are listed which are a guidance for configuring different cameras. A point clouds based perpendicular distance calculation method is proposed. This method improves point to plane distance measurements. Based on point to plane distance's comparison among different cameras, relationships between measurements and ground truth are built. These comparison results and the obtained parameters provide useful advice for researchers and developers when choosing a camera for a specific use.

5. A new method for semi-transparent objects detection. A colour pixels clustering method for semi-transparent ball detection is proposed, which is a new method to solve problems of semi-transparent object detection. The method improves upon the traditional colour thresh-holding method, which usually misses a large portion of objects body. While the K-Means clusters similar red pixels of a ball into one group, and

successful segments semi-transparent objects. This method provides an efficient strategy for semi-transparent objects detection.

6. A new method for object separation. A method for object separation based on concave defects is developed. The method outperforms the Watershed method when separating objects overlap in the image due to proximity and angle of the view. Experiments prove the method effectiveness compared to a Watershed based separation method. In addition, this method can work on multi-object separation, even when the object is occluded. This object separation method contributes to closely connected objects' separation.

1.4 Publications

The following publications are the results of research carried during this PhD:

A Colour Feature Detection and Objects Separation Method for VEX Robot. Jing, C., Potgieter, J., & Noble, F. (2014, July). The Third International Conference on Robot Intelligence Technology and Applications 2014.

A Comparison and Analysis of RGB-D Cameras' Depth Performance for Robotics Application. Jing, C., Potgieter, J., & Noble, F. (2017, Sepetember). 2017 24th International Conference on Mechatronics and Machine Vision in Practice (M2VIP).

1.5 Thesis Structure

Chapter 2 presents the necessary background to this thesis by discussing common SLAM methods and their pros and cons when used on a mobile robot. The navigation and control methods for robots are presented; also a discussion of visual sensors and image processing techniques is presented.

Chapter 3 gives a detailed introduction of frequently used RGB-D cameras, e.g. PrimeSense, Kinect V1 and Kinect V2. Cameras' functions, including colour image quality, depth sensor accuracy and camera sensor performance efficiency, are compared through experiments. Based on comparison results, a relationship between measurements and ground truth is built.

Chapter 4 discusses Visual SLAM method - how iterated observation frames affect the robot's self-localization. A relationship between number of observation frames and SLAM estimates is demonstrated through experiments. A video frame selection strategy is proposed to maximise time efficiency and output accuracy.

Chapter 5 proposes a SLAM-O method for a mobile robot real-time navigation and object detection. This chapter addresses two issues of SLAM and object detection in an indoor environment. The first is semi-transparent object detection and connected objects separation. The second concerns objects tracking and data association.

Chapter 6 examines the proposed SLAM-O method in a case study, and presents a complete SLAM-O method on a mobile robot navigating in a VEX competition field. Experimental results are demonstrated, and the robot is shown to permit very robust self-localization.

Chapter 7 presents conclusions and suggests future direction for the completion and extension of this work.

Chapter 2 Background and Related Work

In this chapter, a review of techniques and previous work related to VEX robotics, RGB-D camera, Visual SLAM, robot navigation, and image processing, are presented. The main sections in the chapter include, but are not limited to, the following:

- VEX Robotics. Here, VEX is used as a case study for the proposed Visual SLAM method. The VEX Robotics Competition and VEX field are introduced; current issues in a VEX competition are discussed.
- RGB-D camera. Here, RGB-D cameras' development, applications, and their usage in robot navigation and Visual SLAM are introduced.
- Visual SLAM. Here, this chapter presents developments of SLAM methods, Visual SLAM methods and robot navigation, and discusses how Visual SLAM methods work in a robot's navigation process.
- Image processing. Feature detection, object detection and machine vision are discussed.

2.1 VEX Robotics

2.1.1 Introduction of VEX Robotics

VEX Robotics Competition (VRC) is one of the largest, fastest growing education robotics programmes in the world (Foundation, 2017). In New Zealand there are over 100 teams actively participating in the VRC programme. It is designed to "increase student interest and involve-

ment in science, technology, engineering, and mathematics (STEM)" (VEX Robotics) and with a particular focus in New Zealand to "inspire a passion for Science and Technology" (KIWIBOTS).

To engage in a competition, a VEX robot is required be programmed with a degree of flexibility, so that it can perform tasks and interact with its environment smoothly (An example of a VEX robot is in Figure 2- 1). A normal VEX Robot system Design includes several different subsystems, namely, structure, motion, power, sensor, logic and control (Robotics, 2017b). These systems are described below:

Structure subsystem: This is responsible for physical support. It includes the elements, such plates, bars, rails, and bumpers etc. which construct the mechanical body of a robot.

Motion subsystem: This inacts responses to a robot are driving functions and involves a motor module clutch and servo module with clutch. The motor module is used whenever a continuous rotation is needed, and the servomotors are only used in cases where the boundaries of motion are well defined.

Power subsystem: this includes 7.2-volt batteries and a battery box.

Sensor subsystem: Two sorts of switch are used in this system, namely a limit or bumper switch. Limit Switches are used to cut power to motors when depressed and enable power to pass when released, and bumper Switches are used to detect when a robot has made contact with an obstacle.

Figure 2- 1: Mechanical structure of a VEX robot

Logic Subsystem: This is the Central Processing Unit (CPU) of a robot that directs a robot's behaviour in response to different circumstances or inputs. In VEX Robotics, the CPU is a Programming Kit, which is a combination of hardware and software that enable users to write programs on a computer and download them to the robot's microcontroller.

Control subsystem: This includes a software program that tells a robot how to act in different circumstances and hardware electronics that process the information. In VEX Robotics, a transmitter is used to accept different frequency signals to allow for operations of multiple

robots simultaneously; a Receiver Module is used to accept different receiver signals from the matching transmitter frequency.

2.1.2 VEX Robot Competition

In a VRC program, each tournament includes practice, qualifying, and elimination matches. After the qualifying matches, teams will be ranked based on their performance. The top teams will then participate in the elimination matches to determine the tournament champions.

Each match consists of two teams – the RED and BLUE alliances – played out on a 12' x 12' field competing against each other for two minutes. The first 45 seconds of each match is dedicated to an autonomous period, where each team' robots are programmed to play the game on their own. The remaining 75 seconds is dedicated to a driver period, where the teams' robots are programmed to responds to a driver's controls (New Zealand KiWibots). A bonus is awarded to the alliance that has the most total points at the end of the autonomous period. The competition field is dynamic as a result of the robots' interaction. The competition field utilised in 2013-2014's international VRC is illustrated in Figure 2- 2. This game utilised a total of twenty buckyballs and eight large semi-tranparent balls available as scoring objects.

VEX Robotics releases a new game each year, to challenge the programming skills of competetors. Figure 2- 3 illustrates the 2014 - 2015 game "Skyrise". Within this game a total of forty-four cubes, twenty-two red and twenty-two blue, and fourteen "Skyrise" sections were utilised as scoring objects. Figure 2- 4 illustrates the 2015 - 2016 game "Nothing But Net". Within this game ninety-four (94) balls and ten (10) bonus balls were utilised as scoring objects. Figure 2- 5 illustrates the 2016-2017 game "Starstruck". The goal objects of the VRC have features in terms of colour and shape, here are summerized in Table 2- 1.

Figure 2- 2: Game of 2013-2014 (Robotics)

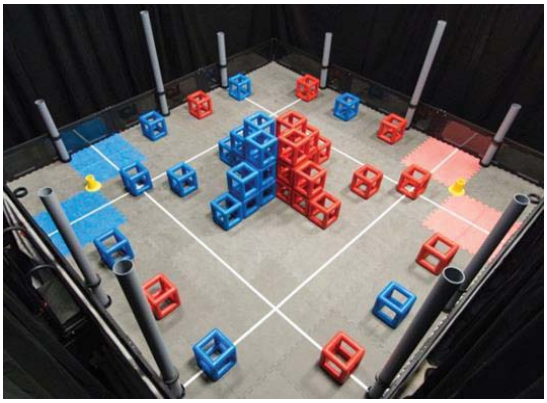


Figure 2- 3: Game of 2014-2015 (VEX Robotics)

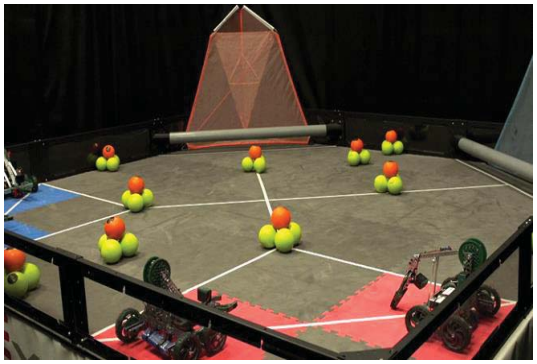


Figure 2- 4: Game of 2015-2016 (Robotics)

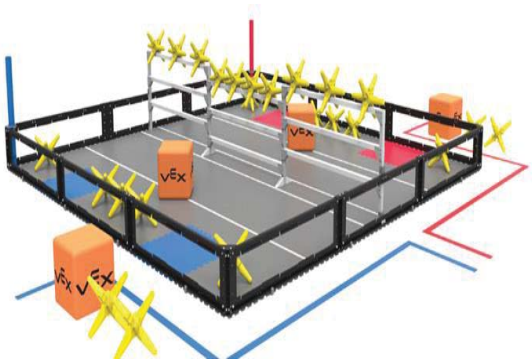


Figure 2- 5: Game of 2016-2017 (Robotics)

Table 2- 1: Features of the goal objects in VEX fields

Games and goal objects for each season	Features of objects
"Toss Up": Buckyballs	Colour: blue and red; Shape: circle
"Skyrise": Cubes	Colour: blue; Shape: polygon
"Nothing But Net":Balls	Colour: green and orange; Shape: circle
"Starstruck":Stars	Colour:yellow; Shape:star

2.1.3 Problems Need to Solve in VEX

The autonomous period of a robotics game yields four predominant obstacles to overcome, namely object identification, encoder drifting, real-time navigation and strategy.

- Objects identification: most of the existing methods for and shape detection, such as colour threshold, Hough line and angle detection, seem to work well under certain limited image condition. However, variations in illumination, orientation, noise, scale, viewpoint and elements' material reflection, etc. will bring challenges to achieve invariant object detection.
- Encoder drifting: the integrated VEX encoder indicates relative movements by detecting light that is reflected from a series of slots in a sensing wheel. The encoder detects false signals in the presence of mechanical vibration and electrical noises. As a result, the positions drift and accuracy degrades.
- Real-time navigation method: at the moment, competitors use offline navigation methods for a VEX robot programming for an autonomous period. However, real-time navigation is essential for a robot to operate in a competition where environments are subject to change over time, because failure and new path requirements are frequent. Thus, it is inefficient to accomplish a task using predefined path.
- Strategy: a strategy is needed to win the match in the autonomous period. As a VEX robot exists in a field of dynamics and uncertainty, navigation tasks may not be reliably carried out. In order to score, a robot must be able to optimise between observing objects and navigating. If the robot spends more time observing objects, other robots may take goal objects and end up lose the game. To obtain more scores and win the match in a limited time, the robot should reliably navigate with optimal observation of environments.

To summarize, current VEX needs reliable navigation, observation and localization strategies. A predominant methodology capable of this is the Visual Simultaneous Localization And Mapping (SLAM) method that embraces the problem of observed uncertainty.

2.2 RGB-D Cameras

2.2.1 Overview

As a new generation of sensing technology, RDB-D cameras capture RGB images along with depth information for each pixel. In this section, a number of these cameras are introduces and discussed, including the Microsoft Kinect, Asus Xtion Pro Live and SR4000.

Microsoft Kinect sensors have been widely used since the launch of the Windows V1 sensor in November 2010. This was one of the first inexpensive depth sensors consisting of an IR camera, a RGB camera and an IR projector that casts a fixed speckle pattern. Conversion of the pattern to a depth map is through a triangulation process described by the inventors (Freedman et al., 2012), which is based on a "light coding" technology from PrimeSense. "Light coding" is described as three steps: step1: the IR laser source emits a single beam which is split into multiple beams by a diffraction grating to create a constant pattern of speckles projected onto the scene; step2: this pattern is captured by the infrared camera; step3: the capture pattern is correlated against a reference pattern, which is obtained by capturing a plane at a known distance from sensor and stored in the memory of the sensor. Other visual sensors using "light coding" techniques include PrimeSense sensors (i.e. the Carmine 1.08, Carmine 1.09 and Capri 1.25) and ASUS Stion (Boehm, 2012).

In July 2014, Windows has released a new sensor, the Kinect for Windows v2 sensor (Kinect V2). The new sensor uses a Time of Flight (ToF) technology, which use active sensors to measure the distances of surfaces by calculating the jorgon time of a pulse of light (Kolb et al., 2009). The ToF method is supposed to reduce the amount of noise and improve the accuracy of measurements. Kinect V2 is composed of two cameras, namely a RGB and an IR camera. To enable the use of the sensor, official Microsoft Software Development Kit (SDK) 2 was published for developers and researchers.

There are several RGB-D cameras using ToF technology, e.g. the PMD and SR4000. The CamBoard nano from PMD was introduced as the smallest RGB-D camera, aiming at close-range applications. Its next generation, CamBoard picoS, can detect a large scale and is designed for professional use in places, such as research and development facilities (PMDec, n.d.). The size of picoS module is 4mm thick, 39.5mm long and 15.5mm wide, which is 1.5mm thinner and half-length of module nano. Those smart cameras are at a higher price, which is up to 59,000 European dollars.

Swiss Ranger SR 4000 from MESA also uses ToF technology. Different from Microsoft which is initially designed for computer games, MESA's SR 4000 is built for machine vision use. One advantage of SR 4000 is that scene illumination is provided by camera, and does not depend on illumination from external light (Ltd, n.d.), which is ideal for outdoor usage. A disadvantage is that the resolution is relatively low (176*144) and nearly half size of a normal Microsoft camera.

2.2.2 Application of RGB-D Cameras

RGB-D cameras have the advantages of low price, small size and light-weight, and visual systems using these are capable of providing high quality synchronized videos of both colour and depth. The application areas of these are mainly in computer vision and robotics, such as 3D reconstruction, human pose detection and analysis, object detection and tracking.

In 3D reconstruction, Cappelletto et al. (2013) presented a pipeline to use the Kinect sensor as an handled scanner. Reliable 3D reconstructions are achieved from their study. With monocular image and depth stream provided by a Kinect V1 camera, Weiss et al.(2011) presented a method for human shape reconstruction. Without RGB-D cameras, human shapes reconstruction relies on recording videos with synchronized cameras (Balan et al., 2007). These setups require special hardware, and cannot make use of commodity camera hardware with limited frame rates. Skeleton tracking is a popular used application in Kinect SDK. Dutta (2012) used Kinect SDK to capture 3D motion, and provided a portable 3-D motion capture system for performing ergonomic assessments. Obdrzalek et al. (2012) examined the accuracy of joint localization and pose estimate in their Kinect system. Object recognition and tracking is another hot topic in RGB-D-based applications. However, this approach in Kinect SDK presumes the background is static, which is not suitable for dynamic environment. Han et al. (2013) proposed a Gaussian mixture model (GMM) for dynamic background modelling. Instead of RGB images, they used depth images for foreground mask and obtained better quality results. Depth images are also used for analysis of human activity, e.g. pose detection. Xia et al. (2011) used a 2-D contour model from a RGB images and a 3-D surface model from depth images for human pose detection.

RGB-D cameras have limitations when it comes to accuracy in centimetres, because they provide depth only up to a limited distance (typically less than 5 m), with depth estimates containing noise (~0.03m at 3m depth). Khoshellham and Elberik (2012) concluded that the Kinect V1 sensor data should be acquired in a range between 1m-3m to the sensor, and they claimed the data quality is degraded by the noise and low resolution of the depth measurements. Due to these constraints, RGB-D cameras are not usually adopted in big open areas. There are studies that discuss solutions for depth correction, such as Smisek et al. (2011) who discovered an offset between the measured depth and expected depth, and Herrera et al. (2012) who used the inverse of depth and disparity, for depth correction.

2.2.3 RGB-D Cameras in 3D Mapping

3D mapping refers to sense the 3D world and represent 3D spatial information in a map. Previously, 3D mapping approaches relied on laser scanners (Besl & McKay, 1992), which are expensive and heavy. After a consecutive scan, the usage of iterative-closet-point (ICP) matching proved capable of obtaining an estimate at millimetre accuracy (Röwekämper et al., 2012). However, a good initial guess is a preliminary to solve ICP problems, where ICP algorithms can converge quickly to desired transformations and yield correct correspondences; otherwise, ICP algorithms would converge to incorrect solutions. In contrast, RGB-D cameras offer a valuable alternative for distance scanning, such as KinectFusion, which is an approach for mapping indoor scenes using RGB-D cameras (Newcombe et al., 2011). In this approach, a scene is represented as a signed distance field (SDF), which is defined on a volumetric grid. Real-time performance of indoor mapping can be achieved if high performance graphics is equipped. Some later proposed frameworks have improved real-time capabilities but lack effective mechanisms for error propagation. Thus, Kinectfusion is suitable for scene reconstruction, but not for cam-

era trajectory estimation. Hornung et al. (2013) developed an open-source 3D mapping framework to construct a 3D geometric representation of the environment. This framework is based on probabilistic theory for representation of reflections and dynamic obstacles. Their mapping results achieved a resolution of 0.02m. Steinbrücker et al. (2014) developed a volumetric mapping method that represents a scene at multiple scales, which computes faster than Kinectfusion and runs on a CPU.

With RGB images provided by RGB-D cameras, visual features from image processing methodologies can be used for 3D mapping. Heredia et al. (2015) used feature descriptors for 3D mapping, they used the correspondences between images and their feature maps to compute a spatial matching. Taguchi et al. (2013) used both point and plane features as primitives for the registration of 3D data. Alternatively, dos Santos et al. (2016) used coarse-to-fine registration of RGB-D data. In their framework, point features, disparity values and plane surfaces were used. Though registrations of RGB and depth images, camera-body transformation can be computed. The transformation results can be integrated into Simultaneous Localization And Mapping (SLAM) algorithm, which employ encoders, GPS, loop closure etc. techniques to optimise a camera's position.

2.3 SLAM and Navigation

2.3.1 Introduction of SLAM Method

SLAM is used in the robotics community to describe the problem of building a model of an environment and estimating the trajectory of the robot simultaneously. Construction of an

environment model and estimation of current location are crucial for a robot's task accomplishment.

The SLAM method was created and presented in 1986 at IEEE Robotics and Automation Conference where probabilistic methods were introduced into robotics and artificial intelligence (AI). Since then, a number of researchers have started to use estimation-theoretic methods for consistent mapping and localization problems. Smith and Cheeseman (1986) presented relationships between landmarks and manipulating geometric uncertainty, correlations of different landmarks were considered related to successive observation. The idea of correlations was further developed by Smith et al. (1990) who discovered it is the errors in the robot's self-localization that cause the correlations among landmark' estimates. A consistent estimate of localization and mapping would require a joint state composed of the robot's pose and landmarks' position, and the estimate needs to be updated following each landmark's observation. Thus, the initial formulation of a SLAM problem was established, this is illustrated in Figure 2- 6.

Due to this problem, true locations of the robot and landmarks are unknown and could not be measured directly. Estimated locations are updated by observations between the robot and landmarks. Thus, there are errors between estimates and true location. Convergence of these errors is one of the issues in the early stage of the SLAM problem. Durrant-Whyet et al. (1996) found that growing correlations between landmarks lead errors to converge, and the more these correlations grew, the better the convergence. Correlations between landmarks increased when more observations were made (M. G. Dissanayake et al., 2001a). In Figure 2- 6, when the robot moves from X2 to X3, it will observe the landmark L2 again. Both robot and landmarks' locations will be updated at this stage, even the unseen landmarks (e.g. landmark L4) which will update because of the correlations.

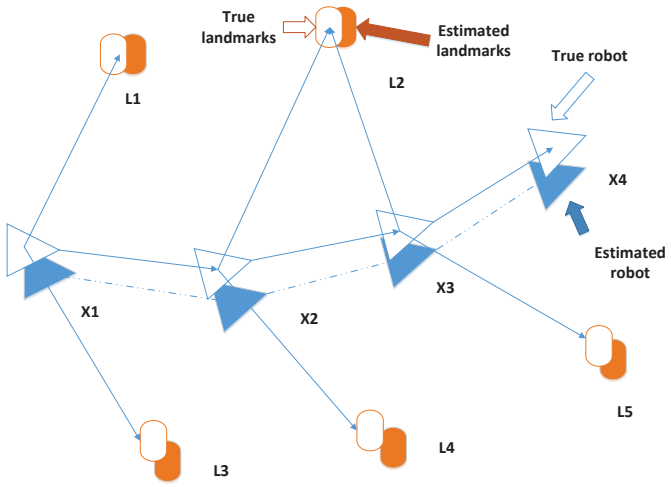


Figure 2- 6: The essential SLAM problem

Probabilistic methods were used to solve the SLAM problem at an early stage. The most common representations are Extended Kalman Filter (EKF) and Rao-Blackwellized particle filter. Another method was proposed by Thrun et al. (1998) who used a maximum-likelihood method for locations' estimate. These three methods all delineate the convergence problem, together with two other issues which improve computational efficiency and data association. As the correlation of landmarks is updated every time a new observation is made, computation requirements grow in quadratic manner with the number of landmarks. EKF formulation uses an innovation process: landmarks and landmark covariance are approximated by Gaussian distributions (M. G. Dissanayake et al., 2001b). As EKF SLAM employs linearized models of non-linear motion and observation models, convergence and consistency can be guaranteed. One disadvantage of the EKF solution is fragile to incorrect association of landmarks' observations (Neira & Tardós, 2001). The EKF is prone to failure where significant vehicle uncertainty induces linearization errors (Bailey, Nieto, et al., 2006a), or where significant clutter induces ambiguity in data association. In contrast of the EKF, FastSLAM uses Monte Carlo sampling or particle

filtering directly to represent the non-linear process model. FastSLAM does not suffer from the linearization problem, and is much more robust in situations of association ambiguity (Montemerlo et al., 2002). In a FastSLAM, joint state is used, and factored into a vehicle and a conditional map component. When a map is conditioned, landmarks become independent. Thus, the map can be represented as a set of independent Gaussians with linear complexity. The disadvantage of FastSLAM is its inability to maintain particle diversity over long periods of time (Bailey, Nieto, & Nebot, 2006). The fundamental problem is that the sampling filter is operating in a very high-dimensional space and the number of particles needed is therefore exponential in the length of the trajectory.

The second stage of SLAM algorithms' development focuses on algorithmic analysis, including developing efficient solvers for SLAM problems, and analysing fundamental properties of SLAM, such as observability, convergence and consistency. Thrun et al. (Thrun & Montemerlo, 2006) developed GraphSLAM to solve SLAM problems offline. Similarly, Olson et.al (2006) presented a pose graph method. Both of these solutions are based on a graphical factor (Kschischang et al., 2001). The Graphical SLAM method is popular because of it enables flexible data association and being capable of handling nonlinearities. Initially a stochastic gradient decent was used to solve the SLAM problem and minimize output errors. The solver was later improved by Grisetti et al. (2007) who introduced a tree-based parameterization for graph nodes and obtained a faster convergence. Folkesson and Christense (2007) presented a framework for Graphic SLAM and conducted experiments on a mobile robot equipped with a laser scanner. Instead of iterative solver, Kaess et al. (2008) introduced a new method by employing smoothing. Later a Bayes tree was implemented (Kaess et al., 2012). Another popular method for GraphSLAM optimization is g^2o (Kümmerle et al., 2011), which is also an open source-library

provided on OpenSLAM (Cyrill Stachniss, n.d.). Other proposed libraries include GTSAM (Dellaert, 2012), SLAM++ (Polok et al., 2013), etc. Filtering is another way to solve a SLAM problem; EKF is widely used because of its ability to represent uncertainties. Filtering methods marginalise out past estimates and summarise the information gained over time (Strasdat et al., 2012). Filtering methods only add new features when a new area is explored, thus a map will not increase arbitrarily for movement in a restricted area. For this reason, EKF SLAM is suitable for indoor applications. The observability of EKF was first studied by Huang et al (2008) in a 2D system. Hesch et al. (2014) analysed the obsevability in a vision aid navigation system. The analysis proved to contribute to estimate accuracy, and was carried out over a range of studies on EKF consistency (Bailey, Nieto, et al., 2006a; Castellanos et al., 2007; Sola, 2010).

Although there is large amount of research on SLAM algorithms, SLAM still deserve further investigation, as algorithms can easily fail when environments are challenging (Cadena et al., 2016a).

2.3.2 Visual SLAM Methods

Within robotics predominent, widely used sensors include laser scanners, sonar range finders, infrared sensors, vision sensors and GPS. Currently, range sensor based SLAM problems in a 2D mapping condition is considered solved. At the same time, Visual SLAM is becoming an active area these years. One reason is visual feature points have the advantage of being more informative which also simplifies the data association process (Davison, 2003). Another reason is that vision sensors provide comparatively cheaper solutions and great flexibility in interpreting environments. In contrast, a laser scanner is normally used to locate terrestrial robots, but not feasible when the environment is rough. Laser scanners are also expensive and slow in operation due to low update rate. Computer vision techniques such as feature detection tools allow

for “structure mapping and motion estimation” (Jin et al., 2000). Popular key-point detectors and descriptors include SIFT (Lowe, 2004), SURF (Bay et al., 2008), and ORB (Oriented fast and rotated brief) (Rublee et al., 2011).

Visual SLAM methods include monocular SLAM, dense monocular SLAM, stereo vision based SLAM, RGB-D based SLAM, etc. Monocular SLAM is a challenging area as only one camera is used. To solve monocular SLAM problems, there are two steps to accomplish tracking and mapping: Firstly, discrete feature observations are extracted and matched; secondly, the camera and feature poses are calculated and optimised through a set of observations. Without depth information, the absolute scale of a map cannot be determined, so additional normalization steps are required during optimisation (Strasdat et al., 2010). Instead of operating solely on visual feature positions, dense monocular SLAM methods reconstruct and track on the whole image using surface-based approaches (Engel et al., 2014; von Stumberg et al., 2016), where the world is modelled as a dense surface. As whole-image alignment methods are not real-time capable on standard CPUs, semi-dense methods are used. Stereo vision based SLAM systems do not suffer computation limits, as depth information can be calculated from the disparity between two images (Konolige et al., 2008). Additionally, the stereo configuration has the advantage of being able to accurately calculate the 3D position of landmarks and features, as colour camera calibration techniques are well developed. Some of the calibration tools are: OpenCV libraries for 3D reconstrction (OpenCV, n.d.), Calibration Toolbox for Matlab (Bouguet, 2015), and Tsai Camera Calibration Software (Dias, 2003). Multi-camera rigs were also developed to provide a wide viewpoint and a robust operation (Kaess & Dellaert, 2010; Tribou et al., 2016).

RGB-D cameras are widely used in Visual SLAM systems, as they can provide colour images and depth images simultaneously. In Henry et al.'s 3D mapping method, visual features in combination with shaped-based alignment were used for mapping optimisation. As depth data is normally noisy, a view-based loop-closure detection was implemented (Henry et al., 2012; Segal et al., 2009). Kerl et al. (2013) estimated the camera motion by registering two consecutive RGB-D frames and minimizing the photometric error with a coarse-to-fine scheme. This method used depth measurements directly. To allow noise and outliers, they developed a robust error function that reduces the influence of large residuals. Hu et al. (2012) proposed a SLAM system which uses a map joining algorithm to combine all the local maps. An advantage of their method is the reduction of computation cost. Regarding tests of these Visual SLAM methods, RGB-D camera based 3D reconstruction and mapping techniques become mature in computer vision. Hu et al.'s system was tested on a robotics platform; Henry et al.'s mapping method was tested by a person carrying a camera, and Kerl et al.'s method was tested in video. Thus, it is well regarded that a real-time implementation of SLAM algorithms for practical environments application is a difficult task. The computation cost of a SLAM algorithm is huge, and it is not easy for a robot's CPU to accomplish large amount of computation.

In the recent years, some research has improves algorithm efficiency, some research developing towards Semantic SLAM, such as object-based reasoning, place recognition, etc. Maier et al. (2014) proposed a sub-mapping technique for saving computation cost in bundle adjustment. Newcombe et al. (2015) presented a scenes tracking method in real-time. This method can analyse occluded scenes without a prior modelling. Ma et al. (2016) worked on dense image alignments for accurate tracking. Dharmasiri et al. (2016) proposed a MO-SLAM method for duplicate object detection. Convolution Neural Network (CNN) is adopted into SLAM frame-

work (Kendall et al., 2015), a PoseNet method was developed and achieved successful camera localization accuracy. Most recent work relies on GPU assistance to achieve real-time performance.

2.3.3 Robot Navigation

A navigation process should be able to perform four stages: perception, localization, planning and actuation. The perception stage obtains and processes the information of sensors. Through this information, a robot can recognize objects around it, and further can recognize each object as a target or as an obstacle. Localization is a stage of determining the position and orientation of a robot with respect to its surroundings. Once the robot's position has been determined, the path planning can be conducted. This includes finding a collision free path from a starting point to a desired point. Finally, the actuation stage is responsible for executing the plan. A sequence of actions to reach the desired position is determined in the actuation stage. As the first two stages are usually finished by the SLAM method, this section will discuss path planning and actuation.

In general, the research of navigation can be classified into two major areas: the global path planning and the local motion planning. A global path planning algorithm calculates optimal path to a desired position(Kambhampati & Davis, 1985). In global path planning, the environment surrounding the robot and the position of obstacles are well known, and the robot is required to navigate to its destination by avoiding any obstacles. There are many popular global path planning methods, such as A^* (Russell et al., 1995), D^* (Stentz, 1994) and focused D^* algorithm (Stentz, 1995). Local motion planning methods take into account unknown and changing characteristics of the environment based on the local sensory information, and dynamically guide the robot according to the local information. Therefore, the local motion plan-

ning methods are more suitable and practical for autonomous robots, if the environment is dynamic and too complicated to be known.

The earliest algorithms for robot navigation were developed in a completely known environment, filled with stationary obstacles whose positions were also known. Popular methods are the artificial potential field (APF) approach (Borenstein & Koren, 1989), the distance function (DF) approach (Gilbert & Johnson, 1985), vector field histogram (VFH) and VFH+ techniques (Ulrich & Borenstein, 1998), dynamic window approach (Tolman) (Fox et al., 1997), and rule based (RB) approach (W. Li, 1994).

The APF approach was introduced by Borenstein and Koren (1989), and it represents a good solution to achieve a fast and reactive response to a dynamically changing environment (Borenstein & Koren, 1991). This method acts to fill an observed environment with a potential field in which a robot is attracted to a target position and is repelled away from obstacles. At any position, a robot can calculate another position that has the global minimum repulsive force. The robot moves toward positions with minimum repulsive force and repeats this until it reaches the target. A limitation of the potential field method is the local minima problem: when attractive and repulsive forces on a robot are zero, the robot has to stop moving. As a result, the robot cannot reach the target. Since a robot's motion in a dynamic field has a certain amount of randomness due to the nature of a real circumstance, obstacle methods normally cannot perform under all conditions.

Currently, control strategies occur as navigation processes. Reactive control methods are used in obstacle avoidance systems. Minguez and Montano (2000) presented a number of motion commands, which generate directions for a mobile robot to head in. A limitation of this meth-

od is the inability to process environmental changes. Thus, this method cannot work in a dynamic environment. Boundary-following methods also use a common control structure (Chatterjee & Matsuno, 2001). In this method, an escape criterion is defined based on analysing surroundings. With this constraint, a robot will follow an obstacle boundary until the escape criterion is satisfied. Behaviour integration approaches were also developed. Those approaches are comprised of a map that models a robot is surroundings, a planning module that outputs a correct direction or path, and a reactive module that makes the robot reactively avoiding the collision with any obstacles (Minguez & Montano, 2005; Philippsen & Siegwart, 2003). In the study of M. Wang and Liu (2008) , the local navigation problem was resolved by recording obstacle and trajectory information into a “memory grid” of the environment. Fuzzy systems were adopted, as they have the ability to treat uncertain and imprecise information using linguistic rules, also an advantage of not requiring a precise analytical model of the environment.

Noise is one factor that affects accuracy of a navigation process. When mapping an environment, a mobile robot generally has to cope with different kinds of noise: noise in the odometry and sensor data. G. Dissanayake et al. (2000) proposed to remove a few landmarks to eliminate noise. Their results demonstrated that it is possible to remove a large percentage of landmarks without making a map building process statistically inconsistent. In their experiments, the SLAM method maintained efficiency by selecting of landmarks. Thrun (2001) used raw sensor data and performed a dense matching of scans. Navigational accuracies weredepent on the quality of images and the number and accuracy of points used in the resection. Although all approaches possess the ability to cope with a certain amount of noise in the sensor data, they assumed that the environment is almost static during the mapping process. To

deal with the noise and be robust to imprecision of sensory measurements, a fuzzy controller was adopted to design a behaviour based steering scheme (Jasmine Xavier & Shantha Selvakumari, 2015). Simulation results were used to demonstrate to the efficiency of a robot's activities.

Earlier studies such as APF and VFH require quantitative formulation of the behaviour and additional computational effort in the implementation. These methods can perform well in static environment, but are not suitable for dynamic environments where there are a large amount of uncertainties. Modern behaviour based control strategies are computationally efficient. A fuzzy controller is commonly used to generate velocity and heading angles. Mostly simulation results of the controller are demonstrated, real-time experiments need to be further conducted. Range sensors are mostly used in a navigation process. As visual sensors can supply informative data, a vision based robot controller for navigation is an increasing relevent research field.

2.3.4 SLAM in a Navigation Process

This section illustrates SLAM applications in a robot navigation process. Navigation is an important topic in robotics. SLAM has been implemented in a number of domains, such as indoor and outdoor environments, underwater, and airborne systems. SLAM's capability of perceiving spatial information is a key prerequisite for autonomous navigation. SLAM is able to interpret and synthesize the external sensory information into a representation of the environment that can be used by a mobile robot to operate autonomously. For different kinds of environmental structures, different types of sensor are used. Laser sensors are utilised predominatly in grounded navigation (Nieto et al., 2006), and for water-based navigation, radar and sonar are normally used respectively (Benjamin et al., 2006; Eustice et al., 2005). The main difficulties of

grounded navigation approaches are that they suffer from unbounded drift (e.g., odometry) (Vaganay et al., 2006), or they require an external infrastructure (e.g., acoustic beacon) that needs to be set up and calibrated (Trimble & Belcher, 2002).

In grounded navigation fields, a number of SLAM methods are used in both indoor and outdoor environments. For example, Grisetti et al. (2007) used a particle SLAM approach to map an indoor building and outdoor campus. In their study, dynamic features of the outdoor campus were ignored or filtered and only features known to be static were used for localization. Their office mapping accuracy was up to a resolution of 0.01m. To correct the map of the static environment, the robot ran around the external perimeter in order to close the out loop.

Several approaches have been proposed for mapping in dynamic environments (C.-C. Wang et al., 2003; Wolf & Sukhatme, 2004). In these approaches, a robot can track movements and build a map simultaneously. If the environment has semi-dynamic objects, which may rarely change poses when a robot performs mapping, the robot will find it difficult to detect these changes. Zhou and Sakane (2008) proposed a SLAM-SD method for objects. They attached RFID tags to semi-dynamic objects to distinguish them from static objects. They conducted experiments in an office, where two cabinets were moved occasionally. Biswas et al. (2002) found that changes in the environment were detected using a straightforward mapping differencing technique.

A disadvantage of outdoor mapping is the computation cost. Kim and Eustice (2013) used a visual SLAM solution for autonomous undersea hull inspection. A sonar sensor and a camera were used in their study, and an iSAM framework was adopted. The whole SLAM calculation was achieved on a process server, due to the size of observing area and the computation

amounts. Consequently, computational demand is dependent on the size of the environments. Visual SLAM also assists Micro Aerial Vehicles for path planning. Sadat et al. (2014) utilised MonoSLAM for path planning and obstacle avoidance. The experiments were carried out in an indoor environment. In flight navigation, Dalen et al. (2016) used a particle filter based framework for absolute position estimation; Magree and Johnson (2015) use a EKF based framework for stability improvements.

In indoor SLAM, the scanning of large planar regions is problematic due to a lack of depth variation, such as wall surfaces in a room. To solve the problem, Trevor et al. (2012) used a range camera as a sensor, and adopted colour feature descriptors for data alignment. Range cameras are suitable sensors for indoor SLAM, as they can provide detailed information up close at a high frame rate and allow users to map planes in any orientation, including horizontal surfaces such as tables or desks. RGB-D sensors were originally designed for gaming and user interaction rather than mapping, so they are more suitable for mapping smaller scale areas with many features. Wieser et al. (2014) used optical sensors for SLAM algorithm, and collision avoidance was achieved. Creating more efficient paths and preventing redundancy in visited locations should be improved.

SLAM is also applied in surgical environments (Grasa et al., 2014; Grasa et al., 2011). Grasa et al. (2014) integrated a monocular SLAM algorithm into ventral hernia repair surgeries. Feature detection methods were used to measure datasets in medical endoscopic sequences, and SLAM results were validated with in vivo human medical sequences. Their typical map sizes were between 50 and 100 points and up to 40 map features were measured per frame, so their SLAM method can be applied in real-time. However, there were also failures due to the lack of stable texture in the defect boundary area and the particular point detection method

they used. Thus, real-time Visual SLAM can be used in a small size area, and mapping accuracy can be improved if there are enough features in this area and those features can be detected.

Thus, for indoor SLAM, lasers and cameras are suitable sensors. Depth information is necessary in indoor navigation. Laser sensors are less useful in highly cluttered environments and they are expensive and heavy. RGB-D cameras are more suitable for small areas, such as indoor environments. The computational requirements of small areas are relatively low. Most existing robotic mapping algorithms possess one important limitation - they all assume that the world is static. However, most natural environments are not stationary. Mapping in semi-dynamic environments is a new trend.

To effectively navigate in an environment and avoid repeatedly visiting a location, adequate robotic time management is important. There is limited research about how to balance the time between navigation and mapping environment. Few studies related to time management skills have been reviewed.

In the field of robotic mapping, people presented methods of adjusting observation time. Krajnil et al.(2014) introduces a spectrum (spectrum, i.e. it is a condition that is not limited to a specific set of values but can vary infinitely within a continuum) method to observe a dynamic environment. They gathered datasets every five minutes, and predicted the changes in the environment. As their study was based on the events which had a daily routine, the spectral method can be used to predict the periodical changing of the environments.

Biber and Ducket (2005) introduced a dynamic map for mobile robots that adapts continuously over time. Their map represented the environment over multiple timescales simultaneously, and older memories faded at different rates depending on the timescale. Their solution to dy-

namic environment mapping was effective, but the method was towards long-term mapping. Their global map was updated after one day observation.

Yang et al. (2005) proposed an optimisation controller for robot navigation. This controller utilizes a hybrid neuro-fuzzy method where a neural network effectively chooses the optimum number of activation rules for real-time applications. Initially, a classical fuzzy logic controller has been constructed for the path planning problem. The neural network is implemented to choose the optimum number of the activation rules based on the input crisp values. Simulation experiments were conducted to test the performance of the developed controller and the results proved that the approach is practical for real time applications.

2.4 Image Processing

Image processing is a form of signal processing, where the input image is treated as a two-dimensional signal. The output of the process is usually an image or a set of characteristics.

Image processing usually refers to digital image processing; however this can also refer to analogue image processing. Digital occurs through the use of computer algorithms to conduct image processing on digital images, while analogue image processing occurs as the processing of a two-dimensional analogue signal. Compares to analogue processing, the digital method allows a much wider range of algorithms to be applied to the input data and can avoid problems such as the build-up of noise and signal distortion during processing.

With the fast development of computers processing and signal processing in the 2000s, digital image processing has become the most common form of image processing and generally, allows the use of much more complex algorithms, and hence, can offer both more sophisticated

performances at simple tasks. As a practical technology, it is widely used in classification, feature extraction, pattern recognition, projection and multi-scale signal analysis.

2.4.1 Feature Detection

Feature detection in image processing refers to methods that aim at computing abstractions of image information and making local decisions at every image point whether there is an image feature or not (Wah, 2007). Features are usually taken as intermediate results to make decisions about the content in a given image (Zhang et al., 2008). There are a large number of features which have been developed, and those features are in the form of isolated points, continuous curves or connected regions (Tuytelaars & Mikolajczyk, 2008). How to select a feature to use depends on the type of applications (Kadir & Brady, 2001). Mainly four kinds of features are commonly used: edges, interest points, regions of interest and ridges.

(A) Edge

An edge is the boundary between two areas of different colours in an image. Edges can be detected by looking for the points within an image with higher gradient with respect to neighbouring pixels (Kaur & Malhotra). An edge feature can be used to find low-level symbolic representations; it is usually the boundary, such as lines and contours, and can be used to detect shapes. Edges are used to find boundaries of objects' surfaces, the contrast between different colour areas, shadow and texture (Senthilkumaran & Rajesh, 2009).

One of the most commonly used edge detector is Canny. Canny is good at finding optimal edges, and make these similar to those in the real image (Shin et al., 1998). But it is sensitive to the edges of noise as well. Sobel is also a popular edge detector, it can avoid noise better, because it is based on convolving the image with a small, separable, and integer valued filter in

horizontal and vertical. The result gradient approximation in Sobel is relatively crude, in particular for high frequency variations in an image, so that this operator can avoid some low frequency noise to some extent (Ziou & Tabbone, 1998).

The stages of Canny algorithm are showed in the following (Deriche, 1987):

I. **Smoothing:** To avoid the noise present in the raw unprocessed image data, a blur filter is used first. Gaussian filters have the properties of having no overshoot to a step function input while minimizing the rise and fall time, and these properties make them popular as filters in image processing. Mathematically, a Gaussian filter works by convolving the input image with a Gaussian filter, as equation (3-4), where 273 is the sum of the numbers in mask.

$$\frac{1}{273}\begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{bmatrix} \tag{3-4}$$

II. **Finding Gradients:** Edges should be marked where the gradients of the image has large magnitudes. The gradients in the x and y direction are obtained by applying the kernels shown in Equation(3-5), which is also known as Sobel-operator (Al-Amri et al., 2010).

$$K_{GX} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$K_{GY} = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{3-5}$$

The largest gradient can be determined as a Euclidean distance measured as shown in equation (3-6), and the direction of the edges can be determined as equation (3-7)

$$|G| = \sqrt{G_x^2 + G_y^2} \tag{3-6}$$

$$\theta = acrtan\left(\frac{|G_y|}{|G_x|}\right) \tag{3-7}$$

III. **Non-maximum Suppression**: this method preserves all local maxima in the gradient image, and deletes everything else (Neubeck & Van Gool, 2006). The algorithm for each pixel in the gradient image acts in the following stages:

Stage a: Round the gradient direction θ to nearest 45◦, corresponding to the use of 8-connected neighbourhoods.

Stage b: Compare the edge strength of the current pixel with the edge strength of the pixel in the positive and negative gradient direction, compare with the pixels to the north and south.

Stage c: If the edge strength of the current pixel is largest; preserve the value of the edge strength. If not, suppress the value.

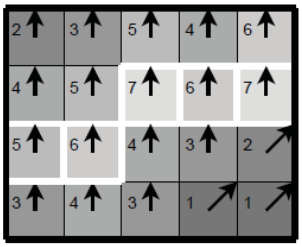


Figure 2- 7: Illustrate of Non-Maximum suppression

A simple example of non-maximum suppression is shown in Figure 2- 7. Almost all pixels have gradient directions pointing north. They are therefore compared with the pixels above and below. The pixels that turn out to be maximal in this comparison are marked with white borders. All other pixels will be suppressed.

IV. **Tracing Edges through the Image and Hysteresis Thresh-holding**: Thresh-holding with hysteresis requires two thresholds – high and low. Edge pixels stronger than the high threshold are marked as strong, important edges should be along continuous curves in the image. Starting from these, using the directional information derived earlier, edges can be traced through the image. While tracing an edge, a lower threshold is used, so edge pixels which are weaker than the low threshold are suppressed. Those occurring between the two thresholds are marked as weak. Once this process is complete a binary image where each pixel is marked as either an edge pixel or a non-edge pixel (Lopez-Molina et al., 2013).

(B) Interest Points and Interest Region

The term interest points refer to point-like features in an image, which have a well-defined mathematical structure; those features are quite stable even if there are perturbations like illumination variations and scale changes. Compared to interest point's detection, interest regions are more point-like, and they can detect areas in an image which are too smooth to be detected by interest points. Interest points and region methods can be divided into three categories: contour based, intensity based and parametric model based methods. Contour based methods usually search for maximal curvature or inflexion points along the extracted contour chains, or do some polygonal approximation and look for intersection points. Intensity based

methods compute a measure that indicates the presence of an interest point directly from grey values. Parametric model methods fit a parametric intensity model to signals. They often provide sub-pixel accuracy, but are limited to specific types of interest points (Schmid et al., 2000).

Interest points were first developed based on the analysis of basic pixel correlation. Moravec (1977) measured grey value differences and the correlation between windows in several directions, and threshold the minimum of direction differences to detect interest points. Schmid et al. (2000) used the second derivatives of the signal to compute the measured Hessian matrix of the image, and points where this measure was maximal were detected interest points.

Later interest points are extracted based on edges, corners and regions. Asada and Brady (1986) extracted interest points according to the changes in curvature from planar curves, and they categorized them into junctions, endings etc. They also integrated their algorithm into a framework to make the extracting method work robust. Medioni and Yasumoto (1986) used B-splines to approximate the contours, and extracted the interest points with maxima curvature. Horaud et al. (1990) grouped line segments from Image contours and took their intersections as interest points. Cooper et al. (1991) measured contour directions locally and computed their differences, then used noise characteristics to determine whether the differences along the contour directions are sufficient to indicate an interest point. Pikaz and Dinstein (1994) detected interest points based on a decomposition of noisy digital curves into a minimal number of convex and concave sections. Based on the correlation matrix, Förstner (1994) detected interest points by classifying image pixels into region, contour and interest point categories. By analysis on the location of local gradient field, Interest points can be classified into junctions or circular features. By looking at the magnitude and the direction of the derivatives of neigh-

bouring points, Reisfeld et al. (1995) computed a symmetry map, and selected points with high symmetry as interest points. The algorithms of Mokhtarian and Suomela (1998) are based on two sets of interest points. One set are T-junctions, and the second set obtained through tracking curvature maxima of contour from a coarse level locally up to the finest level. The approach proposed by Parida et al. (1998) is based on a variant of the morphological closing operator which successively applies dilation/erosion with different structuring elements. And this method can find vertical/horizontal corners.

Parameter based interest points detectors were developed in the next stage. Baker et al. (1998) proposed an algorithm that automatically constructs a detector for an arbitrary parametric feature. Each feature is represented as a densely sampled parametric manifold in a low dimensional subspace. SIFT is a popular method for interest points detection; it detects using a Gaussian scale pyramid, and makes the computation process efficient (Lowe, 2004).

(C) Ridge

A ridge descriptor is usually used for elongated objects, and it can be thought of as a one-dimensional curve that represents an axis of symmetry. But it is algorithmically harder to extract than the edge and interest points features. Ridge descriptors are usually used for road extraction in aerial images and for extracting blood vessels in medical images. Schmid et al. (2000) first tried ridges detection. They pointed out interesting point as high curvature points along ridges or troughs, and intersection points. They also believe such points are more appropriate for tracking as they are less likely to lie on the occluding contours of an object.

2.4.2 Object Detection

Object detection is a key task in fields of computer vision and robotics. Some object detection methods are well developed, such as feature analysis (Comaniciu & Meer, 2002), segmentation (Sumengen et al., 2003), blob clustering (Ana & Jain, 2003), object recognition (Lowe, 2004), and illumination invariance (Finlayson et al., 2001). Some well-developed image processing tools are integrated in OpenCV, Matlab etc. as libraries for usage. These libraries brought a lot of advantages to robotics. For example, some libraries in OpenCV include: template matching to find matched objects in an image, Haar-classifier which is a strong learner for Haar-like features.

Context based image processing enhances image analysis technologies by incorporating contextual information. Heitz and Koller (2008) grouped regions based on appearance and co-occurrence relationships to the detected objects and introduced a “things and stuff” context model. This model is capable of producing interpretable clusters. Yao and Li (2010) proposed a random field model to encode the mutual context of objects and human poses for human-object interaction activity detection. Depth features from RGB-D cameras are adopted in object detection. Gupta et al. (2014) proposed a geocentric embedding method for feature representation. Instead of single object detection, Gall and Lempitsky (2013) introduced an object class identification method by applying a Hough forest based classifier. Dollár et al. (2014) proposed a fast feature pyramids for object detection. They created multi-scale gradient histograms using gradients computed at a single scale and used their scheme to compute finely sampled feature pyramids.

Deep Neural Networks (DNNs) performed efficiency in object classification (Krizhevsky et al., 2012),and later adopted in object detection. Szegedy et al. (2013) use DNNs detect a class of

objects and also locate the positions of the objects. Redmon et al. (2016) took object detection as a regression problem and classified objects spatially using separated bounding boxes and associated class probabilities. Instead using DNNs, Zagoruyko et al. (2016) proposed a multi-path network for object detection. They modified the network to exploit objects at multiple solutions and accordingly adjusted the loss function.

In robotics, Sridharan and Stone (2007) proposed a colour learning method, where a set of images were labelled and a robot learned the range of pixel values that map to each colour. Their results showed the robot can plan its motion autonomously; however, the pixel-colour mapping was dependent on the original training image set and should have the prior knowledge made available of the scene. Browning and Govindaraju (2003) presented two algorithms, Histogram Threshold Vector Projection and Histogram Threshold Ellipsoid Distance, for fast colour-based detection of objects under variable illumination. Their algorithms are for a soccer ball detection, which requires lower computation.

Human motion recognition is a popular area in object detection. It uses the prior knowledge of a human body to establish the human body's model structure, and then extracts underlying features in the image to match the body's model. Normally, model-based methods can obtain a more accurate and more complete feature data (Wei & Yunxiao, 2009). Feng and Perona (2002) made a two-dimensional model, which generally extracts apparent characteristics directly from the bottom of an image, separates the face, trunk, limbs, etc. and estimates two-dimensional human body model parameters. Background segmentation is a commonly used method to extract dynamic motion features. The rationale in this approach is detecting a moving object from differences between the current frame and a reference frame (Piccardi, 2004). This method is mostly used in a video. Tamersoy (2009) proposed a robust background sub-

traction algorithm to handle lighting changes, repetitive motions from clutter and long-term scene changes. The Hidden Markov model (HMM) was first proposed to describe forward-backward procedure (Mitkov, 2005). HMM is especially known for its robust application in temporal pattern recognition, as opposed to template matching which is sensitive to noise and the variations in the movement duration (Starner, 1995). Optical flow is commonly used to extract time-space domain information, as adjacent frames contain the process of movement. It was introduced by James J in 1940s to describe visual stimulus. It can be used to calculate and estimate motion velocities or image displacements (Beauchemin & Barron, 1995), and can detect motions without exacting features. Little and Boyd (1998) extracted frequency and phase features from moments and recognized different people by their gaits. Silhouette is another popular feature to capture the motion, and it is represented as a solid shape of a single colour with its edges matching the outline of the subject. This matching is reasonably independent of the clothing worn by people and it also supports night vision capability (Kale et al., 2002).

2.4.3 Machine Vision Techniques

Machine vision (MV) is concerned with achieving visual perception electronically. MV techniques provide image-based automatic inspection, process control, and robot guidance in industry (Graves & Batchelor, 2003). MV plays an important part in many fields, which include fruit detection, traffic surveillance, products' quality test, process control methods in modern control theory, robotic guidance, etc. MV is also used to detect events, such as visual surveillance or people counting; and in modelling objects or environments, such as industrial inspection, medical image analysis and topographical modelling.

One of the important features of MV in manufacturing is providing greater flexibility to a manufacturing process and accomplishing tasks more efficiently. Li and Zhu (2011) used vision sensors to look for the LED chip electrode in a LED chip automatic measurement system. The image matching method was used in their MV system, and the efficiency of their productivity was improved. In the field of Multi-axis machine process, it is important to make the CAD-CAM/CNC multi-axis safe trajectory generation process optimal. Rafiq et al. (2013) integrated vision based image processing to make the manufacturing process intelligent and automatic. Sitthi-Amorn et al. (2015) integrated MV into a multi-material 3D printing platform. The integrated MV simplified the overall platform design and enabled new applications such as 3D printing over auxiliary parts. Sun et al. (2016) presented an intelligent system that incorporates MV with artificial intelligent to inspect thermal fuses. The system was operating in case study, the cost of human visual inspection was reduced.

MV is also developed as an objective inspection tool in food control field, Dowlati et al. (2012) integrated machine and imaging technologies in fish-quality assessment, they used the image sensors to measure the size and volume. With the obtained parameters, they used shape and skin colour features to recognize fish species. They concluded machine vision technology provided a useful tool for automated, objective, rapid and hygienic detection. Unay and Gosselin (2007) proposed a novel application for grading apples. They extracted features from apple's images, and used these features to exercise a classifier to distinguish multi-category apples. Their results manifested the feature selection based method outperform its syntactical counterparts. Matured mangoes were also sorted by MV techniques (Nandi et al., 2014). With a SVM classifier, the average performance of the proposed MV system was found to be better than the human experts. T. Berge et al. (2012) used MV to do weed detection. With the ability

to detect specific weed species, farmers can apply herbicide directly to cereals. Celik et al. Çelik et al. (2014) used MV for fabric inspection and defect classification. Wavelet transform and morphological operation etc. image processing method were used; five types of defects were detected and classified.

Depth information is a trend of vision application. The conventional way of obtain depth information is stereo vision, which is a process of solving correspondence between a pair of images. However, this process usually requires a substantial amount of computation. Another way to acquire depth information is through range cameras. Comparisons of two kinds of methods are shown in Table 2- 2. Donoser and Bischof (2008) used video for hand tracking test. Without depth sensor, a statistical model was presented for skin colour to give clues about possible hand segments. Lighting variance is one disadvantage of colour threshold. In Hackenberg et al. (2011)'s study, a Time of Flight (ToF) camera was employed for reliable hand gesture recognizion. Elmezain et al. (2010) combined skin colour and 3D depth images to segment hand regions; Manders et al. (2008) computed hand probability map for a Camshift tracker from a joint probability function derived from both depth and skin colour information. The depth information can not only improve the hand localization, but also enable the estimation of 3D hand positions instead of 2D positions. Nanda and Fujimura (2004) extracted depth information and achieved an ellipse fitting method for head tracking.

Table 2- 2: Stereo camera method to derive depth information

Method	Input sensor	Resolution	FPS
Donoser and Bischof, 2008	RGB Video	640*480	25
Hackenberg, 2011	ToF(SR4000)	176*144	50
Manders, et al. , 2008	Stereo + RGB	320*240	30
Bergh and Gool 2011	ToF + RGB	174*144	30
Nanda and Fujimura, 2004	ToF	320*240	14
S.-W. Sun et al. 2013	Kinect V1	320*240	30

Chia-ping Chen, 2011	Kinect V1	320*240	30

With the advantages of low cost and computation efficiency, RGB-D cameras are more suitable than normal 2D cameras and laser sensors for a robotic application.

2.5 Conclusion and Discussion

This chapter reviewed techniques and previous work that related to VEX robotics, RGB-D camera, Visual SLAM, robot navigation, and image processing.

Firstly, VEX Robot and VEX Competition were introduced, problems related to complete a VEX competition were discussed. As the objects on the Vex field change every year, object detection techniques are important for a VEX robot. To win an autonomous period, environmental recognition and path planning abilities are necessary. With the knowledge of VEX, a case study in Chapter 6 is conducted on the VEX field.

Secondly, RGB-D cameras' development, applications, and their usages in robot navigation and Visual SLAM were discussed. RGB-D cameras have been developed and brought convenience to robotic applications. RGB-D cameras provide depth information directly, so computation of depth is saved. However, the depth data may not be evenly distributed, or the depth is not as accurate as the data computed from stereo vision. As such a study of RGB-D cameras properties should be carried out before using it.

Thirdly, developments of SLAM methods, Visual SLAM methods and robot navigation were introduced, how Visual SLAM methods work in a robot's navigation process was discussed. A fundamental and critical research area in mobile robotics is navigation. How to plan a path and

traverse to the desired location, without prior complete knowledge of the environment, is an important task. SLAM provides tools to recognize the environment and localize a robot's and objects' positions. Currently, SLAM problems concerning indoor mapping with 2D range sensors in a static environment are considered solved. Current trends of SLAM algorithms are developing towards efficient mapping in large areas, dynamic environments, loop-closure and convergence analysis, Semantic SLAM, etc. SLAM methods are developing across fields including image processing, signal control, etc.

Most SLAM methods are tested with a handle-hold camera or compared with benchmarks in a video for accuracy analysis. A practical SLAM solution for a mobile robot on a critical level of effectiveness and robustness is still lacking. Most robot applications are based on human-constructed maps or external infrastructures. This study integrates EKF SLAM in to a robot's navigation process. As discussed, EKF has the ability to represent uncertainties. This thesis uses an EKF SLAM method for navigation.

Finally, the feature and object detection methods were introduced and discussed. Simple features, such as key-point, edge are well developed. Object and motion detection methods are incorporated with depth information. In the following work, feature detection techniques were integrated into the SLAM method. A Visual SLAM method for a mobile robot real-time navigation and object detection (SLAM-O) is proposed and developed, in order to establish the position of the mobile robot and interest objects in an unknown indoor environment. Thus, this study targets the improvement of the SLAM algorithm to provide more robust and consistent estimations for a robot's trajectory in real-time applications.

Chapter 3 RGB-D Cameras Comparison

In this chapter, the performance of frequently used cameras, e.g. PrimeSense, Kinect V1 and Kinect V2 is compared. Comparisons include colour image quality, depth sensor accuracy and camera sensor performance efficiency. Using OpenNI and Microsoft SDK, depth measurements and colour image detection results for different cameras is evaluated. A Point Cloud Library (PCL)-based, perpendicular distance calculation method is also discussed in this chapter. Based on the comparison's results, a relationship between measurements and ground truth is built.

3.1 Introduction

RGB-D cameras were initially designed for home entertainment and gaming (Microsoft, n..d.). These cameras have the advantage of being light weight, reliable and operate higher speed of measurement (normally 30 fps). This 3D sensing technology offers convenience for indoor robotics (Basso et al., 2013), 3D scene reconstruction (Izadi et al., 2011), and object detection (Herbst et al., 2011), etc.

Before determining which RGB-D camera to use, an analysis of measurement accuracy is necessary. These measurements include RGB-D data, depth data, and the mapping from a RGB image to its corresponding depth clouds. A number of camera calibration methods have been developed as open-source libraries, but most of the provided libraries are only suitable for specific types of cameras or operation environments. For example, OpenCV provides calibration libraries for radial distortion in a RGB image and depth alignment in a depth point cloud (OpenCV, 2017). However, depth calibration libraries are based on OpenNI, which is not com-

patible with the Kinect V2. As a mature technique, RGB image calibration libraries are also provided in Matlab (MathWorks, n.d.). Studies of depth clouds and RGB pixels mapping to depth clouds have being developed by a number of researchers. For example, Raposo (n.d.) developed software EasyKinCal for Kinect calibration in Matlab and Herrera et al. (2012) developed a Kinect Calibration toolbox. These two methods need the use to collect RGB and disparity (inverse of depth) images and saving these images as inputs of the software. This software is limited by the type of disparity image, portable graymap format (pgm), which can only be obtained from a Linux system. As such, Wiedemeyer (2016) developed a calibration method for both Kienct v1 and Kinect V2. This method operates in Robot Operating System (ROS) and has been studied and used by a number of researchers (Fankhauser et al., 2015; M. Li et al., 2016; Wu & Wetzstein). This method requires Kinect images to meet certain conditions, because factors, such as light, shadows, the point of view, etc. all affect the measurements. The obtained radial distortion coefficients from calibration results are in a range (-1, 1), these coefficients could be easily affected by these environmental factors. As a result, calibration procedures need to be followed strictly to ensure measurement improvement.

The followings studies have been concluded on how to improve depth measurements and their noise properties. Smisek et al. (2013) proposed a geometrical model for Kinect V1 and analysed measurements properties. They developed a calibration method based on the analysis and tested the depth correction on a 3D reconstruction scene. Nguyen et al. (2012) derived a noise model for Kinect V1 and analysed the data in both lateral and axial directions. The proposed noise model improved the Kinectfusion method. A similar noise analysis approach performed on Kinect V2 was conducted by Fankhauser et al. (2015), but how to improve the Kinect V2 depth measurements was not mentioned. From the analysis of noise data, the point of

view affects depth measurements. Yang et al. (2015) used a multiple camera measurement method to improve the depth accuracy of the Kinect V2 camera. It is concluded that, depth noise varies with the view of point, and the noise grows with the distance away from the camera. To compare depth measurement ability, this study uses point to plane distance as a criterion. Another conclusion is the limited study of the efficiency of mapping colour pixels to their depth clouds. As depth points vary with the view of point from the object, whether these pixels are being correctly and efficiently mapped to their corresponding depth points should be considered. As such, this study conducts an analysis on RGB images quality of RGB-D cameras.

This chapter presents a comparison of three different and frequently used sensors, namely, the PrimeSense, Kinect V1 and Kinect V2. The objective of this chapter is to provide an insight into measurement data and an analysis of the accuracy and density of RGB and depth points. Hardware and software for these cameras is discussed. An experimental method and a mathematical model are proposed to obtain the perpendicular distance from an observing surface to the camera, finally, fitting functions are obtained for depth error correction.

3.2 RGB-D Camera Development Framework

This section discusses how to implement RGB-D sensors in Windows and how to access data from RGB-D cameras using the libraries from OpenNI, Kinect SDKs and OpenCV.

3.2.1 Hardware

PrimeSense cameras require low power (maximal power consumption is 2.35 Watt), this can be supplied by a USB cable. The sensor has a field of view 57.5^{o} x 45^{o}, and the depth map and colour images occurs as arrays of 640 x 480 pixels. This sensor allows for implementation of

OpenNI process. This Chapter will utilise the camera illustrated in Figure 3- 1 conjunction with the Carmine 1.08 model. This model has a range limit from 0.8m to 3.5m with a depth output resolution of 0.012m in a format of 16 bits (PrimeSense, 2011).

The Kinect V1 is based on a sensor design developed by PrimeSense Ltd. Both PrimeSense and Kinect V1 work on Window 7 (or above). The default RGB stream uses an 8-bit VGA resolution of 640 x 480 pixels with a ranging limit of 0.4-4.5m and angular field of view of 57^{o} x 43^{o}. A USB 2.0/3.0 is used to transfer data, and requires an extra adapter to work properly (Jana, 2012). This camera is illustrated in Figure 3- 2.

A Kinect V2 sensor includes an Xbox Kinect sensor and a Kinect Adapter for Windows as illustrated in Figure 3- 3 and Figure 3- 4 respectively. This is connected to a PC for data transfer and a 220 volts power supply is used. The Kinect V2 sensor features a 512 x 424 pixel depth, and records video of areas of 1080p resolution, with an operation range of 0.5-4.5m. In addition to allowing the sensor to see in the dark, the new IR capabilities produce a lighting-independent view—and allow for IR and colour to be utilised simultaneously (Microsoft).

The Kinect V2 requires the operation system to be Windows 8 (or above) and needs the connectivity of a USB 3.0 interface. A 64-bit, Dual-core 3.2 GHz and faster processor is required(Microsoft, n.d.). The hardware requirements and main features of each camera are listed in Table 3- 1.

Figure 3- 1: A PrimeSense sensor

Figure 3- 2: A Kinect V1 sensor

Figure 3- 3: An Xbox Kinect sensor

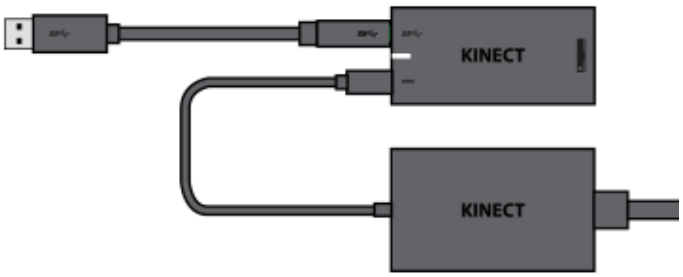


Figure 3- 4: A Kinect adapter for Windows

Table 3- 1: Hardware features for RGB-D cameras

Features	PrimeSense	Kinect V1	Kinect V2
Power supply	USB	Extra adapter	Extra adapter
USB interface	USB 2.0/3.0	USB 2.0/3.0	USB 3.0
Operating system	Windows 7 above;32/64-bit	Windows 7 above; 32/64-bit	Windows 8 above ; 64-bit
Field of view	57.5° H, 45°V	57° H, 43°V	70° H, 60°V
RGB resolution	640 x 480	640 x 480	1920 x 1080
Depth resolution	640 x 480	320 x 240	512 x 424
Depth distance	0.8-3.5m	0.4-4.5m	0.5-4.5m

Table 3- 2: Software compatibility for each RGB-D camera

Software Libraries	PrimeSense	Kinect V1	Kinect V2
OpenNI	√	√	
NITE	√	√	
Kinect SDK		√	√
OpenCV	√	√	√

Table 3- 3: Software libraries used for each RGB-D camera

Cameras	Software Libiraies
PrimeSense	Visual Studio 2013 + OpenNI 2.2 + NITE 2.2 + OpenCV 2.4.12
Kinect V1	Visual Studio 2013+ OpenNI 2.2 + NITE 2.2 + OpenCV 2.4.12
Kinect V2	Visual Studio 2013 + Kinect SDK 2.0

3.2.2 Software

This section presents software that can be used to build applications for RGB-D sensors.

1) OpenNI

OpenNI libraries provide a framework that can convert raw sensor data from a compliant device to application-ready data. Both the PrimeSense sensor and Kinect V1 are OpenNI-compliant devices. OpenNI libraries are written in C/C++, and they can be used across different platforms, such as Ubuntu systems. OpenNI2 libraries are developed with new classes and functions, and are adopted in this study.

The PrimeSense NiTE™ is a computer vision middleware for 3D data processing (OpenNI). It works with OpenNI for natural interface. The newest version is NITE 2.2, which is suitable for OpenNI2.2.

2) Kinect for Windows SDK

Microsoft Research announced the non-commercial Kinect for Windows SDK v1.8 and v2.0 for Kinect V1 and V2 respectively. These tool kits work with a GUI interface and provide examples for colour and depth processing. These also provided libraries which can work in Visual Studio for further developments. A few commonly used functions are illustrated below:

MultiFrameSourseReader is used to access both the depth and colour frame at the same time. *CoordinateMapper* will be used for RGB and depth images registration. *MapDepthFrameToColorSpace* method can project the depth data into their corresponding colour pixel.

3) OpenCV

OpenCV is a library for image processing. It is designed for computational efficiency with a strong focus on real-time applications. OpenCV is used in this section for interesting point detection and other image processing utilities.

Table 3- 2 illustrates a summary of software compatibility for each RGB-D camera.

3.2.3 Hardware and Software Implementation

Integration of the hardware and software is discussed in this section.

By adding OpenNI and NITE libraries to Visual Studio, the PrimeSense sensor can transfer data to Windows. OpenNI2, NITE2.2 and Visual Studio 2013 are used in this chapter.

The Kinect SDK v1.8 is designed for Kinect V1 and enables developers to create applications. If the Kinect SDK's library is added to Visual Studio, application from this IDE can process data from the Kinect V1. OpenNI and NITE libraries are also compatible with Kinect V1. OpenNI2 and NITE2.2 are used to access the Kinect V1 in this chapter.

Kinect V2 only works with Kinect SDK library. Developers can use Kinect SDK v2.0 for programming, or use Visual Studio by adding Kinect SDK v2.0 libraries to the current workspace. In this chapter, Kinect SDK v2.0 and Visual Studio 2013 are used.

A summary of software libraries used for each RGB-D camera is illustrated in Table 3- 3.

3.3 Experimental Approach

A number of experiments are described for sensors, PrimeSense, Kinect V1 and Kinect V2's performance comparison, in terms of colour image quality, depth points' accuracy and depth clouds distribution.

3.3.1 Experiment Design

A scheme of the experiment design is showed in Figure 3- 5. The camera is placed at the base-line. A PC is placed behind the camera for data processing. The centre of the camera is placed directly to the centre of a 7 x 9 chessboard, which is stuck on a whiteboard. The whiteboard moves from 0.5m to 3.5m away from the camera at 0.5m each time.

3.3.2 Experiment Setup

Real world experiments were performed in a closed and clear hallway with electrical lights, where sun light did not interrupt. Distance from the camera and whiteboard was measured by a grid-map. The grid-map had a canvas of 0.5 x 4m and each grid was printed as 0.1m width and 0.1m length. These experiments were performed from 4 pm to 6 pm. An image of the experiment setup is Figure 3- 6, and apparatus of this experiment are listed in Table 3- 4.

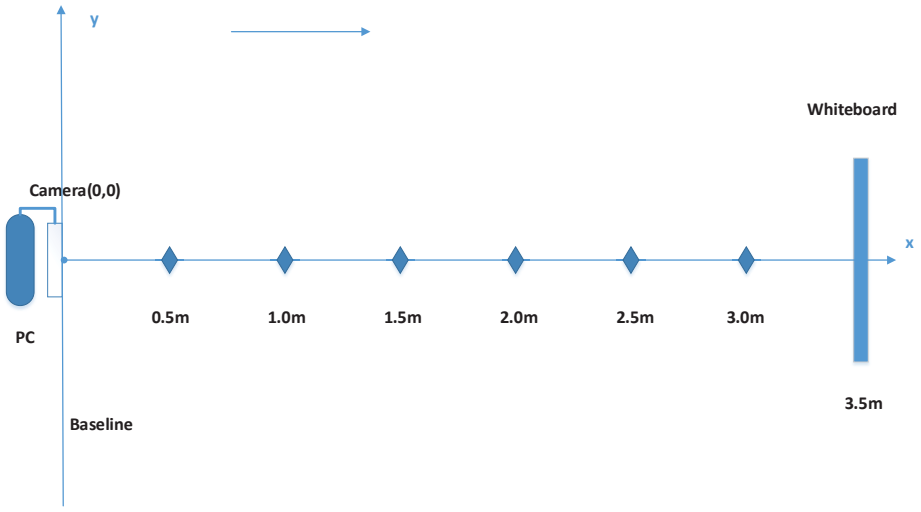


Figure 3- 5: Experiment design

Figure 3- 6: Experiment setup

Table 3- 4: Apparatus in the experiment setup

RGB-D camera	Kinect for Winows v2 sensor
Whiteboard	2m height x 1.84m width
Grid-map	a canvas of 0.5 x 4m printed on an ISO A1 wide page
Reference image	An 7 x 9 chessboard printed on an A2 paper
PC	a TOSHIBA laptop, model Tecra R950
Power supply	220 v power supply and transformer for the corresponding camera

3.4 Results

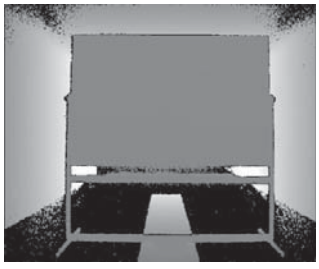
3.4.1 Depth Quality Comparison

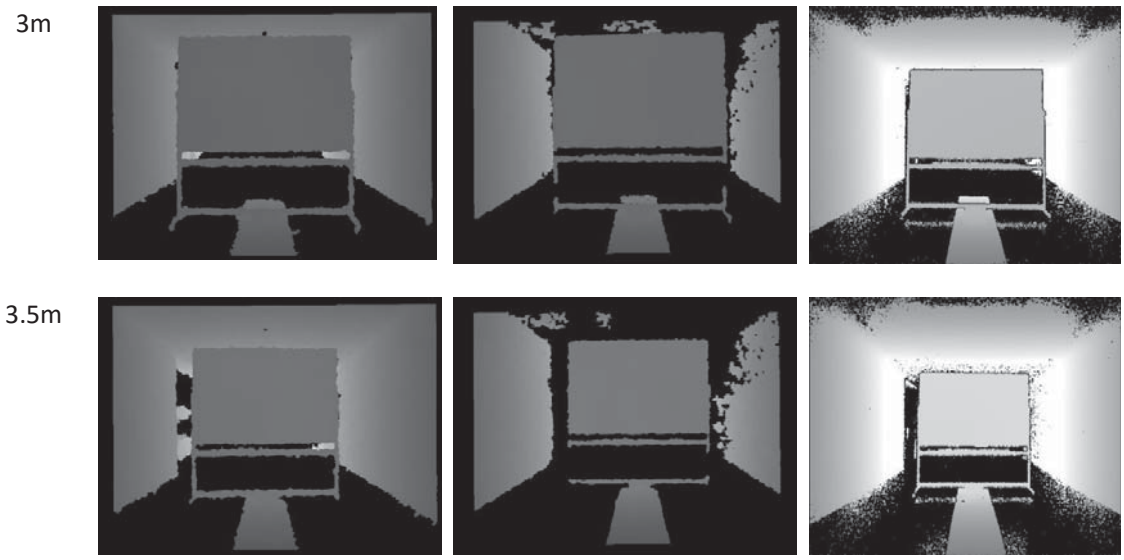
As the whiteboard moved from 0.5m to 3.5m away from the camera, depth data was captured and shown in Table 3- 5. This section compares the obtained depth data in terms of density of distribution and field of view.

Depth loss is an important parameter for a depth camera, because a depth frame is considered invalid and discarded if there is a certain percentage of depth points missing. Table 3- 5 shows that both the PrimeSense and Kinect V2 received a higher quality of depth data than the Kinect V1. In the depth images of Kinect V1, there is depth lost on the left and right edges of the white board, and the corner of the wall behind the board. This is because of the different depth image resolutions (PrimeSense = 640 x 480, Kinect V1 = 320 x 240, and Kinect V2 = 512 x 424) and that the Kinect V1 has the lowest depth resolution and this may cause depth data lost.

In terms of field of view, the Kinect V2 has a larger view angle than the Primesense and Kinect V1 camera. The field of view for the PrimeSense is 57.5^{o} for horizontal and 45^{o} for vertical, for the Kinect V1 is 57^{o} for horizontal and 43^{o} for vertical, while the Kinect V2 is 70^{o} for horizontal and 60^{o} for vertical. Further, the Kinect V1 has a loss in depth image when registrating with RGB image, because of its low resolution. The Kinect V2 data has a higher accuracy and resolution, as its depth images demonstrate more details of the scene than the Kinect V1 and the PrimeSense.

Table 3- 5: Depth distribution comparison

	PrimeSense	Kinect V1	Kinect V2
0.5m			
1m			
1.5m			
2m			
2.5m			



3.4.2 RGB Images Quality and Chessboard Corners Detection Comparison

In this section, colour image quality is compared with the corners of a chessboard used to test each camera's ability. The depth points of the corners on a chessboard are used to analyse the accuracy and efficiency of mapping colour pixels to their depth clouds.

The experiments were carried out in the similar environment utilised in previous experiments. As the whiteboard moved from 0.5m to 3.5m away from the camera, corner detection was performed and repeated for 50 times. During this process, images were captured, and corresponding depth point for each corner was recorded.

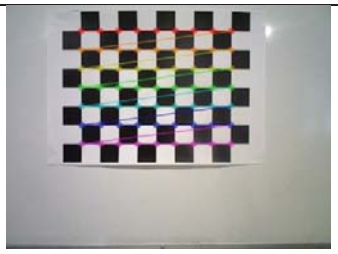
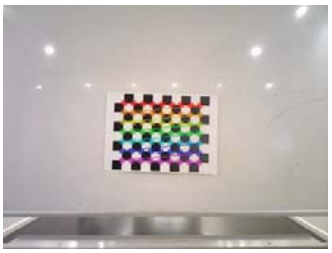
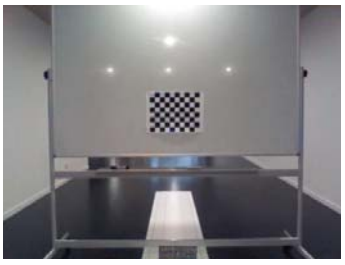
Corners on the chessboard could not be detected when the distance was closer than 1.5m. At the distance of 1.5m, corners were occasionally detected. For the Kinect V2, corners could be detected from 0.5m to 3m. The results of corner detection and RGB images at each distance

are demonstrated in Table 3- 6. The different corner detection results are caused by colour resolution. PrimeSense has 640 x 480 for colour images, Kinect V1 has 640 x 480, and Kinect V2 has 1920 x 1080. None of the cameras can recognize corners on the chessboard at distances greater than 3.5m.

A comparisons of the obtained depth points for each corner is demonstrated in Figure 3- 7 to 3- 10. In a range from 0.5m to 1.5m, depth points of corners were obtained from PrimeSense, Kinect V1 and Kinect V2 cameras. For Kinect V2, the depth of each corner can be obtained; however, a number of depth points were lost in the cases of PrimeSense and Kinect V1. A common reason for depth is occlusions, is due to RGB-D cameras requiring different views of an object to obtain depth. When an occlusion occurs, the object is present in only one view and it is not possible to obtain depth (Kadambi et al., 2014). Another possible reason is a matching ambiguity which occurs when point correspondences cannot be found. For this study, different depth and colour correlation methods can result in different ambiguities. Both PrimeSense and Kinect V1 use OpenNI2 for registration of colour pixels and their corresponding depth points, and Kinect V2 used Kinect SDK to map colour pixels to depth point.

The PrimeSense camera lost fewer depth points than the Kinect V1. This may be because of the different resolutions between depth points and the RGB image, resulting in lost data during the registration process. PrimeSense has the same resolution parameters for the depth points and RGB image, 640 x 480, while Kinect V1 has 640 x 480 for RGB and 320 x 240 for depth images.

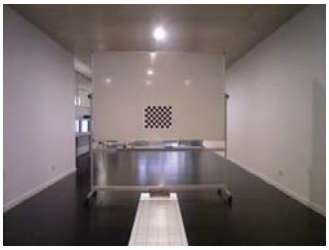
Table 3- 6: RGB images comparison

	PrimeSense	Kinect V1	Kinect V2
0.5m			
1m			
1.5m			
2m			
2.5m			

3m



3.5m



The Kinect V2 camera is able to locate each corner at distances greater than 2m. Figure 3- 10 demonstrates the evenly distributed depth points collected from 2m to 3m. The Kinect V2 is able to map colour pixels to their depth clouds accurately and efficiently. Another conclusion from Figure 3-7 to 3- 9 is that depth data from PrimeSense is not as accurate as that from Kinect V1 and V2, this will be discussed in the next section.

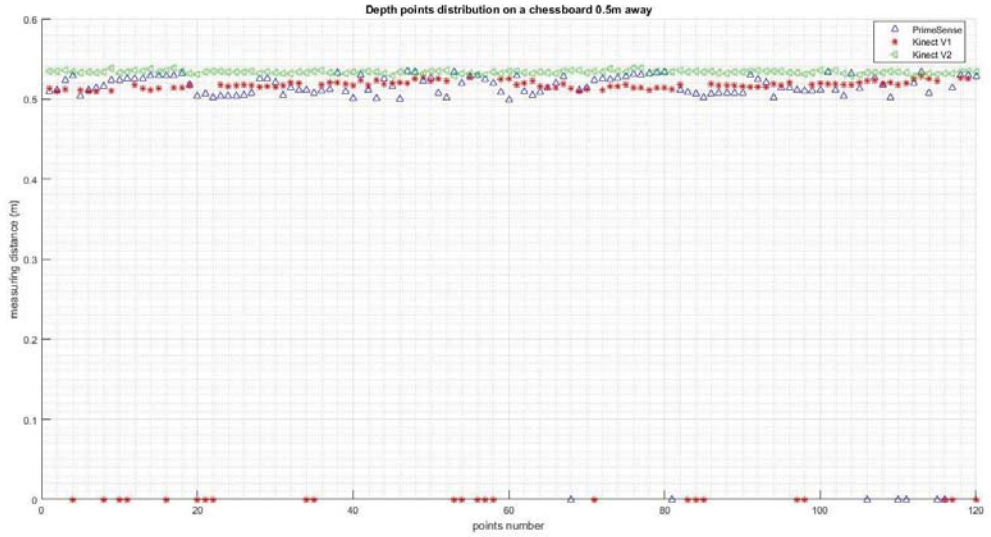


Figure 3- 7: Depth points distribution on a chessboard 0.5m away

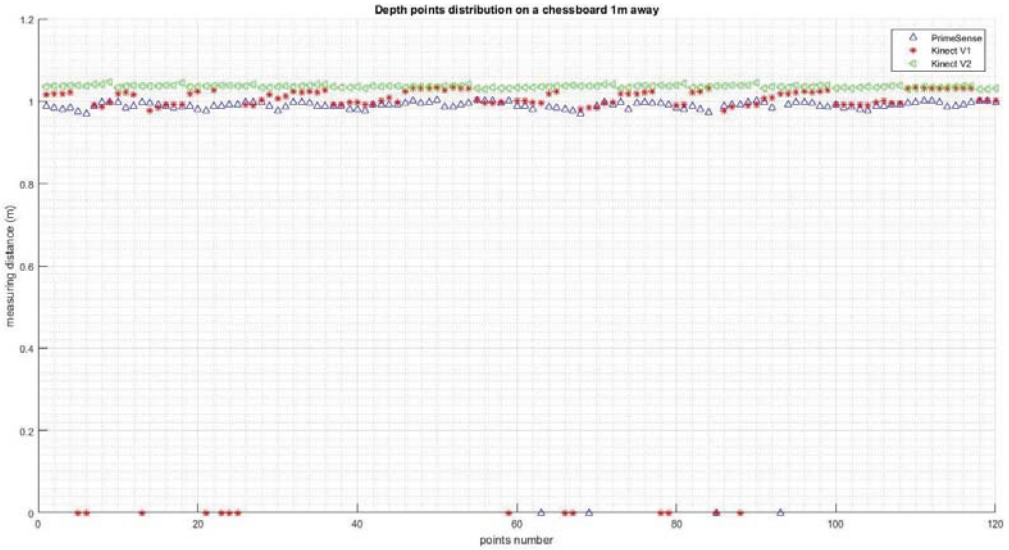


Figure 3- 8: Depth points distribution on a chessboard 1m away

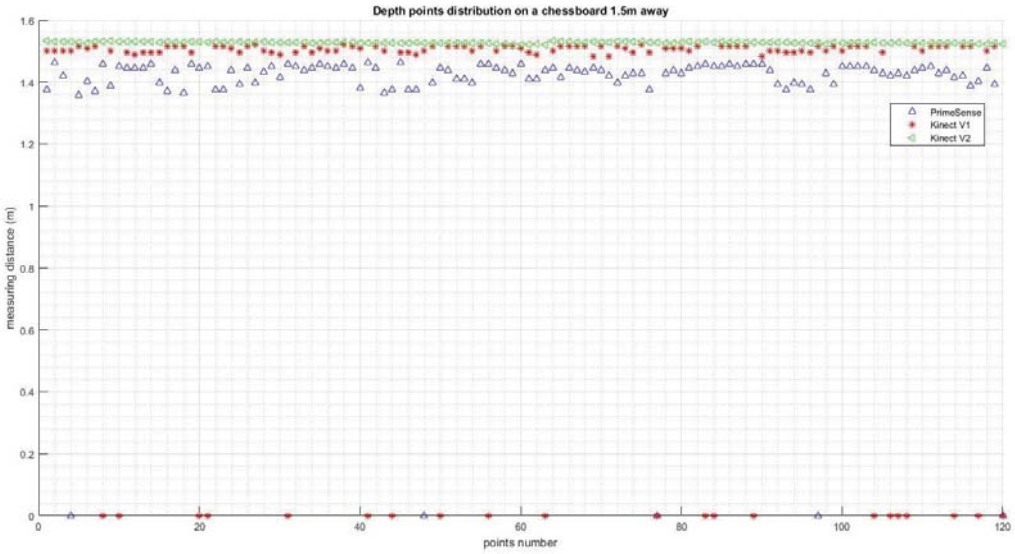


Figure 3- 9: Depth points distribution on a chessboard 1.5m away

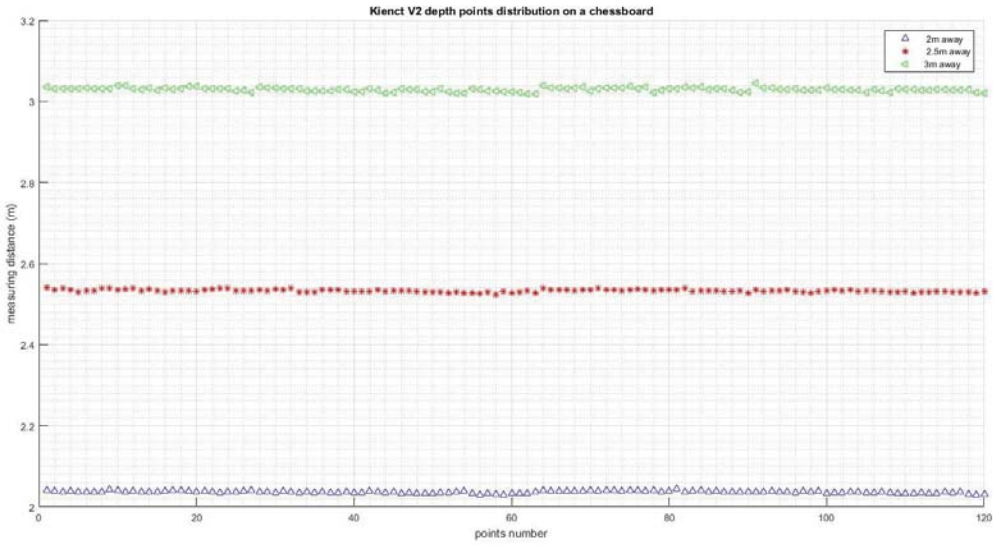


Figure 3- 10: Kinect V2 depth points in a chessboard at 2m, 2.5m and 3m away

3.4.3 Experimental Method for Depth Measurements

This section discusses point to plane distance which is the perpendicular distance from camera centre to the white board.

As discussed in section 3.3.2, measurements started from 0.5m to 3.5m away with a 0.5m gap, and the distances were measured by the grid-map. Figure 3- 11 describes the experiment set-up. The camera centre was placed perpendicular to the centre of chessboard. The camera, whiteboard and the chessboard were all parallel to the grid-map to make sure the camera was facing perpendicular to the chessboard. The centres of both the camera and chessboard were on the centre line of the grid-map. Thus, the blue line from the camera centre to chessboard centre was the perpendicular distance between camera and white board when the camera and chessboard centres were the same height from ground.

For each distance, 400 measurements were repeated and the average of the measurements considered as the final measurement results, as shown in Figure 3- 12. From 1m to 1.5m, all three cameras performed well, with error less than 0.05m. The Kinect V2 camera did not obtain depth data when it was at 0.5m. This is because it is black in on the measured chessboard centre and there is no infrared data reflected back. The Primesense camera had a 0.2m error when the distance was set to 3m, unlike the Kinect cameras which still maintained errors within 0.05m. All three cameras did not obtain accurate results at a distance at 3.5m. Errors for Primessense, Kinect V1 and V2 cameras are 0.18m, 0.18m and 0.2m respectively. This accuracy is probably affected by the depth range of cameras. The maximum range of the PrimenSense is 3.5m, which is also closer to the maximum ranges of the Kinect V1 and V2. It is concluded depth measurements accuracy declines when the distance is further and this applies to all the three cameras used in this thesis. This conclusion is reiterated in Chapter 6 where 0.1-0.2m

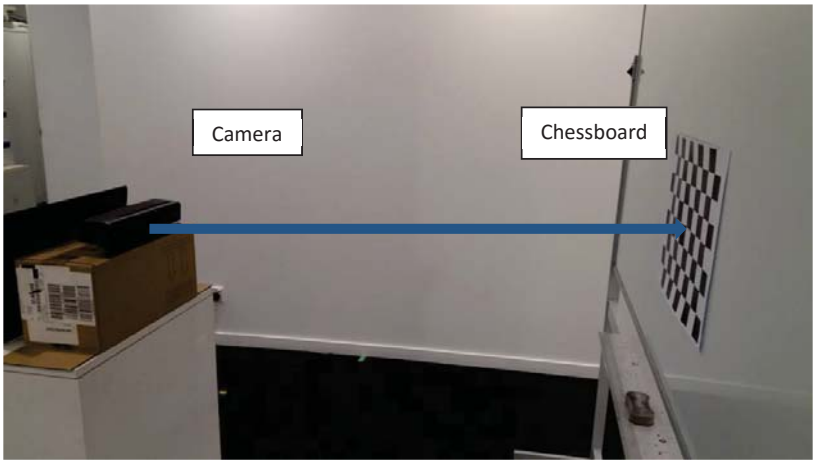


Figure 3- 11: Camera and chessboard position description

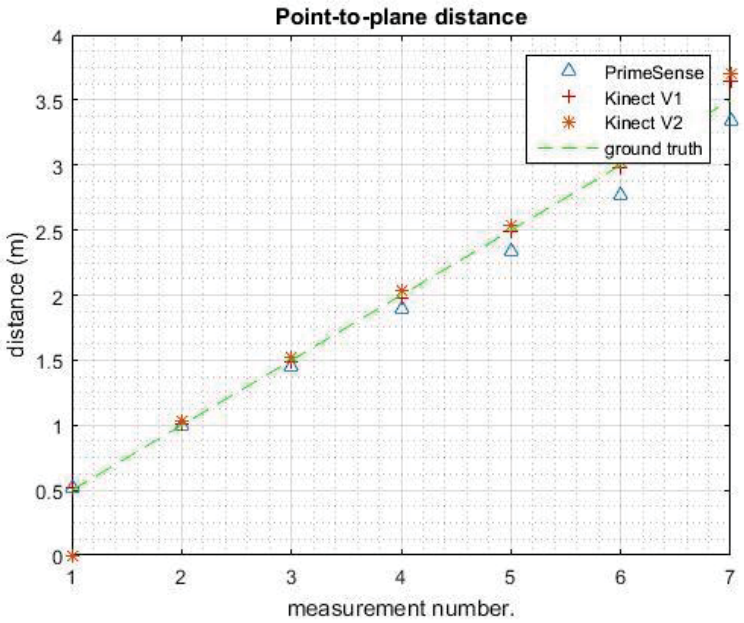


Figure 3- 12: Point-to-plane distance

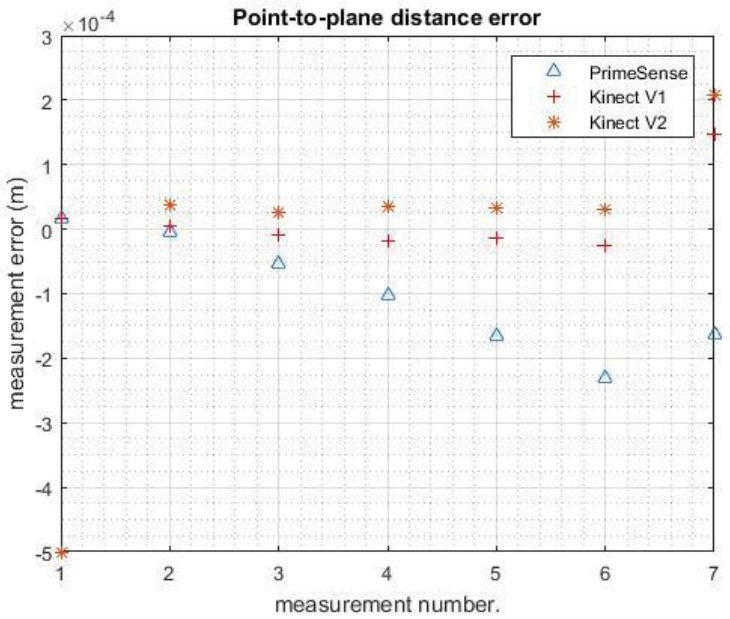


Figure 3- 13: Point-to-plane distance errors

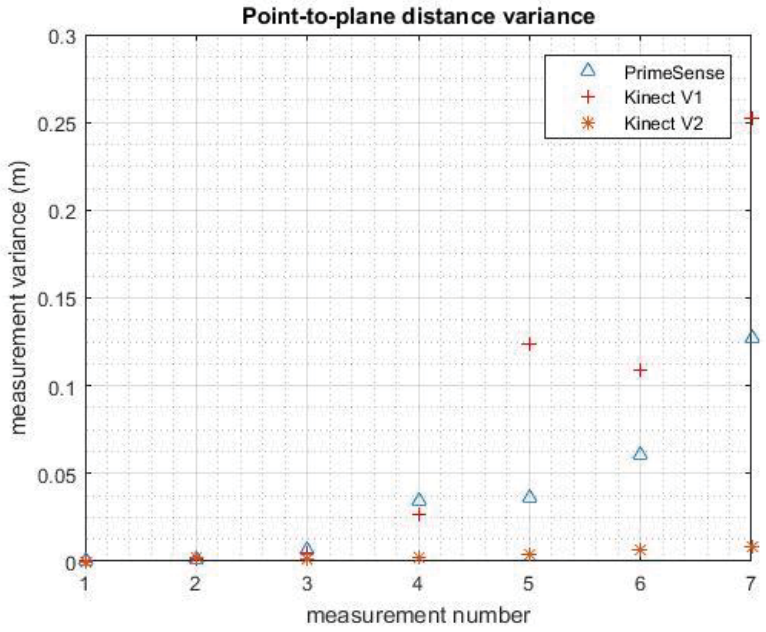


Figure 3- 14: Point-to-plane covariance

Table 3- 7: Covariance comparison

Variance(McManus et al.)	0.5m	1m	1.5m	2m	2.5m	3m	3.5m
PrimeSense	0.002	0.012	0.063	0.342	0.365	0.609	1.270
Kinect V1	0.002	0.013	0.052	0.266	1.237	1.091	2.525
Kinect V2	NA	0.015	0.022	0.023	0.042	0.066	0.080

errors exist when localising objects 3m away.

Accumulated least square error for each camera was calculated utilising the following equation (3-1):

$$E_{primesense} = \frac{1}{n}\sqrt{\sum_{i=1}^{n}\sum_{j=1}^{m}(d_{ij-}d_{i0})^2} \quad (3\text{-}1)$$

Where, n indicates set number of distance in the experiments, which is 7 (from 0.5m to 3.5m with a 0.05m gap increase each time); m means measurement times for a single distance, which is 400 in this experiment. d_{ij} Indicates the distance read from these three cameras; d_{i0} means the ground truth value of each single measurement. The results for three cameras are:

$$E_{primesense} = 0.04938\text{m} \quad (3\text{-}2)$$

$$E_{kinectv1} = 0.021771\text{m} \quad (3\text{-}3)$$

$$E_{kinectv2} = 0.031418\text{m} \quad (3\text{-}4)$$

Each calculation demonstrated that, the Kinect camera outperforms the PrimeSense camera. The Kinect V1 camera has a smaller least square error. To further compare Kinect cameras, variance of measurement are also calculated as equation (3-5)

$$V_{PrimeSense} = E(d_{ij} - d_{imean}) \tag{3-5}$$

Covariance data is described in Figure 3- 14. With the Kinect V2 has the most stable outputs.

Fitness functions for three cameras were obtained in Figure 3- 15, 3- 16 and 3- 17. Equations to fit real distance and measurements are obtained. With fitting function, measurements can be corrected when a raw depth data is read from the sensors.

$$d_c^{PrimeSense} = (d_{measure}^{PrimeSense} - 5.7157)/0.92145 \tag{3-6}$$

$$d_c^{Kinect\ v1} = (d_{measure}^{Kienct\ v1} + 3.2255)/1.0235 \tag{3-7}$$

$$d_c^{Kinnect\ v2} = (d_{measure}^{Kinect\ v2} + 32.064)/1.151 \tag{3-8}$$

Where, d_c is the corrected distance and $d_{measure}$ is the raw data from cameras.

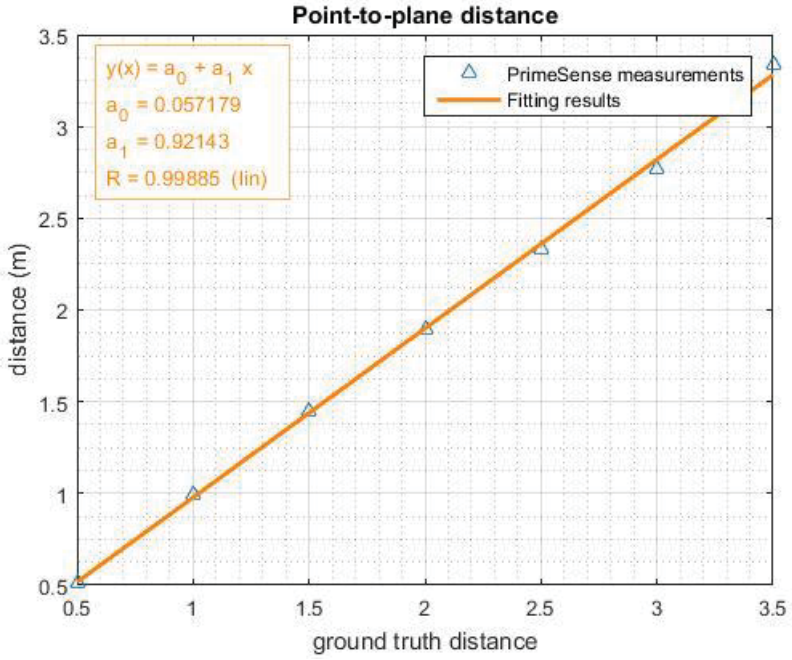


Figure 3- 15: Primesense measurements fit equation

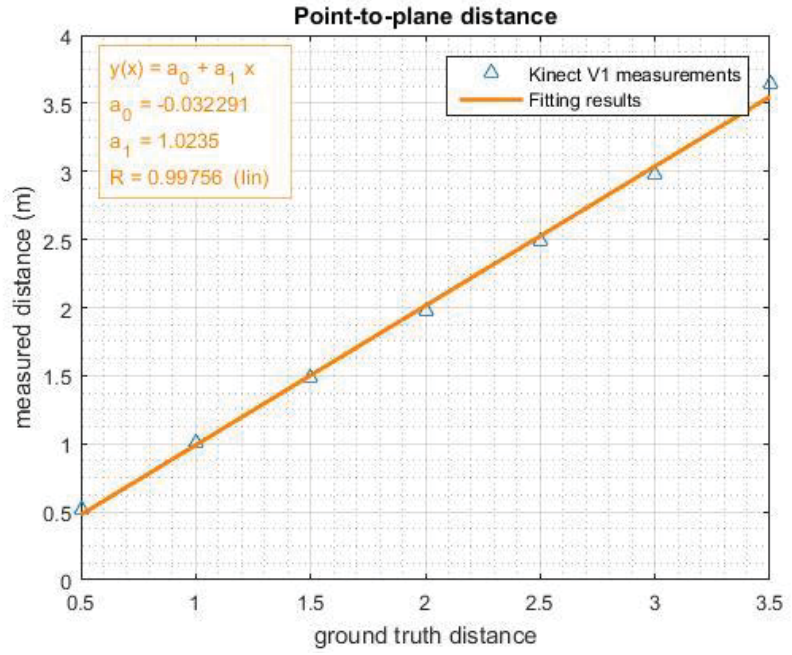


Figure 3- 16: Kinect V1 measurements fit equation

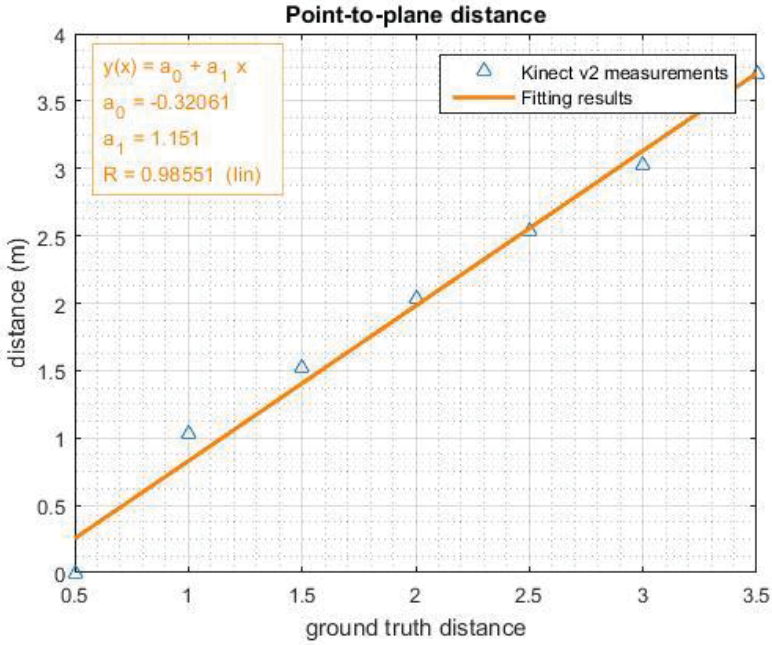


Figure 3- 17: Kinect V2 measurements fit equation

3.4.4 Point to Plane Distance by Point Cloud Library

A Point Cloud Library (PCL) based method for point to plane distance is discussed in this section. The proposed method computes perpendicular distance to a point cloud surface instead of a point, and avoids the failure when the measured point is missed as in section 3.4.3.

In order to obtain a point to plane distance to the whiteboard, a plane model which is perpendicular to the camera needs to occur for the whiteboard. The plane model is then used to deduce a point to plane distance from the camera to whiteboard. Steps to obtain the plain model are described in Figure 3- 18. The initial point cloud is obtained from a frame of depth stream. The current observing camera is set as the origin point as $P(0,0,0)$, which has three dimensional coordinate. In PCL, x, y and z arises are in the forward, right and up direction, respectively. Thus, the camera orientation is set along the x axis as $(1,0,0)$, and the orientation for

the perpendicular plane model is along z axis $(0, 0, 1)$. Parameters for the plane model also include a distance threshold for the depth point (which is 0.05 in this section); an angle constraint for the direction of the plane of 10^0 is used in this study. A Random sample consensus (RANSAC) method is used to look for inliers for the perpendicular plane model estimation, with 1000 maximum iterations. The obtained plane model is following the equation (3-9):

$$ax + by + cz + d = 0 \tag{3-9}$$

Where, a, b, c and d are plane model parameters. (a, b, c) is a normal vertor of the plane. d means the distance if the plane is moved to its original point. Absolute value of d is the camera to whiteboard distance as the plane is perpendicular to the camera.

Point to plane distances obtained from the PrimeSense, Kinect V1 and Kinect V2 cameras are obtained from are described in Figure 3- 19. These are similar to the results of section 3.4.3, Kinect cameras have a better accuracy than the Primense camera. With the PCL method, the Kinect V2 camera can detect at a distance of 0.5m from the camera (this failed in the previous study in section 3.4.3).

Errors of the measurements are also plotted in Figure 3- 20. When the PCL method is used to fit the points on the whiteboard to a plane, a threshold of 0.05 m and 10 degrees of variances are used which determines how close a point must be to the plane model in order to be considered acceptable. The following the equation (3-1) is used to calculate the accumulate least square error:

$$E^{pcl}_{PrimeSense} = \ 0.056392\text{m} \tag{3-10}$$

$$E^{pcl}_{Kinect\ v1} = \ 0.021312\text{m} \tag{3-11}$$

$$E_{Kinect\ v2}^{pcl} = \ 0.032305\text{m} \tag{3-12}$$

These errors are calculated with an uncertainty of 0.05m and 10°. Using the PCL method, the PrimeSense camera has a lager error of 0.03m than the results in section 3.4.3. Alternatively, Kinect V1 has a smaller error of 0.005m when measuring distance with the PCL based method. The Kinect V2 camera obtains a similar error of 0.21m using both methods.

Fitness functions for three cameras are obtained in Figure 3- 21, 3- 22 and 3- 23. Equations to fit real distance and measurements are also obtained in this section.

$$d_c^{PrimeSense\prime} = (d_{measure}^{PrimeSense'} - 6.6512)/0.90966 \tag{3-13}$$

$$d_c^{Kinect\ v1\prime} = (d_{measure}^{Kienct\ v1'} + 1.741)/1.0176 \tag{3-14}$$

$$d_c^{Kinnect\ v2\prime} = (d_{measure}^{Kinect\ v2'} + 1.9835)/1.0384 \tag{3-15}$$

Where, $d_c{}'$ is the corrected distance and $d_{measure}{}'$ is the raw data from cameras.

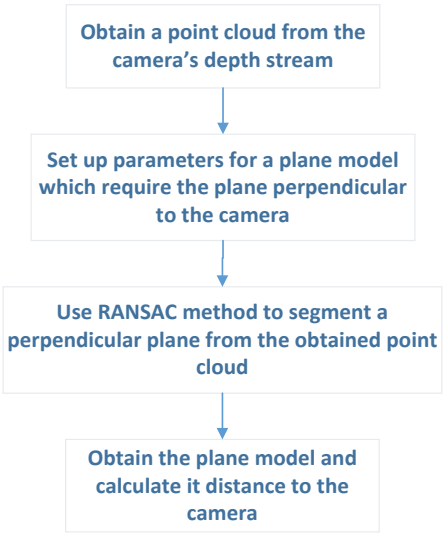


Figure 3- 18: A flow chart for obtaining a perpendicular plane with PCL

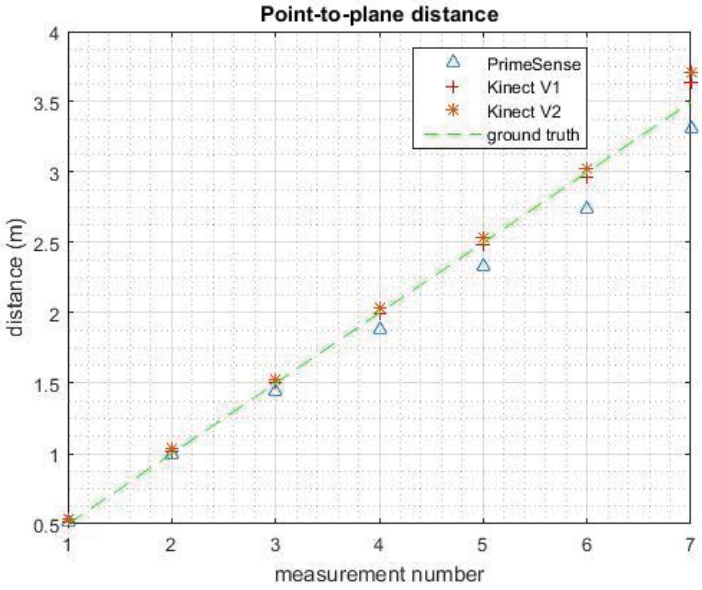


Figure 3- 19: Point-to-plane distance with PCL method

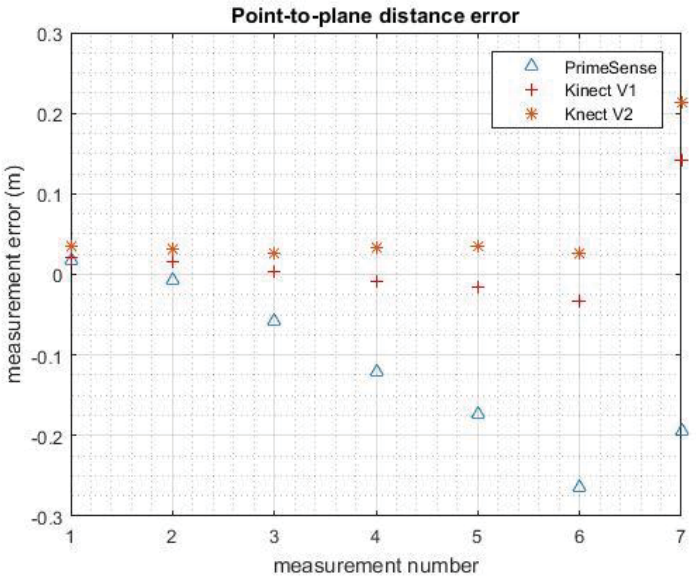


Figure 3- 20: Point to plane distance errors with PCL method

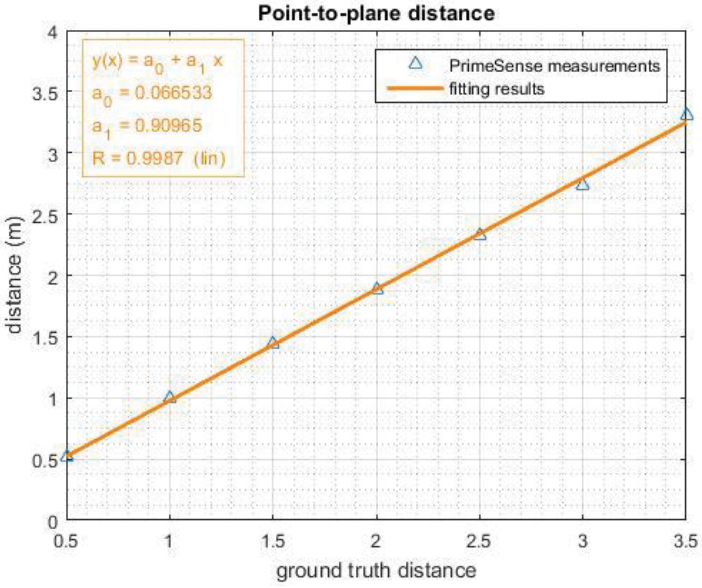


Figure 3- 21: Primesense measurements fit equation with PCL

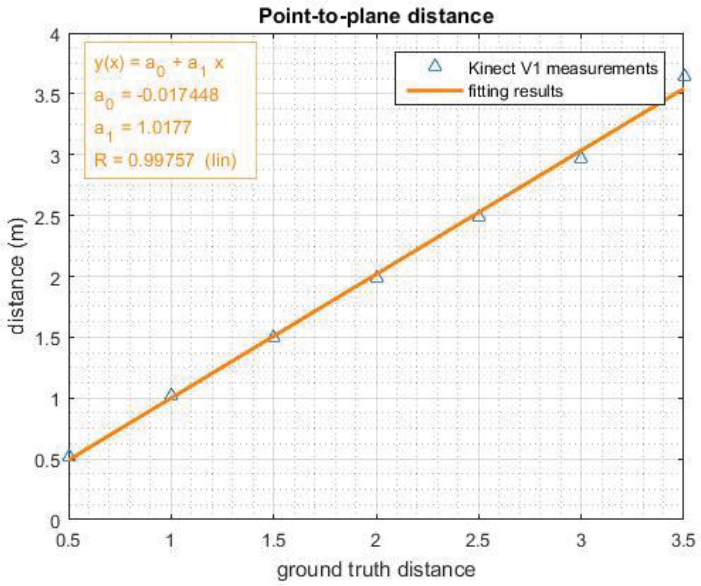


Figure 3- 22: Kinect V1 measurements fit equation with PCL

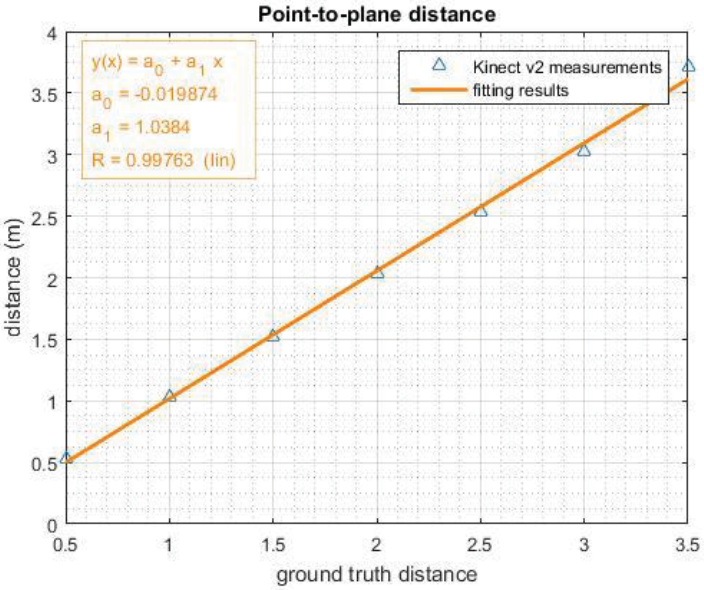


Figure 3- 23: Kinect V2 measurements fit equation with PCL

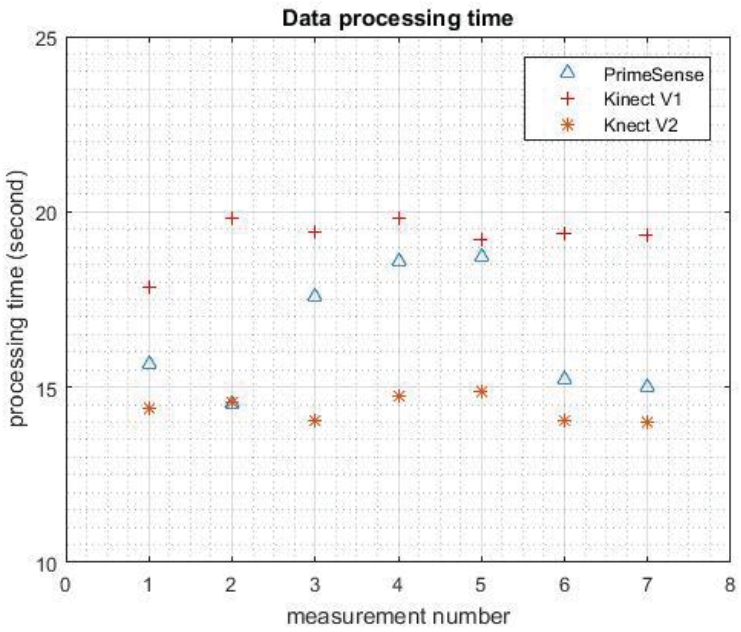


Figure 3- 24: Data processing time for three cameras

3.4.5 Processing Time

The data processingt time for section 3.4.3 is demonstrated in Figure 3- 24. For each measurement, the Kinect V1 camera requires an additional 5 seconds more than the Kinect V2 camera. The time cose of the Kinect V1 camera is between 18s and 20s, while the Kinect V2 camera only requires 14-15s. This may be because of light coding technology requiring cost more time than ToF for data processing. The time cost for both Kinect cameras is distributed evenly in a certain area bound. This means these two cameras work robustly. However, time costs of the PrimeSense camera fluctates between 15s and 18s. This is probably because that the camera's working condition was not stable.

3.5 Discussion

In terms of depth point clouds, the PrimeseSense and Kinect cameras can obtain relatively good quality clouds. Experimentation demonstrated the Kinect V2 camera obtains a high resolution cloud, this agrees with a study by Yang et al. (2015). Their study found that objects with reflective material can lead to a problem in that IR light cannot reflect back to the camera. This finding is also proved in section 3.4.3, where one depth value (at 0.5m) from Kinect V2 is unable to be determined.

In addition, this study found that only the Kinect V1 loses depth points in boarder area. This could be because of slight light interruption, although experiments were carried out in a stable light environment. Fankhauser et al. (2015) found a rapid increase in the amount of invalid measurements when there was direct sunlight.

The Kinect V2 has a high quality colour lense compared to PrimeSense and Kinect V1, which makes the Kinect V2 camera a stronger tool for feature detection. The other two cameras cannot detect corner features from a chessboard when the distance is further than 2m. Thus, RGB-D cameras are suitable for near field indoor area, as recommended in Langmann et al. (2012)'s study. This study also conducted a measurement on obtaining the depth for each corner on a chessboard. The Kinect V2 has a strong ability to map colour pixels to their corresponding depth points.

The Kinect V1 performs surprisingly better than V2 cameras when measuring point to plane distance. When the distance is within 0.5m to 3m, these have errors around 0.01m and 0.02m respectively. When the distance is set at 3.5m, Kinect V1 has an error of 0.15m, compared to Kinect V2 with a 0.2m error. This could be because of different techniques used inside of the

cameras, namely structured light and ToF. Smisek et al. (2011) conducted an experiment on planar target measurements, they found Kinect V1 performs better than SR 4000, which is another ToF camera discussed in section 2.2.1. However, other research concluded that the Kinect V2 is more accurate than the Kinect V1 in point cloud comparison (Zennaro et al., 2015), this could because the Kinect V2 obtains a better qualified depth points as approved in section 3.4.1. Additionally, when a point cloud from a laser scanner is set as ground truth, it is supposed to be same with that from Kinect V2, if the laser scanner also uses ToF techniques. To conclude, the Kinect V2 obtains a better quality point cloud and the Kinect V1 performs better for point to plane distances.

In addition, this study compared the cameras' "robustness" and efficiency which other studied have not done before. The Primesense camera is robust in registration of RGB frame and depth frames, as they have the same resolution. The Kinect V1 camera is efficient in point to plane measurement in the range from 0.5m to 3m when there is no direct light interruption. The Kinect V2 performs robustly when mapping depth points for each colour pixel, and is also efficient in depth data retrieved.

3.6 Conclusion

From the specification, PrimeSense, Kinect V1 and v2 cameras have their own advantages. The PrimeSense camera does not need a transformer and charger, so it is more portable for a mobile robot to use. It also has the same field of view for colour and depth frames, which will avoid some misalignments compared to cameras having different fields of view, such as the Kinect V1. The Kinect V1 camera has a higher point-to-plane measurement accuracy compared to Kinect V2. Depth cloud quality, point-to-cloud distance and colour images quality all need

evaluation when using each camera in different conditions. The Primesense camera works more robustly within the distance of 2m than the distance is further than 2m. The Kinect V1 and V2 can obtain accurate depth data. Kinect V1 camera obtained the most accurate point-to-plane distance in these three cameras, and is recommended for point cloud calculation when working in the range from 0.5m to 3m. this camera does however have depth loss. Depth quality is important, because frames with some depth points missing will be discarded from visual processing. As such the Kinect V2 is recommended if it is applied for areas using both image processing and depth point clouds.

3.7 Contribution

This chapter presents the performance of the popular used cameras and technology. Comparisons of these cameras provides useful advice for researchers and developers when choose an appropriate camera. Comparison outputs, such as errors level, covariance and computation time are important parameters when choosing cameras for a system.

Chapter 4 A Real-time Visual SLAM Method for Robot Navigation

This chapter presents a real-time Visual SLAM method for accurate state estimates used in a robot navigation process. This approach makes use of an RGB-D camera, an Extended Kalman Filter (EKF) and wheel encoders, and does not require any other sensors or odometry. In addition, this chapter integrates an iterated video frames module, into the EKF model. In contrast to classic EKF approaches, this module achieves more accurate state estimates. A number of experiments iterating different numbers of video frames are demonstrated. How the number of video frames affects SLAM estimates is discussed and analysed. Additionally, a frame selection method is proposed. Compared with a fixed number iteration of frames, the selection strategy maximises time efficiency and output accuracy.

4.1 Introduction

Simultaneous Localization And Mapping (SLAM) methods are adopted as tools for robot self-localization in navigating processes, due to their abilities of position estimation and environment mapping. While previously many SLAM methods relied on expensive and heavy laser scanners (Bosse & Zlot, 2008; Grisetti et al., 2010), the commercial launch of RGB-D cameras provided an attractive, powerful alternative for their advantages of light weight, lower price and ability of providing informative video frames at a high speed (30 fps) (Smisek et al., 2013).

RGB-D camera based SLAM methods were initially designed to reduce or tolerate the noise brought by depth point clouds, such as the loop-closure method (Henry et al., 2012; Segal et al., 2009), coarse-to-fine scheme (Kerl et al., 2013), local maps method (Hu et al., 2012), etc. In the recent two or three years, research has continued improving algorithm efficiency, such as sub-mapping method (Maier et al., 2014), some research develops towards Semantic SLAM, such as object-based reasoning (Dharmasiri et al., 2016; Ma et al., 2016), place recognition (Newcombe et al., 2015), etc. A Convolution Neural Network (CNN) is adopted into a SLAM framework, a PoseNet method was developed and achieved successful camera localization accuracy (Kendall et al., 2015). However, most of this recent work relies on GPU assistance to achieve real-time performance. This study develops a Visual SLAM method based on Extended Kalman Filter (EKF) for real-time robot navigation, and discusses how to obtain more accurate state estimates for robot self-localization.

To improve an EFK, this study develops a method of iterating renewed observations to linear-ize state estimates. This method is inspired by Julier and Uhlmann (2001) , who found that a stationary robot measuring the relative position of a new landmark multiple times, resulted in a decreased of estimated variance. However, their study did not demonstrate whether the decreased variance affects state estimates. Ling et al. (2013) found repetitive measurements can linearize the measurement in a EKF model when the state causes nonlinearities in the measurements function. However, their study did not explain how to determine measurement times. Instead of a free-floating camera used by Ling et al., a mobile robot carrying camera was used in this study. This study reveals the relationship between measurement times and state estimates; both theoretic explanation and mobile robot experiments will be discussed.

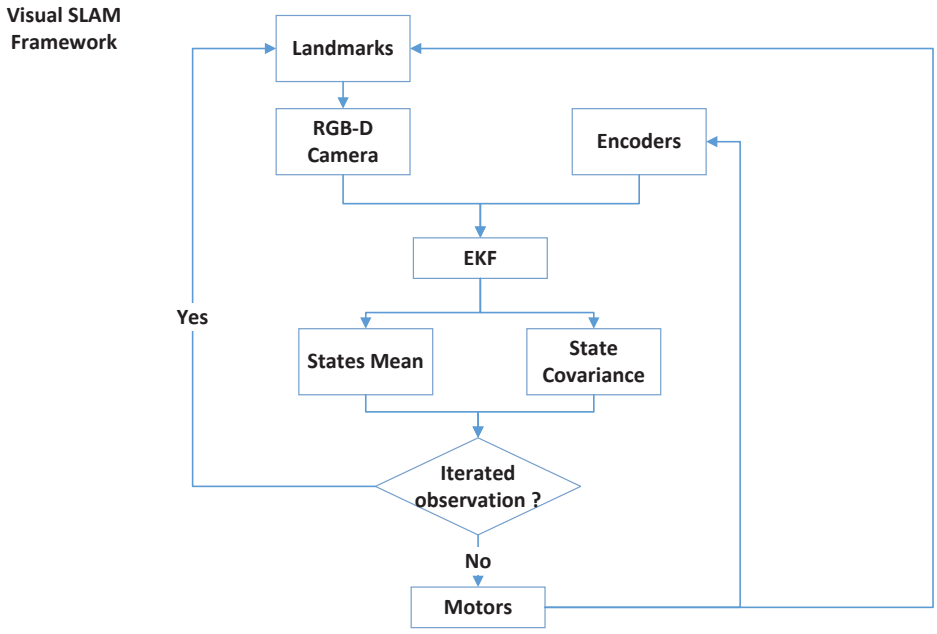


Figure 4- 1: A Visual SLAM framework

A framework of the proposed Visual SLAM method is demonstrated in Figure 4-1. Landmarks are used as interesting features. A RGB-D camera detects the landmark's features by colour video stream and obtains the distances of the corresponding feature points by depth video stream. Data from the RGB-D camera and encoders are the inputs of the EKF, which calculate the mean and covariance values for the current state estimates. The framework will be in the process of observing landmarks if there are not enough video frames iterated. Otherwise, the iteration stops and motors start to perform another movement. When the movement is finished, another round of landmarks observation and observation iteration will start again. The iteration process aims to obtain an optimal state estimate. This paper discusses how to determine a proper number of video frames for the iteration process. For this purpose, different numbers of iterating frames are used and the corresponding state estimates are obtained and compared; an automatic frame number selection method is developed and tested, also the

accuracy of the obtained states estimate and the efficiency of the processing time are compared with curtained frames number methods.

4.2 Preliminaries

EKF has been used as an observer of a nonlinear dynamic system to model robot motion and sensor perception, and provide an effective solution to the SLAM problems. The EKF SLAM theory is discussed in this section.

4.2.1 Robot and Landmarks Description

The goal of a typical EKF-SLAM solution is to use the information obtained from the observation and a robot's motion model to estimate the mean state vector $X = [X_v, X_f]^T$ and error covariance matrix P. The derivations and equations in this section are cited and summarized from Julier and Uhlmann (2001):

$$X = \begin{bmatrix} X_v \\ X_f \end{bmatrix}; \qquad P = \begin{bmatrix} P_R & P_{RF} \\ P_{FR} & P_F \end{bmatrix} \tag{4-1}$$

Where, X_v is the estimated location of the robot, X_f is the estimated location of stationary landmarks. The robot state vector $X_v = [x_v, y_v, \theta_v]^T$ represents the robot position and orientation. Landmarks are described as $X_f = [f_1, f_2, \dots f_n]^T$, which consists of individual landmark coordinates in the same reference coordinate frame of the robot.

P_R is the estimated error covariance of the location of robot, P_F is the estimated error covariance of the location of the landmarks, and finally, P_{FR} represents the cross-covariance between the different elements of the state vector.

A robot used for experiments was designed and built as it is shown in Figure 4- 2. As discussed in section 1.2.4, structure frames and driving systems are from VEX robotics. The driving system is controlled by an Arduino board. To read encoders' data in Arduino, a I2CEncoder library (Henning, n.d.) is used for data transportation. More details of the robot structure are listed in appendix B. A Kinect V2 camera is used as a visual sensor, and a laptop with a processer of core i5 is used as PC for SLAM programme processing.

A Landmark is illustrated in Figure 4- 2. Colour detection is used for feature extraction. The centre of the extracted area is considered as a point position in the colour frame coordinate. Transformation from colour frame to depth frame is achieved by a function called "$MapColorFrameToDepthSpace$" from Kinect SDK. The depth data for each colour pixel is saved in an array called "$depthBufferMat$". For each Point (u, v) at "$depthBufferMat$", its depth value, fZ, can be accessed by equation (4-1), which is the relative distance from the landmark to the camera, Equation (4-2) and (4-3) describe how to obtain the horizontal axis fX and vertical axis fY of the Point (u, v) , respectively. The derivations and equations below are cited and summarized from Khoshelham and Elberink (2012).

$$fZ = depthBufferMat.at(u, v) \tag{4-2}$$

$$fX = (u - C\_x) * fZ/F\_x \tag{4-3}$$

$$fY = (v - C\_y) * fZ/F\_y \tag{4-4}$$

Where, C_x and C_y are principle points, F_x and F_y are the focal lengths of the lens in x and y directions.

Mobile Robot Structure

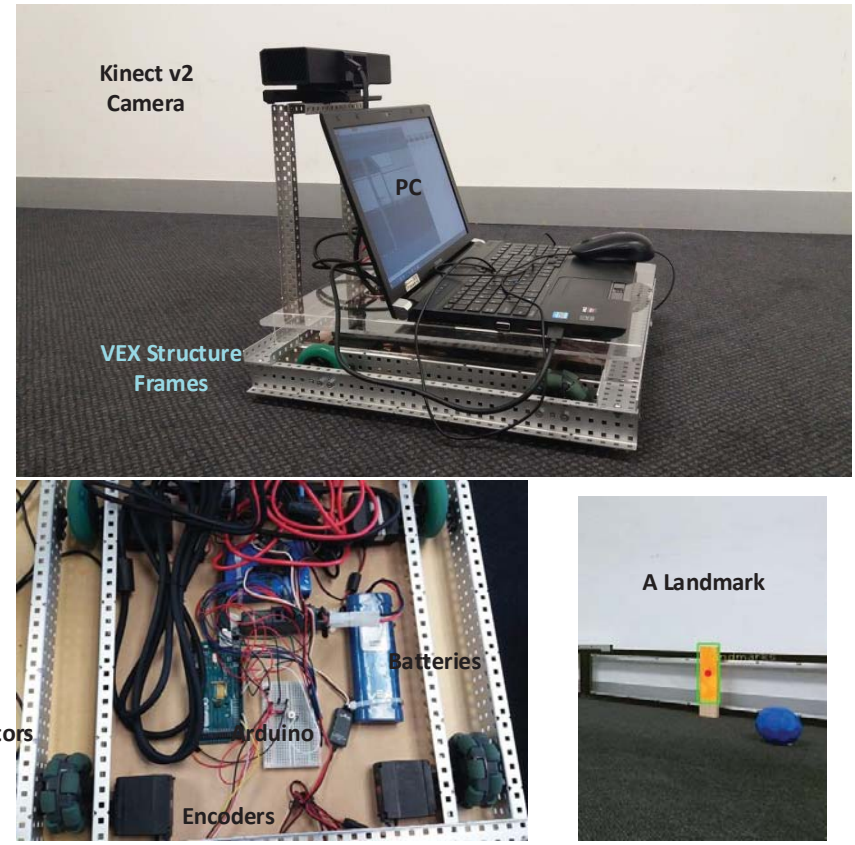


Figure 4- 2: The designed mobile robot in this study

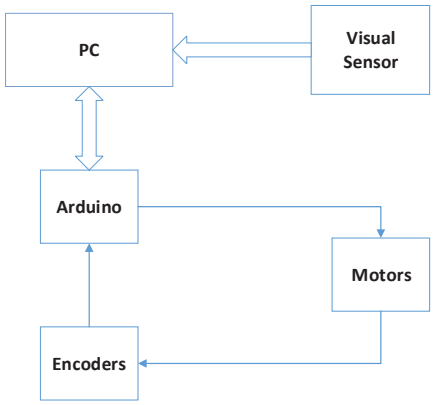


Figure 4- 3: Data flow in the robot

These parameters can be obtained through Kinect V2 calibration (Wiedemeyer, 2016), the results are as follows:

$$C\_x = 254.878;$$

$$C\_y = 205.390;$$

$$F\_x = 365.456;$$

$$F\_y = 365.456;$$

Measurement Z of landmark f is saved in the following format:

$$Z = h(f, X_v) = (r_f, \emptyset_f)^T \tag{4-5}$$

Where, h is an observation model, r_f and $\emptyset_f$ are relative distance and angle from the landmark to the observing camera. Parameters for a landmark representation are obtained from the following equation:

$$\begin{pmatrix} r_f \\ \emptyset_f \end{pmatrix} = \begin{pmatrix} fZ \\ arcsin\frac{fX}{fZ} \end{pmatrix} \tag{4-6}$$

Data flow in the mobile robot is described in Figure 4- 3. The Visual sensor, Kinect V2 transfers depth and RGB video frames to the PC. The PC sends orders to the Arduino, through which robot movements are controlled. Motion speed and distance are controlled by a PID module in the Arduino. Feedback signals for the PID module are from encoders.

4.2.2 EKF Prediction and Update

State prediction: In this stage, the robot control inputs are processed to obtain a prediction for state vector X and the covariance of the system state, which is propagated through the

linearized motion. The '+' and '-'superscript are used to identify the state vector and covariance matrix values after update and after prediction stages, respectively. All the equations and deviations in this section are cited and summarized from S. Huang and Dissanayake (2007). To describe the state update, the general motion model for system state vector becomes:

$$X^{-}(k) = f(X^{+}(k-1), u_k) \tag{4-7}$$

In which, u_k is a control input with zero mean Gaussian noise with a covaria nce Q, andprediction model $f(\cdot)$, which defines how to compute an expected position of the robot given the old position and the control input. k indicates the times of robot' movements.

In prediction step, covariance P becomes:

$$P^{-}(k) = F(k-1)P^{+}(k-1)F(k-1)+Q(k-1) \tag{4-8}$$

F is the Jacobian of the prediction model $f(\cdot)$.

Observation Prediction: with the results of the previous observation and the observation model, whether a feature should be detected at the current step can be determined.

Observed landmarks' features are fed into a data association procedure to find out possible correspondence between previously seen features and currently observed ones. After this step, observed features fall into two categories: newly observed features and previously observed features. The former is stored and to be augmented to the system vector in the following augmentation step. Prediction vector for landmark features $Z^{-}(k)$ in the second category is calculated as the stacked vector of the individual sensor measurement predictions $z^{-}(k)$. For observation model h, observation Jacobian $H = \partial h/\partial X|_{(f^{-}, X_v^{-})}$ is calculated by substituting (f^{-}, X_v^{-}) in the following equation.

$$H = \begin{pmatrix} -cos(\emptyset) & -sin(\emptyset) & 0 & cos(\emptyset) & sin(\emptyset) \\ \frac{sin(\emptyset)}{r} & \frac{-sin(\emptyset)}{r} & -1 & \frac{-sin(\emptyset)}{r} & \frac{cos(\emptyset)}{r} \end{pmatrix} \quad (4\text{-}9)$$

$$Z^{-}(k) = \mathrm{h}(x^{-}(\mathrm{k}), \mathrm{P}^{-}(\mathrm{k})) \quad (4\text{-}10)$$

$$e(k) = Z^{+}(k-1) - Z^{-}(k) \quad (4\text{-}11)$$

Update: The covariance update rule is expressed in the information form as:

$$S(k) = H(k)P^{-}(k)H^{T}(k) + R(k) \quad (4\text{-}12)$$

$$X^{+}(k) = X^{-}(k) + W(k)e(k) \quad (4\text{-}13)$$

$$P^{+}(k) = P^{-}(k) - W(k)S(k)W^{T}(k) \quad (4\text{-}14)$$

$$W(k) = P^{-}(k)H^{T}S^{-1}(k) \quad (4\text{-}15)$$

$R(k)$ is the measurement covariance. $W(k)$ is Kalman gain.

Augmentation: In this stage, first-ever seen landmark features are included in the system state vector through the inverse observation model. Meanwhile, the covariance matrix of the system vector is expanded in order to include the correlation information of newly observed features. In this step, new observed features are initialized into the system.

4.2.3 Covariance Analysis

In EKF SLAM, the pair (X^{+}, P^{+}) is considered as a consistent estimate for the variable X if the two following conditions are satisfied (Tamjidi et al., 2009) :

$$E[X - X^{+}] = 0 \quad (4\text{-}16)$$

$$E[(X - X^{+})(X - X^{+})^{T}] = 0 \quad (4\text{-}17)$$

Researchers studied the case of a stationary robot making m times of observations on one static landmark from one or two poses. For a robot with initial uncertainty P_0 observing a feature m times with covariance R, if all Jacobians are calculated at their ideal state, the general expression for covariance matrix of the augmented system state vector becomes (S. Huang & Dissanayake, 2007):

$$P^+(m) = \begin{pmatrix} P_0 & P_0 H_e^T \\ H_e P_0 & H_e P_0 H_e^T + \frac{H^{-1} R H^{-T}}{m} \end{pmatrix} \tag{4-18}$$

Where, matrices H_e and H are obtained from the Jacobians of the observation model. If a robot is stationary and observe a single feature m times, the robot state vector and its variance should decrease if m is not infinite.

4.3 Approach

An iterated video frames module is discussed in this section. During this process, the robot is stationary, while the camera continuously observing frames and the EKF continues working and iterating observation frames till a desired frame number is reached. The following equations describe the calculation process that the robot comes to the kth waypoint, and perform the ith observation.

$$Z_i^-(k) = \mathrm{h}(X_i^-(\mathrm{k}), P_i^-(\mathrm{k})) \tag{4-19}$$

$$e_i(k) = Z_{i-1}^-(k) - Z_i^-(k) \tag{4-20}$$

$$S_i(k) = H_i(k) P_i^-(k) H_i^T(k) + R_i(k) \tag{4-21}$$

$$P_i^+(k) = P_i^-(k) - W(k) S_i(k) W^T(k) \tag{4-22}$$

$$X_i^+(k) = X_i^-(k) + W(k)e(k) \tag{4-23}$$

$$W(k) = P_i^+(k)H_i^T(k)S_i(k) \tag{4-24}$$

Where, $X_i^-(k) = X_{i-1}^+(k)$, $P_i^-(k) = P_{i-1}^+(k)$. Because the robot does not move when the observation is repeated, there is no variation of state estimates. The predicted state of i is the same as the update state of $i-1$. The same explanation applies on to covariance P as well.

The computation cycle of the proposed algorithm is as follows:

1) Set up frame processing number i_p.
2) Calculate predicted states for X and P.
3) Calculate update states for X and P.
4) Increase current processing frames number i.
5) If i is below i_p, repeat 2) to 5)
6) Obtain final X and P; move to another waypoint if the route is not finished.

Algorithm 1: Visual SLAM with iterated observations

Define iterated video frames number $i\_p(5,10,15,20,25\ or\ 30)$

Define robot total move waypoints k

X, P=setzero();

for $k = 1$ to steps **do**

$x_i^-(k)$, $P_i^-(k)$ from equation(4-7)(4-8)

for $i < i\_p$ **do**

$Z_i^-(k)$ from equation (4-19)

$P_i^+(k)x_i^+(k)$from equation(4-22)(4-23)

$i++$

$X_i^-(k)$=$X_{i-1}^+(k)$, $P_i^-(k)$=$P_{i-1}^+(k)$

End for

Robot moves, read from encoders, $k++$, $i = 0$

End for

Return X, P

4.4 Simulation Results

This section discusses the application of Algorithm 1 in a simulated environment. Tim Bailey's SLAM package (Bailey, N.A.) is used to test the iterated video frames module. Waypoints and landmarks are predefined in the package.

The average Normalized Estimation Error Squared (NEES) is used to characterize the iterated video frame based EKF SLAM method. The NEES is obtained in the following format:

$$D^2(k) = \left(x^+(k) - \hat{x}(k)\right)^T (P^+(k))^{-1} \left(x^+(k) - \hat{x}(k)\right) \quad (4\text{-}24)$$

In which, $\hat{x}(k)$ is the true state and $x^+(k)$ is the estimated value at a random time step k.

A simulation of NEES among 50 times of Monte Carlo run is demonstrated. For a random time step, NEES results of standard EKF SLAM (1 iteration) and 10 frames' iteration based SLAM averaged over the 50 simulation repetitions is obtained as follows:

$D^2_{1iteration}$= 262.6806

$D^2_{10iteration}$= 104.0406

It is clear that standard EKF SLAM has more than two time the NEES value than the 10 times iteration method. Time costs are compared in Figure 4- 5. It is demonstrated that the 10 times iteration method costs 10 s more than standard EKF SLAM.

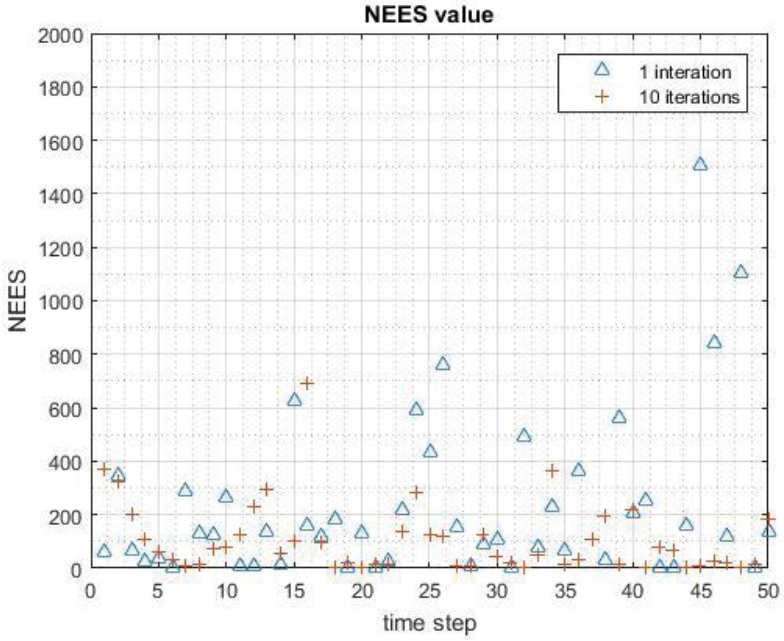


Figure 4- 4 NEES comparison

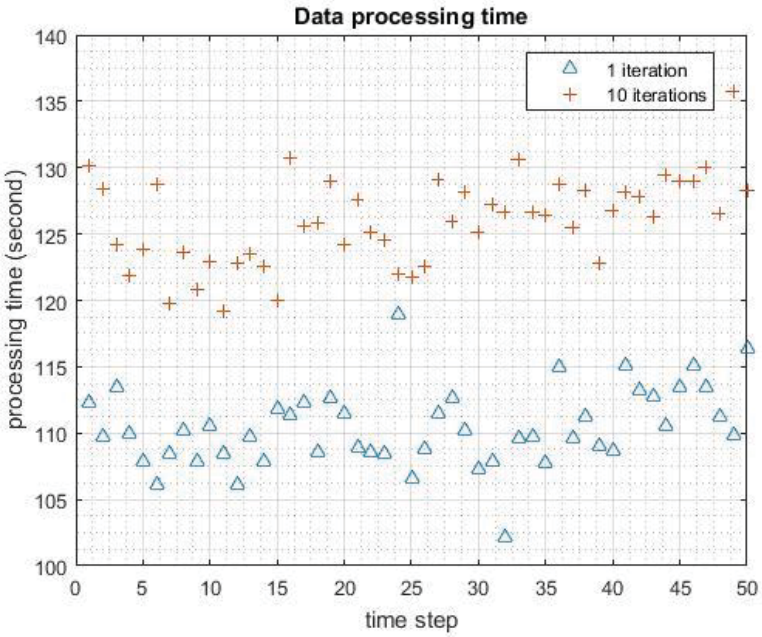


Figure 4- 5: Time costs comparison

4.5 Experimental Results

This section discusses the application of Algorithm 1 in an experimental environment with a different number of landmarks. The robot starts from the original point (0, 0), and moves forward a certain distance each time. The fixed distance is determined by a report from Konolige and Agrawal (2008), who found that a key frames selection is typical based on the distance between frames about 0.1m-0.5m. Thus, the robot is designed to move forward every 0.3m from the origin point every time. The robot moves forward to four waypoints as described in Figure 4- 6. For each of the waypoints, the encoder results, SLAM results and ground truth are recorded.

4.5.1 One Landmark

This section discusses the application of Algorithm 1 with one landmark situation. Cases using a different numbers of iteration frames, 5, 10, 15, 20, 25 and 30, were tested and compared. For each test, the robot stopped when encoder results were 0.3m, 0.6m, 0.9m and 1.2m. At each stop, the robot observed the landmark and computed current state and covariance.

Figure 4- 7 describes the state estimates of different iterations when the robot stopped at 0.3m. The SLAM estimate results were closer to ground truth when the number of iterated frames increased. It was a 0.02m distance to ground truth when iterating 30 observation frames. But the encoder results were the closest, being 0.005m close to ground truth. SLAM estimate results became closer to ground truth when the robot moved to 0.6m, as is demonstrated in Figure 4- 8. 25 frames processing resulted in fluctuation around encoder results, while 30 frames processing results came close to the ground truth steadily. But the encoder results were still more accurate than SLAM results at this stage.

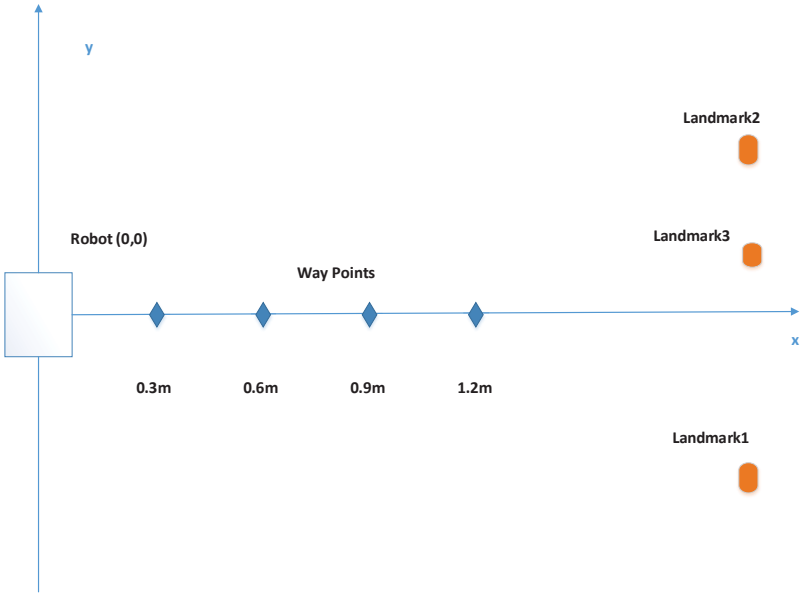


Figure 4- 6: Experiment design

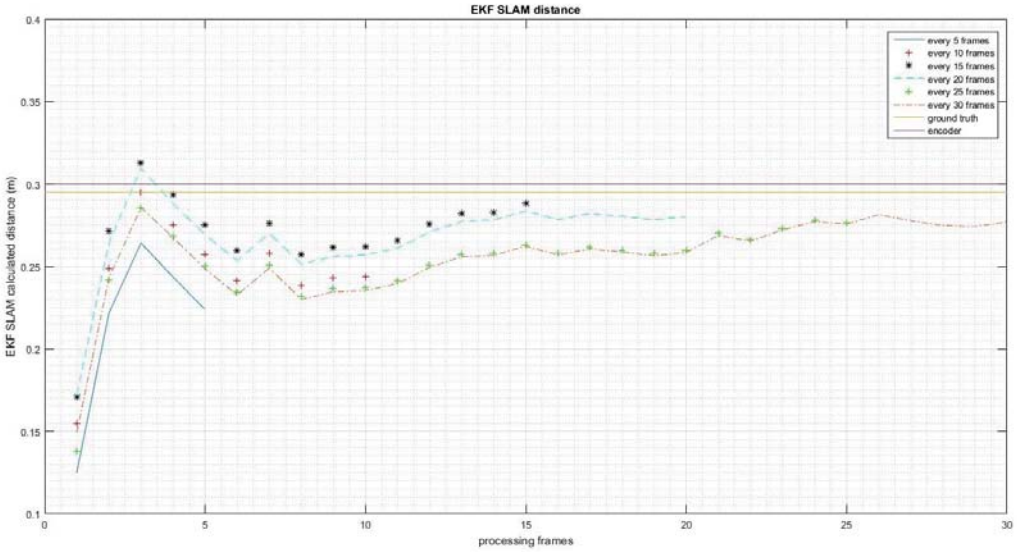


Figure 4- 7: Distance comparison from 0 to 0.3m

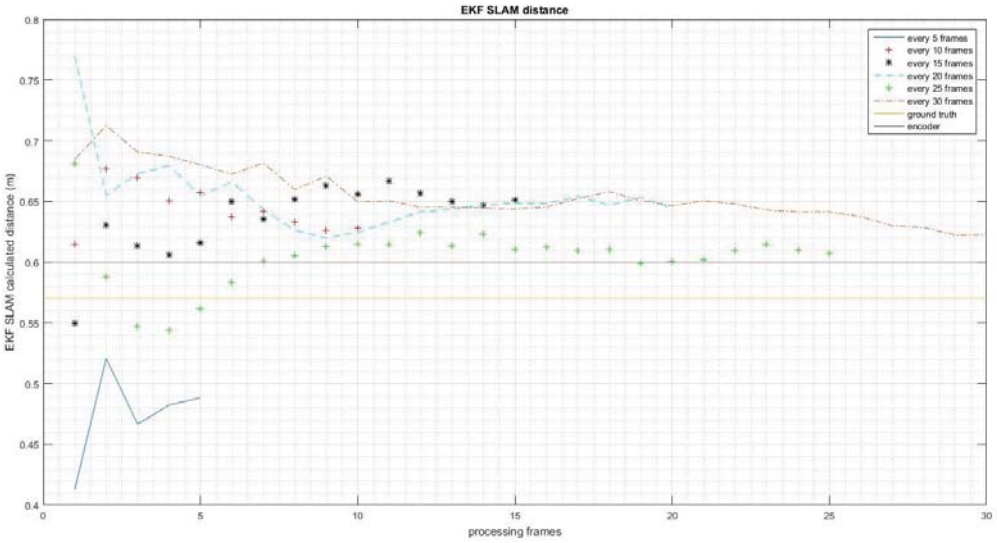


Figure 4- 8: Distance comparison from 0.3m to 0.6m

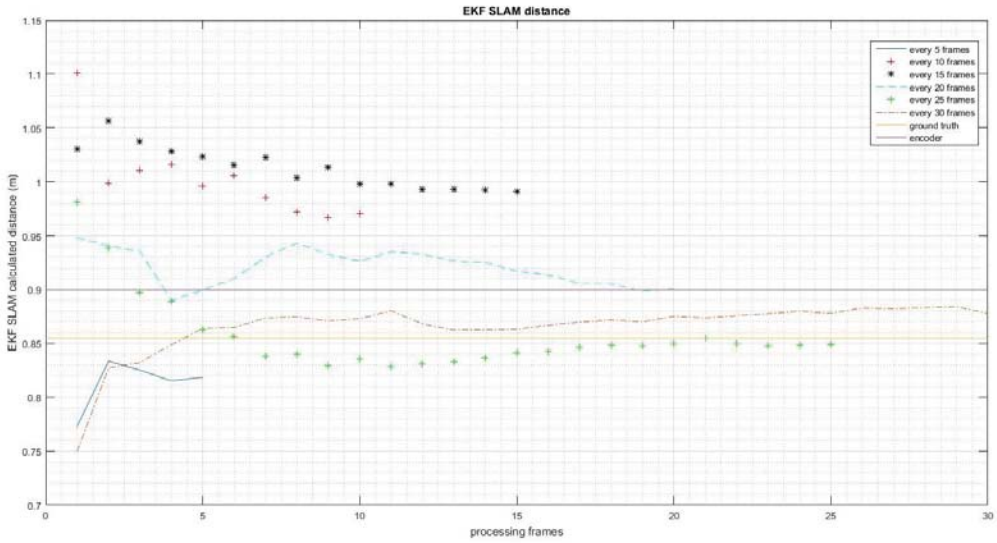


Figure 4- 9: Distance comparison from 0.6m to 0.9m

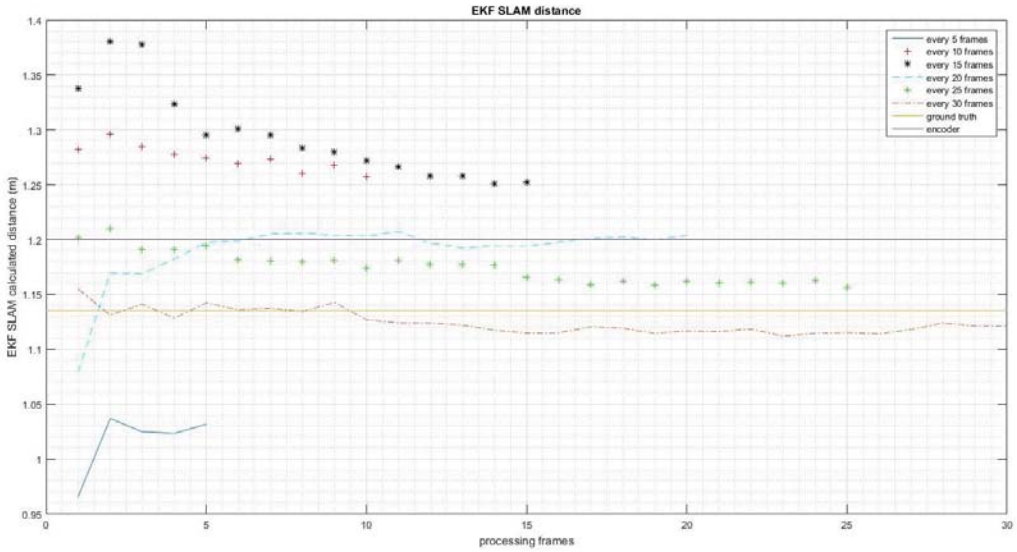


Figure 4- 10: Distance comparison from 0.9m to 1.2m

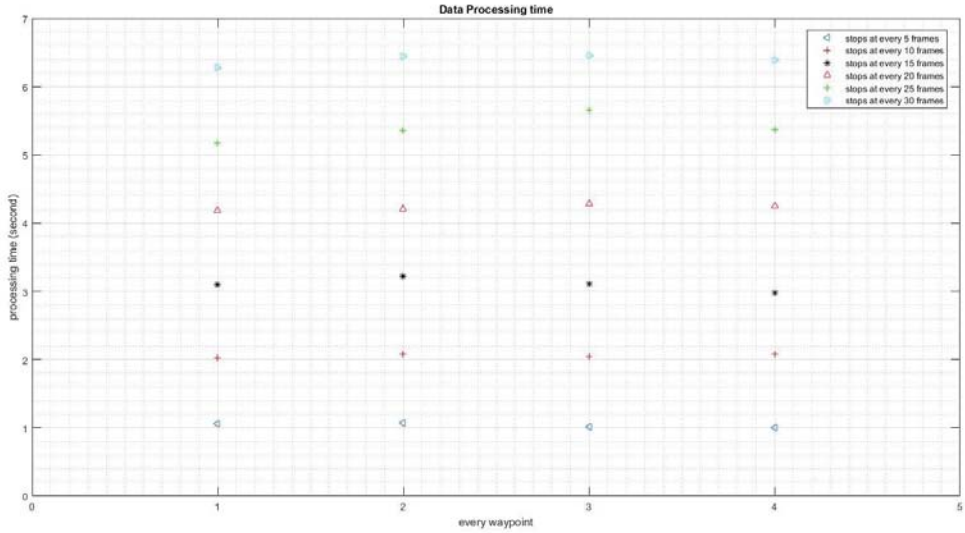


Figure 4- 11: Time comparison for each stop

The SLAM method obtained more accurate results than encoders when the robot moved forward to 0.9m. Figure 4- 9 shows that both 25 frames and 30 frames iteration methods performed better than the encoders. The error of SLAM estimates were around 0.023m while the error of encoders were at 0.05m. When the robot moved to 1.2m, SLAM estimate was 0.01m close to ground truth when frames were iterated 30 times. These results are demonstrated Figure 4- 10. 25 and 30 frames are both recommended for an accurate robust estimate in one landmark condition. While the 30 frames iteration has an accurate estimate, computation is costly.

Time cost of different frames of iteration is compared in Figure 4- 11. Time costs increased when more frames were iterated. Nearly every 5 frames cost around 1 second. When it was 30 frames, the processing time grew up to 6.4s.

4.5.2 Two Landmarks Analysis

This section discusses when two landmarks are used for localization. The two landmarks situation performed better than one landmark.

SLAM estimates from 30 frames of iteration were already more accurate than the results from encoders when the robot stopped at the 0.3m waypoint. The results are demonstrated in Figure 4- 12. Other SLAM estimates were also distributed around ground truth steadily. When moving from 0.3m to 0.6m, most of the SLAM results performed better than encoders. These results are demonstrated in Figure 4- 13. The SLAM method outperformed encoders when the iteration time was greater than 15. Estimates of 30 frames of iteration were nearly the same with ground truth. Similarly in Figure 4- 14, the SLAM method kept on performing well when the robot moved from 0.6m to 0.9m. Estimates from 20 frames, 25 frames and 30 frames of

iteration all grew smoothly in the direction of ground truth. However, the 15 frames of iteration method at 0.9m did not perform as well as when the robot moved to 0.6m. This could because that 15 times of observing is not robust for a steady SLAM estimate. The SLAM estimates outperformed encoders when the robot came to the last waypoint, 1.2m. Even 5 times of iteration obtained more accurate results than encoders. The results are shown in Figure 4-15. Time consuming for observing two landmarks is demonstrated in Figure 4- 16. It cost nearly 1.1 seconds for processing 5 frames, and 6.8 seconds for 30 frames.

To summarise, from 5 frames to 15 frames of iteration, results fluctuated around ground truth with a large offset up to 0.08m. Estimates from frame numbers of 25 to 30 achieved a result which is as close as 0.01m accurate to ground truth, but the computation cost 6s for every 0.3m movements. Thus, it is not a wise idea to choose 25 or 30 frames as a processing threshold. Additional, compared with one landmark, observing two landmarks can enhance SLAM method performance while maintaining the same level of time cost.

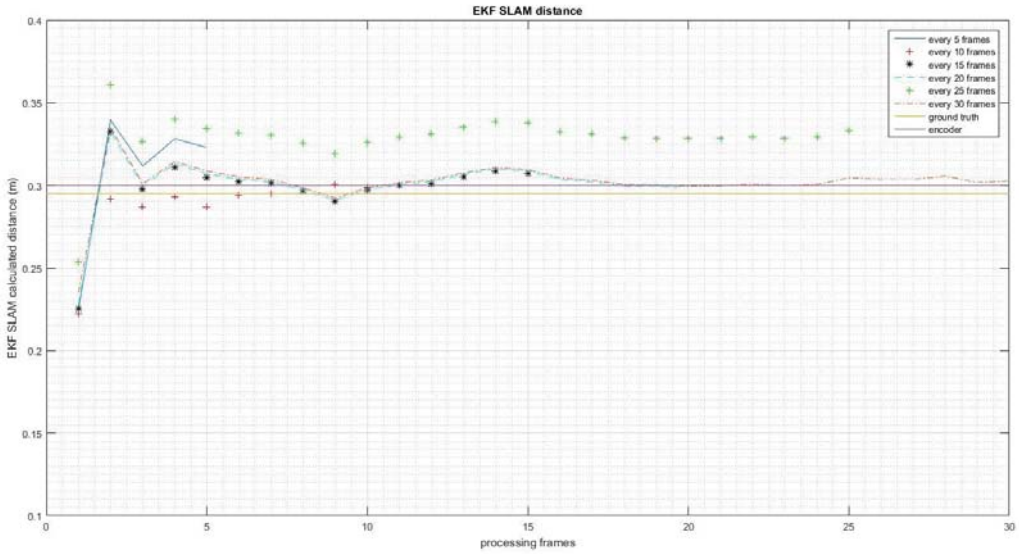


Figure 4- 12: Distance comparison from 0 to 0.3m with two landmarks

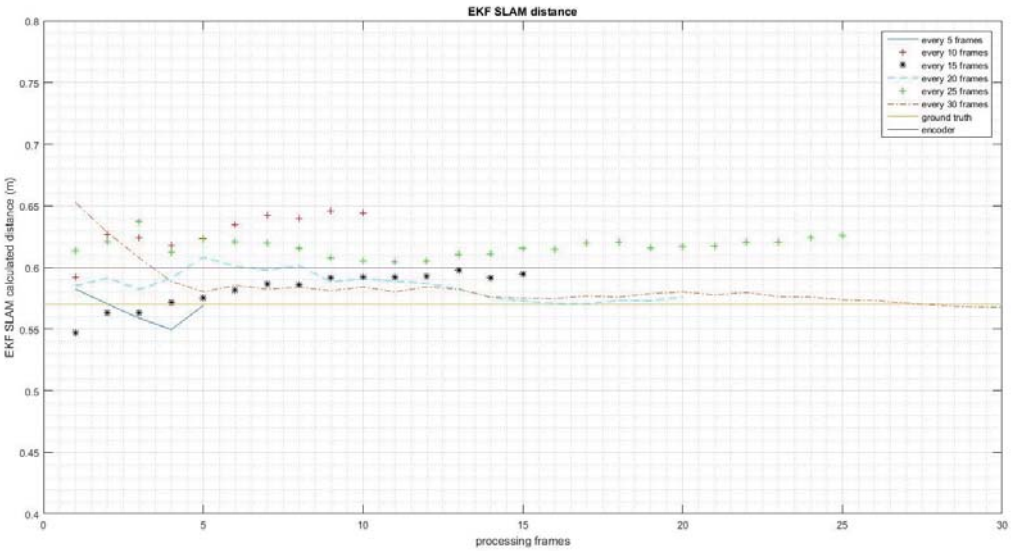


Figure 4- 13: Distance comparison from 0.3m to 0.6m with two landmarks

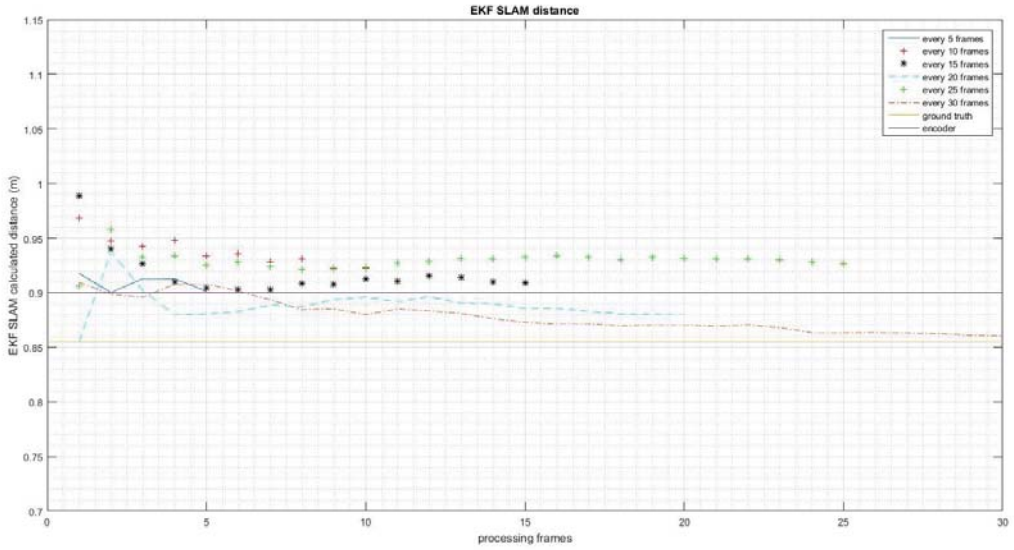


Figure 4- 14: Distance comparison from 0.6m to 0.9m with two landmarks

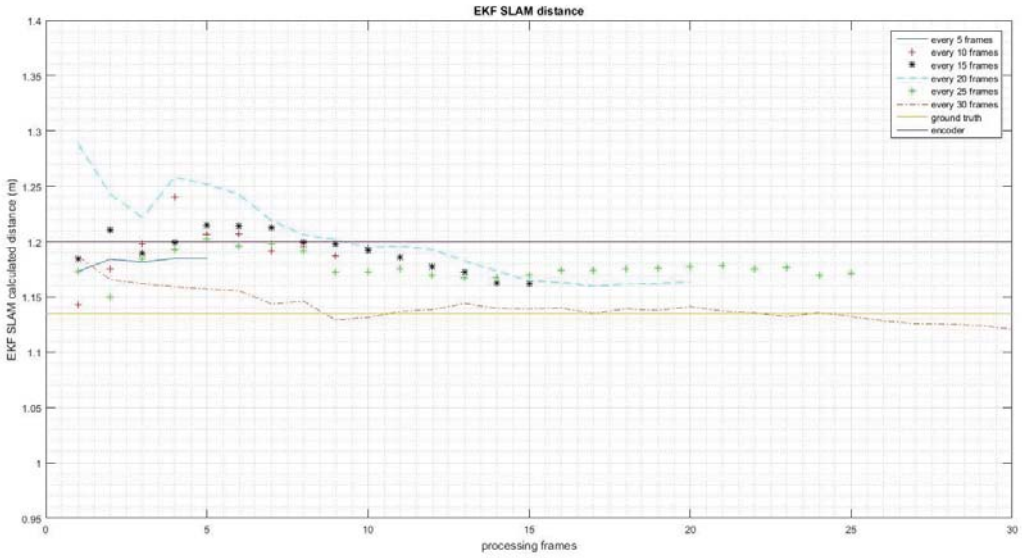


Figure 4- 15: Distance comparison from 0.9m to 1.2m with two landmarks

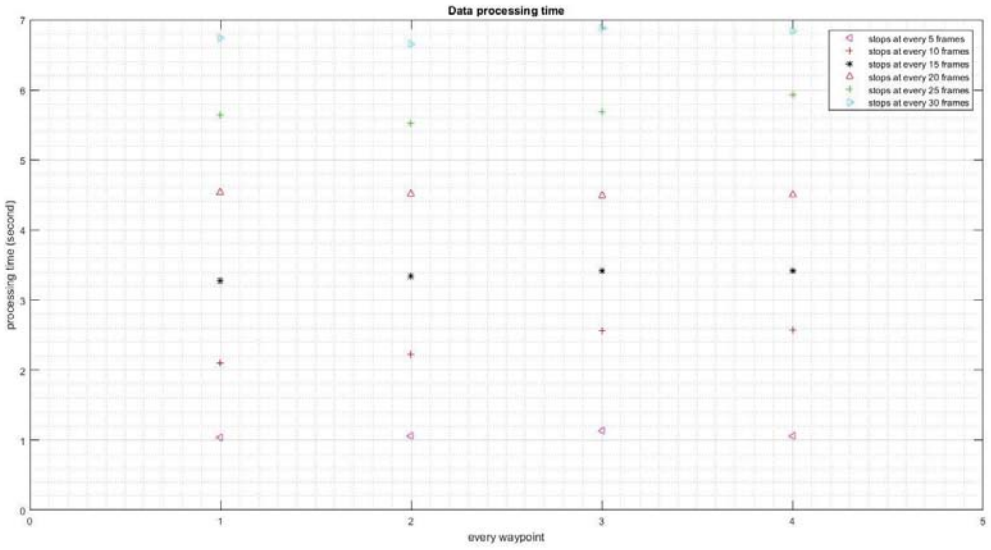


Figure 4- 16: Time comparison for each stop when observing two landmarks

4.5.3 Three Landmarks Analysis

This section discussed experiments conducted on three landmarks.

Similar to the result with one landmark condition, encoder results were closer to ground truth than SLAM estimates when the robot moved a short distance to 0.3m. These results are demonstrated in Figure 4-17. For a distance up to 0.6m, encoders obtained the most accurate result in all conditions (shown in Figure 4- 18). For distances up to 0.9m and 1.2m, SLAM estimates of 30 frames of iteration were as close as the encoder results.

The SLAM method in three landmarks condition did not perform as well as two landmarks' condition. Compared to one landmark' condition, data points received from three landmarks condition are more evenly spread in a reasonable area. But the SLAM with three landmarks did

not receive results as accurate as one landmark condition. Three landmarks also cost 0.5 seconds more for image processing as shown in Figure 4- 20.

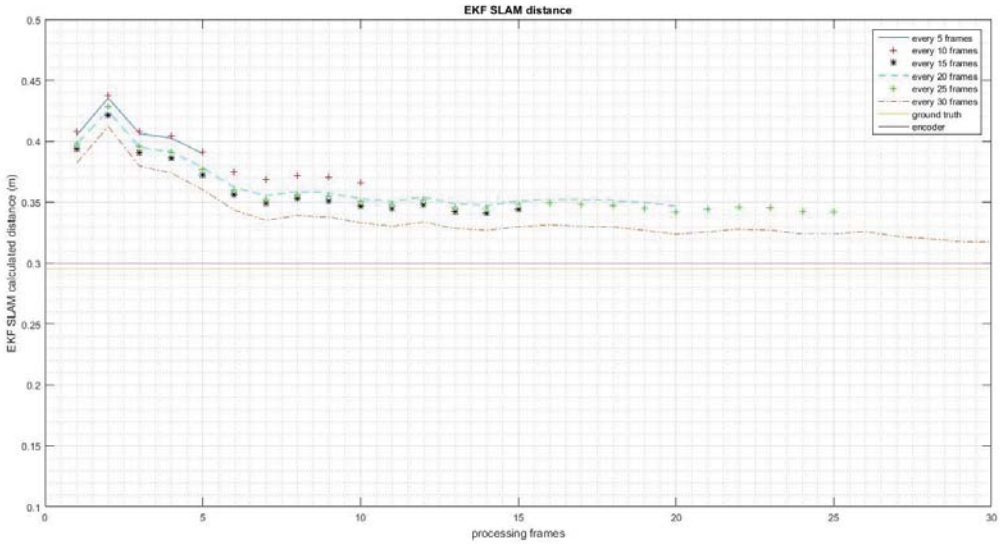


Figure 4- 17: Distance comparison from 0 to 0.3m with three landmarks

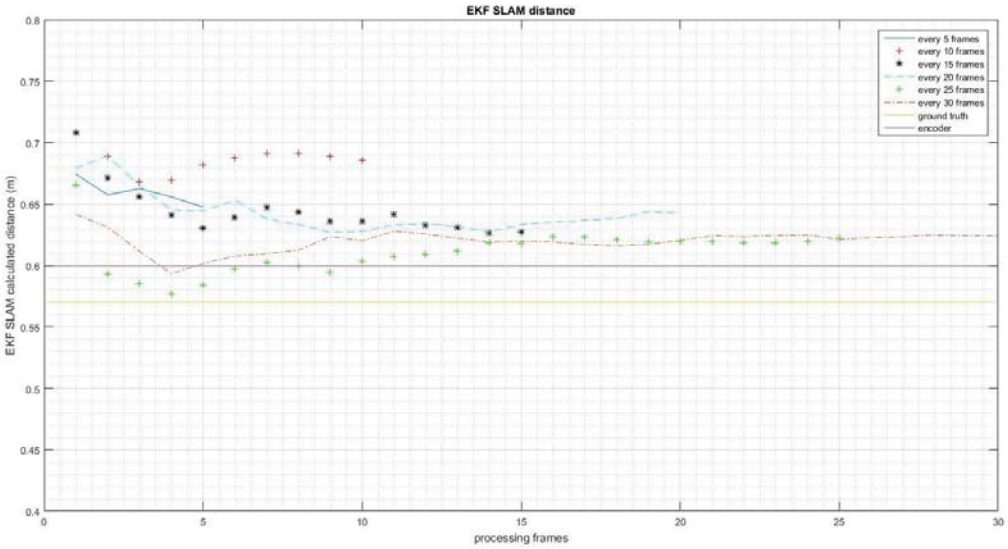


Figure 4- 18: Distance comparison from 0.3m to 0.6m with three landmarks

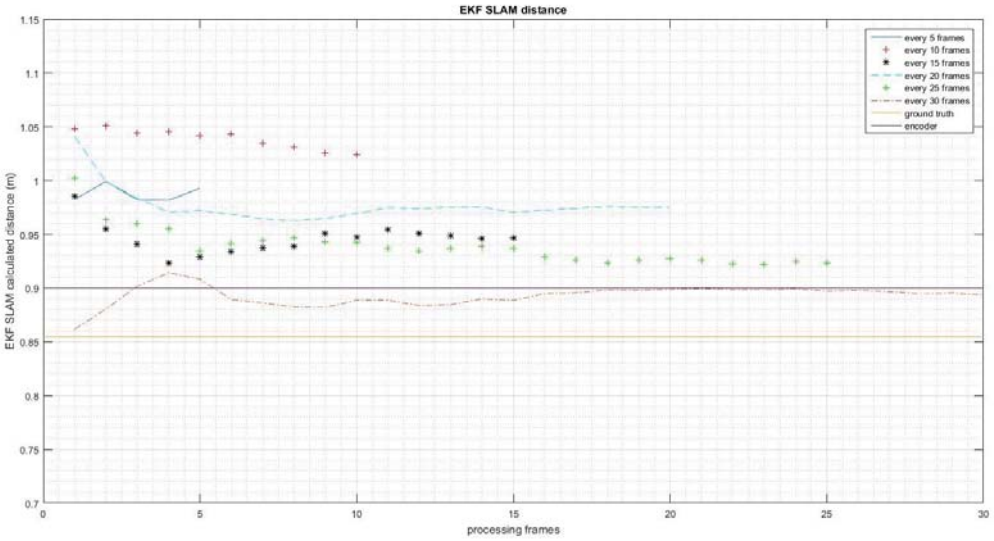


Figure 4- 19: Distance comparison from 0.6m to 0.9m with three landmarks

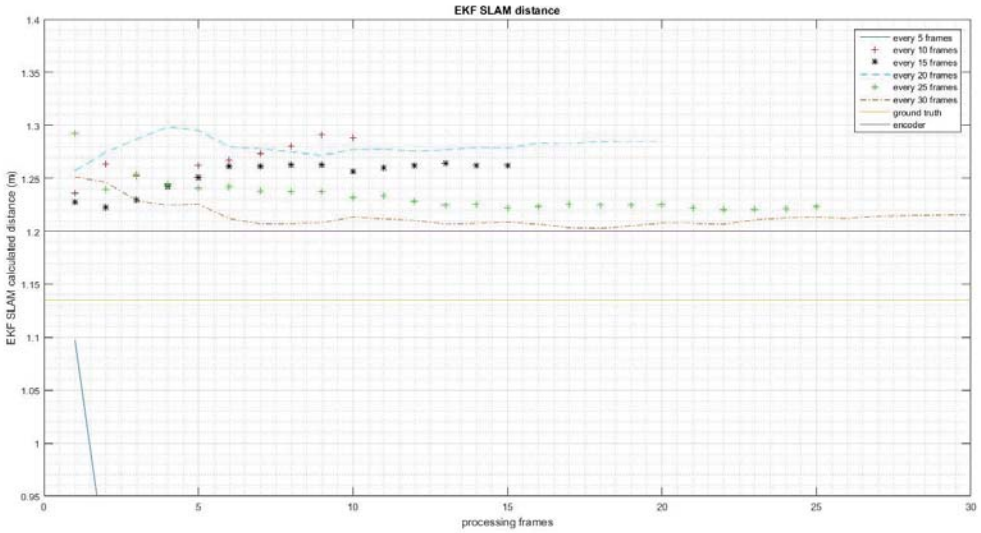


Figure 4- 20: Distance comparison from 0.9m to 1.2m with three landmarks

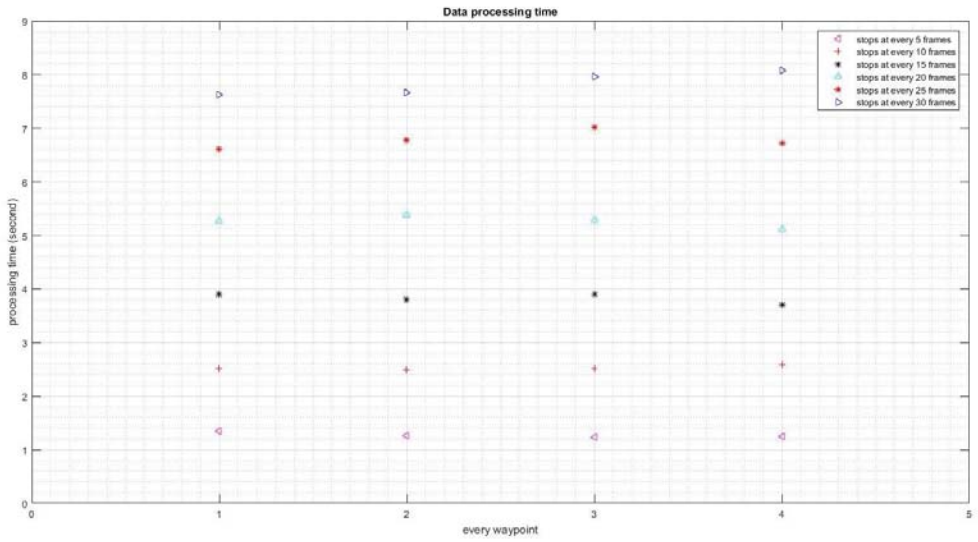


Figure 4- 21: Time comparison for each stop when observing three landmarks

4.5.4 Selected Frames Processing

A frame selection method is proposed in this section. State estimates $X_i^+(k)$ and their corresponding covariance $P_i^+(k)$ from previous experiments in section 4.2.2 are drawn in Figure 4-22. The state estimates were closer to ground truth when the processing frame number grew up to 30. Meanwhile, the factor P_R (see equation (4-6)) of covariance $P_i^+(k)$ decreased to a certain value every time state estimates were closer to ground truth. The value where P_R dropped to around $3.8*10^{-3}$.

The initial value of P_R is determined by measurement noise Q in equation (4-7). An experiment of 30 frames of iteration can be used to test and determine a threshold for P_R.

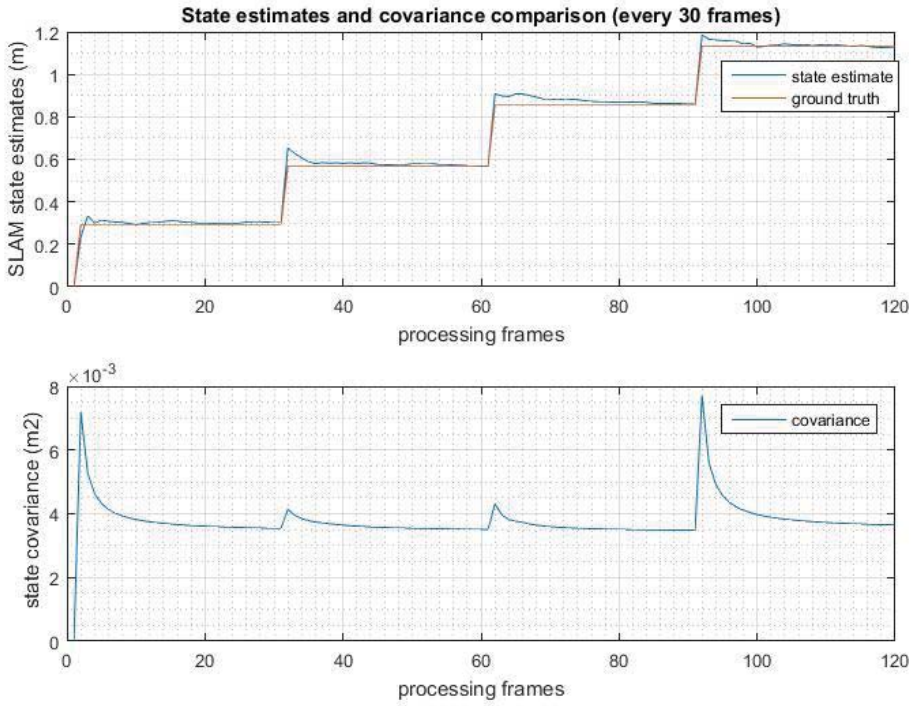


Figure 4- 22: State estimates and covariance (every 30 frames)

The P_R value from 15 frames of iteration is plotted in Figure 4- 23 for comparison with 30 frames of iterations, as state estimates 15 frames of iteration are not very close to ground truth. The covariance factor P_R dropped in a range from $3.8*10^{-3}$ to $4*10^{-3}$. The maximum error between ground truth and state estimates at the 15th iteration is 0.075m. It is assumed that the lower value P_R drops, the more accurate the obtained estimate. Thus, a hypothesis is proposed in this section, if a number of iteration causes the covariance factor P_R below a certain threshold, then an accurate estimate can be obtained at this waypoint.

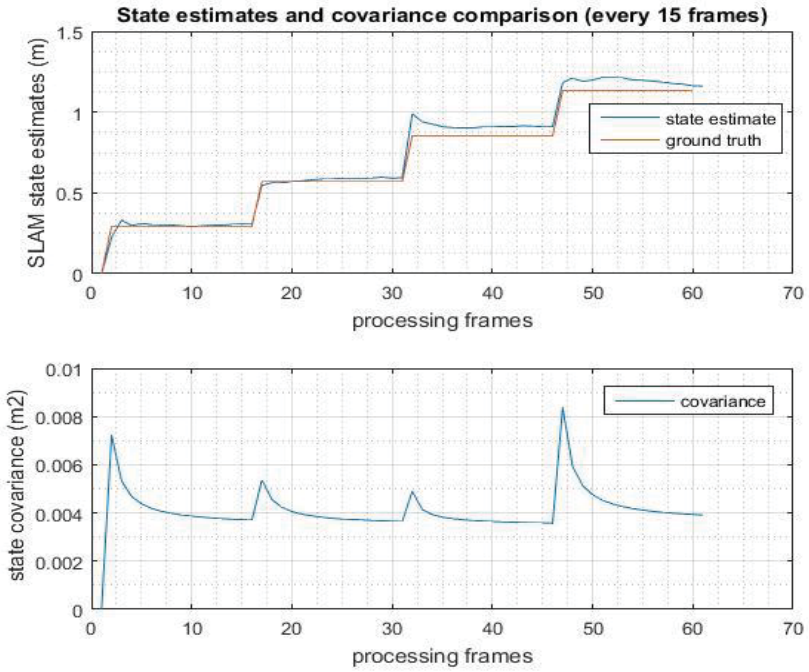


Figure 4- 23: State estimates and covariance (every 15 frames)

A second algorithm is proposed, the threshold of P_R is set to 4*10^{-3}. The computation cycle of the proposed algorithm is as follows:

1) Set waypoints number, initialise X and P.
2) Calculate predicted states for X and P.
3) Observe landmarks and calculate update states for X and P.
4) If P_R is below the threshold 4*10^{-3}, repeat 2) to 4)
5) Obtain final X and P of the current waypoint and move to the next waypoint if the current way point is not the final

Algorithm 2: Visual SLAM with selected times of observation

Define threshold $4*10^{-3}$

Define robot total move waypoints

X, P=setzero();

for k=1 to steps **do**

$X_i^-(k)$, $P_i^-(k)$ from equation(4-7)(4-8)

for $P_{Ri}^-(k)$ <$4*10^{-3}$**do**

$Z_i^-(k)$ from equation (4-19)

$P_i^+(k)X_i^+(k)$from equation(4-22)(4-23)

$i++$

$X_i^-(k)$=$X_{i-1}^+(k)$, $P_i^-(k)$=$P_{i-1}^+(k)$

End for

Robot moves, read from encoders, $k++$, $i=0$

End for

Return X, P

State estimates are demonstrated in Figure 4- 24. State estimates were more accurate than encoder results when the robot moves further than 0.6m. The frame selection method obtained accuracy similar with 20 frames as the errors are shown in Figure 4- 25. With regards to time comparison (in Figure 4- 26), the frame selection method costs a similar time with 10 frames method.

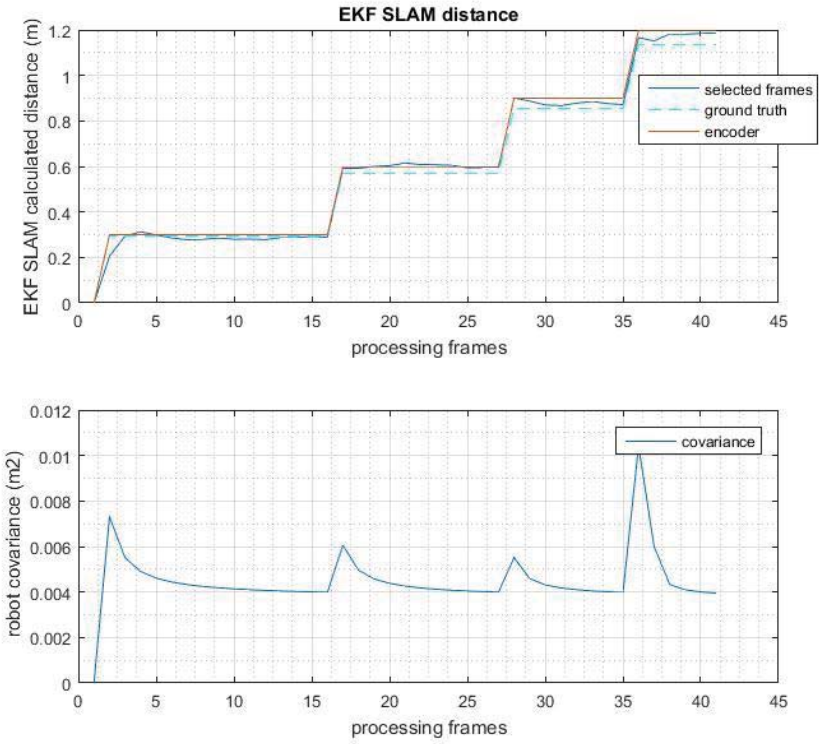


Figure 4- 24: Two landmarks select frame method moving 1.2m

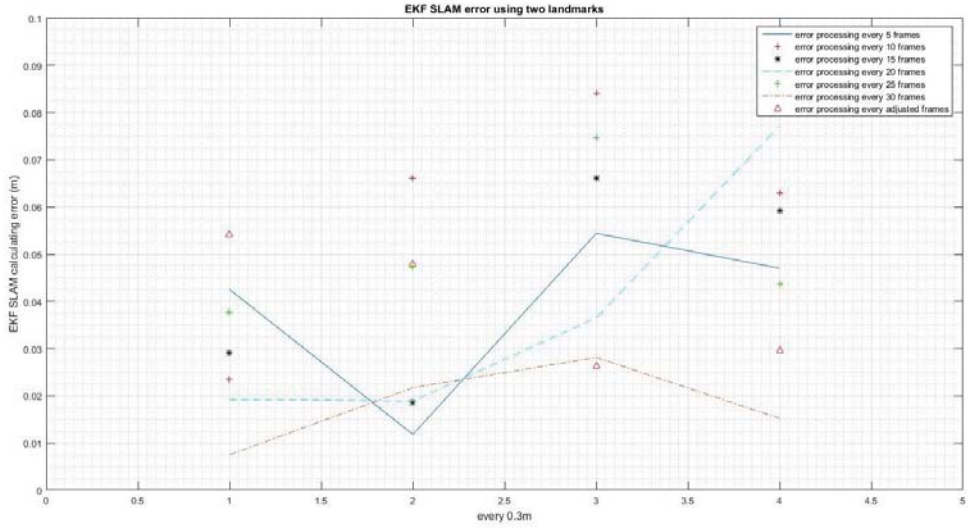


Figure 4- 25: Error comparison

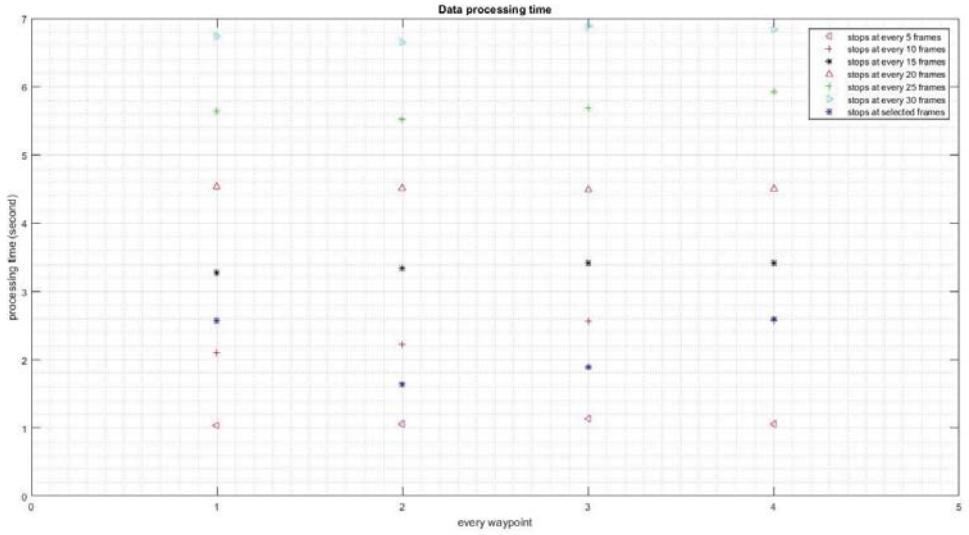


Figure 4- 26: Time costs

Table 4- 1: Comparison among selected frames, 10 frames and 20 frames

	Selected frames	10 frames	20 frames
Accumulated errors	0.1578m	0.2367m	0.1517m
Total time costs	8.709s	7.459s	18.054s

A comparison of output accuracy and time costs among frames selection method, 10 and 20 frames iteration methods is listed in Table 4- 1. The frame selection method maximised time efficiency and output accuracy.

4.6 Discussion

This chapter presented a real-time Visual SLAM method for accurate state estimates in a robot navigation process. The Visual SLAM method made use of an EKF and a RGB-D camera. Compared to a typical laser sensor based EKF SLAM method, the presented system is equipped with higher environmental recognition ability due to an assisted RGB-D camera. The informa-

tive visual data was used for feature identification and data association, which is more efficient than traditional scan matching.

An iterated video frames module was integrated in the EKF. State estimates were closer to ground truth and more accurate when more frames were iterated. This is probably because the EFK parameters were updated after each time of iteration, and these updates improved the linearization during the state and measurement prediction process, and thus errors were minimized. Additionally, the experiment data showed that covariance of the state estimates decreased when more frames were iterated, which agrees with the findings of Ling et al. (2013). Moreover, when the covariance dropped to its lowest point, the state estimates also became closest to ground. This finding was used as a hypothesis to decide if the frame iteration times were enough.

The proposed frame selection method used covariance as a threshold to select an optimal number of frames to iterate. The obtained accuracy was similar to a 20 frames method, and the time costs were nearly as much as the costs of 10 frames method. This selection method can be used in other scenarios where an EKF is used, because this selection method is developed based on the analysis of covariance in the EKF process. The proposed frame selection method is recommended to balance accuracy and computation costs in an EKF process.

The methods used in this study achieved an accurate estimate. The maximum offset from ground truth was 0.02m when two landmarks were used and the iterated frames number was 30. Compared to previous study, Haverinen and Kemppainen (2009) ran their robot through a corridor and mapped with a Monte Carlo Localization for 25m, the maximum error was 0.28m which is 14 times of this study where the maximum error is 0.02m. This error will decrease

when the robot travel distance is as far as 25m, because the EKF parameters renew during every frame iteration period, this estimate will be more accurate.

Another finding in this study is that the encoder could obtain a more accurate result than SLAM methods if slippage dose not accumulate too much (this is when the robot moved to a 0.6m distance in this study). Thus, in a relatively short distance, the encoder can outperform SLAM methods. Besides small slippage, another reason could be that the SLAM methods had not received a set of ideal parameters for the EKF model yet, so the outputs from SLAM were not close enough to ground truth when only a few frames were iterated.

For different number of landmarks, SLAM with one landmark and three landmarks did not perform as well as that in two landmarks' condition. This could be because one landmark does not provide enough information for SLAM estimates. For three landmarks, it could be because these landmarks introduced an outlier and this outlier is used as inlier in EKF, as there is no outlier identification method (i.e. RANSAC) in the used EKF.

4.7 Conclusion

To summarize, iterating frames is an effective method to update EKF parameters. Changing different numbers of iterated video frames can change robot self-localization results in an EKF framework. Selecting frames method can maximise time efficiency and optimise output accuracy, and is recommended for other EFK SLAM methods.

4.8 Contribution

The contribution of this chapter is threefold. First of all, an EKF based Visual SLAM method was developed and extensive experimental tests on the proposed method have been achieved. As a second contribution a relationship between processing frames and estimation accuracy was discovered. As a third contribution, a frame selection strategy was devised, which can achieve accuracy to a certain degree and maximise time efficiency compared to previous fixed number frames methods. The effectiveness of the method was demonstrated.

Chapter 5 A SLAM-O Method for Robot Navigation and Object Detection

This chapter presents a SLAM-O method for robot real-time navigation and object detection. The SLAM-O method integrates object detection into the Visual SLAM framework presented in Chapter 4. The aim of this chapter is to enable a mobile robot to navigate in an indoor environment, localise its position as it moves along, recognizes objects on its way, and put them in a map. Semi-transparent object detection and connected object separation methods for recognizing and localizing objects are developed, which are robust in variations of illumination, viewpoint, objects' material and surface reflections. Through experiments, utilities of the presented SLAM-O system are illustrated; effectiveness of describing a robot's trajectory and object locations is demonstrated. Location accuracy is also compared between a fixed number of frames and selected frames methods.

5.1 Introduction

Visual SLAM is a key method that allows mobile robots to create maps as they explore environments. Conventional SLAM systems typically focus on providing reliable camera localization (Cadena et al., 2016b), but poorly describes the observed scene. This study integrates object detection into SLAM, which provides a vital component in a robot's repertoire of skills. Compared to traditional object recognition methods, which function on a frame-by-frame basis (Ta

et al., 2009), the developed SLAM-O method enables object detection and tracking from multiple view-points when a robot moves around.

To approach invariant object detection during a robot navigation process, feature detection methods should be further developed for application-specific scenes. Most of the existing methods, such as colour threshold, Hough line and angle detection, etc., seem to work well under certain limited image condition; however, variations in illumination, orientation, noise, scale, viewpoint and elements' material reflection. will bring challenges to achieve object detection. Within the VEX game, there are two issues that normal image processing methods cannot solved: one is to identify semi-transparent coloured balls, and the other one is to separate connected buckyballs.

For semi-transparent coloured objects, there are light reflections on transparent balls' surfaces, so that colour-threshold cannot extract all the ball area because the colour pixels are not evenly distributed; also, a normal trained classifier easily mistakes the reflections on the transparent balls as separate targets. One reason for this is that the profile of a ball is too simple or not unique enough to distinguish balls from other similar round objects. Another cause is the nature of semi-transparent object, which is made of thin material and reveals the background. Other approaches, such as segmentation blob clustering and illumination invariance require a substantial amount of computational and/or memory resources (Finlayson et al., 2001). However, mobile robotic systems typically have strict constraints on the computational and memory resources available, but still demand real-time processing. This chapter presents a K-Means clustering method which classifies the semi-transparent ball by accumulating the various colour tints.

For object separation, segmentation methods are widely used. Active contour method is greatly used to obtain contours of the object of interest. The active contour is an energy-minimizing spline influenced by external constraint and images forces that pull it toward the contours(Trier & Jain, 1995). This method is not suitable for shape detection, as it gets one smooth regular shaped contour losing the feature of peak points, and thus is unusable in analysing a polygon's shape. The watershed transformation is another segmentation method, which based on topological gradient (Sezgin, 2004). This approach does not work well when two objects overlap. The results of the Watershed method will be discussed in this chapter. A concave based method for separation is proposed, the experimental demonstration outperforms the general Watershed method.

The following sections present the SLAM-O framework, and describe how to implement a robot's self-localization and object detection in a Visual SLAM system, and discuss how to detect coloured semi-transparent and connected objects.

5.2 SLAM-O Method Framework

In this section, the SLAM-O method is discussed. The SLAM-O method is developed based on the Visual SLAM method, which was discussed previously in Chapter 4.

5.2.1 Object Localization Method in EKF-SLAM

Feature detection for object localization is integrated in the SLAM-O framework. A flowchart for the framework is shown in Figure 5- 1. The detected objects are separated, and then located according to robot's self-localization. Objects' locations are associated with previous saved objects to achieve object tracking.

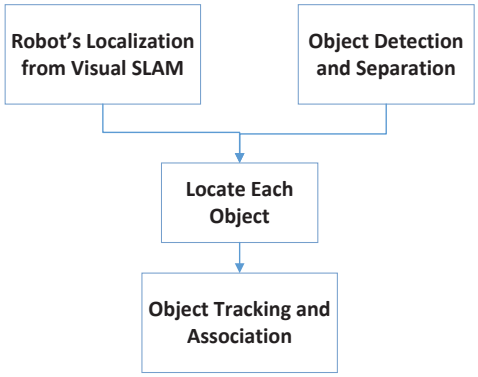


Figure 5- 1: SLAM-O framework

The objects' localization is saved in a stacked state vector X_o, and describes as $X_o = [o_1^t, o_2^t, \dots o_n^t]^T$. A newly observed object vector will be initialized as a new vector, and already saved vectors are used in data association process for object tracking. Data association for newly observed and already saved objects is based on nearest neighbourhood method (Andoni & Indyk, 2006). For an object $o = (x_o, y_o)^t$ in a robot's reference coordinate, the observation model is as follows:

$$\begin{pmatrix} r_o \\ \emptyset_o \end{pmatrix} = \begin{pmatrix} \sqrt{(x_o - x_v)^2 + (y_o - y_v)^2} \\ arctan\left(\frac{y_o - y_v}{x_o - x_v}\right) - \theta_v \end{pmatrix} \quad (5\text{-}1)$$

Where, r_o and $\emptyset_o$ are relative distance and angle from the object to the observing camera respectively.

5.2.2 K-Means Based Objects Detection

In this section an objects separation method is discussed.

A general approach for colour detection is to capture a RGB image, convert it into the Hue-Saturation-Value (HSV) colour space, and threshold for a specific hue value, resulting in a binary image, where only the desired hue is evident.

The objects are the field elements from the VRC Toss Up competition: a small solid coloured ball and large transparent coloured ball (see Figure 5- 2). The general approach would work well for the small solid ball; but not for the large transparent ball due to the hue value being a combination of the foreground and background colours; widening the corresponding hue range necessary to threshold. Furthermore, the nature of the transparent material, is reflective therefore results in low saturation spots.

The proposed approach is to accumulate the various colour hues, and classify the large transparent balls as separate objects. The approach utilises the K-Means clustering method, which finds partitions such that objects within each cluster are as close to each other as possible and as far from objects in other clusters as possible. For example, given a set of observation ($m_1, m_2, m_3 \dots m_n$), where each observation is d-dimensional real vector, K-Means clustering aims to partition the n observations into k sets (k ≤ n), as to minimize the within-cluster sum of squares:

$$\arg\min \sum_{i=1}^{k} \sum_{m_j \in S_i} \left\| m_j - \mu_i \right\|^2 \tag{5-2}$$

Here μ_i is the mean of points in S_i.

(a)Small solid colour ball (b) Large semi-transparent coloured ball

Figure 5- 2: VRC Toss Up field elements.

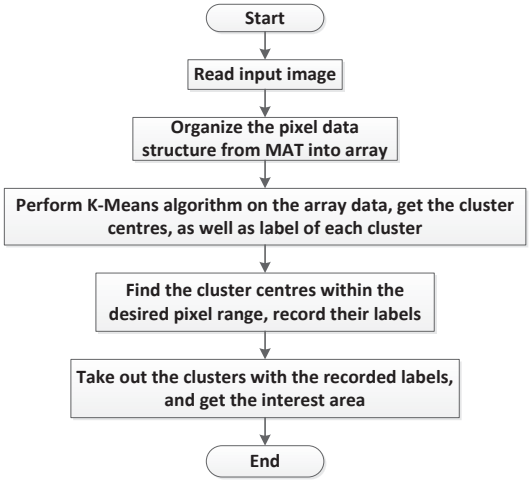


Figure 5- 3: Flowchart of K-Means procedures

The OpenCV library is used to implement the K-Means method for colour detection. OpenCV uses the MAT data type to store images. When processing an RGB image, there are three channels, i.e. one channel per colour. In order to use OpenCV's K-Means function, Mat data is first converted from three channels into one array. And then K-Means is performed on the array, several clusters' centres as well as their labels are obtained. Thus, the interested colour area is obtained by collecting pixels with the same label.

The Figure 5- 3 provides an overview of implementation in C++ using OpenCV's library:

5.2.3 Connected-Object Separation

This section describes the connected-object separation method.

A general approach for connected-object separation is to use a Watershed method, where an image is thresholded for a specific hue and then a morphological filter is applied to erode/dilate the image, resulting in two close objects being separated.

Two buckyballs are used to test connected-object separation. The general the Watershed method would work well when the view angle is nearly vertical; however, this situation is quite rare, because VRC field elements keep moving during a match. The Watershed method would not work very well for most cases, as shown in Figure 5- 4, where the balls overlap due to proximity and angle of the view.

The proposed approach is to use a defect as a clue for object separation. A defect is the point on concave hull, and the defects can be derived by thresh-holding the distance from concave hull to convex, which is smallest region enclosing the given polygon. And separation can be done by connecting the defect pair. The following flowchart (see Figure 5- 5) provides an overview of the proposed method, each contour is checked for defects. A flag is set for multi-object separation, and it is used to decide whether it needs another separation. If the flag equals to 0, it is the last instance of separation; otherwise another defects finding loop is performed through the process again.

(a) Original Image (b) Red Detection (c) Distance Transforms (d) Watershed Separation

Figure 5- 4: Watershed separations transform

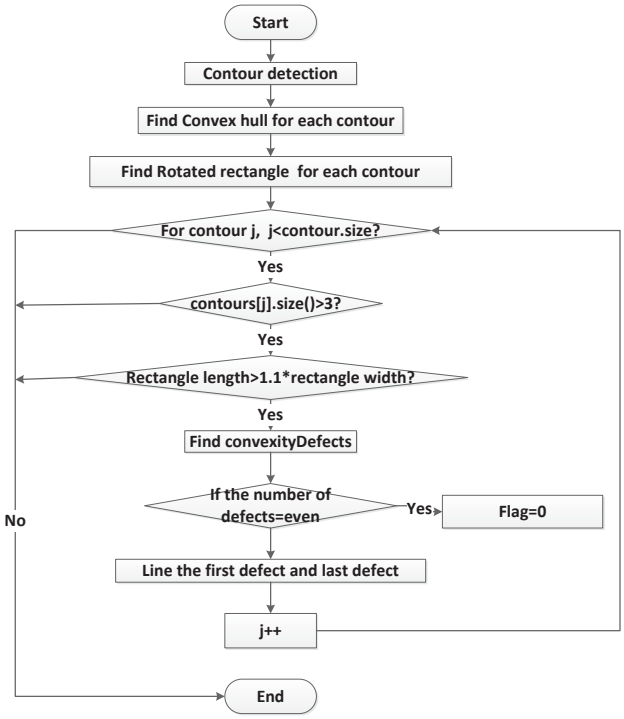


Figure 5- 5: Flowchart of object separation procedure

(a) Find defect pair in occluding condition (b) Twice separation for three connected objects

Figure 5- 6: Multiple connected-objects separation

Rules are also defined for defect pair selection and locating missing defects (created by occlusion). The first is that the defect pairs should be the first point and the last point of the array, or the second point and the second to last point of the array. As defect points are stored in a sequence of array, starting from the lower left travelling anti clockwise to end at the upper left. The rule makes sure the defects are from the same objects. The other rule is that the direction of the line of the defect should be perpendicular to the ground. For the direction choice, a rectangle is drawn enclosing the target region, and the longer side of the rectangle will contain the orientation to the ground. This is followed up with dot production to check if either direction is vertically aligned.

Multiple connected-object separation is demonstrated. Figure 5- 6 (a) illustrates two red balls and one blue ball in contact, which is a plausible possibility in reality. After detecting the area of interest, the contact area with blue ball left an additional defect, and the proposed method correctly selected the defect pair. In Figure 5- 6 (b), there are three balls, as only three defects were found in the first separation loop; a second separation procedure is performed. And every ball is identified after the second operation.

5.3 SLAM-O Method Implementation

5.3.1 Play-ground Description

This section discusses how SLAM-O method is implemented.

The playground is in an indoor environment of 3m length and 2m width, as is shown in Figure 5- 7. The designed robot starts to move from the original point $(0,0)$, and moves along the x axis to $(1.5,0)$. During the movement, the robot stops every 3m and observes the

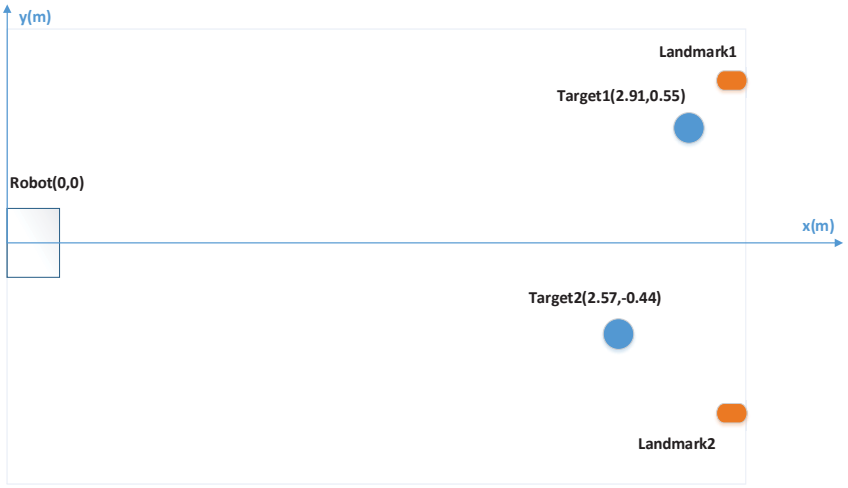


Figure 5- 7: Play-ground designed

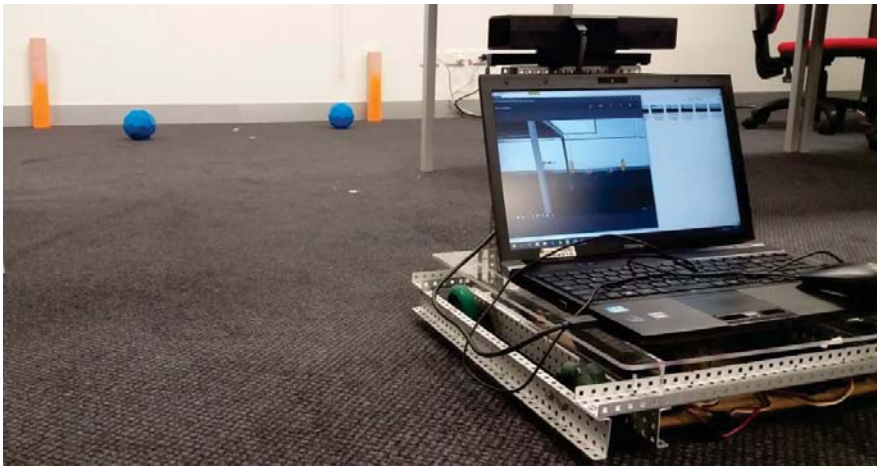


Figure 5- 8: Play-ground example of SLAM-O method

environment. The robot will observe two landmarks for self-localization and targeting for object detection.

A real-time Visual SLAM system on a mobile robot is demonstrated in Figure 5- 8. Two landmarks are placed in front of the robot. Two blue buckballs are spread in the field as target objects.

5.3.2 SLAM-O Method Implementation

There are three methods designed for SLAM-O method implementation. The first implementation is based on a 20 iterated frames processing scheme. As discussed in Chapter 4, the robot moves 0.3m and observes landmarks for 20 times to update EKF parameters. Object detection is implemented every time after EKF updates. The second implementation is based on Gaussian Mixture Models (GMM) frame selection method. This is different from the first implementation, where the GMM method selects frame number. Object detection also performs after EKF updates. The third implementation is a covariance based frame selection. The frame number is determined by threshold the covariance value. Object detection is only implemented when the frame number is determined. Thus, for each 0.3m movement, target detection only performs once. For these three implementations, target detection is performed by both colour threshold and K-Means clustering.

5.4 Results

5.4.1 Mapping of SLAM-O Method with 20 Iterated Frames

With 20 frames iterated, self-localization and object detection results are demonstrated in Figure 5- 9. SLAM estimates converged around ground truth data at the first three waypoints. The fourth waypoint estimate was 0.05m offset from ground truth. Object detection results were closer to the ground truth when the robot moved closer to the object. This may be because the camera performed better at a distance from the landmarks around 2m compared to 3m. Figure 5- 10 depicts the estimated position of target objects with ground truth of (2.91,0.55) and (2.57,-0.44), along with the variance of each estimate.

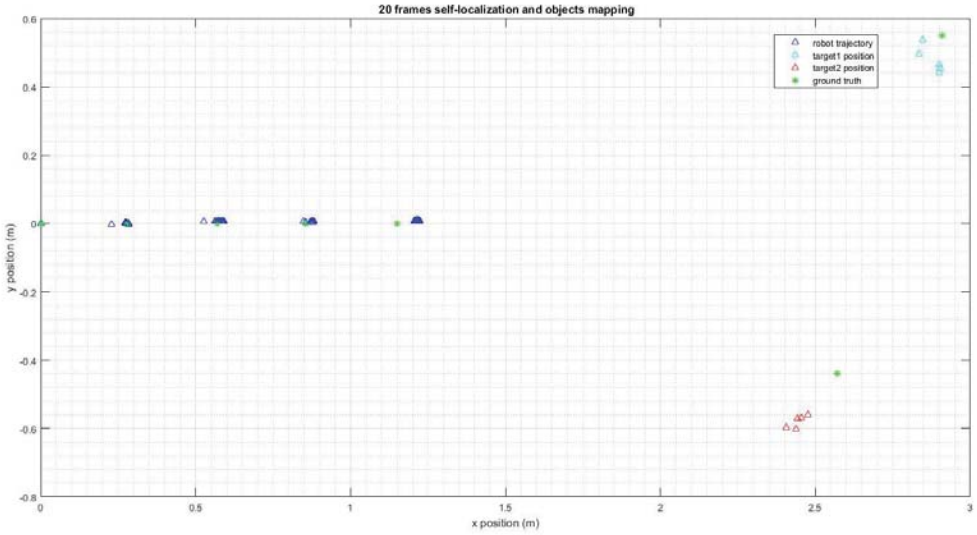


Figure 5- 9: Mapping with iterated 20 frames

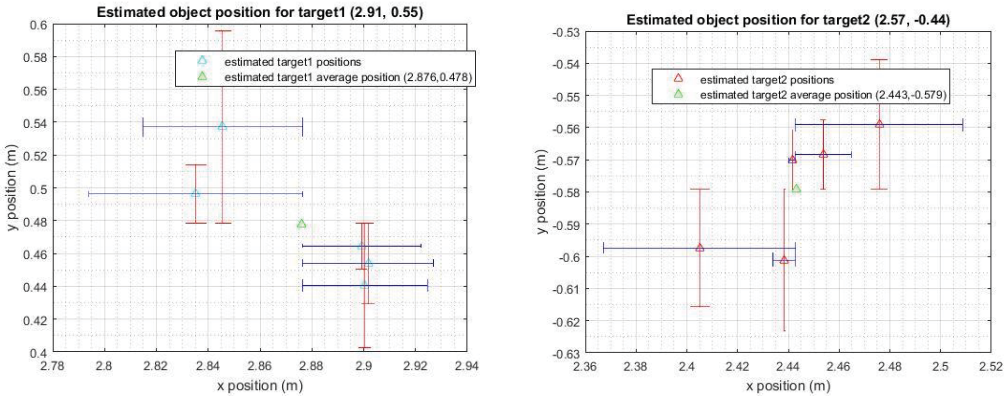


Figure 5- 10: Object variance with 20 frames iteration

Measurements of target1 were more accurate than target2: the average measurement of target1 was 0.04m to its ground truth in x direction and 0.08m in y direction; and the average measurement of target2 was 0.13m to its ground truth in x direction and 0.14m in y direction. Also, Average measurements in x direction were more accurate than those in y direction. For

variance of the measurements, the largest offset was 0.05m, and the offset became smaller when the robot moves closer to the objects.

The K-Means method is also used for object detection in the process of 20 frames based self-localization. K-Means method based object detection did not observe the targets when the robot was more than 2m away. This is because the clustered target points were small when they were relatively far, and thus were accidently erased by an erosion process, which was used for noise clearance. The SLAM estimates were converged within 0.05m around ground truth in Figure 5- 11. However, targets' localization results do not show obvious advantage in terms of position estimates. Figure 5- 12 depicts the estimated position of target objects with ground truth of (2.91, 0.55) and (2.57,-0.44): the average measurement of target1 was 0.04m to its ground truth in x direction and 0.14m in y direction; average measurement of target2 was 0.1m to its ground truth in x direction and 0.18m in y direction. Variance of the measurements had a smaller offset, which was within 0.02m.

To summarize, the 20 frames iteration method can give accurate state estimates for robot self-localization, with a maximum offset of 0.05m. Object measurements were less accurate compared to the robot localization, especially in y direction, which could be caused by a nearly 0.1m gap between the colour camera an IR cameras. The K-Means method improved boundary's detection which can be used for object separation, as discussed in section 5.2.3; however, when the objects are separately distributed, threshold is preferred for colour detection.

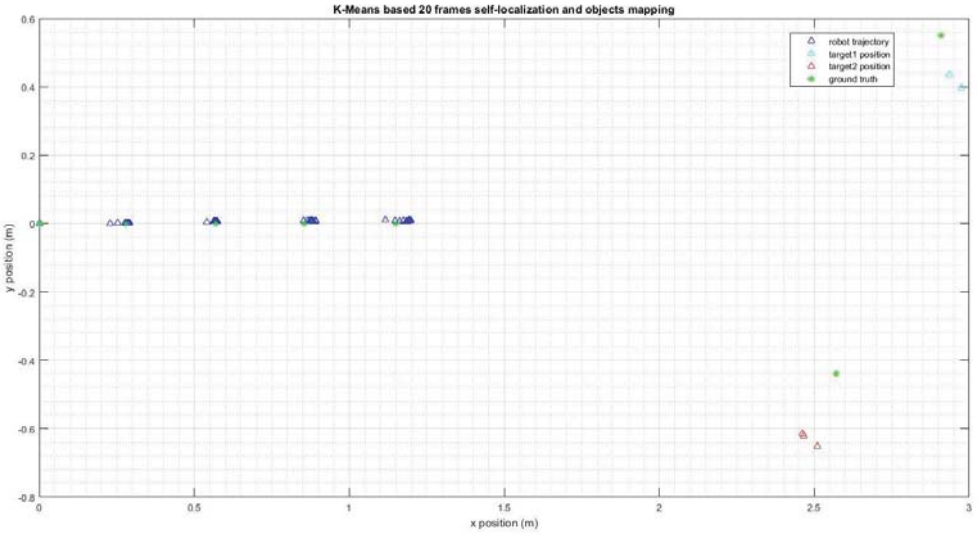


Figure 5- 11: K-Means based mapping with iterated 20 frames

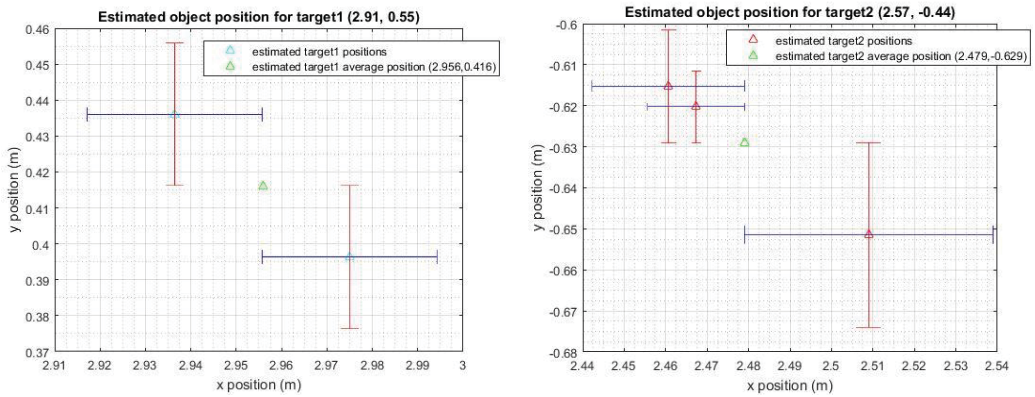


Figure 5- 12: Object variance using K-Means method and with iterated 20 frames

5.4.2 Mapping using Covariance based Frame Selection Method

The covariance based frame selection (as discussed in section 4.4.3) is incorporated with the object detection method in this section. Figure 5- 13 demonstrates the robot trajectory and object mapping results. The iterated frame numbers for each stop were 14, 9, 15 and 15. The best estimate was achieved when current covariance dropped to the threshold value. The

maximum offset from the ground truth was 0.02m. Figure 5- 14 depicts the estimated position of target objects with ground truth of (2.91, 0.55) and (2.57,-0.44), along with the variance of each estimate. The average measurement of target1 was much more accurate than target2: for target1, it was 0.03m error in x direction and 0.3m in y direction; for target2, it was 0.2m in x direction and 0.23m in y direction. But the target measurements had a small variance which is 0.01m; the variance for the target1 was up to 0.04m.

K-Means was also used for object detection in this section. Figure 5- 15 depicts the self-localization and object detection results. Frame numbers for each stop were 21, 17, 16 and 14. For self-localization, the maximum offset from the ground truth was 0.1m. Figure 5- 16 depicts the estimated position of target objects: average measurement of target1 was 0.13m to its ground truth in x direction and 0.24m in y direction; average measurement of target2 was 0.34m to its ground truth in x direction and 0.18m in y direction. Variance of the measurements had a smaller offset, which was within 0.03m.

To summarize, covariance based frames selection method can obtain more accurate state estimates than 20 frames method. Object measurement in both methods need improvements.

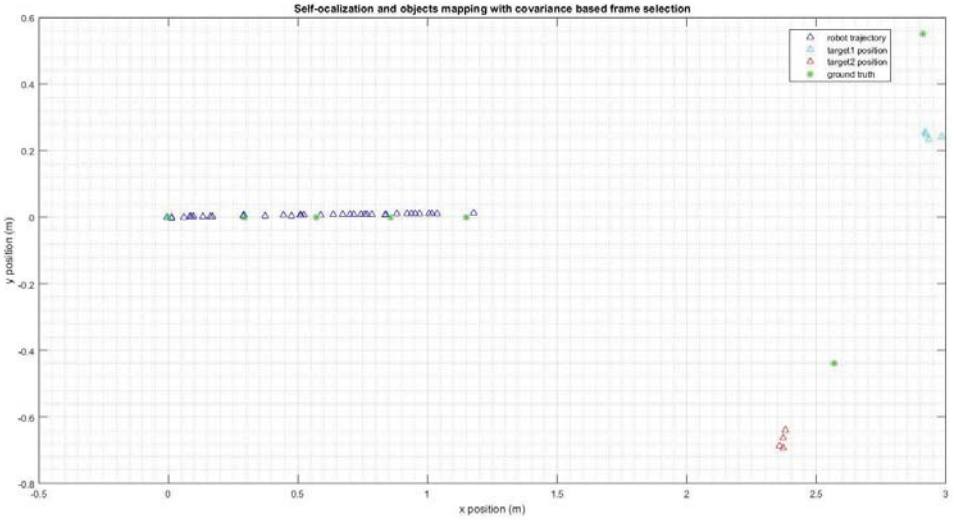


Figure 5- 13: Mapping with covariance frame selection

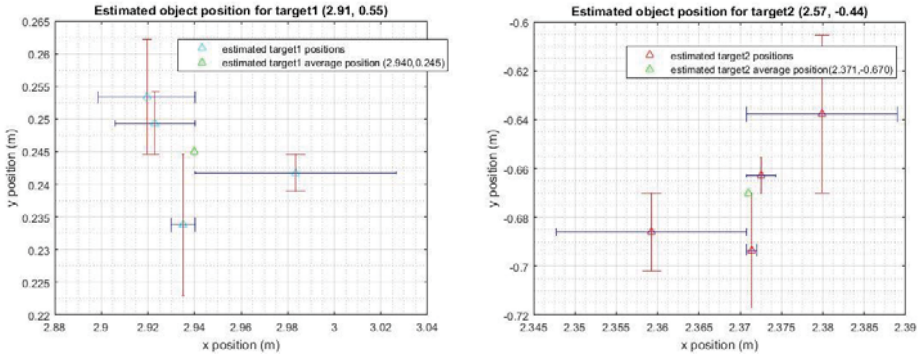


Figure 5- 14: Target1 and target2's position variance

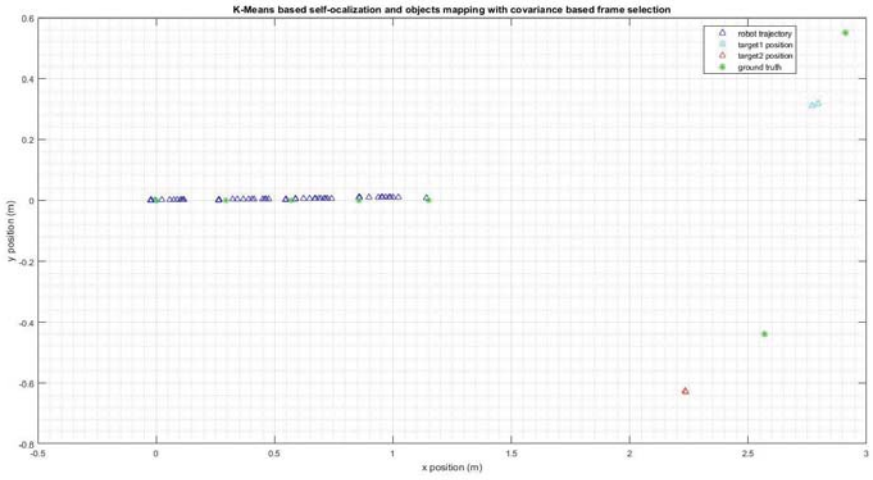


Figure 5- 15: Mapping with covariance frame selection and K-Means object detection

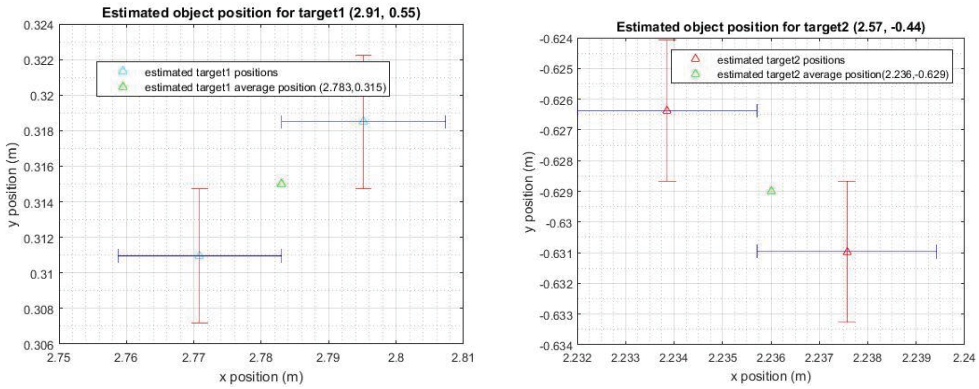


Figure 5- 16: Object1 and object2's position variance with K-Means object detection

5.4.3 Mapping using GMM Frame Selection Method

Based on the observation of 20 frames method and covariance based frame selection methods, it is found that SLAM estimates become accurate when a number of estimates cluster in a range. According to the above analysis, a GMM based frame selection strategy is proposed in this section. The method is described as: state vector x is assumed to converge when $X^{+}(k)$ falls into a reasonable range around the mean value of current GMM. A flow chart for GMM

based frame selection method is demonstrated in Figure 5- 17. For each waypoint, a Gaussian distribution is initialized at the first frame. The initial mean value of the Gaussian is the current state estimate. After obtaining a new states estimate from the following frame, the Gaussian parameters are updated. If the state estimate is within a threshold of current Gaussian mean, and the current processed frame number is greater than 5, the state estimate is considered as converged. Otherwise, another frame is processed and Gaussian parameters calculation is repeated. Frame number of 5 is used because of dead reckoning. Encoders will obtain an accurate result in the first few step of movement; however, dead reckoning occurs when the robot moves a further distance. Thus, the selected frame numbers should be at least 5. If one Gaussian is converged, a new Gaussian will be initialized when the robot moves to a new waypoint and processes a new frame.

Robot self-localization and object detection results using GMM are demonstrated in Figure 5-18. In the experiment, 20 frames were set as the top number of frames which can be iterated. There were four waypoints and therefore four times the frame number selection. Frame numbers were 20, 13, 20 and 20. SLAM estimates converged and the each GMM centre had around 0.02m offset from ground truth. Experiment results had the similar accuracy of 20 frames method. Object detection results are illustrated in Figure 5- 19. The average measurement of target1 was 0.02m to its ground truth in x direction and 0.11m in y direction; and the average measurement of target2 was 0.2m to its ground truth in x direction and 0.1m in y direction.

The K-Means method is also incorporated in GMM frames selection for object. As discussed in 20 frames method, the K-Means method did not improve the accuracy of solid coloured ball localization. The 5 frames constraint is removed to check if the selection method maintains state estimates accuracy. In Figure 5- 20, the frame number selection results are 6, 1, 1, and 2.

The state estimates were up to 0.1m away from ground truth, which proved that the 5 frames constraint is a necessary constraint for GMM method.

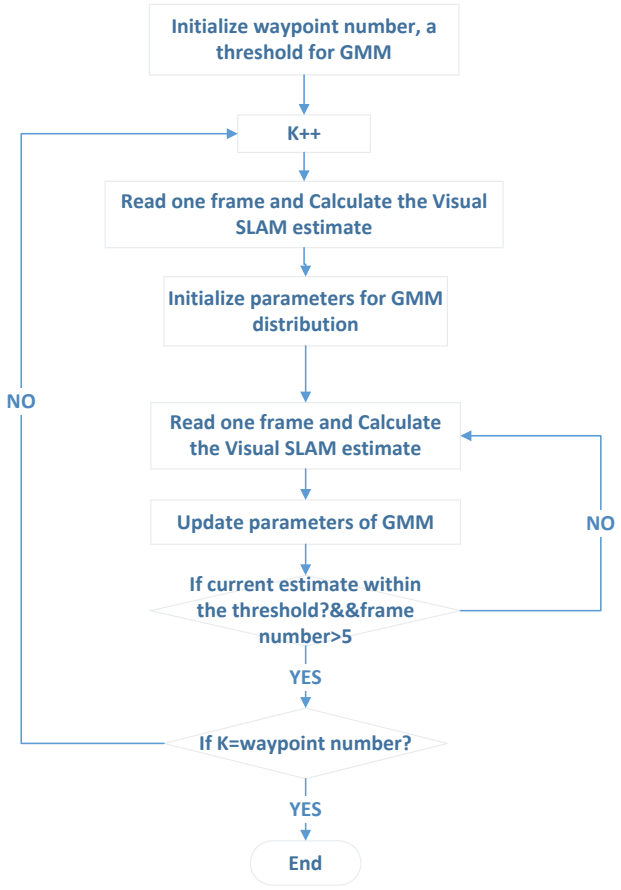


Figure 5- 17: GMM based frame selection method

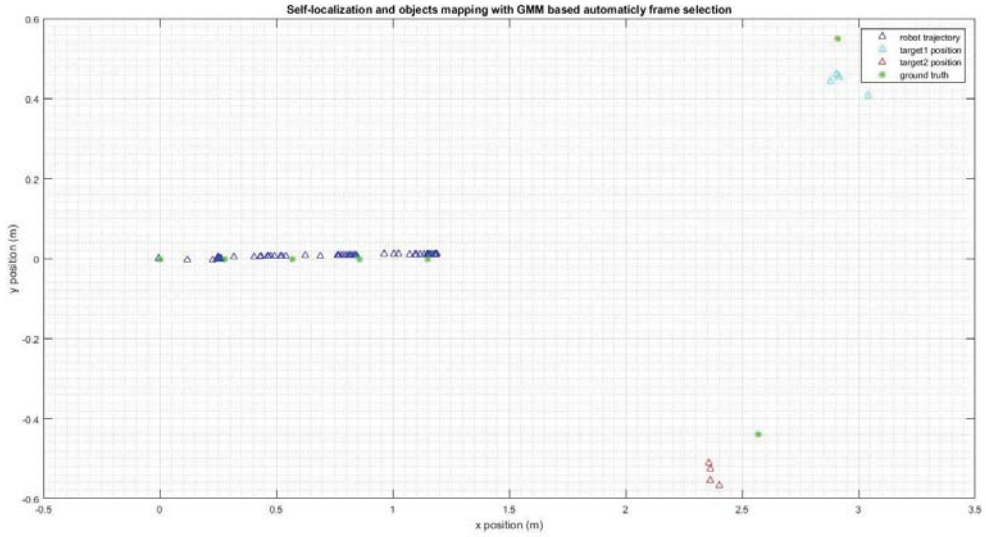


Figure 5- 18: Mapping with GMM frame selection

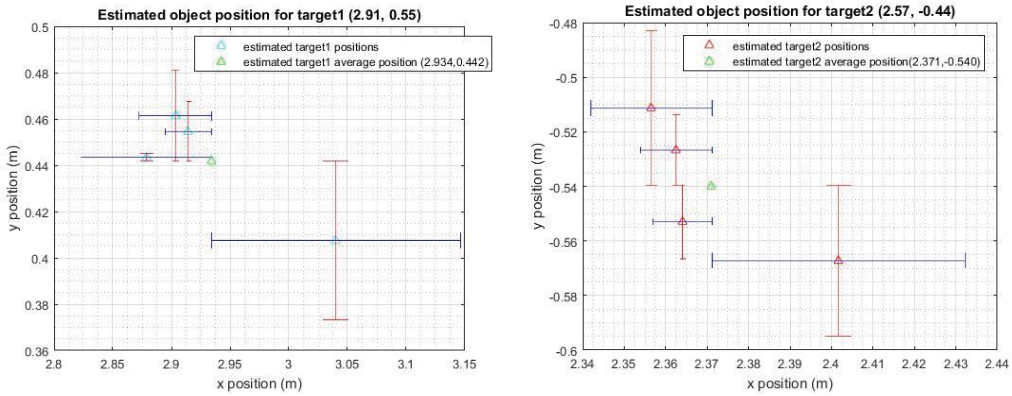


Figure 5- 19: Objects variance with GMM frame selection

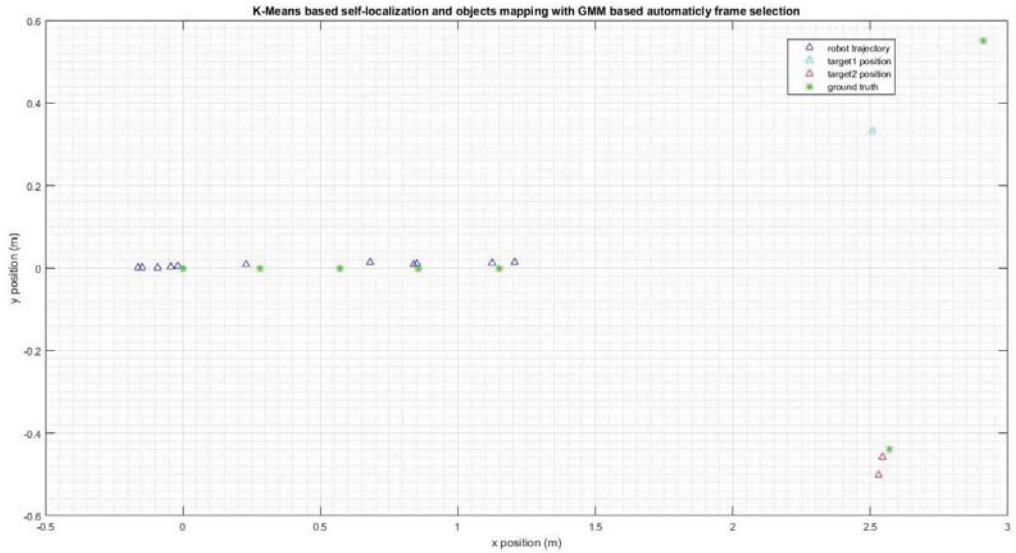


Figure 5- 20: K-Means based mapping with GMM frame selection

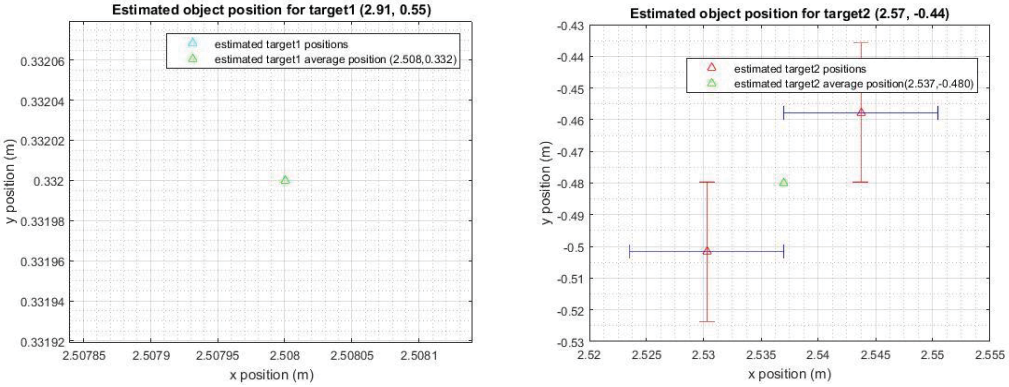


Figure 5- 21: Object variance using K-Means method and with GMM frame selection

Figure 5- 21 depicts the estimated position of target objects: average measurement of target1 was 0.4m to its ground truth in x direction and 0.22m in y direction; average measurement of target2 was 0.04m to its ground truth in x direction and 0.04m in y direction.

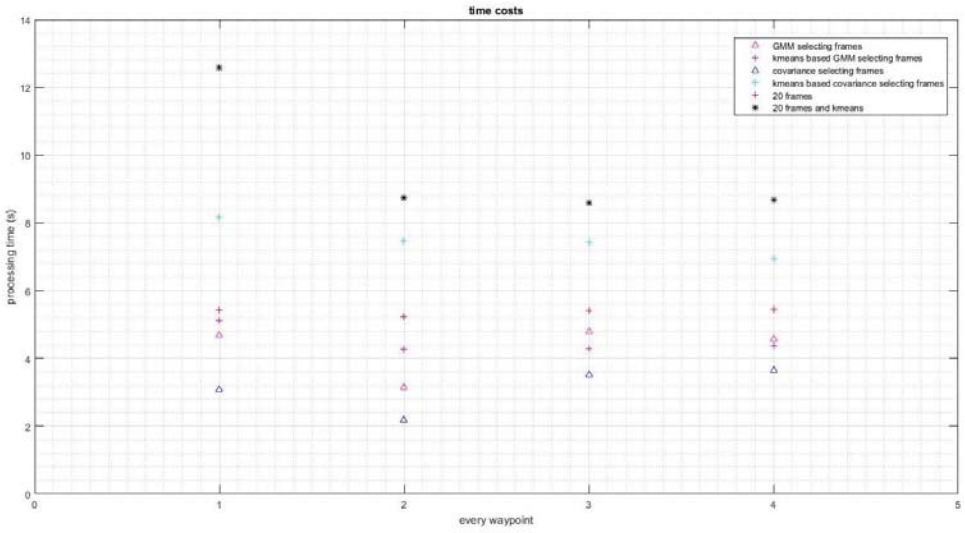


Figure 5- 22: Time costs

To summarize, GMM based method can obtain accurate results when a 5 frame constraint is added. Objects on the y positive side of the camera are more likely to obtain accurate estimates.

The time costs of six mapping experiments in section 5.4.1, section 5.4.2 and section 5.4.3 are compared in Figure 5-21. K-Mean based methods cost more time in image processing. The K-Means and 20 frames is the most time costly combination, which is almost three times the least time intensive method (covariance based frame selection). Normally, K-Means method brings as twice time cost as its corresponding method without K-Means clustering. ALL of these selection methods (20 frames, GMM and covariance) cost less than 5 seconds for each waypoint stop.

5.4.4 Colour Identification and Connected-Object Separation methods

This section presents the results of colour identification and connected-object separation methods. Objects that used to demonstrate the method are the field elements from the VRC Toss Up season: a solid coloured ball and a transparent coloured ball.

a. Solid and Transparent Object Colour Identification

The first case is solid and semi-transparent objects colour detection. In Figure 5- 22, the pixels on semi-transparent ball were grouped together, after the pixels in red range were taken out, shapes could be identified.

b. Connected-Object Separation

The second case is object separation; two overlapped balls which cannot be separated by general method in In Figure 5- 23 , by finding the defects in In Figure 5- 23(b), those two balls were successfully separated.

c. Multiple Connected-Object Separation

The third case is multiple connected-object separation. This is demonstrated in Figure 5- 24 (a) where two red balls and one blue ball are in contact (which is a plausible possibility in reality). After detecting the area of interest, the contact area with blue ball left an addition defect, and the proposed method correctly selected the defect pair. In Figure 5- 24 (b), there were three balls, as only three defects were found in the first separation loop; a second separation procedure was needed. And every ball was identified after the second operation.

Figure 5- 25 shows the combination of our colour detection and separation methods, experimenting on solid balls and semi-transparent balls, the results were successful and shows the robustness of this method.

d. Objects in Noisy Environment

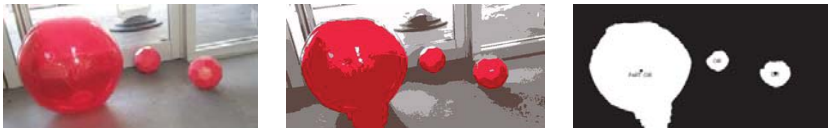
The last case is objects in a noisy environment. In In Figure 5- 26, there were balls of different sizes and shape in the court. The feature used in detection is colour and shape. In Figure 5- 27 (c) (three balls with no background), shape detection was performed which recognized the circular shape and ignored the blue mat.

5.4.5 Object Tracking in SLAM-O

Object tracking is demonstrated in this section. As the robot moved forward 0.3m each time, objects were identified and labelled after every movement. The nearest neighbourhood method was used for tracking. Object positions were calculated by the SLAM-O method, and those positions were association with the nearest previous positions. The tracking results are demonstrated in Figure 5- 28. After a movement step, objects were identified and tracked successfully.

5.4.6 Object Separation in SLAM-O

The proposed connected objects separation method is used in SLAM-O method. The separation results are illustrated in Figure 5- 30. The connected objects cannot be detected in contrast to two objects discussed in section 5.2.3. When the objects were relatively close as demonstrated in Figure 5- 29, the proposed method successfully separated the objects.



(a) Original image (b) K-Means clustering (c) Object detection

Figure 5- 23: Solid and semi-transparent objects detection

(a) Original image (b) Defects detection (c) Separation (d) Shape detection

Figure 5- 24: Connected-object separation

(a) Find defect pair in occluding condition (b) Twice separation for three connected objects

Figure 5- 25: Multiple connected-object separation

Figure 5- 26: Five balls and semi-transparent balls separation results

(a) Original image (b) K-Means feature selection (c) Object shape detection

Figure 5- 27: Object detection in noisy environments

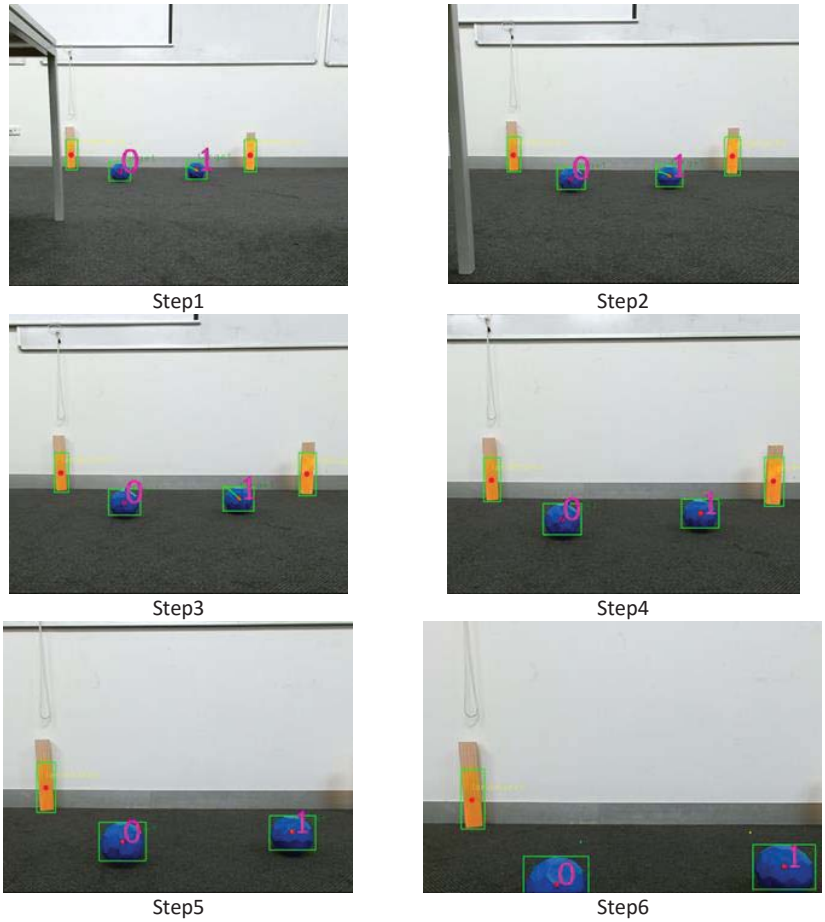


Figure 5- 28: Objects tracking

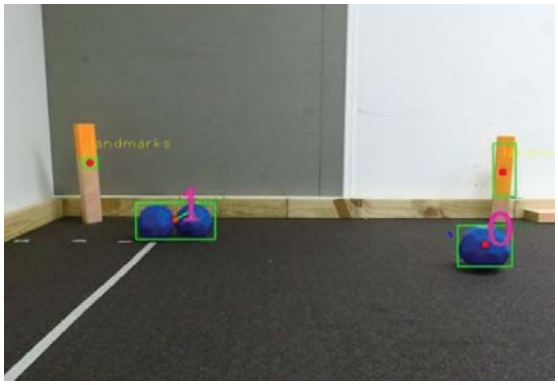


Figure 5- 29: Object detection when two objects connected

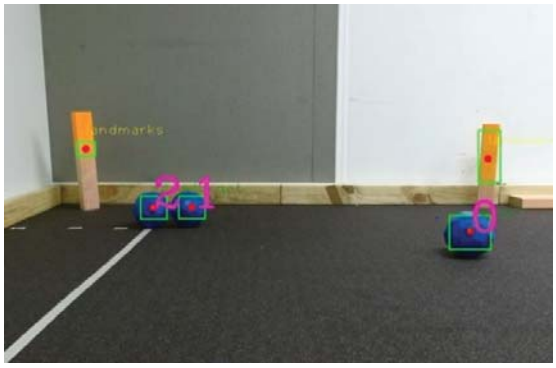


Figure 5- 30: Object detection with the proposed separation method

5.5 Discussion

This chapter discussed the integration of object detection method with the Visual SLAM method and presented a SLAM-O framework. Three methods for frame selection were demonstrated. The outputs of 20 frames method was within 0.05m close to the ground truth, relatively robust compared to the other two methods. The covariance based method needs a threshold to be calculated in advance. The threshold value needs to be computed using sample time, robot speed, and measurement noise, etc. and tested by experiments to make sure that the

threshold is reliable. In these three frame selection methods, the covariance based method can obtain the most accurate result if the EKF is not in the process of divergence. The GMM based method stops iterating when the states estimates do not have large increase or decrease. This method presumes that the EKF is converged if there is no big fluctuation in the state estimates. The experimental results proved that GMM based SLAM estimates were close to the ground truth. An accurate estimate can be obtained when the robot moves a further distance. In short distance movements, GMM based method is not encouraged to use, because encoders can obtain an accurate result when moving in a short distance due to less dead reckoning occurs.

This chapter presented a colour clustering method for semi-transparent ball detection, which is a new method to solve problems of semi-transparent object detection. This method improved upon the traditional colour detection methods; the colour thresh-holding usually misses a large portion of the object body, and the classifier method in computer vision normally deals with complicated features with area caused by light refraction to be wrongly classified as objects in some conditions. While the K-Means method can cluster similar red pixels of a ball into one group, this chapter demonstrated how semi-transparent balls can be detected successfully.

A method for object separation based on concave defects was also developed. This outperforms the Watershed method when the balls overlap in the image due to proximity and angle of the view. Experiments proved the effectiveness compared to the result of Watershed separation. In addition, this method can work on multi-object separation, even when the object is occluded. Additionally, objects tracking and separation method also performed well in SLAM-O method.

5.6 Conclusion

As discussed, the proposed SLAM-O methods of incorporating SLAM and object detection performed well in the indoor environment. The K-Means based colour clustering method can improve performance of the tradition colour thresh-holding method. Successful transparent object detection best illustrated the efficiency of K-Means method. The proposed new object separation method worked robustly compared to watershed method and the proposed method performs reasonably well in SLAM-O method. Three frame selection methods were used and discussed: 20 frames method can work robustly, covariance based method need observation and calculation to obtain a threshold, while GMM based method can give better localization estimates.

5.7 Contribution

The contribution of this chapter is threefold. First of all, a SLAM-O method is developed for robot self-localization and objects detection; also achieved extensive experimental testing on the proposed method. A second contribution method for semi-transparent object detection and closely connected object separation were proposed. The third contribution is frames selection strategies which can achieve accuracy to a certain degree and time efficiency comparing to fixed number of frames iteration method.

Chapter 6 Case Study

6.1 Introduction

One of the difficult aspects of SLAM is that the robot starts out with no information about its location, or about its environment. To accurately map the observations, a robot must know its location. Two Case studies are carried out in this chapter. An iterated video frames module is used to maximize a robot's self-localization accuracy. With information from SLAM method, interesting objects can be detected and tracked when the robot moves. The proposed SLAM-O method was used for object detection. In each case study, robot follows a different path.

6.2 Case Study One

A VEX field of 12"*12" is used for robot navigation. There are eight landmarks placed on the boarder of VEX field. Bule and red buckyballs are spread randomly and the blue buckyballs are selected as target objects. Figure 6-1 depicts the design of the experiment field.

In case study one, the robot is supposed to move forward 1.5m towards diagonal direction. The robot stops when encoder results come to every 0.3m. During the stop, a 20-frames iterated method is used to compute the robot's pose; the SLAM-O method from Chapter 5 is used to detect and localize target objects. The robot's local coordinate is used for localization. Ground truth of target objects' positions in the robot coordinate are demonstrated in Figure 6-1.

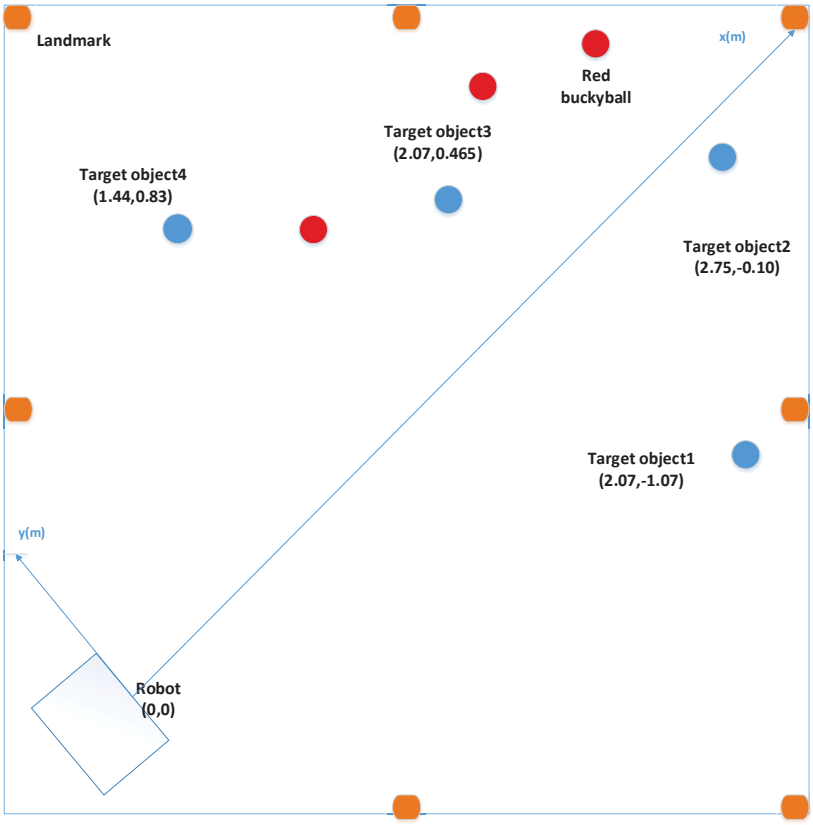


Figure 6- 1: Case study1 field description

Figure 6- 2: VEX field when moving diagonal towards a landmark

One of the robot's observation scenes is showed in the Figure 6- 2. Three landmarks were observed when the robot moved. Mapping results are described in Figure 6- 3. The robot's self-localization results were within 0.05m from ground truth. Location estimates were closer to ground truth when the movement distance was larger than 1m. This is because those EKF parameters were updating during the whole movement process. The new received estimates were closer to ground truth due to the updated parameters which can better simulate the SLAM process. The current estimate yields a good initial estimation of the next step, the following estimate will converge faster. The proposed SLAM method should obtain reasonable results even when the robot moves in an outdoor environment.

Figure 6- 4 depicts the estimated position of target objects with ground truth of (2.07,-1.07), (2.75, -0.10), (2.07, 0.456) and (1.44, 0.83), along with the variance of each estimate. Object locations were close to ground truth in the x direction, except target1 which had a 0.05m error compared to 0.03m error of other targets. In the y direction, locations had an offset around 0.02m. Localization results were more accurate when objects were about 2m away and had a small angle deviation to the robot. Target1 and Target4 did not obtain accurate results, because their distances to the robot were in a 1.2m radius and their angle to the robot were more than 40^{o} deviations.

Variance of each objects' measurements were within 0.02m to their average value, except target1 which had a 0.1m variance in x direction. This could be because of the large angle deviation to the robot, which may introduce errors when Kinect V2 maps depth to colour. As there are more colour pixels (1092*1080) than depth (512*424), a large deviation may occur when a colour pixel could not map to its corresponding depth point.

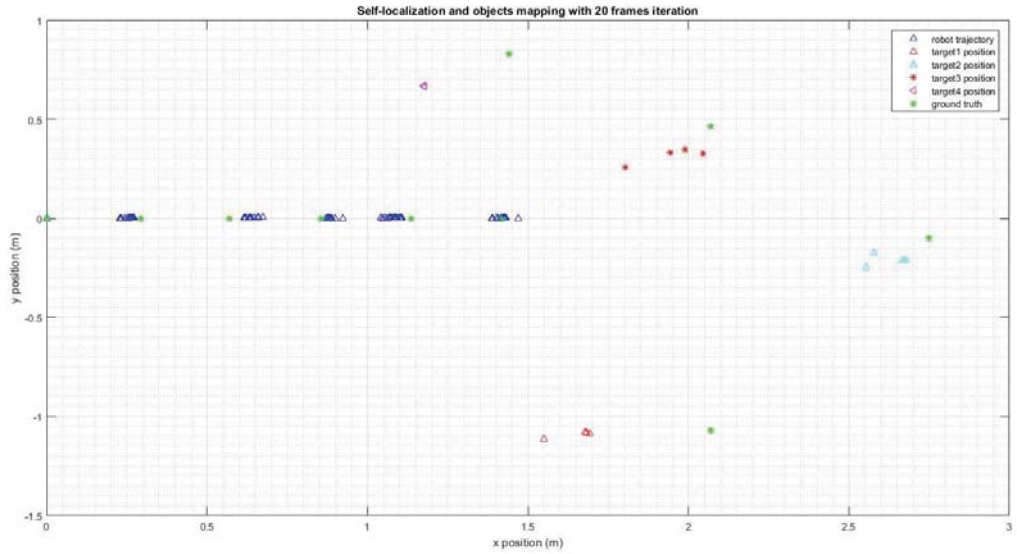


Figure 6- 3: Mapping results of case study one

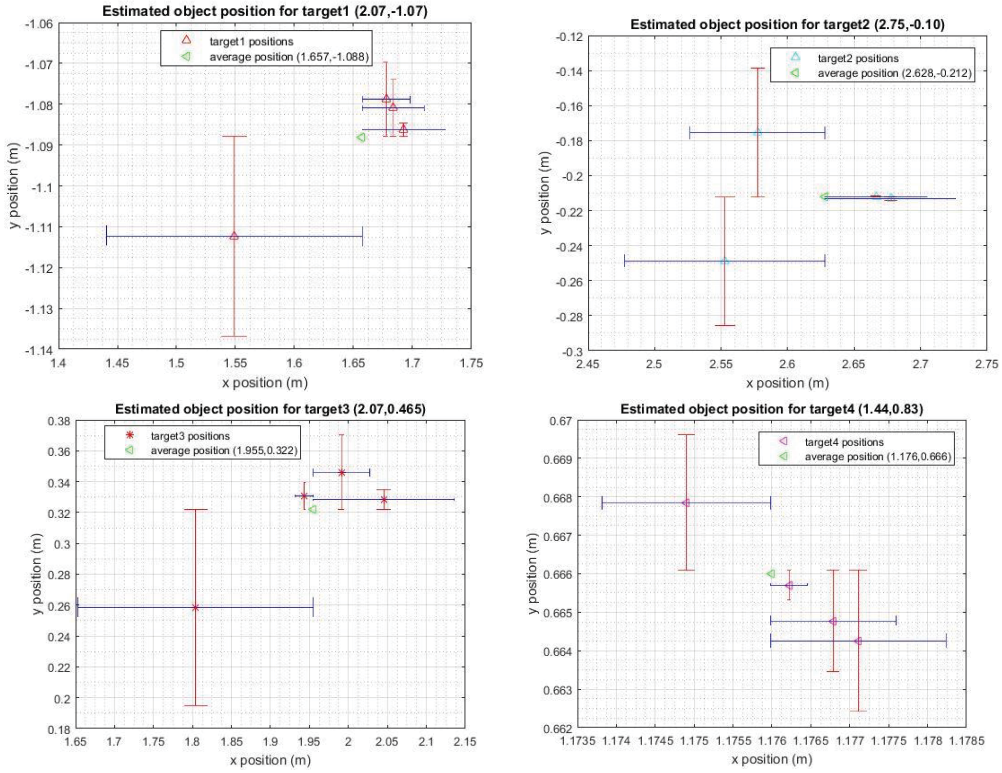


Figure 6- 4: Object variance in case study one

6.3 Case Study Two

In case study two, the robot follows a different route. At first the robot moves forward 1.2m, and then turns 90 degree towards left, then keeps on moving 0.6m. The field description is in Figure 6- 5. The positions of target objects are in the robot's local coordinate system.

The mapping results are demonstrated in Figure 6- 6. In the first three waypoints, the SLAM estimates were 0.2m away from the ground truth. These results agree with discussion in section 4.4.2, where three landmarks based self-location obtained a 0.16m error. After moving to the fourth waypoints, self-localization results started to converge closer to ground truth. This is similar to case study one (the estimates were within 0.05m of ground truth). Although accurate estimates were obtained after approximately 1m movements, the object detection and landmarks association in the first 1m movements could be affected. It is suggest that two landmarks should be used for SLAM in the beginning. There was a 0.1m error in x direction during the robot turning process; this could because of the turning method which was based on movement difference between left and right sides of wheels. Both encoder mistakes and slippages could cause errors during a turning process.

Figure 6- 7 depicts the estimated position of target objects with ground truth of (2.22,-0.23), (1.96, 0.95), (1.06, 0.9) and (0.42, 0.71), along with the variance of each estimate. Similar with case study one, errors in object detection exist when objects were too close to the robot and had large angle deviation. Target1, target3 and target4 had errors up to 0.1m in x direction and 0.5m in the y direction, while target2 had a much small error, 0.06m in the x direction and 0.22m in the y direction. Measurement variance was within 0.02m, except for target2 which

had 0.06m variance. This may be because of the turning process, where big errors exist in robot self-localization.

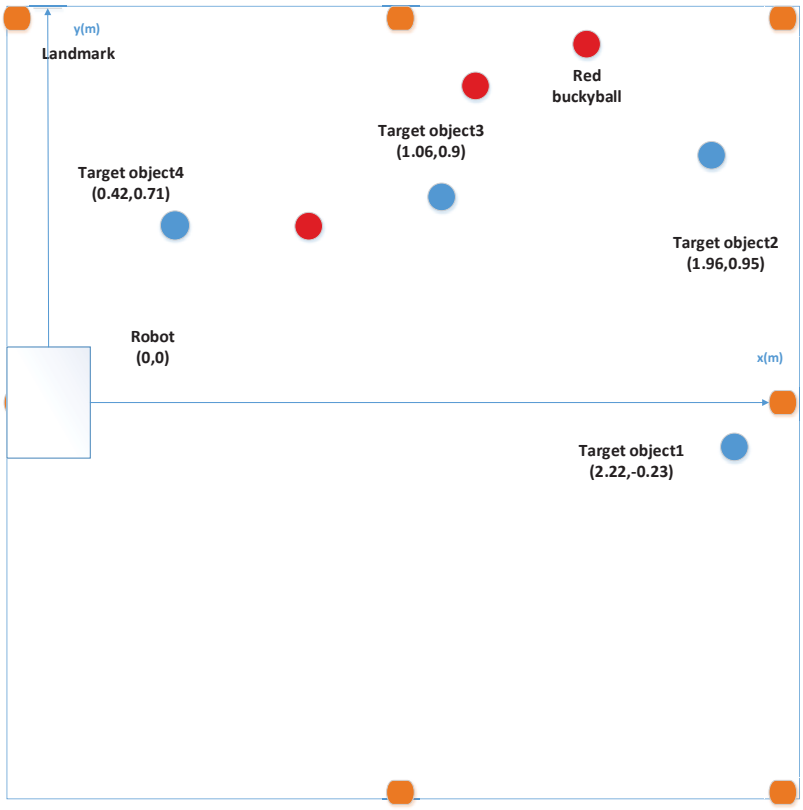


Figure 6- 5: Case study2 field description

Efficiency of the iterated video frames module is checked using a chi-square test:

$$D^2 \leq \chi^2_{r,1-\alpha} \tag{6-1}$$

Where $r = dim(x)$ and α is the desired significance level (it is considered that the usual $\alpha = 0.05$). The case studies use (x, y, θ) coordinates of the robot localization and covariance matrix P is of size 3×3, so the value D^2 has a three degrees of freedom. The threshold for efficiency is $D^2 \leq 0.352$.

NEES of both Case Studies is computed using equation (4-24) and obtained as follows:

$$D^2_{casestudy1} = 0.0057$$

$$D^2_{casestudy2} = 0.0026$$

Both NEES results are within threshold, thus the localization method is effective.

RGB-D cameras are mostly used for dense mapping or 3D reconstruction, what researchers considered is if the model can accurately represent an environment. Whether the model is close to ground truth is rarely discusses. This study suggests that a number of issues should be considered when using a RGB-D camera to locate a single object:

- Locate an single object with a dense feature or patch feature
- If the camera's self-localization is accurate before locating another object
- Make sure the target object does not have a reflective or transparent surface, which can bring in a false range measurement
- Consider if the camera should be calibrated

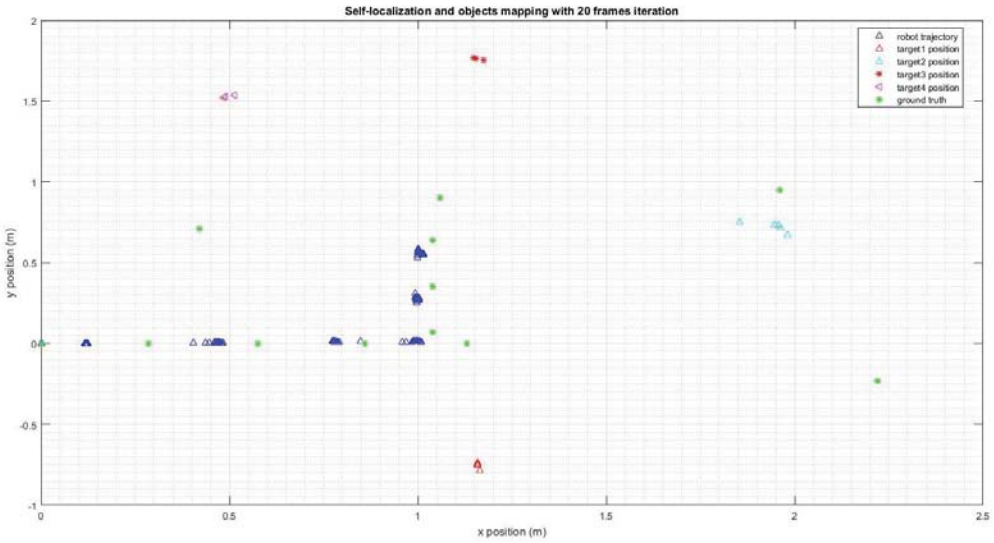


Figure 6- 6: Mapping results of case study two

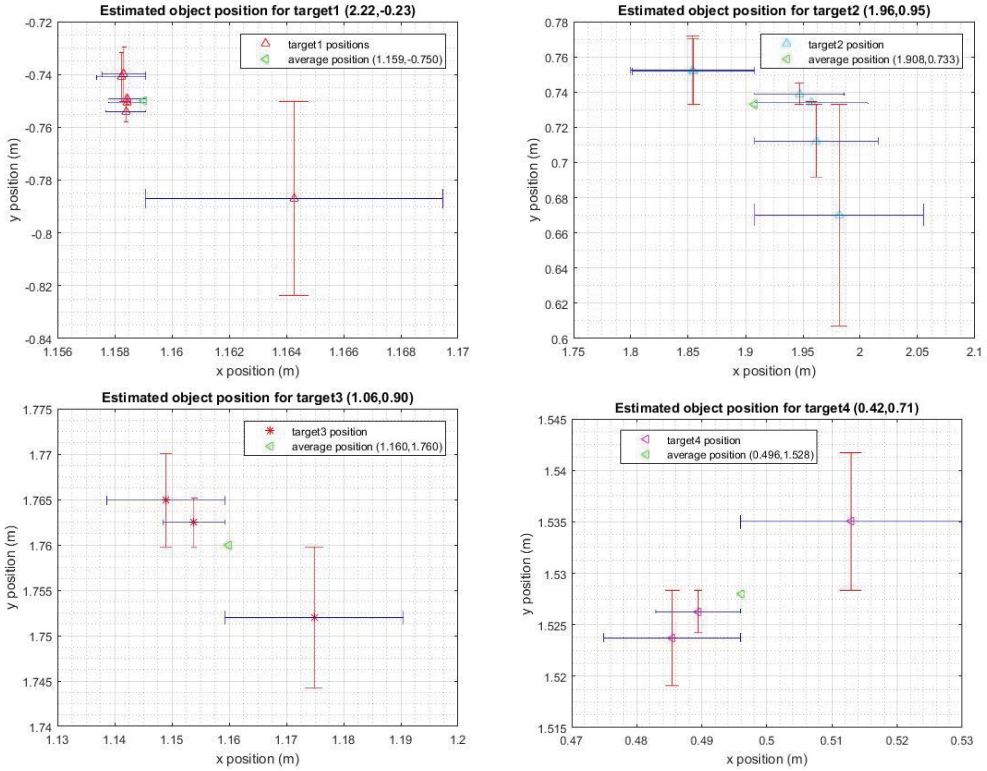


Figure 6- 7: Object variance in case study two

6.4 Conclusion

Two case studies have been carried out in this chapter. The robot moved according to a pre-defined path. Self-localization and object detection results were obtained. Self-localization method has been proved to be effective, while object localization needs to be improved. Improvement methods were discussed.

Chapter 7 Discussion

This chapter discusses possible directions for future research and concludes this thesis.

7.1 Future Work

This thesis has presented a SLAM-O system for a mobile robot real-time navigation and object detection. The approach exploits the relationship between iterated video frames and EKF state estimates, where the iterated image frames update EKF model parameters and thus improve the model estimate accuracy. There are many directions in which the system could be developed. Some interesting possibilities include:

1. **Making greater use of video frames selection method**. In this thesis frames selection method has been tested on a straight route of 1.5m length. In a practical system, there are normally several way points (e.g. a path generate by path planning), time spent on Visual SLAM and navigation could be improved considerably by adopting video frames selection method. While an optimisation function is required for time balance, frame selection methods provide a reference for time balancing on environment observation.

2. **Optimise frame selection method**. In this thesis, an experienced number has been used to determine the threshold of covariance. In principle this term could instead be calculated, and indeed it is determined by operation noise, movement speed and sample intervals. Further investigations on covariance threshold for different movement environments can be performed. Subsequent analysis is likely to yield an automatic threshold for various robots' behaviour.

3. **Use of camera calibration**. Often a RGB-D camera is calibrated before a scientific study, which provides a more accurate depth (Herrera et al., 2012; Raposo et al., 2013). Unlike the direct measurements, the calibration method relocates depth pixels and colour pixels with calculated intrinsic and extrinsic parameters. It should be possible to map a depth pixel to a colour pixel by transition and rotation matrixes from calibration of the particular camera used, rather than using functions from Kinect V2 libraries which are for all the Kinect cameras. It is suggested that calibration may significantly improve the performance of object localization.

4. **Insert a tracker for object localization**. In this thesis, data association has been used to track objects. The solution is effective but not ideal. A better solution might be to track static and dynamic objects at the same time. Data association is based on distance comparison. An alternative approach should be proposed.

5. **Stronger localization to long distance navigation**. One of the greatest challenges for Visual SLAM systems will be dealing with long distance movements where localization may fail due to lost accuracy. As mentioned in Chapter 4, experiments in this thesis about SLAM on a robot has performed longer distance navigation than previous studies. A comparison with Normal EKF model and this thesis would be interesting.

6. **Robustness to dynamic environment**. It seems likely that a mobile robot will be dealing with a dynamic environment where the robot encounters some new features or interesting objects will be a challenge. It does not seem feasible to learn a-priori that these are the features to identify interesting objects, different from the case of Chapter 5 where objects in a competition are known. Possible solutions included rely more

heavily on a robot's learning ability, a robust image processing and analysis, etc. to locate and tracking interesting objects.

7.2 Concluding Remark

This study aims to provide an innovative and practical approach for robot navigation, and set out to provide a coherent Visual SLAM framework in which to locate the robot and interesting objects. The approach of iterating observation frames for the best estimate of the robot poses has proven to be remarkably successful. It should be firmly established as an essential component of EKF SLAM. The proposed approach is useful for improving the robustness and efficiency of an EKF based SLAM systems and provides a framework integrated with object detection techniques. The approach will be used for balance time spent on environment observation and robot navigation, and can be further developed to enable an autonomous robot against human participants in competition environments.

Appendix A

This section is on the design, construction, and testing a robot to test Visual SLAM method. The primary goal is to design and build a robot that could navigate freely in an indoor environment. After researching existing indoor robot designs, a robot prototype was built using concepts from the existing designs. The prototype was then tested to determine the effectiveness of the design. The prototype proved to be successful, being capable of moving straight forward, backward, and turning 90^{o} to the left and the right.

The following requirements are used as the baseline for the construction of the robot.

- Flexibility: the driving system of robot should be access by open software. It should be programmed in C, and flexible to communicate with windows.
- Speed: The robot must move at a suitable speed to observe its environment. It is deemed that a speed of 0.05-0.4m/s is sufficient to achieve this.
- Visual observation ability: visual sensor should be able to observe environments in front of the robot. Visual sensor must be place in the front part of the robot. For visual processing, a PC should be behind the sensor.
- Height: The height of the robot is required to enable a RGB-D camera to detect objects 0.5m in the front. Considered a normal vertical field of view is 45^{o} -60^{o}, a height of 0.5m is considered appropriate for the task.
- Stability: the robot must keep balanced when moving forward and turning.
- Weight: The robot needs to be solid enough to carry. The weight of the robot is required to be around 10 kg to move swiftly.

- Power supply: The robot must carry its own power supply, which will be required to last for at least two hours. The supply must be rechargeable and cost effective.
- Manoeuvrability: The robot is required to manoeuvre in an office, corridor or VEX field. This may involve turning and direction reversal in spaces as small as 0.8m diameter.
- Cost: the cost should be lower than a commercial robot. The budget is 1000 NZD.

After determining the features needed on the robot, a development process is conducted to design and build a robot that could fulfil the requirements. The development process is shown in Figure A- 1.

A. Frame structure and driving system

Frame structure should be low price; driving system should be available to public. VEX is chosen for a few reasons. As VEX has its own product supply in New Zealand, it is easy to consume the products and save transportation from overseas; VEX driver is compatible with Arduino software, which make it easy to control; the VEX hardware kit is flexible to user's design and easy to make modifications.

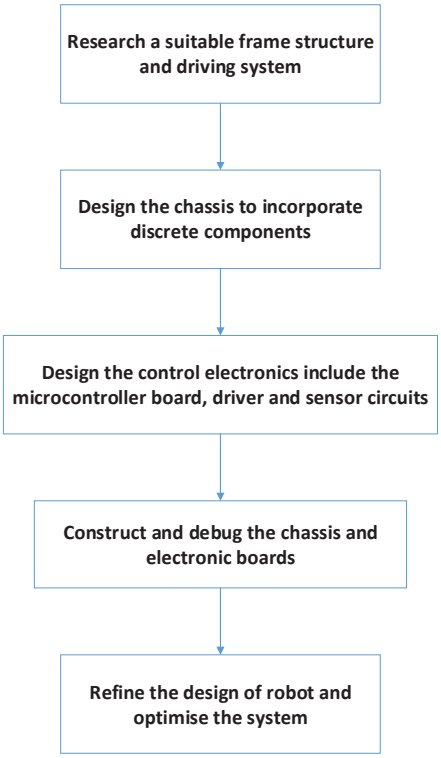


Figure A- 1: Flowchart of development process

Table A- 1: Frame structure and driving system costs from Kiwibots (KIWIBOTS, n.d.)

Name	Item Number	Amount (pack)	Costs(NZD)
Aluminium c-channel	4359	1	60.79
Batteries	1491	2	100.50
Motor controllers	2193	4	66.96
Motors	2177	4	97.36
Power Expander	2271	1	83.76
Drive shafts	2011	1	15.02
Bearing flats	1209	1	8.37
Spacers	2019	1	5.02
Back wheels	1902	1	33.50
Keps	1026	1	6.47
Screws	1004	1	12.56
Shaft collar	2010	1	17.58
Front wheels	1496	1	16.74
		Total costs	524.63

Wheeled driven robots are selected for navigation as they are easier to design, build and program than using treads or legs. Two, three and four wheeled robots are commonly used in indoor environments. Two wheeled robots are harder to balance than other types because they must keeping moving to maintain upright, such as vacuum robot Roomba. In a three wheeled robot's condition, the robot direction may be changed by varying the relative rate of rotation of the two driven wheels. The centre of rotation is not easy to balance. Four wheeled robot is easily to keep balanced, also has a tight turning circle. For the stability, a four-wheeled design is selected. 2 pairs of powered wheels are used and each pair turns in the same direction. A list of VEX parts used for frame structure driving systems is in Table A- 1.

B. Power supplies

For driving motors, VEX provides its own batteries and power expander for power supply. For visual sensors, the power supply must be portable, and batteries must supply enough power. To supply power for Visual sensor, Kinect V2 in this study, battery must have outputs 12 volts and 2000-8000 mAh. Search for 12 volts batteries, there are mainly three types of battery available, which are Lipo, Ni-MH and Acid batteries as shown in Figure A- 2.

The Lipo battery has a capacity of 2200mah, with a small size 5.5mm * 2.1mm. the price is 25 NZD. The Ni-MH battery also has a compact size, but each one can only supply 1.2v. To obtain a output of 12 v,ten batteries should be connected together to output 12v. Four of them cost around 25 NZD. Acid battery can output 12v, but this battery has a big size, 150mm*65mm* 95mm, which will take a large space in chassis and not convient for elctronic diagram design. Charge of each Acid battery is around 50 NZD. Thus, Lipo battery is selected for its small size, cheaper price and power supply ability.

Lipo battery

Ni-MH battery

Acid battery

Figure A- 2: 12 Volts rechargeable batteries

C. Chassis design

The robot should be of a suitable size to navigate in indoor environment; also need to be large enough to support the weight of PC, power supplies and electronic parts.

The chassis is designed and constructed in three major parts, the base, the cover and the frame. The base is designed to support the wheels, frame and batteries directly attached to it. To strongly support the parts, a wood base is used. The cover is designed to support the PC. A glass cover is used and providing a space of approximately 318*445 mm. A hole is designed in the glass to provide wire connections.

Aluminium c-channel parts from VEX are used as frame, as it is demonstrated in Figure 2-8. The upper frame in front of the laptop is used for holding a camera; the lower frame is used to attach motors and their encoders.

Four wheels of each motor are mounted in a rectangle. Two front wheels designed with a smooth grippy tire, which has high traction on smooth surface; two back wheels are omni-direction which will roll forward like normal wheels, but also able to slide sideways with almost no friction. Turning is achieved by letting the left hand side wheels rotate with an opposite

velocity from the right hand side wheels. This gives a fast rotating motion with an advantage of time saving.

D. Control electronic

The control of the driving motors is accomplished with the use of a microcontroller, which can be programmed from a PC. The microcontroller will deliver commands to the motors of the device via the driver circuitry.

The microcontroller used to control the driver system is the Arduino Mega 2560. Power supply of the microcontroller is from USB, which is connected with PC.

The VEX motor 393 and motor control 29 is used to drive wheels. The VEX Motor Controller 29 uses a standard Pulse Width Modulation (PWM) signal to drive a VEX Motor. PWM is a method of controlling the amount of current supplied to driving motors. The motor rotates forward when a PWM value is 0, stops when PWM is 90 and rotates backwards when PWM is 180. Motor speed changes nonlinear with PWM value. A PID controller is programmed in microcontroller for a steady speed.

E. Sensors

The robot must be able to observe its surroundings and sense its own location, also to calculate its position, information on the distance and heading that it has travelled. A Kinect V2 camera is used as visual sensor, and integrated encoders from VEX are used to determine its absolute location with respect to an arbitrary start point.

Arduino is designed to read integrated encoders' data by I2C protocol. A I2CEncoder library (Henning, n.d.) is used to obtain speed and position of the encoders.

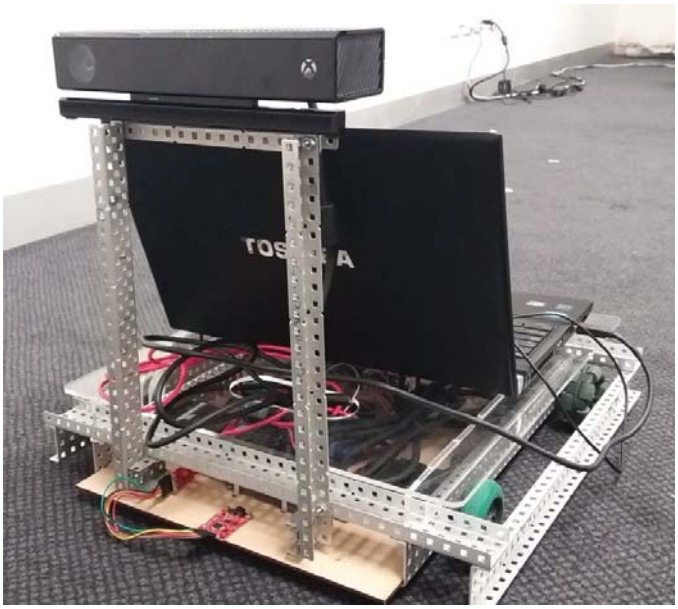


Figure A- 3: The mobile robot used in experiments of this thesis

With the using of encoders and a PID controller, the movement is maintained at a speed of 0.22 feet/s, approximately 0.067 m/s. Turning degree is achieved by calculating rotation of the left and right wheels.

The built robot is demonstrated in Figure A- 3. Evaluation of the robot is based on tests which include: moving straight, moving to a designed destination, and turning an angle of 90^{o}.

1) For straight moving, the robot is tested on a flat road. 3 m movement in a forward direction was achieved.

2) The second test is to check if the robot can move to a required destination. Way points of 0.3m, 0.6m, 0.9m, 1.2m and 1.6m are designed. The robot stops when the encoders

achieved the designed destination. Testing results are demonstrated in Figure A- 4. For every 0.3m, there is a 0.05m error caused by drifts.

3) The third test is to test a robot can turn a 90^{o}. After 20 times of turnings, a variance of 2^{o} in a 90^{o} turning is detected.

4) A final test combing 2) and 3) is carried. The testing shows a slippery of around 0.1m during the turning process. Testing results are demonstrated in Figure A- 5.

To summarize, the robot is able to accomplish movements and turnings commands. Control parts in electronic and communication are function well. There are errors when moving to a designed a destination, which are due to slippery and drifts. These errors can be improved by using Visual SLAM method, which will be discussed in Chapter 4.

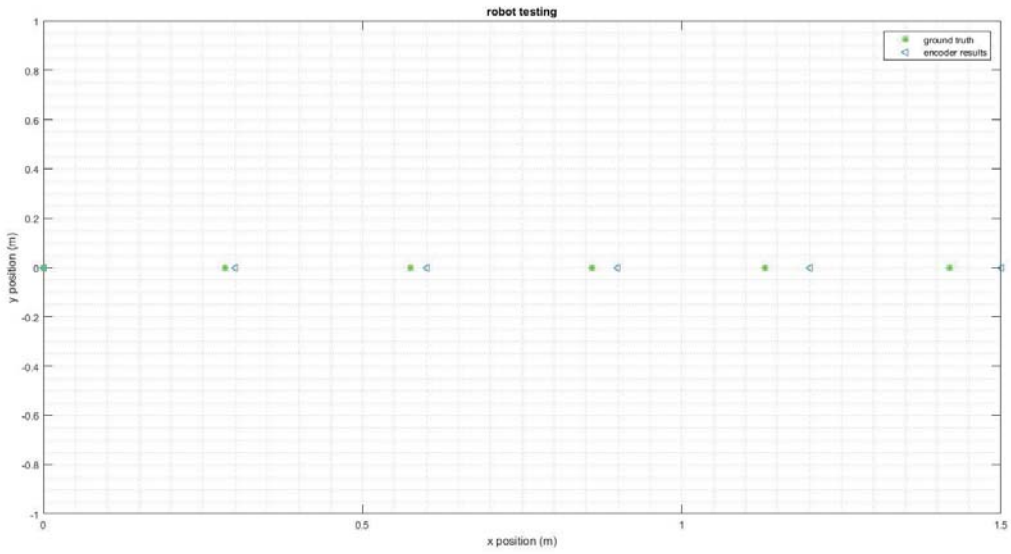


Figure A- 4: Robot straight moving testing

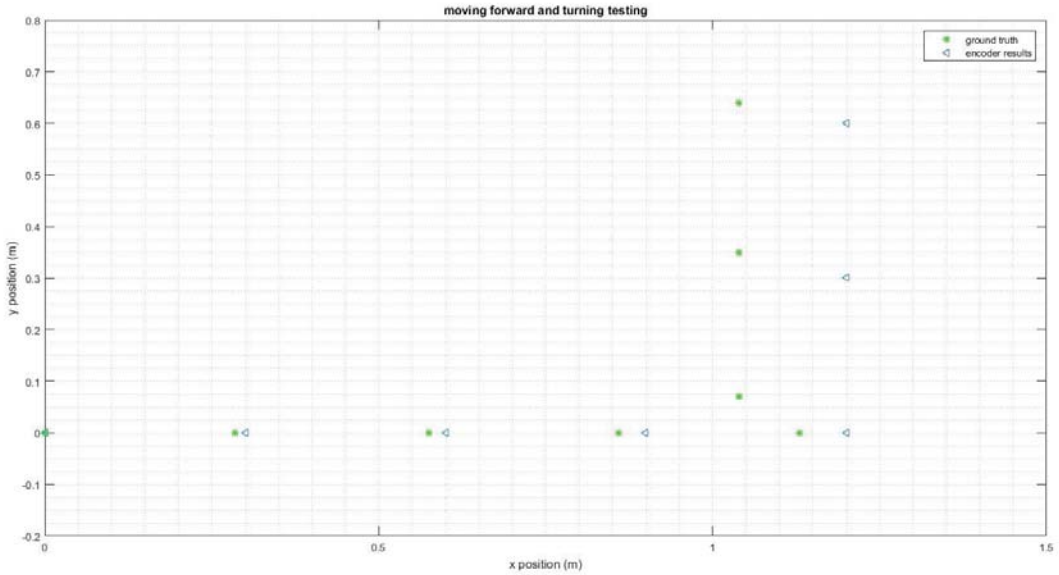


Figure A- 5: Robot straight moving and turning

Appendix B

This section lists the details of the robot's structure. Electronics parts for motion control are demonstrated in Figure B- 1.

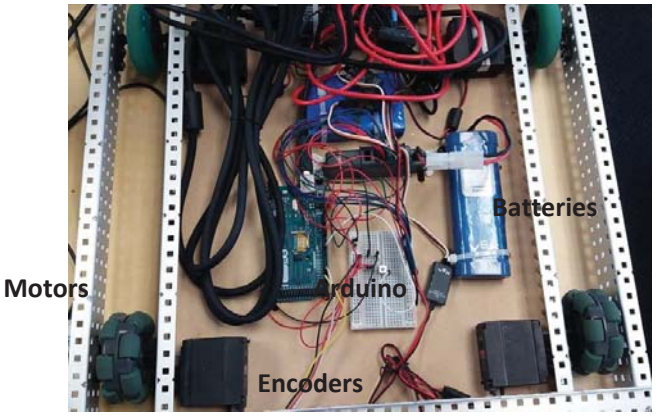


Figure B- 1: Electronics parts of the robot

Sizes of the electronics parts are in Figure B- 2.

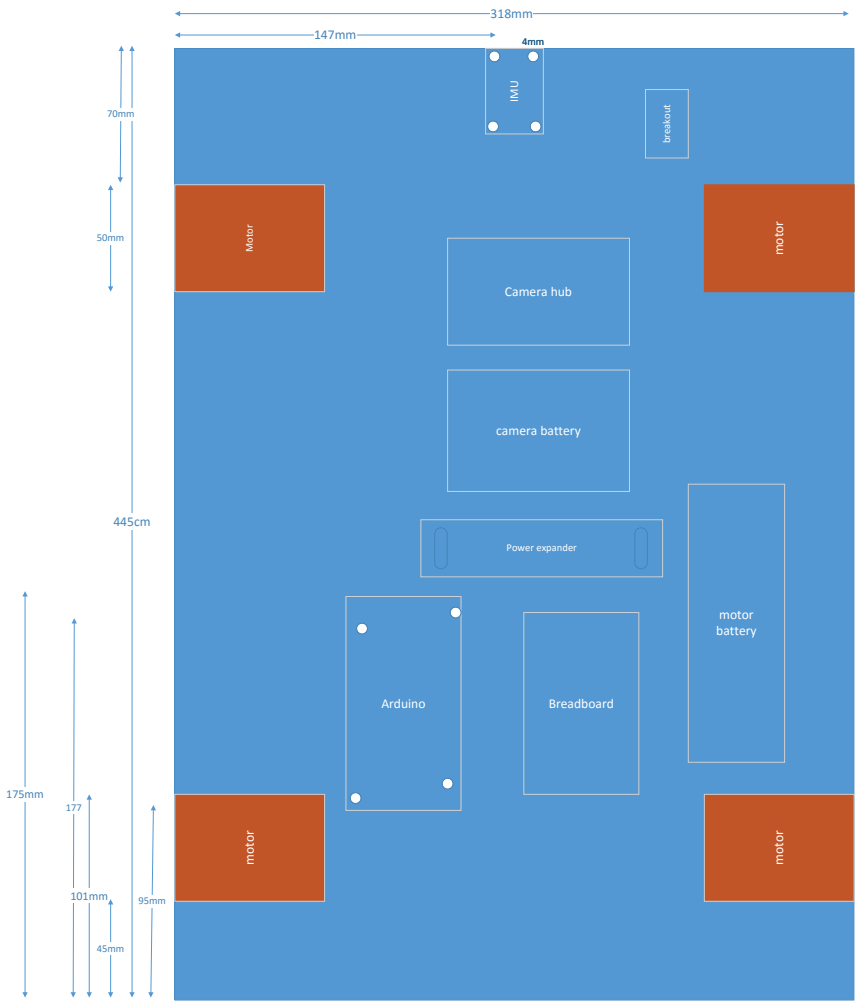


Figure B- 2: Sizes of the electronics parts

Hardware parts for the mobile robot structure are in Table B- 1, electronics parts are in Table B- 2.

Table B- 1: Hardware parts for the mobile robot structure

NO.	Name	Model
1	RGB-D camera	Kinect for Xbox one
2	Kinect adapter	Kinect Adapter for Xbox One S and Windows PC
3	laptop	TOSHIBA, model Tecra R950
4	2 front wheels	VEX 2.75" diameter Wheels
5	2 back wheels	2.75" Omni Directional Wheel - Double Roller
6	4*393 drivers	2-Wire VEX Motor 393
7	4*encoders	VEX Motor 393 integrated encoder module
8	4* motor controller	VEX Motor Controller 29
9	VEX batteries	7.2V Robot Battery NiMH 3000mAh
10	A battery for camera	12V
11	Power expander	VEX Power Expander
12	Metal kit	Aluminium C-Channel 1x3x1x35(6-pack)
13	Drive shaft	Drive Shaft 12" (4-pack)
14	Bearing Flat	Bearing Flat, Delrin (10-pack)
15	Spacers	Spacer, 8mm (20-pack)
16	Keps	Nut 8-32 Keps (100-pack)
17	screw	Screw 8-32 x 0.500
18	shaft	Shaft Collar

Table B- 2: Electronics parts

NO.	Name	Model
1	Arduino	Arduino Mega 2560 R3
2	USB cable	USB Cable A to B - 6 Foot
3	Breadboard	Breadboard - Self-Adhesive (White)
4	Jumper wires	Dupont Wire Male to Male + Male to Female + Female to Female Jumper Wire
5	3-set jumper wires	12pcs 0.3m 3pin Female to 1pin Male jumper wire
6	Push button	Tactile Push Button Switch Tact Switch 6X6X5mm 4-pin DIP

References

Al-Amri, S. S., Kalyankar, N., & Khamitkar, S. (2010). Image segmentation by using edge detection. *International Journal on computer science and engineering, 2*(03), 804-807.

Ana, L., & Jain, A. K. (2003). *Robust data clustering.* Paper presented at the Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on.

Andoni, A., & Indyk, P. (2006). *Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions.* Paper presented at the Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on.

Asada, H., & Brady, M. (1986). The curvature primal sketch. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*(1), 2-14.

Bailey, T. (N.A.). Source Code. Retrieved from http://www-personal.acfr.usyd.edu.au/tbailey/software/

Bailey, T., Nieto, J., Guivant, J., Stevens, M., & Nebot, E. (2006a). *Consistency of the EKF-SLAM algorithm.* Paper presented at the Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on.

Bailey, T., Nieto, J., Guivant, J., Stevens, M., & Nebot, E. (2006b). *Consistency of the EKF-SLAM algorithm.* Paper presented at the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems.

Bailey, T., Nieto, J., & Nebot, E. (2006). *Consistency of the FastSLAM algorithm.* Paper presented at the Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on.

Baker, S., Nayar, S. K., & Murase, H. (1998). Parametric feature detection. *International journal of computer vision, 27*(1), 27-50.

Balan, A. O., Sigal, L., Black, M. J., Davis, J. E., & Haussecker, H. W. (2007). *Detailed human shape and pose from images.* Paper presented at the 2007 IEEE Conference on Computer Vision and Pattern Recognition.

Basso, F., Munaro, M., Michieletto, S., Pagello, E., & Menegatti, E. (2013). Fast and robust multi-people tracking from RGB-D data for a mobile robot *Intelligent Autonomous Systems 12* (pp. 265-276): Springer.

Bay, H., Ess, A., Tuytelaars, T., & Van Gool, L. (2008). Speeded-up robust features (SURF). *Computer vision and image understanding, 110*(3), 346-359.

Beauchemin, S. S., & Barron, J. L. (1995). The computation of optical flow. *ACM Computing Surveys (CSUR), 27*(3), 433-466.

Benjamin, M. R., Curcio, J. A., Leonard, J. J., & Newman, P. M. (2006). *Navigation of unmanned marine vehicles in accordance with the rules of the road.* Paper presented at the Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on.

Berge, T., Goldberg, S., Kaspersen, K., & Netland, J. (2012). Towards machine vision based site-specific weed management in cereals. *Computers and Electronics in Agriculture, 81*, 79-86.

Besl, P. J., & McKay, N. D. (1992). *Method for registration of 3-D shapes.* Paper presented at the Robotics-DL tentative.

Biber, P., & Duckett, T. (2005). *Dynamic Maps for Long-Term Operation of Mobile Service Robots.* Paper presented at the Robotics: science and systems.

Biswas, R., Limketkai, B., Sanner, S., & Thrun, S. (2002). *Towards object mapping in non-stationary environments with mobile robots.* Paper presented at the Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on.

Boehm, J. (2012). Natural user interface sensors for human body measurement. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 39*, 531-536.

Borenstein, J., & Koren, Y. (1989). Real-time obstacle avoidance for fast mobile robots. *Systems, Man and Cybernetics, IEEE Transactions on, 19*(5), 1179-1187.

Borenstein, J., & Koren, Y. (1991). The vector field histogram-fast obstacle avoidance for mobile robots. *Robotics and Automation, IEEE Transactions on, 7*(3), 278-288.

Bosse, M., & Zlot, R. (2008). Map matching and data association for large-scale two-dimensional laser scan-based slam. *The International Journal of Robotics Research, 27*(6), 667-691.

Bouguet, J.-Y. (2015). Camera Calibration Toolbox for Matlab. Retrieved from https://www.vision.caltech.edu/bouguetj/calib_doc/

Browning, B., & Govindaraju, D. (2003). Fast, Robust Techniques for Colored Object Detection in Variable Lighting Conditions. *United State Army*.

Cadena, C., Carlone, L., Carrillo, H., Latif, Y., Scaramuzza, D., Neira, J., . . . Leonard, J. J. (2016a). Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on Robotics, 32*(6), 1309-1332.

Cadena, C., Carlone, L., Carrillo, H., Latif, Y., Scaramuzza, D., Neira, J., . . . Leonard, J. J. (2016b). Past, Present, and Future of Simultaneous Localization And Mapping: Towards the Robust-Perception Age. *arXiv preprint arXiv:1606.05830*.

Cappelletto, E., Zanuttigh, P., & Cortelazzo, G. M. (2013). *Handheld scanning with 3D cameras.* Paper presented at the Multimedia Signal Processing (MMSP), 2013 IEEE 15th International Workshop On.

Castellanos, J. A., Martinez-Cantin, R., Tardós, J. D., & Neira, J. (2007). Robocentric map joining: Improving the consistency of EKF-SLAM. *Robotics and Autonomous Systems, 55*(1), 21-29.

Çelik, H., Dülger, L., & Topalbekiroğlu, M. (2014). Development of a machine vision system: real-time fabric defect detection and classification with neural networks. *The Journal of The Textile Institute, 105*(6), 575-585.

Chatterjee, R., & Matsuno, F. (2001). Use of single side reflex for autonomous navigation of mobile robots in unknown environments. *Robotics and Autonomous Systems, 35*(2), 77-96.

Chung, H., Ojeda, L., & Borenstein, J. (2001). *Sensor fusion for mobile robot dead-reckoning with a precision-calibrated fiber optic gyroscope.* Paper presented at the Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on.

Comaniciu, D., & Meer, P. (2002). Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 24*(5), 603-619.

Cooper, J., Venkatesh, S., & Kitchen, L. (1991). *Early jump-out corner detectors.* Paper presented at the Computer Vision and Pattern Recognition, 1991. Proceedings CVPR'91., IEEE Computer Society Conference on.

Cyrill Stachniss, U. F., Giorgio Grisetti. (n.d.). OpenSLAM. Retrieved from https://openslam.org/

Davison, A. J. (2003). *Real-time simultaneous localisation and mapping with a single camera.* Paper presented at the Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on.

Dellaert, F. (2012). *Factor graphs and GTSAM: A hands-on introduction*. Retrieved from

Deriche, R. (1987). Using Canny's criteria to derive a recursively implemented optimal edge detector. *International journal of computer vision, 1*(2), 167-187.

Dharmasiri, T., Lui, V., & Drummond, T. (2016). *MO-SLAM: Multi object SLAM with run-time object discovery through duplicates.* Paper presented at the Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on.

Dias, P. (2003). Tsai Camera Calibration. Retrieved from http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/DIAS1/

Dissanayake, G., Durrant-Whyte, H., & Bailey, T. (2000). *A computationally efficient solution to the simultaneous localisation and map building (SLAM) problem.* Paper presented at the Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on.

Dissanayake, G., Huang, S., Wang, Z., & Ranasinghe, R. (2011). *A review of recent developments in simultaneous localization and mapping.* Paper presented at the Industrial and Information Systems (ICIIS), 2011 6th IEEE International Conference on.

Dissanayake, M. G., Newman, P., Clark, S., Durrant-Whyte, H. F., & Csorba, M. (2001a). A solution to the simultaneous localization and map building (SLAM) problem. *IEEE transactions on robotics and automation, 17*(3), 229-241.

Dissanayake, M. G., Newman, P., Clark, S., Durrant-Whyte, H. F., & Csorba, M. (2001b). A solution to the simultaneous localization and map building (SLAM) problem. *Robotics and Automation, IEEE Transactions on, 17*(3), 229-241.

Dollár, P., Appel, R., Belongie, S., & Perona, P. (2014). Fast feature pyramids for object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 36*(8), 1532-1545.

Donoser, M., & Bischof, H. (2008). *Real time appearance based hand tracking.* Paper presented at the Pattern Recognition, 2008. ICPR 2008. 19th International Conference on.

dos Santos, D. R., Basso, M. A., Khoshelham, K., de Oliveira, E., Pavan, N. L., & Vosselman, G. (2016). Mapping indoor spaces by adaptive coarse-to-fine registration of RGB-D data. *IEEE Geoscience and Remote Sensing Letters, 13*(2), 262-266.

Dowlati, M., de la Guardia, M., & Mohtasebi, S. S. (2012). Application of machine-vision techniques to fish-quality assessment. *TrAC Trends in Analytical Chemistry, 40*, 168-179.

Durrant-Whyte, H., Rye, D., & Nebot, E. (1996). Localization of autonomous guided vehicles *Robotics Research* (pp. 613-625): Springer.

Dutta, T. (2012). Evaluation of the Kinect™ sensor for 3-D kinematic measurement in the workplace. *Applied ergonomics, 43*(4), 645-649.

Elmezain, M., Al-Hamadi, A., & Michaelis, B. (2010). *A robust method for hand gesture segmentation and recognition using forward spotting scheme in conditional random fields.* Paper presented at the Pattern Recognition (ICPR), 2010 20th International Conference on.

Endres, F., Hess, J., Engelhard, N., Sturm, J., Cremers, D., & Burgard, W. (2012). *An evaluation of the RGB-D SLAM system.* Paper presented at the Robotics and Automation (ICRA), 2012 IEEE International Conference on.

Engel, J., Schöps, T., & Cremers, D. (2014). *LSD-SLAM: Large-scale direct monocular SLAM.* Paper presented at the European Conference on Computer Vision.

Eustice, R., Singh, H., Leonard, J. J., Walter, M., & Ballard, R. (2005). *Visually Navigating the RMS Titanic with SLAM Information Filters.* Paper presented at the Robotics: Science and Systems.

Fankhauser, P., Bloesch, M., Rodriguez, D., Kaestner, R., Hutter, M., & Siegwart, R. (2015). *Kinect v2 for mobile robot navigation: Evaluation and modeling.* Paper presented at the Advanced Robotics (ICAR), 2015 International Conference on.

Feng, X., & Perona, P. (2002). *Human action recognition by sequence of movelet codewords.* Paper presented at the 3D Data Processing Visualization and Transmission, 2002. Proceedings. First International Symposium on.

Finlayson, G. D., Hordley, S. D., & Hubel, P. M. (2001). Color by correlation: A simple, unifying framework for color constancy. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 23*(11), 1209-1221.

Folkesson, J., & Christensen, H. I. (2007). Graphical SLAM for outdoor applications. *Journal of Field Robotics, 24*(1‐2), 51-70.

Förstner, W. (1994). A framework for low level feature extraction *Computer Vision—ECCV'94* (pp. 383-394): Springer.

Foundation, R. (2017). VRC Overview. Retrieved from http://www.roboticseducation.org/vex-robotics-competitionvrc/

Fox, D., Burgard, W., & Thrun, S. (1997). The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine, 4*(1), 23-33.

Freedman, B., Shpunt, A., Machline, M., & Arieli, Y. (2012). Depth mapping using projected patterns: Google Patents.

Gall, J., & Lempitsky, V. (2013). Class-specific hough forests for object detection *Decision forests for computer vision and medical image analysis* (pp. 143-157): Springer.

Gilbert, E. G., & Johnson, D. W. (1985). Distance functions and their application to robot path planning in the presence of obstacles. *Robotics and Automation, IEEE Journal of, 1*(1), 21-30.

Grasa, O. G., Bernal, E., Casado, S., Gil, I., & Montiel, J. (2014). Visual SLAM for handheld monocular endoscope. *Medical Imaging, IEEE Transactions on, 33*(1), 135-146.

Grasa, O. G., Civera, J., & Montiel, J. (2011). *EKF monocular SLAM with relocalization for laparoscopic sequences.* Paper presented at the Robotics and Automation (ICRA), 2011 IEEE International Conference on.

Graves, M., & Batchelor, B. (2003). *Machine vision for the inspection of natural products*: Springer Science & Business Media.

Grisetti, G., Kummerle, R., Stachniss, C., & Burgard, W. (2010). A tutorial on graph-based SLAM. *IEEE Intelligent Transportation Systems Magazine, 2*(4), 31-43.

Grisetti, G., Stachniss, C., & Burgard, W. (2007). Improved techniques for grid mapping with rao-blackwellized particle filters. *Robotics, IEEE Transactions on, 23*(1), 34-46.

Grisetti, G., Stachniss, C., Grzonka, S., & Burgard, W. (2007). *A Tree Parameterization for Efficiently Computing Maximum Likelihood Maps using Gradient Descent.* Paper presented at the Robotics: Science and Systems.

Gu, L., & Zhou, M. (2011). *Based on the research of the chip electrode positioning system about machine vision in the LED chip automatic measurement.* Paper presented at the Electronics and Optoelectronics (ICEOE), 2011 International Conference on.

Gupta, S., Girshick, R., Arbeláez, P., & Malik, J. (2014). *Learning rich features from RGB-D images for object detection and segmentation.* Paper presented at the European Conference on Computer Vision.

Hackenberg, G., McCall, R., & Broll, W. (2011). *Lightweight palm and finger tracking for real-time 3D gesture control.* Paper presented at the Virtual Reality Conference (VR), 2011 IEEE.

Han, J., Shao, L., Xu, D., & Shotton, J. (2013). Enhanced computer vision with microsoft kinect sensor: A review. *IEEE transactions on cybernetics, 43*(5), 1318-1334.

Haverinen, J., & Kemppainen, A. (2009). *A global self-localization technique utilizing local anomalies of the ambient magnetic field.* Paper presented at the Robotics and Automation, 2009. ICRA'09. IEEE International Conference on.

Heitz, G., & Koller, D. (2008). *Learning spatial context: Using stuff to find things.* Paper presented at the European conference on computer vision.

Henning, A. (n.d.). I2CEncoder. Retrieved from https://github.com/alexhenning/I2CEncoder

Henry, P., Krainin, M., Herbst, E., Ren, X., & Fox, D. (2010). *RGB-D mapping: Using depth cameras for dense 3D modeling of indoor environments.* Paper presented at the the 12th International Symposium on Experimental Robotics (ISER).

Henry, P., Krainin, M., Herbst, E., Ren, X., & Fox, D. (2012). RGB-D mapping: Using Kinect-style depth cameras for dense 3D modeling of indoor environments. *The International Journal of Robotics Research, 31*(5), 647-663.

Herbst, E., Henry, P., Ren, X., & Fox, D. (2011). *Toward object discovery and modeling via 3-d scene comparison.* Paper presented at the Robotics and Automation (ICRA), 2011 IEEE International Conference on.

Heredia, M., Endres, F., Burgard, W., & Sanz, R. (2015). Fast and Robust Feature Matching for RGB-D Based Localization. *arXiv preprint arXiv:1502.00500*.

Herrera, D., Kannala, J., & Heikkilä, J. (2012). Joint depth and color camera calibration with distortion correction. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 34*(10), 2058-2064.

Hesch, J. A., Kottas, D. G., Bowman, S. L., & Roumeliotis, S. I. (2014). Camera-IMU-based localization: Observability analysis and consistency improvement. *The International Journal of Robotics Research, 33*(1), 182-201.

Horaud, R., Veillon, F., & Skordas, T. (1990). Finding geometric and relational structures in an image *Computer Vision—ECCV 90* (pp. 374-384): Springer.

Hornung, A., Wurm, K. M., Bennewitz, M., Stachniss, C., & Burgard, W. (2013). OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots, 34*(3), 189-206.

Hu, G., Huang, S., Zhao, L., Alempijevic, A., & Dissanayake, G. (2012). *A robust rgb-d slam algorithm.* Paper presented at the Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on.

Huang, G. P., Mourikis, A. I., & Roumeliotis, S. I. (2008). *Analysis and improvement of the consistency of extended Kalman filter based SLAM.* Paper presented at the Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on.

Huang, S., & Dissanayake, G. (2007). Convergence and consistency analysis for extended Kalman filter based SLAM. *IEEE Transactions on Robotics, 23*(5), 1036-1049.

Izadi, S., Kim, D., Hilliges, O., Molyneaux, D., Newcombe, R., Kohli, P., . . . Davison, A. (2011). *KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera.* Paper presented at the Proceedings of the 24th annual ACM symposium on User interface software and technology.

Jan Smisek, M. J. a. T. P. (2011). *3-D with Kinect.* Paper presented at the IEEE ICCV Workshops.

Jana, A. (2012). *Kinect for windows SDK programming guide*: Packt Publishing Ltd.

Jasmine Xavier, A., & Shantha Selvakumari, R. (2015). Behavior architecture controller for an autonomous robot navigation in an unknown environment to perform a given task.

Jin, H., Favaro, P., & Soatto, S. (2000). *Real-time 3D motion and structure of point features: a front-end system for vision-based control and interaction.* Paper presented at the Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on.

Julier, S. J., & Uhlmann, J. K. (2001). *A counter example to the theory of simultaneous localization and map building.* Paper presented at the Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on.

Kadambi, A., Bhandari, A., & Raskar, R. (2014). 3d depth cameras in vision: Benefits and limitations of the hardware *Computer Vision and Machine Learning with RGB-D Sensors* (pp. 3-26): Springer.

Kadir, T., & Brady, M. (2001). Saliency, scale and image description. *International journal of computer vision, 45*(2), 83-105.

Kaess, M., & Dellaert, F. (2010). Probabilistic structure matching for visual SLAM with a multi-camera rig. *Computer Vision and Image Understanding, 114*(2), 286-296.

Kaess, M., Johannsson, H., Roberts, R., Ila, V., Leonard, J. J., & Dellaert, F. (2012). iSAM2: Incremental smoothing and mapping using the Bayes tree. *The International Journal of Robotics Research, 31*(2), 216-235.

Kaess, M., Ranganathan, A., & Dellaert, F. (2008). iSAM: Incremental smoothing and mapping. *IEEE Transactions on Robotics, 24*(6), 1365-1378.

Kale, A., Rajagopalan, A., Cuntoor, N., & Kruger, V. (2002). *Gait-based recognition of humans using continuous HMMs.* Paper presented at the Automatic Face and Gesture Recognition, 2002. Proceedings. Fifth IEEE International Conference on.

Kambhampati, S., & Davis, L. S. (1985). *Multiresolution path planning for mobile robots*. Retrieved from

Kaur, K., & Malhotra, S. A Survey on Edge Detection Using Different Techniques.

Kendall, A., Grimes, M., & Cipolla, R. (2015). *Posenet: A convolutional network for real-time 6-dof camera relocalization.* Paper presented at the Proceedings of the IEEE international conference on computer vision.

Kerl, C., Stuckler, J., & Cremers, D. (2015). *Dense continuous-time tracking and mapping with rolling shutter RGB-D cameras.* Paper presented at the Proceedings of the IEEE International Conference on Computer Vision.

Kerl, C., Sturm, J., & Cremers, D. (2013). *Robust odometry estimation for RGB-D cameras.* Paper presented at the Robotics and Automation (ICRA), 2013 IEEE International Conference on.

Khoshelham, K., & Elberink, S. O. (2012). Accuracy and resolution of kinect depth data for indoor mapping applications. *Sensors, 12*(2), 1437-1454.

Kim, A., & Eustice, R. M. (2013). Real-time visual SLAM for autonomous underwater hull inspection using visual saliency. *IEEE Transactions on Robotics, 29*(3), 719-733.

KIWIBOTS.

KIWIBOTS. (n.d.). STORE. Retrieved from https://www.kiwibots.co.nz/store

Klein, G., & Murray, D. (2007). *Parallel tracking and mapping for small AR workspaces.* Paper presented at the Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on.

Kolb, A., Barth, E., Koch, R., & Larsen, R. (2009). *Time-of-flight sensors in computer graphics.* Paper presented at the Proc. Eurographics (State-of-the-Art Report).

Konolige, K., & Agrawal, M. (2008). FrameSLAM: From bundle adjustment to real-time visual mapping. *IEEE Transactions on Robotics, 24*(5), 1066-1077.

Konolige, K., Agrawal, M., Bolles, R. C., Cowan, C., Fischler, M., & Gerkey, B. (2008). *Outdoor mapping and navigation using stereo vision.* Paper presented at the Experimental Robotics.

Krajnık, T., Fentanes, J. P., Cielniak, G., Dondrup, C., & Duckett, T. (2014). *Spectral analysis for long-term robotic mapping.* Paper presented at the International Conference on Robotics and Automation (ICRA).

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). *Imagenet classification with deep convolutional neural networks.* Paper presented at the Advances in neural information processing systems.

Kschischang, F. R., Frey, B. J., & Loeliger, H.-A. (2001). Factor graphs and the sum-product algorithm. *IEEE Transactions on information theory, 47*(2), 498-519.

Kümmerle, R., Grisetti, G., Strasdat, H., Konolige, K., & Burgard, W. (2011). *g 2 o: A general framework for graph optimization.* Paper presented at the Robotics and Automation (ICRA), 2011 IEEE International Conference on.

Kümmerle, R., Ruhnke, M., Steder, B., Stachniss, C., & Burgard, W. (2015). Autonomous robot navigation in highly populated pedestrian zones. *Journal of Field Robotics, 32*(4), 565-589.

Langmann, B., Hartmann, K., & Loffeld, O. (2012). *Depth Camera Technology Comparison and Performance Evaluation.* Paper presented at the ICPRAM (2).

Lee, Y.-J., & Song, J.-B. (2007). *Autonomous selection, registration, and recognition of objects for visual SLAM in indoor environments.* Paper presented at the Control, Automation and Systems, 2007. ICCAS'07. International Conference on.

Li, M., Zhang, M., Fu, Y., Guo, W., Zhong, X., Wang, X., & Chen, F. (2016). *Fast and robust mapping with low-cost Kinect V2 for photovoltaic panel cleaning robot.* Paper presented at the Advanced Robotics and Mechatronics (ICARM), International Conference on.

Li, W. (1994). *Perception-action'behavior control of a mobile robot in uncertain environments using fuzzy logic.* Paper presented at the Intelligent Robots and Systems' 94.'Advanced Robotic Systems and the Real World', IROS'94. Proceedings of the IEEE/RSJ/GI International Conference on.

Li, Y., Zhu, J., & Hoi, S. C. (2015). *Reliable patch trackers: Robust visual tracking by exploiting reliable patches.* Paper presented at the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.

Ling, L., Cheng, E., & Burnett, I. S. (2013). *An iterated extended Kalman filter for 3D mapping via Kinect camera.* Paper presented at the 2013 IEEE International Conference on Acoustics, Speech and Signal Processing.

Little, J., & Boyd, J. (1998). Recognizing people by their gait: the shape of motion. *Videre: Journal of Computer Vision Research, 1*(2), 1-32.

Lopez-Molina, C., De Baets, B., Bustince, H., Sanz, J., & Barrenechea, E. (2013). Multiscale edge detection based on Gaussian smoothing and edge tracking. *Knowledge-Based Systems, 44*, 101-111.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision, 60*(2), 91-110.

Ltd, A. T. P. (n.d.). SwissRanger SR4000. Retrieved from http://www.adept.net.au/cameras/Mesa/SR4000.shtml

Ma, L., Kerl, C., Stückler, J., & Cremers, D. (2016). *Cpa-slam: Consistent plane-model alignment for direct rgb-d slam.* Paper presented at the Robotics and Automation (ICRA), 2016 IEEE International Conference on.

Magree, D. P., & Johnson, E. N. (2015). *A monocular vision-aided inertial navigation system with improved numerical stability.* Paper presented at the AIAA Guidance, Navigation, and Control Conference.

Maier, R., Sturm, J., & Cremers, D. (2014). *Submap-based bundle adjustment for 3D reconstruction from RGB-D data.* Paper presented at the German Conference on Pattern Recognition.

Manders, C., Farbiz, F., Chong, J. H., Tang, K. Y., Chua, G. G., Loke, M. H., & Yuan, M. (2008). *Robust hand tracking using a skin tone and depth joint probability model.* Paper presented at the Automatic Face & Gesture Recognition, 2008. FG'08. 8th IEEE International Conference on.

MathWorks. (n.d.). Camera Calibration. Retrieved from https://au.mathworks.com/help/vision/camera-calibration.html

McManus, C., Churchill, W., Maddern, W., Stewart, A. D., & Newman, P. (2014). *Shady dealings: Robust, long-term visual localisation using illumination invariance.* Paper presented at the Robotics and Automation (ICRA), 2014 IEEE International Conference on.

Medioni, G., & Yasumoto, Y. (1986). *Corner detection and curve representation using cubic B-splines.* Paper presented at the Robotics and Automation. Proceedings. 1986 IEEE International Conference on.

Microsoft. Kinect hardware. Retrieved from https://developer.microsoft.com/en-us/windows/kinect/hardware

Microsoft. (n..d.). Kinect for Xbox one. Retrieved from http://www.xbox.com/en-US/xbox-one/accessories/kinect

Microsoft. (n.d.). Xbox Support. Retrieved from http://support.xbox.com/en-US/xbox-on-windows/accessories/kinect-for-windows-v2-setup

Minguez, J., & Montano, L. (2000). *Nearness diagram navigation (nd): A new real time collision avoidance approach.* Paper presented at the Intelligent Robots and Systems, 2000.(IROS 2000). Proceedings. 2000 IEEE/RSJ International Conference on.

Minguez, J., & Montano, L. (2005). Sensor-based robot motion generation in unknown, dynamic and troublesome scenarios. *Robotics and Autonomous Systems, 52*(4), 290-311.

Mitkov, R. (2005). *The Oxford handbook of computational linguistics*: Oxford University Press.

Mokhtarian, F., & Suomela, R. (1998). Robust image corner detection through curvature scale space. *Pattern Analysis and Machine Intelligence, IEEE Transactions on, 20*(12), 1376-1381.

Montemerlo, M., Thrun, S., Koller, D., & Wegbreit, B. (2002). *FastSLAM: A factored solution to the simultaneous localization and mapping problem.* Paper presented at the AAAI/IAAI.

Moosmann, F. (2013). *Interlacing self-localization, moving object tracking and mapping for 3d range sensors* (Vol. 24): KIT Scientific Publishing.

Moravec, H. P. (1977). *TOWARDS AUTOMATIC VISUAL BBSTACLE AVOIDANCE.* Paper presented at the International Conference on Artificial Intelligence (5th: 1977: Massachusetts Institute of Technology).

Nanda, H., & Fujimura, K. (2004). *A robust elliptical head tracker.* Paper presented at the Automatic Face and Gesture Recognition, 2004. Proceedings. Sixth IEEE International Conference on.

Nandi, C. S., Tudu, B., & Koley, C. (2014). A machine vision-based maturity prediction system for sorting of harvested mangoes. *IEEE Transactions on Instrumentation and measurement, 63*(7), 1722-1730.

Neira, J., & Tardós, J. D. (2001). Data association in stochastic mapping using the joint compatibility test. *Robotics and Automation, IEEE Transactions on, 17*(6), 890-897.

Neubeck, A., & Van Gool, L. (2006). *Efficient non-maximum suppression.* Paper presented at the Pattern Recognition, 2006. ICPR 2006. 18th International Conference on.

New Zealand KiWibots. KiwiBots,. Retrieved from http://www.kiwibots.co.nz/

Newcombe, R. A., Fox, D., & Seitz, S. M. (2015). *Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time.* Paper presented at the Proceedings of the IEEE conference on computer vision and pattern recognition.

Newcombe, R. A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A. J., . . . Fitzgibbon, A. (2011). *KinectFusion: Real-time dense surface mapping and tracking.* Paper presented at the Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on.

Nguyen, C. V., Izadi, S., & Lovell, D. (2012). *Modeling kinect sensor noise for improved 3d reconstruction and tracking.* Paper presented at the 3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT), 2012 Second International Conference on.

Nieto, J., Guivant, J., & Nebot, E. (2006). Denseslam: Simultaneous localization and dense mapping. *The international journal of Robotics Research, 25*(8), 711-744.

Obdržálek, Š., Kurillo, G., Ofli, F., Bajcsy, R., Seto, E., Jimison, H., & Pavel, M. (2012). *Accuracy and robustness of Kinect pose estimation in the context of coaching of elderly population.* Paper presented at the 2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society.

Olson, E., Leonard, J., & Teller, S. (2006). *Fast iterative alignment of pose graphs with poor initial estimates.* Paper presented at the Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on.

OpenCV. (2017). Camera calibration with OpencV. Retrieved from http://docs.opencv.org/2.4/doc/tutorials/calib3d/camera_calibration/camera_calibration.html?

OpenCV. (n.d.). Camera Calibration and 3D Reconstruction. Retrieved from http://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html

OpenNI. NITE 2.2.0.11. Retrieved from http://openni.ru/files/nite/index.html

Parida, L., Geiger, D., & Hummel, R. (1998). Junctions: Detection, classification, and reconstruction. *Pattern Analysis and Machine Intelligence, IEEE Transactions on, 20*(7), 687-698.

Philippsen, R., & Siegwart, R. (2003). *Smooth and efficient obstacle avoidance for a tour guide robot.* Paper presented at the None.

Piccardi, M. (2004). *Background subtraction techniques: a review.* Paper presented at the Systems, man and cybernetics, 2004 IEEE international conference on.

Pikaz, A., & Dinstein, I. H. (1994). Using simple decomposition for smoothing and feature point detection of noisy digital curves. *Pattern Analysis and Machine Intelligence, IEEE Transactions on, 16*(8), 808-813.

PMDec. (n.d.). the new CamBoard pico flexx. Retrieved from http://pmdtec.com/picoflexx/

Pollefeys, M., Nistér, D., Frahm, J.-M., Akbarzadeh, A., Mordohai, P., Clipp, B., . . . Merrell, P. (2008). Detailed real-time urban 3d reconstruction from video. *International journal of computer vision, 78*(2-3), 143-167.

Polok, L., Ila, V., Solony, M., Smrz, P., & Zemcik, P. (2013). *Incremental Block Cholesky Factorization for Nonlinear Least Squares in Robotics.* Paper presented at the Robotics: Science and Systems.

PrimeSense. (2011). PrimeSense Natural Interactor.

Rafiq, A., Makroo, H. A., Sachdeva, P., & Sharma, S. (2013). Application of Computer Vision System in Food Processing - A Review. *Journal of Engineering Research and Applications, 3*(6), 1197-1205.

Raposo, C. (n.d.). EasyKinCal. Retrieved from http://arthronav.isr.uc.pt/~carolina/kinectcalib/

Raposo, C., Barreto, J. P., & Nunes, U. (2013). *Fast and accurate calibration of a kinect sensor.* Paper presented at the 3DTV-Conference, 2013 International Conference on.

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). *You only look once: Unified, real-time object detection.* Paper presented at the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.

Reisfeld, D., Wolfson, H., & Yeshurun, Y. (1995). Context-free attentional operators: the generalized symmetry transform. *International journal of computer vision, 14*(2), 119-130.

Robotics, V. Competion Resource. Retrieved from http://www.vexrobotics.com/vex/competition/competition-resources

Robotics, V. Toss Up. Retrieved from http://www.vexrobotics.com/wiki/Toss_Up

Robotics, V. (2017a). Compete on the world stage. Retrieved from https://www.vexrobotics.com/vexedr/competition

Robotics, V. (2017b). VEX IQ. Retrieved from https://www.vexrobotics.com/vexiq/

Röwekämper, J., Sprunk, C., Tipaldi, G. D., Stachniss, C., Pfaff, P., & Burgard, W. (2012). *On the position accuracy of mobile robot localization based on particle filters combined with scan matching.* Paper presented at the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems.

Rublee, E., Rabaud, V., Konolige, K., & Bradski, G. (2011). *ORB: an efficient alternative to SIFT or SURF.* Paper presented at the Computer Vision (ICCV), 2011 IEEE International Conference on.

Russell, S., Norvig, P., & Intelligence, A. (1995). A modern approach. *Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs, 25*.

Sadat, S. A., Chutskoff, K., Jungic, D., Wawerla, J., & Vaughan, R. (2014). *Feature-rich path planning for robust navigation of mavs with mono-slam.* Paper presented at the Robotics and Automation (ICRA), 2014 IEEE International Conference on.

Schmid, C., Mohr, R., & Bauckhage, C. (2000). Evaluation of interest point detectors. *International journal of computer vision, 37*(2), 151-172.

Segal, A., Haehnel, D., & Thrun, S. (2009). *Generalized-ICP.* Paper presented at the Robotics: Science and Systems.

Senthilkumaran, N., & Rajesh, R. (2009). Edge detection techniques for image segmentation–a survey of soft computing approaches. *International journal of recent trends in engineering, 1*(2).

Sezgin, M. (2004). Survey over image thresholding techniques and quantitative performance evaluation. *Journal of Electronic imaging, 13*(1), 146-168.

Shin, M. C., Goldgof, D., & Bowyer, K. W. (1998). *An objective comparison methodology of edge detection algorithms using a structure from motion task.* Paper presented at the Computer Vision and Pattern Recognition, 1998. Proceedings. 1998 IEEE Computer Society Conference on.

Sitthi-Amorn, P., Ramos, J. E., Wangy, Y., Kwan, J., Lan, J., Wang, W., & Matusik, W. (2015). MultiFab: a machine vision assisted platform for multi-material 3D printing. *ACM Transactions on Graphics (TOG), 34*(4), 129.

Smisek, J., Jancosek, M., & Pajdla, T. (2013). 3D with Kinect *Consumer depth cameras for computer vision* (pp. 3-25): Springer.

Smith, R., Self, M., & Cheeseman, P. (1990). Estimating uncertain spatial relationships in robotics *Autonomous robot vehicles* (pp. 167-193): Springer.
Smith, R. C., & Cheeseman, P. (1986). On the representation and estimation of spatial uncertainty. *The international journal of Robotics Research, 5*(4), 56-68.
Sola, J. (2010). *Consistency of the monocular ekf-slam algorithm for three different landmark parametrizations.* Paper presented at the Robotics and Automation (ICRA), 2010 IEEE International Conference on.
Song, S., Lichtenberg, S. P., & Xiao, J. (2015). *Sun rgb-d: A rgb-d scene understanding benchmark suite.* Paper presented at the Proceedings of the IEEE conference on computer vision and pattern recognition.
Sridharan, M., & Stone, P. (2007). Structure-based color learning on a mobile robot under changing illumination. *Autonomous Robots, 23*(3), 161-182.
Starner, T. E. (1995). *Visual Recognition of American Sign Language Using Hidden Markov Models*. Retrieved from
Steder, B. (2013). Feature-based 3D perception for mobile robots. *Universitätsbibliothek Freiburg, Doctoral dissertation*.
Steinbrücker, F., Sturm, J., & Cremers, D. (2014). *Volumetric 3d mapping in real-time on a cpu.* Paper presented at the Robotics and Automation (ICRA), 2014 IEEE International Conference on.
Stentz, A. (1994). *Optimal and efficient path planning for partially-known environments.* Paper presented at the Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on.
Stentz, A. (1995). *The focussed D^* algorithm for real-time replanning.* Paper presented at the IJCAI.
Strasdat, H., Montiel, J., & Davison, A. J. (2010). *Scale Drift-Aware Large Scale Monocular SLAM.* Paper presented at the Robotics: Science and Systems.
Strasdat, H., Montiel, J. M., & Davison, A. J. (2012). Visual SLAM: why filter? *Image and Vision Computing, 30*(2), 65-77.
Sumengen, B., Manjunath, B., & Kenney, C. (2003). *Image segmentation using multi-region stability and edge strength.* Paper presented at the Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on.
Sun, T.-H., Tien, F.-C., Tien, F.-C., & Kuo, R.-J. (2016). Automated thermal fuse inspection using machine vision and artificial neural networks. *Journal of Intelligent Manufacturing, 27*(3), 639-651.
Szegedy, C., Toshev, A., & Erhan, D. (2013). *Deep neural networks for object detection.* Paper presented at the Advances in Neural Information Processing Systems.
Ta, D.-N., Chen, W.-C., Gelfand, N., & Pulli, K. (2009). *Surftrac: Efficient tracking and continuous object recognition using local feature descriptors.* Paper presented at the Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on.
Taguchi, Y., Jian, Y.-D., Ramalingam, S., & Feng, C. (2013). *Point-plane SLAM for hand-held 3D sensors.* Paper presented at the Robotics and Automation (ICRA), 2013 IEEE International Conference on.

Tamersoy, B. (2009). Background subtraction. *University of Texas. http://userweb. cs. utexas. edu/~ grauman/courses/fall2009/slides/ lecture9_background. pdf. Diakses tanggal, 5*.

Tamjidi, A. H., Taghirad, H. D., & Aghamohammadi, A. A. (2009). *On the consistency of EKF-SLAM: Focusing on the observation models.* Paper presented at the IROS.

Thrun, S. (2001). A probabilistic on-line mapping algorithm for teams of mobile robots. *The international journal of Robotics Research, 20*(5), 335-363.

Thrun, S., Burgard, W., & Fox, D. (1998). A probabilistic approach to concurrent mapping and localization for mobile robots. *Autonomous Robots, 5*(3-4), 253-271.

Thrun, S., & Montemerlo, M. (2006). The graph SLAM algorithm with applications to large-scale mapping of urban structures. *The International Journal of Robotics Research, 25*(5-6), 403-429.

Tolman, E. C. (1948). Cognitive maps in rats and men. *Psychological review, 55*(4), 189.

Trevor, A. J., Rogers, J., & Christensen, H. I. (2012). *Planar surface SLAM with 3D and 2D sensors.* Paper presented at the Robotics and Automation (ICRA), 2012 IEEE International Conference on.

Tribou, M. J., Wang, D. W., & Waslander, S. L. (2016). Degenerate motions in multicamera cluster SLAM with non-overlapping fields of view. *Image and Vision Computing, 50*, 27-41.

Trier, O. D., & Jain, A. K. (1995). Goal-directed evaluation of binarization methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 17*(12), 1191-1201.

Trimble, G. M., & Belcher, E. (2002). *Ship berthing and hull inspection using the CetusII AUV and MIRIS high-resolution sonar.* Paper presented at the OCEANS'02 MTS/IEEE.

Tuytelaars, T., & Mikolajczyk, K. (2008). Local invariant feature detectors: a survey. *Foundations and Trends® in Computer Graphics and Vision, 3*(3), 177-280.

Ulrich, I., & Borenstein, J. (1998). *VFH+: Reliable obstacle avoidance for fast mobile robots.* Paper presented at the Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on.

Unay, D., & Gosselin, B. (2007). Stem and calyx recognition on 'Jonagold'apples by pattern recognition. *Journal of Food Engineering, 78*(2), 597-605.

Vaganay, J., Elkins, M., Esposito, D., O'Halloran, W., Hover, F., & Kokko, M. (2006). *Ship hull inspection with the HAUV: US Navy and NATO demonstrations results.* Paper presented at the OCEANS 2006.

Van Dalen, G. J., Magree, D. P., & Johnson, E. N. (2016). *Absolute localization using image alignment and particle filtering.* Paper presented at the AIAA Guidance, Navigation, and Control Conference.

Vandewouw, M. M., Aleman, D. M., & Jaffray, D. A. (2016). Robotic path-finding in inverse treatment planning for stereotactic radiosurgery with continuous dose delivery. *Medical Physics, 43*(8), 4545-4557.

VEX Robotics, I. Skyrise. Retrieved from http://www.vexrobotics.com/wiki/Skyrise

VEX Robotics, I. VEX Robotics. Retrieved from http://www.vexrobotics.com/?ref=logo

von Stumberg, L., Usenko, V., Engel, J., Stückler, J., & Cremers, D. (2016). Autonomous Exploration with a Low-Cost Quadrocopter using Semi-Dense Monocular SLAM. *arXiv preprint arXiv:1609.07835*.

Wah, B. W. (2007). *Wiley encyclopedia of computer science and engineering*: Wiley-Interscience.

Wang, C.-C., Thorpe, C., & Thrun, S. (2003). *Online simultaneous localization and mapping with detection and tracking of moving objects: Theory and results from a ground vehicle in crowded urban areas.* Paper presented at the Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on.

Wang, M., & Liu, J. N. (2008). Fuzzy logic-based real-time robot navigation in unknown environment with dead ends. *Robotics and Autonomous Systems, 56*(7), 625-643.

Wang, Y.-T., Lin, M.-C., & Ju, R.-C. (2010). Visual SLAM and moving object detection for a small-size humanoid robot. *International Journal of Advanced Robotic Systems, 7*(2), 133-138.

Wei, W., & Yunxiao, A. (2009). *Vision-based human motion recognition: A Survey.* Paper presented at the Intelligent Networks and Intelligent Systems, 2009. ICINIS'09. Second International Conference on.

Weiss, A., Hirshberg, D., & Black, M. J. (2011). *Home 3D body scans from noisy image and range data.* Paper presented at the 2011 International Conference on Computer Vision.

Wiedemeyer, T. (2016). Kinect2 Calibration. Retrieved from https://github.com/code-iai/iai_kinect2/tree/master/kinect2_calibration

Wieser, I., Ruiz, A. V., Frassl, M., Angermann, M., Mueller, J., & Lichtenstern, M. (2014). *Autonomous robotic SLAM-based indoor navigation for high resolution sampling with complete coverage.* Paper presented at the Position, Location and Navigation Symposium-PLANS 2014, 2014 IEEE/ION.

Wolf, D., & Sukhatme, G. (2004). *Online simultaneous localization and mapping in dynamic environments.* Paper presented at the Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on.

Wu, C.-H. D., & Wetzstein, G. Automated Restyling of Human Portrait Based on Facial Expression Recognition and 3D Reconstruction.

Xia, L., Chen, C.-C., & Aggarwal, J. K. (2011). *Human detection using depth information by kinect.* Paper presented at the CVPR 2011 WORKSHOPS.

Yang, L., Zhang, L., Dong, H., Alelaiwi, A., & El Saddik, A. (2015). Evaluating and improving the depth accuracy of Kinect for Windows v2. *IEEE Sensors Journal, 15*(8), 4275-4285.

Yang SX, M. M., Patel RV (2005). A layered goal-oriented fuzzy motion planning strategy for mobile robot navigatio. *IEEE transactions on systems, man, and cybernetics—part b: cybernetics. , 35*(6), 1214-1224.

Yao, B., & Li, F.-F. (2010). *Modeling mutual context of object and human pose in human-object interaction activities.* Paper presented at the Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on.

Zagoruyko, S., Lerer, A., Lin, T.-Y., Pinheiro, P. O., Gross, S., Chintala, S., & Dollár, P. (2016). A multipath network for object detection. *arXiv preprint arXiv:1604.02135*.

Zennaro, S., Munaro, M., Milani, S., Zanuttigh, P., Bernardi, A., Ghidoni, S., & Menegatti, E. (2015). *Performance evaluation of the 1st and 2nd generation Kinect for multimedia applications.* Paper presented at the Multimedia and Expo (ICME), 2015 IEEE International Conference on.

Zhang, D., Liu, Y., & Hou, J. (2008). *Digital image retrieval using intermediate semantic features and multistep search.* Paper presented at the Digital Image Computing: Techniques and Applications (DICTA), 2008.

Zhou, H., & Sakane, S. (2008). *Localizing objects during robot SLAM in semi-dynamic environments.* Paper presented at the Advanced Intelligent Mechatronics, 2008. AIM 2008. IEEE/ASME International Conference on.

Ziou, D., & Tabbone, S. (1998). Edge detection techniques-an overview. *Pattern Recognition and Image Analysis C/C of Raspoznavaniye Obrazov I Analiz Izobrazhenii, 8*, 537-559.