3D Indoor Mobile Mapping using Multi-Sensor Autonomous Robot

Mostofi, Navid

http://hdl.handle.net/11023/2580

Downloaded from PRISM Repository, University of Calgary
3D Indoor Mobile Mapping using Multi-Sensor Autonomous Robot

by

Navid Mostofi

A THESIS
SUBMITTED TO THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE
DEGREE OF DOCTOR OF PHILOSOPHY

GRADUATE PROGRAM IN GEOMATICS ENGINEERING

CALGARY, ALBERTA

September, 2015

© Navid Mostofi 2015
Abstract

Autonomous indoor mobile mapping has opened up new horizon in the field of surveying and mapping industry. The ability to create a 3D map without user intervention not only reduces labour costs but also provides more flexibility for exploring remote sites. Hence, it is worthwhile to consider the role of robotics in the mapping industry.

The primary demand for autonomous robot systems is to interact with environment for obstacle avoidance and self-localization in six degrees of freedom (x-, y-, z-position, roll, yaw and pitch angle). The later issue requires knowledge of the operating environment, which leads to automatic environment modeling or environment mapping solution.

Two different scenarios for autonomous indoor mobile mapping are investigated in this thesis. The first scenario is based on the use of a single RGB-D sensor to map a small room of size (8x8 meter). In the second scenario RGB-D sensor is used as an aiding sensor for Velodyne HDL-32 LiDAR to map a large corridor of size (33x11 meter). The results shows that the solution of single RGB-D sensor is accurate enough for mapping a small room; however, for large corridor the result of RGB-D aided Velodyne HDL-32 generated more accurate and consistent mapping solution.

The main challenge that should be handled for autonomous mapping is alignment of multiple local scans as they become locally distorted because of the motion of the platform and noise in sensor measurements. The collected scans from multiple locations are associated with the individual sensor locations (the capturing process is done using stop-and-go approach, where the robot is stopped at different locations to capture the scene). Hence, a registration process must be performed in order to combine several scans at different locations. The main goal of the registration process is to estimate the transformation parameters, which will define the relation between the collected datasets from different locations.
The optimization and enhancement of the registration procedure plays a major role for generating indoor mobile mapping solution. The problem of alignment is addressed through several optimization steps, starting from coarse registration, followed by fine registration, segmentation and finally loops closure.
Acknowledgements

I would like to express my sincere thanks to my advisor, Dr. Naser El-Sheimy for his generous advice, inspiring guidance and encouragement throughout my graduate studies, research, and dissertation work. His advice on both research as well as on my career have been priceless.

I would also like to thank my supervisor Dr. El-Habiby for his advice and feedback on my work. This thesis could not have been completed without their generous and professional assistance.

I should say that I am very happy that I involved in a project that force me to dig into different fields and gain valuable knowledge.

I also appreciate my supervisors committee” Dr. Noureldin, Dr. Ayman Habib, Dr. Michael Chapman, and Dr. Mamdouh El-Badry” for carefully reading and providing comments concerning various aspects of this research.

I would like to also thank my colleagues in the MMS research group for their help throughout my project.

A special thanks to Sanam, for her support throughout my Phd. Also thanks to my friends Mohammad and Siavash.

A special thanks to my family. Words cannot express how grateful I am to my mother, father and my sister for all they have done for me.
Dedication

To my beloved Parents

Naser and Esmat

and my sister

Naghmeh
Table of Contents

Abstract .................................................................................................................. i
Acknowledgements .......................................................................................... iii
Dedication ........................................................................................................... iv
Table of Contents ............................................................................................... v
List of Tables .......................................................................................................... viii
List of Figures and Illustrations ........................................................................ ix
List of Symbols ....................................................................................................... xiii
List of Abbreviations ........................................................................................... xv

CHAPTER ONE: INTRODUCTION ....................................................................... 1
  1.1 Motivation ....................................................................................................... 1
  1.2 Problem statement ....................................................................................... 2
  1.3 Research Objective ..................................................................................... 7
  1.4 Contribution ................................................................................................ 9
  1.5 Thesis Outline ............................................................................................. 10

CHAPTER TWO: BACKGROUND ......................................................................... 12
  2.1 Introduction .................................................................................................. 12
  2.2 Indoor Mobile Mapping System using a Robot ........................................... 12
  2.3 Indoor Mobile Mapping System using Handheld devices ......................... 15
  2.4 Indoor Mobile Mapping System using Trolley devices ............................. 17
  2.5 Indoor Mobile Mapping System using an UAV .......................................... 19
  2.6 Overview of Sensors for Indoor Mobile Mapping System ....................... 21
  2.7 Overview of Map Representation ................................................................ 28
      2.7.1 Octree based mapping ........................................................................ 29
      2.7.2 Triangular mesh based mapping ......................................................... 31
      2.7.3 Convex planar based mapping ............................................................ 34
      2.7.4 Segmentation based mapping ............................................................... 35

CHAPTER THREE: COARSE REGISTRATION ..................................................... 36
  3.1 Introduction .................................................................................................. 36
  3.2 Feature based registration (Visual Odometry) ............................................ 36
  3.3 Feature detection ......................................................................................... 38
  3.4 Feature matching ........................................................................................ 40
  3.5 Matching strategy and error rates ............................................................... 40
      3.5.1 Efficient matching .............................................................................. 43
      3.5.2 Feature match verification ................................................................. 43
  3.6 Feature-based pose estimation .................................................................... 44
      3.6.1 2D-to-2D feature based pose estimation ........................................... 45
      3.6.2 3D-to-2D feature based pose estimation ........................................... 48
      3.6.3 3D-to-3D feature based pose estimation ........................................... 49
          3.6.3.1 Stereo based visual odometry ...................................................... 49
          3.6.3.2 RGB-D-based Visual Odometry ............................................... 59
CHAPTER SIX: EXPERIMENTAL RESULTS ............................................. 101
 6.1 Introduction ........................................................................... 101
 6.2 Hardware Description ............................................................ 101
 6.2.1 RGB-D camera .................................................................. 101
 6.2.2 Velodyne HDL-32 ............................................................... 102
 6.2.3 SICK LMS-111 Rangeﬁnder .................................................. 103

3.6.3.3 Surface Normals and Curvature Estimates ................................61
3.6.3.4 Normal Aligned Radial Feature (NARF) ................................. 62
3.7 Calibration of visual sensor .......................................................... 64

CHAPTER FIVE: GLOBAL OPTIMIZATION ............................................ 87
 5.1 Introduction ........................................................................... 87
 5.2 Segmentation ......................................................................... 87
 5.2.1 Segmentation based on clustering of features ......................... 88
 5.2.2 Segmentation based on surface growing .....................................88
 5.2.2.1 Identiﬁcation of seed surface: .............................................. 88
 5.2.2.2 Growing of seed surface ..................................................... 88
 5.2.3 Segmentation by model ﬁtting ................................................ 90
 5.2.3.1 RANSAC ........................................................................ 90
 5.2.4 Hybrid segmentation technique .............................................. 91
 5.3 Transformation from segmentation ............................................ 91
 5.4 Loop closure ......................................................................... 92
 5.4.1 Bag-of-Words (BoW) ............................................................. 93
 5.4.2 FAB-MAP ........................................................................... 95
 5.4.3 Explicit Loop Closing Heuristic (ELCH) ....................................95
 5.4.4 Loop Closing based on manual registration .............................. 96
 5.5 Obstacle avoidance .................................................................. 97

CHAPTER FOUR: FINE REGISTRATION .............................................. 71
 4.1 Introduction ........................................................................... 71
 4.2 Registration .......................................................................... 72
 4.3 General Registration Pipeline ................................................... 74
 4.3.1 Selection of Points .............................................................. 74
 4.3.2 Matching Points .................................................................. 75
 4.3.3 Weighting of Pairs .............................................................. 75
 4.3.4 Rejecting Pairs .................................................................... 76
 4.3.5 Error Metric and Minimization .............................................. 76
 4.4 ICP point-to-point minimization ............................................... 77
 4.4.1 Direct Solutions of the ICP Error Function ............................... 78
 4.4.2 Closed Form Solution .......................................................... 78
 4.4.3 Linearized Solution of the ICP Error Function ......................... 80
 4.5 ICP Point-to-Plane Minimization .............................................. 81
 4.6 ICP Plane-to-Plane Minimization .............................................. 83
 4.7 ICP Edge based Minimization ................................................... 84

6.2 Hardware Description ............................................................ 61
6.1 Introduction ........................................................................... 61
6.2 Introduction ........................................................................... 62
6.3 Calibration of visual sensor ........................................................ 64
List of Tables

Table 6-1. The specifications of the RGB-D camera (Kinect) ........................................... 102
Table 6-2. The specifications of the Velodyne HDL-32E laser scanner ................................. 102
Table 6-3 The specifications of the SICK LMS-111 laser range finder .................................. 103
Table 6-4. Iterative Closest Point (ICP) algorithm ................................................................. 120
Table 6-5: Performance of RGB-D coarse registration over 14 sequences comparing to ICP point-to-plane of LiDAR scans .............................................................. 125
List of Figures and Illustrations

Figure 1.1. The proposed framework for 3D Mapping................................................................. 9

Figure 2.1. "Kurt3D", Intelligent Robot for Mapping Applications in 3D “Irma3D”, “AZIMUT-3”, “Mobile robot”.................................................................................. 13

Figure 2.2. Handheld mobile mapping system: backpacking system Irma3D, Zebedee handheld 3D mapping system...................................................................................... 15

Figure 2.3. Scannect equipped with two sideward facing Kinects, MEMS-based IMU from Xsens and FARO Focus3D terrestrial laser scanning (TLS), Modified Pioneer AT robot.. 18

Figure 2.4. Trimble Indoor Mapping Solution (TIMMS).................................................................. 19

Figure 2.5. Quadrotor micro air vehicle (MAV), CityFlyer micro-UAV platform ......................... 20

Figure 2.6. Seekur Jr .......................................................................................................................... 21

Figure 2.7 Structured light (blue) and time-of-flight sensors (black) time line ......................... 24

Figure 2.8. Point cloud generated from different sensors (SICK LMS (2D laser range finder), RGB-D camera, LiDAR, Velodyne HDL-32, Thermal camera)................................. 28

Figure 2.9. Octree representation of Stanford Bunny point cloud, with a leaf size of 1.5cm...... 30

Figure 2.10. Triangle meshes approximating of Stanford Bunny.................................................. 32

Figure 2.11. Convex planar patches approximating of Stanford Bunny........................................ 35

Figure 3.1. Flowchart of transformation estimation from set of image frames ......................... 38

Figure 3.2. False positives and negatives of feature detection for the matching process .......... 41

Figure 3.3. Fixed threshold, nearest neighbour, and Nearest Neighbour Distance Ratio (NNDR) matching........................................................................................................ 42

Figure 3.4. Feature-based alignment methods .................................................................................. 45

Figure 3.5. Epipolar geometry ........................................................................................................ 46

Figure 3.6: For every planes passing through epipole, there is a corresponding epipolar line $l$ and $l'$, in the respective image planes ................................................................. 47

Figure 3.7. Schematic of the pose estimation algorithm. Initialization to get a key-frame and store the properties. .................................................................................................. 50
Figure 3.8 Schematic of the pose estimation algorithm. Loop to read images, find key-points, find correspondences in the key-frame, then find correspondences between left and right, triangulate and finally estimate the pose. ................................................................. 52

Figure 3.9. Left-right matching using rectified images: allow a small tolerance to compensate for imperfect rectification. ................................................................. 53

Figure 3.10. Feature matching (2 frames, moving camera), colors encode disparities .......... 54

Figure 3.11. Schematic of matching without using descriptors ........................................... 55

Figure 3.12. Feature tracking and matching (5 frames, stereo camera), colors encode track orientation. ............................................................................................................ 55

Figure 3.13. Optical flow tracking on stereo data .............................................................. 56

Figure 3.14. Stereo visual odometry, Results on the Karlsruhe data set (Kitt, 2010) .......... 57

Figure 3.15. Reconstruction of the scene from triangulation (Kitt, 2010) ......................... 58

Figure 3.16. Visual odometry process .................................................................................. 60

Figure 3.17. NARF keypoint computation ........................................................................... 63

Figure 3.18. RGB-D sensor (Khoshelham, 2012) ............................................................... 65

Figure 3.19. Parameters needed for calibration ................................................................... 67

Figure 3.20. Software interface for camera calibration ....................................................... 70

Figure 4.1 Overview of the registering process using a pair of point clouds ....................... 73

Figure 4.2. Point-to-plane error between two surfaces ...................................................... 82

Figure 4.3. Edge extraction from RGB image using canny edge detection ......................... 85

Figure 5.1. Planar patch segmentation on RGB-D point clouds using region growing followed by convexhull ......................................................................................... 89

Figure 5.2. Finding the angle between the normal vectors in two consecutive frames and compensating for accumulated error ................................................................. 92

Figure 5.3. Image appearance-based modeling process ...................................................... 94

Figure 5.4. Loop closure detection ..................................................................................... 97

Figure 5.5. Simulation for obstacle detection using laser range scanning (blue circle is the position that is define for the robot to move there, red lines are laser measurements) ...... 100
Figure 6.1. Images of the experiment room ................................................................. 104

Figure 6.2. Flowchart of 3D mapping using single RGB-D sensor ................................. 105

Figure 6.3 Down-sampling the image using Gaussian pyramid, Left (level 0 no down-
sampling), right (level 1, one level down-sampling), red circle is a radius search for
feature detection ............................................................................................................ 106

Figure 6.4. Left: the locations of features in the image. Right: the corresponding location of
the 2D features in 3D space using depth information ................................................ 107

Figure 6.5 Software architecture for mapping using RGB-D camera ............................. 108

Figure 6.6. 3D modelling before loop closure .................................................................. 109

Figure 6.7. 3D modelling after loop closure .................................................................. 110

Figure 6.8. Accuracy of 3d modelling compare to true value ........................................ 110

Figure 6.9. RMS error of visual odometry and ICP and visual odometry alone ................ 111

Figure 6.10. VO translation estimate (tx, ty, tz) ............................................................. 112

Figure 6.11. Visual odometry rotation estimation (Roll, Pitch, Yaw) ............................. 113

Figure 6.12. University of Calgary CCIT building third floor ........................................ 114

Figure 6.13. Position of the robot at scan time .............................................................. 115

Figure 6.14. Drift in robot pose estimation (blue) due to the drift in visual odometry
(magenta) ..................................................................................................................... 116

Figure 6.15. Drift of single RGB-D sensor for mapping of large corridor ......................... 116

Figure 6.16. (Left) RGB image and (right) depth information captured by an RGB-D camera 117

Figure 6.17. Flowchart of the autonomous 3D mapping ............................................... 118

Figure 6.18. RGB-D 2D and 3D matching Left: 2D-2D matching using SURF from two
consecutive frames. Right: NARF key-point .................................................................. 119

Figure 6.19 Segmentation procedure ............................................................................ 121

Figure 6.20. Velodyne HDL-32E scan, in the right, the velodyne scan after being filtered by a
voxel grid with a leaf size of 0.01 .............................................................................. 123

Figure 6.21. Result of two consecutive LiDAR scans before and after fine registration using
ICP ................................................................................................................................. 124
Figure 6.22. RMS error of standard ICP, point-to-plane ICP and plane-to plane ICP with dmax=1 ................................................................. 125

Figure 6.23. Planar segmentation (10 planar patch segments) from two consecutive RGB-D frames .................................................................................................................. 127

Figure 6.24. Schematic of university of Calgary CCIT building third floor with true values for length and width ................................................................................................. 128

Figure 6.25. 3D Mapping using LiDAR after plane-to plane registration and loop closure ...... 129
**List of Symbols**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \vec{b} )</td>
<td>Image Base (Vector between Two Perspective Centers of the Stereopair)</td>
</tr>
<tr>
<td>( C_i^P )</td>
<td>Covariance of P</td>
</tr>
<tr>
<td>( \mathcal{C} ), ( \mathcal{C}' )</td>
<td>Centres of projection</td>
</tr>
<tr>
<td>( \vec{C}x )</td>
<td>Vectors from the Perspective Center to a Conjugate Point in the Left and Right Images Respectively</td>
</tr>
<tr>
<td>( d_1, d_2 )</td>
<td>First and second nearest neighbor distances</td>
</tr>
<tr>
<td>( d_i^{(T)} )</td>
<td>Difference between the source point cloud and the transform target point cloud (destination cloud)</td>
</tr>
<tr>
<td>( D_A )</td>
<td>Descriptor of A</td>
</tr>
<tr>
<td>( Gx, Gy )</td>
<td>Gradients in x and y direction</td>
</tr>
<tr>
<td>( H )</td>
<td>Homography matrix</td>
</tr>
<tr>
<td>( \vec{n} )</td>
<td>Normal vector</td>
</tr>
<tr>
<td>( \vec{p} )</td>
<td>Centroid of the 3D points</td>
</tr>
<tr>
<td>( q )</td>
<td>Target point cloud</td>
</tr>
<tr>
<td>( R )</td>
<td>Rotation Matrix</td>
</tr>
<tr>
<td>( s )</td>
<td>Scale Factor</td>
</tr>
<tr>
<td>( x, y )</td>
<td>Image Coordinates in the Reference Image</td>
</tr>
<tr>
<td>( X, Y, Z )</td>
<td>Ground Point Coordinates</td>
</tr>
<tr>
<td>( X_0, Y_0, Z_0 )</td>
<td>Exterior Orientation Parameters ((X_0, Y_0) and (Z_0) Represent the Position of Perspective Center with Respect to Ground</td>
</tr>
<tr>
<td>( \omega, \varphi, \kappa )</td>
<td>Position of Perspective Center with Respect to Ground</td>
</tr>
</tbody>
</table>
Coordinate System, where $\omega$, $\varphi$ and $\kappa$ represent the rotation angles between the Ground and Image Coordinate Systems.

$x_p, y_p, c$ Interior Orientation Parameters (Calibrated Principal Point Position and Principal Distance of The Camera with Respect to Image Coordinate System)

$w$ Weighting factor

$\lambda_0, \lambda_1, \lambda_2$ Eigen values

$\sigma_p$ Curvature value
List of Abbreviations

2D \hspace{1cm} \text{Two Dimensional}
3D \hspace{1cm} \text{Three Dimensional}
6D \hspace{1cm} \text{Six Dimensional}
ELCH \hspace{1cm} \text{Explicit Loop Closing Heuristic}
EM \hspace{1cm} \text{Expectation-Maximization}
EOP \hspace{1cm} \text{Exterior Orientation Parameters}
FOV \hspace{1cm} \text{Field of View}
GCP \hspace{1cm} \text{Ground Control Point}
GICP \hspace{1cm} \text{Generalized ICP}
IMU \hspace{1cm} \text{Inertial Measurement Unit}
ICP \hspace{1cm} \text{Iterative closest point}
LiDAR \hspace{1cm} \text{Light Detection and Ranging}
MEMS \hspace{1cm} \text{Micro-Electro-Mechanical Systems}
MMS \hspace{1cm} \text{Mobile Mapping Systems}
MVS \hspace{1cm} \text{Multi-view Stereo}
NARF \hspace{1cm} \text{Normal Aligned Radial Feature}
NNDR \hspace{1cm} \text{Nearest Neighbor Distance Ratio}
PTAM \hspace{1cm} \text{Parallel Tracking And Mapping}
PCA \hspace{1cm} \text{Principal Component Analysis}
PNP \hspace{1cm} \text{Prespective-n-Points}
RANSAC \hspace{1cm} \text{RANdom Sample Consensus}
RMSE \hspace{1cm} \text{Root Mean Square Error}
RO \hspace{1cm} \text{Relative Orientation}
SAC-IA \hspace{1cm} \text{SAmple Consensus Initial Alignment}
SLAM \hspace{1cm} \text{Simultaneous Localization And Mapping}
SVD \hspace{1cm} \text{Singular value decomposition}
SBA \hspace{1cm} \text{Sparse Bundle Adjustment}
SfM \hspace{1cm} \text{Structure from Motion}
SIFT \hspace{1cm} \text{Scale Invariant Feature Transform}
SURF \hspace{1cm} \text{Speeded Up Robust Features}
TORO \hspace{1cm} \text{Tree-based netwORk Optimizer}
TLS \hspace{1cm} \text{Terrestrial Laser Scanner}
TSDF \hspace{1cm} \text{Truncated Signed Distance Function}
UAV \hspace{1cm} \text{Unmanned Aerial Vehicle}
VO \hspace{1cm} \text{Visual Odometry}
Chapter One: Introduction

1.1 Motivation

Nowadays, the interest and demand for 3D indoor mobile mapping has been increased with the continuous improvement in data acquisition systems and the expanding range of potential applications. Currently, 3D data can be obtained through two technologies: photogrammetry and laser scanning. Laser scanning directly provides 3D data, while photogrammetry reconstructs 3D information through photogrammetric techniques such as triangulation process from multiple images of the surveyed area. The advantage of the direct acquisition of 3D data makes laser scanner popular for mapping of indoor and outdoor environment.

Generally, it is not always possible to cover a complete structure with a single Terrestrial Laser Scanner (TLS) scan. Therefore, several TLS scans from different positions/orientations are necessary for complete coverage of the surveyed structure. The collection and processing of TLS scans is a time consuming process; each collected scan has its own coordinate system (Canz, 2012), and 3D models can only be obtained by alignment of the collected scans in a common coordinate system. This manual mapping procedure requires an operator to constantly change the position of TLS until enough coverage obtained which is quite time-consuming and might be dangerous for remote places where it is difficult to be reached by human (e.g. mining environment).

Hence, the motivation of this thesis is to automate the process of mapping using an autonomous robot to increase safety and reduce the time and cost needed for manual mapping operation.
1.2 Problem statement

The volume of research in the field of mobile mapping has been remarkably increased over the past years, especially for outdoor environment. However, this is not the case for indoor environment, as there are still challenges that need to be addressed to achieve a system which can rapidly, reliably, remotely and accurately perform measurements in the three-dimensional space to map any indoor environments with minimum user intervention.

Some of the main problems in mapping of an indoor environment are summarized as follows:

- **Challenges of indoor Mobile Mapping systems**

Having complete mobile mapping system requires at least three parallel stages namely localization, mapping and loop closure to work together in parallel and at the same time. In the context of robotic applications this is called simultaneous localization and mapping (SLAM). Current challenges for indoor mobile mapping systems can be classified in terms of fusion of different sensors for localization and mapping, optimizing the drift in localization, and consistent map solution. In general, there have been three key open questions:

**Q1. What type of sensors should be used for indoor localization and mapping? Is it possible to rely on vision system alone for localization? How other sensors can help in localization?**

Selecting proper sensor is the perquisite requirement for localization and mapping procedure. Because of the diversity of indoor environments (e.g. mining area, building structure) the study should be done first to pick an appropriate sensor for both localization and mapping. Extensive research has been done on a vision based algorithm to enhance navigation and mapping solution especially in robotic field. This is mainly due to the unique and rich information which is provided
by visual sensors in terms of texture, colour, luminosity, contrast, and three-dimensional representations of both observed structure and camera pose.

Computer vision based pose estimation, however, does have some drawbacks that are the focus of ongoing research. Generally, vision-based systems suffer from poor lighting conditions, low texture in the surrounding environment and fast movement that causes image blur (Warren, 2015). Loss of tracking or pose estimation is considered as main drawback of visual systems. Hence, it is highly important for the system to re-localize itself as soon as it detects previously encountered features. One solution to this problem is to keep the previous features and save them in a database, so that the re-localization is done based on comparison of new features with the saved features. Another solution to the previous problem is to integrate the visual system with an IMU or wheel odometer in cases where the system lost tracking of the features.

To date, a significant amount of research has been done to use different visual sensor such as monocular camera; stereovision and RGB-D camera (Endres, 2014) for pose estimation (localization).

In many monocular localization algorithms, even with an accurate initialisation, the scale of the geometry is prone to drift. This limits the use of single camera systems, and often results in the integration of visual sensors with others such as Inertial Measurement Units (IMUs).

One approach to the scale problem is the use of rigid-stereo systems. Having two cameras (stereo vision) with accurate calibration parameters, scale is constrained by the known baseline between the cameras. Another approach is the use of RGB-D sensor that contains depth information of corresponding pixels in the image.

Recently, LiDAR become very popular for mapping of the indoor environment as it directly provides 3D data information. For applications where high level of accuracy (millimetre) is
required, the conventional method is to use a static scanning approach. At the time of static scanning, a LiDAR is placed on a tripod and this mode only a part of the environment is captured with high detail. The LiDAR is then moved around and the process is repeated until the entire area has been scanned. The 3D point clouds captured from different places can then be stitched together to build a single, and unified representation of the environment. While this process is accurate and reliable, it is highly slow and invasive.

To improve acquisition efficiency, the LiDAR is often placed on a mobile platform such as a wheeled platform or human operator. In compare to static scanning, mobile mapping systems are generally faster, but reconstructing the environment from the captured point clouds can be more complex and time consuming.

**Q2. Which method should be used for vision based localization? How to remove the drift in vision based system for pose estimation? What type of features should be extracted from the environment and which one is more stable and accurate?**

Localization or pose estimation of the vehicle, human, or robot using only the input from single or multiple cameras attached to it, is known as visual odometry (VO) process. The name VO came from the term wheel odometry which can incrementally estimates the motion of a vehicle by integrating the number of turns of its wheels over time. Various methods exist for pose estimation using VO. Related works in VO can be divided into three categories: feature-based methods, appearance-based methods, and hybrid methods. Feature-based methods are rely on salient and repeatable features that are tracked over the continuous frames; appearance-based methods use the intensity information of all the pixels in the image or sub-regions of it; and hybrid methods use a combination of the previous two methods. A major drawback with appearance-based approaches is that they are not robust to occlusions. The main draw-back of feature-based method is in cases
where there is no texture in environment and no features can be extracted from the environment. Hence, hybrid method is a good alternative approach for VO pose estimation as it can cover the problems with appearance-based and feature-based method.

Since VO incrementally estimate the pose of the system (pose after pose), the errors introduced by each new frame-to-frame motion accumulate over time. The accumulated error appears as a drift in the estimated trajectory and results in divergence of the system from the true trajectory. In many applications, it is highly important to keep the drift as small as possible. This can be done through local optimization over the last \( m \) camera poses (sliding window bundle adjustment or windowed bundle adjustment). Also the VO drift can be reduced through integration of visual sensors with other sensors such as IMU, laser range finder or wheel odometry.

In localization or mapping applications, the ability to find similar parts in different sets of sensor measurements is a complicated problem (Steder, 2011). A common method in vision based systems is to estimate features that best describe a partition of data in a compact representation. A compact representation of features helps to efficiently perform comparisons between different data regions. The entire process is generally subdivided into two subtasks, which are the identification of interest point or key-point across image frames, and identifying or extracting the information in the vicinity of the key-point which is encoded in a descriptor or description vector. One of the main advantages of key-points is that they highly reduce the search space and calculation time required for finding correspondences among two frames this will bound and focuses the computation on areas where it is more likely relevant for the matching process.

As the basic component of any vision based pose estimation algorithm, an accurate feature detector and matcher is essential.
Initial investigations for feature extraction started in the late 1980s, with the application of Haar features, the Kanade-Lucas-Tomasi (KLT) (Birchfield, 1997) tracker, and Harris (Harris, 1988) corners. The development of the Scale Invariant Feature Transform (SIFT) was a major progress in vision-based feature tracking. The development of a descriptor, which contains brightness gradients and other component of a feature, means that wider-baseline matching between frames is more reliable. This, however, comes at increased cost due to the size of the descriptor and the need for a more complicated matching routine. An evolution on SIFT (Lowe, 1999), Speeded Up Robust Features (SURF) (Bay, 2008), and FAST corner detection (Rosten, 2006) were other major progress in both speed and accuracy. Modern feature extraction technique including Oriented FAST and Rotated BRIEF (ORB) (Rublee, 2011), Centre Surround Extrema (CenSurE) (Agrawal, 2008), Speeded Up Surround Extrema (SUSurE), (Ebrahimi, 2009) have pushed the limits of feature tracking and continued to show improvement in both reliability and speed.

3D descriptors are also useful as they work directly on 3D point clouds. Normal Aligned Radial Feature (NARF) (Stederet al., 2011) contains a key-point extraction step, however; it applies to range images. Hence, it is not possible to use it directly on laser scans. Fast Point Feature Histograms (FPFH) (Rusu et al., 2009) and Signature of Histograms of Orientations (SHOT) (Tombari et al, 2010) are other type of 3D descriptors that directly computable on laser scans.

It is still considered difficult to track features on a pixel-by-pixel basis due to computational cost, while being unreliable and ultimately unnecessary. Various approaches now consider ‘dense’ tracking over every pixel in the image, but such techniques still require huge amounts of computating power (mostly require Graphical processing unit (GPU)). By sub-sampling or down-sampling the image into features that have salient components for tracking across the image frames.
such as corners or high contrast blobs, matching is seen as more reliable and computationally efficient.

**Q3. What type of mapping required by the application?**

Generally, the attention of indoor mapping system is mainly in acquiring the structures that define the 3d space. These structures include walls, floors, ceilings and so forth. In an indoor environment the objects that compose space are also of interest (eg. furniture). Typically, the raw 3D point cloud data from the sensors have to be processed, fused and reformed in order to remove meaningless data or artefacts. In the following step, one must choose an adequate representation of the environment such as topological maps, polygonal layouts, occupancy grid 3-D models, or feature-based maps.

**1.3 Research Objective**

The main objective of this thesis is to develop an autonomous indoor mobile mapping system using a multi-sensor autonomous robot. In order to achieve the main objective of the thesis, the following tasks were implemented as part of the overall software framework for autonomous indoor mobile mapping:

- Evaluation of Stereo and RGB-D camera for Visual odometry (VO) and mapping steps.
- Developing two different approaches for RGB-D visual odometry; the first is based on projecting the 2D image interest points in 3D space using Speed Up Robust Features (SURF) and the second is working directly on the point cloud using Normal Aligned Radial Feature (NARF) for estimating rotation and translation parameters
- Evaluating various Iterative closest point (ICP) methods including (point-to-point, point-to-plane and plane-to-plane) to enhance the pose (rotation and translation) estimation
• Developing segmentation process to extract planar patches such as walls, ceilings, and floors and the normal vector associated with these patches to update the pose (only rotation, geometry of three planar patches are not enough to find the translation) of the system and remove the drift over the long run.

• Developing loop closure algorithm to reduce the drift of registration process and globally update the pose of the system

• Developing RGB-D aided Velodyne for mapping over long trajectories.

• Developing obstacle avoidance using 2D laser range finder to make the mobile mapping system autonomous

The obstacle avoidance is implemented on the Seekur Jr (Mobile Robot, 2014) robot using information obtained by the 2D laser range finder integrated within the robot. The RGB-D camera and Velodyne HDL-32 are added to the robot as external sensors to acquire point cloud data for indoor mapping procedure.

Figure 1.1 shows the propose algorithm for 3D mapping. Two methods are used to generate the map of indoor environment. In the first method single RGB-D sensor is used as a main sensor for 3D mapping. In the second method RGB-D aided Velodyne is used to map a large indoor environment.
It should be noted in this thesis, stereo localization and mapping algorithms is also proposed and investigated. However, this algorithm is not used for integration with Velodyne sensor.

1.4 Contribution

1. The first contribution of this work is the development of autonomous mobile mapping system that is capable of mapping small and large indoor environment with the accuracy ranging up to two centimetres for a small room of size 8x8 meters, and two meters for a large corridor of size 33x11 meters. It should be noted the system is working in the “stop-and-go” mode when collecting the laser scan data. The stop mode of the robot is defined based on time and position of the robot with respect to initial position.
2. The underlying odometry estimation algorithm (coarse registration), and iterative closest point ICP (fine registration), are prone to failure in situations where camera displacement is large between frames or a lack of features poorly constrains the camera pose in the observed scene. Two contributions are considered for solving this problem:

- Switching technique between ICP and visual odometry for short period. This means that the pose is estimated from registration of the point cloud coming from the RGB-D sensor or Velodyne instead of features extraction method using consecutive images.
- The angle between normal vectors associated with planar patch segments of every frame is used to update the pose (only orientation) of the system (the geometry is not enough to find the translation (three planar structures)). To find the translation and update the pose homography method is used as the structure mainly consists of planar geometry.

3- For the accurate alignment of the point cloud, two stage registrations is proposed where the result of the first step (VO (coarse registration)) will be used for the next step which is Fine registration. The main goal for such a process is to register the point cloud as

1.5 Thesis Outline

This thesis consists of seven chapters, which are briefly explained below

Chapter 2 reviews current indoor mobile mapping system including robotic systems, handheld systems, trolley systems and UAVs. Also a comprehensive review is done on the most common sensors used for indoor mobile mapping systems.

Chapter 3 discusses the steps for coarse registration using visual odometry method. Evaluation is done on two different cameras including Stereo and RGB-D for VO pose estimation and map reconstruction. Different feature extraction techniques are presented and the most stable one is
chosen for the pose estimation. The main contribution of this chapter is accurate pose estimation method using VO for coarse registration of point clouds. The concept of key-frame is introduced for robust feature extraction and tracking based on optical-flow.

Chapter 4 deals with the fine registering using different alignment techniques namely point-to-point, point-to-plane and plane-to-plane ICP. Combining the result of the coarse registration from previous chapter and fine registration, point clouds can be aligned accurately and the registration result will be more consistent.

Chapter 5 describes different techniques for optimizing and enhancing the result obtained from coarse and fine registrations. This is highly important to generate consistent 3D maps. The main contribution of this chapter is to use segmentation as an optimization technique to extract planar patches and normal vector associated with the patches to update the pose globally. The last step is loop closure which minimizes the error between first and last acquired scans.

Chapter 6 illustrates the results and discussion for mapping in two scenarios. In the first scenario, it is shown that single RGB-D sensor is enough to map a small room. However, this is not the case for large corridor. Hence, in the second scenario, RGB-D sensor is used as an aiding sensor to generate consistent mapping solution.

Chapter 7 includes the conclusion and future work.
Chapter Two: Background

2.1 Introduction

This chapter reviews current indoor mobile mapping systems including robotic systems, handheld systems, trolley systems and UAVs. Moreover a comprehensive review is done on the most common sensors used for indoor mobile mapping systems. Different map representation including Octree, Triangular mesh, Convex planar and Segmentation based mapping are also discussed. The information provided in this chapter gives a good comparison between current available systems and the developed system in this thesis in terms of algorithm and the used sensors for indoor mobile mapping.

2.2 Indoor Mobile Mapping System using a Robot

Mapping and exploring indoor environment has become a very important topic, particularly in the field of robotics and indoor navigation and locations-based services market. Technology focused on mapping indoor environments creates wide range of potential applications, such as search and rescue, hazardous material handling, collision-free navigation, surveying remote sites or dangerous areas (such as underground mines, tunnels, caves, or channels), as well as exploration and inspection of infrastructure to compare it with its original plans.

This section briefly describes recent indoor mobile mapping systems in the context of robotic applications. This gives a good comparison between current available systems and the developed system in this thesis in terms of algorithm and used sensors for autonomous indoor mobile mapping. Figure 2.1 illustrates the four most well-known robots currently working for indoor mobile mapping using various sensors.
The first robot is Kurt3D (Surmann, 2004) with a size of 45 cm (length) × 33 cm (width) × 26 cm (height) and a weight of 15.6 kg. Considering the rotating 2D laser range finder, the height is increased to about 47 cm and weight is increased to 22.6 kg. The maximum velocity of this robot is 5.2 m/s (autonomously controlled 4.0 m/s). The operation time of the robot is about four hours with one battery (28 NiMH cells, capacity: 4500 mAh) charge. The core of the robot is a Pentium-III-600 MHz with 384 MB RAM running Linux. An embedded 16-Bit CMOS microcontroller is used to control the motor.

In (Surmann et al., 2004) Kurt3D is used to generate 3D volumetric maps of the scene by taking several 3D scans and merging them in one consistent 3D model using pose information from visual sensor, which estimated from 6D SLAM. The main drawback of this system is that the laser scanner should rotate continuously to acquire 3D point cloud data which results in non-uniform and inconsistent point cloud measurement data.
Another indoor mobile mapping system is the Intelligent Robot for Mapping Applications in 3D (Irma3D) developed by (Nüchter, 2012). Irma3D is a small three wheeled robot, with a light weight. With the width of 52 cm it is possible for this robot to pass through narrow doorways. Because of the three-wheeled design the robot is able to do high manoeuvrability such that it can rotate on the spot. These properties make Irma3D ideally suited for indoor environments. The main sensor of Irma3D is the 3D laser scanner VZ-400 by RIEGL Measurement GmbH. The scanner is placed on top of the robot. Attached to the top of the scanner is a Canon 1000D DSLR camera. The mapping algorithm integrate camera with LiDAR to generate point clouds. A similar process was done using the Optris PI160 thermal camera, which is also placed on top of the VZ-400 to captured information about the thermal properties of structures in the point cloud. The optris PI160 thermal camera has an image resolution of 160×120 pixels and a thermal resolution of 0.1 degree centigrade. It captured images at a frame rate of 120 Hz and with an accuracy of 2 degrees centigrade. The laser scanner acquires data with a field of view of 360 × 100 degree. The aim of this robot is to take usage of a constantly rotating 3D scanner, by scanning on the fly methods. The drawback is no loop-closing strategy was considered in this work to optimize the pose drift.

Other system is AZIMUT-3 robot (Ferland, 2010), equipped with a URG-04LX (Kneip, 2009) laser rangefinder and a RGB-D camera. The RGB images from the RGB-D camera are used for the appearance based loop closure detection while the depth images are used to find the 3D position of the visual features and pose estimation. The main drawback of this system is in cases where RGB-D sensor cannot find enough features from environment. Hence, the system cannot localize itself in these cases.

Another system is from Mobile Robot company(Mobile Robot,2014), which is equipped with a forward looking laser range finder aimed for localizing the robot during the mapping process, and
upward-looking laser range finder for structural mapping, and a panoramic camera for recording the texture of the environment. In (Thrun, 2004) a probabilistic model known as expectation-maximization (EM) algorithm is used to simultaneously estimate rectangular surface patches in 3D data, while the robot is moving and exploring the environment. The flat-surface assumption in this work leads to a convenient closed-form solution of the EM algorithm. However, the main drawback of this system is in alignment of the point cloud generated by the two laser range finders in order to build a map.

### 2.3 Indoor Mobile Mapping System using Handheld devices

Other systems for indoor mobile mapping are known as handheld mobile mapping systems. Figure 2.2 shows two well-known handheld system in the form of backpack and handheld system known as Zebedee.

![Figure 2.2. Handheld mobile mapping system: backpacking system Irma3D, Zebedee handheld 3D mapping system](image)

The setup of the backpack system (Nüchter, 2013) is almost similar to Irma3D described in the previous section. The basis is a Tatonka load carrier attached using pipe clamps. Energy is
provided by two Panasonic 12V lead-acid batteries. Similar to the Irma3D, the backpack features a horizontally scanning SICK LMS 100 (Rudan, 2010), which is used to observe the motion of the carrier. To fully make use of the 270 degree field of view of the SICK LMS 100 laser scanner, the sensor head is mounted slightly above the load carrier. The main sensor of the backpack system is the 3D laser scanner RIEGL VZ-400. The VZ-400 is capable to freely rotate around its vertical axis to acquire 3D scans. Due to the setup, however, there is an occlusion of about 100 degree due to the backside of the backpack and the human carrier.

Another system is called Zebedee (Bosse, 2012), which is a handheld mobile mapping system for mapping of indoor and outdoor environment. The primary sensor for Zebedee is a 2D laser scanner, which can captures up to 43200 measurements per second to visible surfaces in the environment. In order to improve two-dimensional field of view of the laser scanner into a three-dimensional field of view, a flexible spring connects the scanner to the device's handle. The spring allows the scanner to pivot freely (sweeping up to 170 degree in amplitude and tuned to a desired frequency of approximately 0.5 Hz) as a result of the operator's natural walking or arm motion. The system also consists of inertial measurement unit (IMU), which is mounted on the laser scanner side of the spring to provide rough estimates of the scanner orientation. The IMU provides three-axis rotational rate, linear acceleration, and heading information at a 100 Hz update rate. A small laptop computer is also considered for recording and saving the data. The system can work for several hours with lithium-ion battery. The mass of the Zebedee is approximately 650 g, and the whole system can be carried in a small pack with a total mass of 3.8 kg (including device, laptop, cables, battery, and the bag itself).

The mapping algorithm include place recognition step to identify revisited regions or areas were mapped more than once in the dataset and apply coarse corrections to enhance the global
registration algorithm. The main drawback of the system is in the sparseness of the point clouds and the amount of the surveying time that should be done in order to map a small environment.

### 2.4 Indoor Mobile Mapping System using Trolley devices

Another type of mobile system is based on using trolley devices (Figure 2.3). Scannect is a trolley based mobile mapping system developed by (Chow, 2014) for indoor application. The system is equipped with two sideward facing Kinects, MEMS-based IMU from Xsens to keep track of the orientation in dark areas and in places where no features can be detected. Scannect is also equipped with FARO Focus3D terrestrial laser scanning (TLS) instrument for starting point initialization. The proposed system attempts to provide the accuracy and stability of stop-and-go approach. Using stop-and-go approach, the laser scanner can capture lower quality data in the non-stop mode and higher quality 3D scans measurement in the stop-mode. Hence, higher mapping quality is obtained in stop-mode and lower quality map is achieved in non-stop mode. For this reason, the system design can be seen as a hybrid between static and kinematic mapping, termed continuous stop-and-go mode.

The main drawback of the system is to use an expensive laser scanner to aid a cheap Kinect sensor for mapping and localization.
Another system is known as Mine mapping cart (Thrun, 2003) consist of four laser range finders, for 4x2-D volumetric mapping. The system consists of two SICK laser range finders, one pointing forward parallel to the floor, and one pointing upward perpendicular to the robot’s heading direction. Moreover, the robot is equipped with two wheel encoders to measure approximate robot motion. The forward pointing laser scanner is used for simultaneous localization and mapping (SLAM) in 2D. Using this data, the robot acquires an accurate 2D map of the environment. The upward pointing laser is used to reconstruct the 3D shape of the walls and the ceiling of the mine, registered in space according to position estimates gathered from the 2D map. Data from the horizontal scans is used to remove artifacts in the vertical scans, and vice versa.

Another system is a rolling cart system by Trimble Indoor Mapping Solution (TIMMS) unit (Trimble, 2013). TIMMS (Figure 2.4). This system equipped with LiDAR and spherical camera as its main sensors for mapping and localization. The Trimble Indoor Mobile Mapping Solution
has been specially designed to capture all interior spaces, allowing floor plans to be as complete as possible.

![Image](image.png)

**Figure 2.4. Trimble Indoor Mapping Solution (TIMMS)**

The main drawback of all rolling cart based systems is that the operator cannot easily map uneven terrain such as staircases or thick carpeting. Moreover, these systems are totally impractical in areas where it is difficult to reach by human.

### 2.5 Indoor Mobile Mapping System using an UAV

Unmanned aerial vehicles (UAV) systems (Figure 2.5) have recently become popular for indoor mobile mapping. Although UAVs payload is one of the main drawbacks of these systems for indoor mapping, there still high demand for using these systems for indoor applications.

One of the first UAV for indoor mapping is developed by (Huang, 2011). The control of a micro air vehicle requires accurate estimation of not only the position of the system but also the velocity
– estimates. Estimating a UAV’s 3D motion requires calculating relative motion at each time step by aligning successive sensor measurements of RGB-D frames, which is known as Visual odometry. The motion estimates computed by the visual odometry are fused with measurements from the onboard IMU in an Extended Kalman Filter. The estimated position and velocity of the system are used by the position controller to stabilize the position of the system. Having the knowledge of the relative motion of the UAV from sensor frame to frame, the 3D trajectory of the system in the environment can be estimated by integrating the relative motion over time. However, the main drawback is that visual odometry methods suffer from long-term drift and are not suitable for building large-scale maps. Hence it requires other methods such as loop closures or Sparse Bundle Adjustment (SBA) to be integrated with it to minimize the error over time.

Figure 2.5. Quadrotor micro air vehicle (MAV), CityFlyer micro-UAV platform

Another system is CityFlyer (Morris, 2011) micro-UAV platform equipped with a Hokuyo laser range-finder and a Swissranger 4000 depth camera, an on-board IMU and altimeter, which are used to provide the mapping solution in the form of multi-volume occupancy grids, or MVOGs. MVOGs save the observations of occupied and free space separately for obstacle avoidance. The
mapping algorithm projects the laser scan measurement using IMU information, in order to make them invariant of the roll and pitch motion of the helicopter.

In this thesis, the Seekur Jr Robot (MobileRobot, 2014) (Figure 2.6) was used for indoor mapping. Seekur Jr is a mobile robot platform with a size of 105 cm (length) \times 84 \text{ cm} (width) \times 50 \text{ cm} (height), a weight of 77 kg and speed of up to 1.2 m/s. This mobile robot platform is equipped with onboard microcontroller server controlling motors and wheels, a PC with SSD hard drive. Seekur Jr is equipped with a 2D laser range finder. RGB-D camera and Velodyne HDL-32 are used as external sensors for indoor mapping.

Figure 2.6. Seekur Jr

2.6 Overview of Sensors for Indoor Mobile Mapping System

Collecting 3D point clouds data is a prerequisite requirement to begin the process of mapping. Selecting the proper sensor is an important step for generating final map solution. This section briefly describes various sensors for 3D mapping of indoor environments.
The first sensor is 2D laser scanners, which uses laser light to measure distances. The laser is rotated, or more often a mirror is rotated, such that the laser beam measures distances in different directions. The apex angle, usually 90, 180, 270 or 360 degree, is discretized with different resolutions, e.g., 1, 0.5 or 0.25 degree.

Various researchers used two 2D laser scanners to acquire 3D information. For example, in the work by (Thrun et al. 2000, Früh and Zakhor, 2001) two 2D laser scanners for collecting 3D data were used. The setup of the system consists of one scanner, which is placed vertically and the other one placed horizontally. The vertical and horizontal scan lines are registered to common coordinated using the current 3D robot pose. The previous work was improved by (Zhao and Shibasaki, 2001) where they used two additional vertically mounted 2D scanners, shifted by 45° to scan sides of objects, and hence reduce occlusions. The scanner mounted in horizontal position used to compute the 3D robot pose. The accuracy of the point clouds obtained in the later method depends on the precision of the scanner, and the pose estimation. Other groups use rotating SICK scanners for acquiring 3D point cloud (Kohlhepp, Walther and Steinhaus, 2003; Wulf, Arros, Christensen and Wagner, 2004). In the work by Wulf et al.,(2004) the scanner is rotating around the vertical axis, while the robot is moving. The quality of the final model highly depends on the pose estimated from aiding sensors (eg. inertial sensors).

Terrestrial laser scanning (TLS) can acquire millions of range measurements in 3D with high accuracy from ground level, and has been employed for different indoor mapping applications. These systems are generally placed on a tripod and require several minutes to generate a 3D scan from a static location. As a terrestrial scanner can only measure surfaces visible from its current position, shadows occur in the data due to occlusions. Therefore, in order to achieve reasonable coverage of a complex site, a scanner must be placed at different (typically carefully selected)
locations. Once the measurements are acquired, they should be aligned to be referenced at one location; however, this step requires enough overlap between the scans and accurately surveying the tripod location or placing reflective targets in the environment.

Mobile mapping technology positions the scanning equipment on a moving platform during data acquisition. The platform's motion ensures that the sensors continuously view the environment from different viewpoints, hence significantly reducing shadowing from occlusions.

Examples of using 3D laser scanners for indoor mobile mapping system are introduced in (Allen, Stamos, Gueorguiev, Gold and Blaer, 2001; Georgiev and Allen, 2004; Sequeira, Ng, Wolfart, Goncalves and Hogg, 1999).

Monocular and stereovision cameras are other common sensors used for generating point clouds and reconstruct the scene. Recently, Multi-view Stereo (MVS) (Bradley, 2008) in the context of structure from motion (SFM) has become very popular for generating dense point clouds. The goal of Multi-view Stereo (MVS) is to extract a dense 3D surface reconstruction from multiple images taken from known camera position.

The main drawback of MVS approach is the time that it required to process multiple images to reconstruct the large scene. Recently, incremental Structure from Motion (SfM) (Snavely, 2006) system address the previous problem by incrementally reconstructs the scene using a windowing approach.

Time-of-Flight camera is another popular sensor for indoor mapping but the main drawback of this sensor is a lack in measurement accuracy and robustness. May (2009) presents a comprehensive approach for 3D environment mapping based on time-of-flight technology. Imprecision of depth measurements are properly handled by calibration and integration of several filters. Robust registration is performed by a novel extension to the Iterative Closest Point
algorithm. Remaining registration errors are reduced by global relaxation after loop-closure and surface smoothing.

Thermal imaging is another sensor that can give information about the 3D thermal model of the environment. Precise thermal 3D models will allow architects and construction engineers to study the model, run simulations of heat and air flow and use the gained information to change existing buildings to reach the estimated energy savings (Nüchter, 2011).

Another common sensor is the RGB-D camera. The RGB-D camera consists of three main components: a projector that projects a pattern, IR camera that detects the returning pattern and RGB camera that provides colour information. The field of view of RGB-D camera is 58 degree horizontally and 40 degree vertically, and it produces 640 × 480 pixels depth images at 30 frames per second. The range of operation is between 0.5 m and ~5.0 m (Khoshelham, 2012).

At close range (0.5–2 m), the accuracy of depth range is between 1 and 6 mm with a spatial XY-resolution of 3 mm at 2 m. Figure 2.7 shows the time line for structure light and time of flight cameras.

Figure 2.7 Structured light (blue) and time-of-flight sensors (black) time line
Several researchers employed RGB-D camera for indoor mapping. The KinectFusion algorithm introduced by Newcombe et al. (2011), is one of the first systems to produce a volumetric reconstruction of a scene in real-time with an unprecedented level of accuracy. The KinectFusion algorithm does not rely on identifying key-points (for example geometric planes, corners etc.) in the incoming data in order to match sparse features between frames for registration. KinectFusion performs registration based on dense correspondences, i.e. correspondences from all data points in each depth frame. As the incoming depth data from the Kinect Sensor is very noisy the first stage is to apply a bilateral filter on the depth image. This can greatly reduces the noise present in the data while maintaining depth discontinuities. The effect of bilateral filters on depth image is equivalent to effect of this filter on normal RGB image, where it can remove noise and smooth an image while preserving edges. After the applying bilateral filter the depth data is replicated twice, followed by sub-sampling the image. This produces three views on a single depth data frame with different resolution. All three views on the data are stored in a depth map pyramid. In the following step each level of the depth data pyramid is converted into a 3D point cloud from the point of view of the sensor by back projecting the depth data through the intrinsic image plane calibration parameters of the sensor. This point cloud representation of the data is stored as a vertex map where each vertex has 3D coordinates in the local coordinate space of the depth sensor. From this vertex map pyramid, a normal map pyramid is calculated based on a nearest neighbour calculation on the vertex map to give each vertex a value for the direction it is facing along with its position. The vertex and normal pyramids are then stored for processing by the next step in the pipeline which is the ICP registration algorithm.

As the ICP process produces an estimated transform by an iterative process rather than an exact solution for the frame to frame pose estimation it by definition introduces some error to the global
pose estimate in each successive frame. Using the raw depth data from the previous frame as the reference for the incoming frame would also negatively impact the accuracy of ICP due to the quality of the raw depth data frames which the low cost depth sensor in the Kinect produces.

In order to avoid these problems Kinect Fusion employs loop closure to remove the drift.

Another example of using RGB-D sensor for indoor mapping is a work done by (Henry et al., 2012). The algorithm begins with RANSAC on the 3D key-points for the initial alignment step and ICP algorithm for refining the point cloud registration. Compared to their previous work (Henry et al., 2010) a global optimization using sparse bundle adjustment (SBA) instead of the Tree-based netwORk Optimizer (TORO)(Grisetti, 2009) is used. The experiment for indoor mapping shows the capability of the system to map a large space. The ICP-only solution showed 15 cm error, while the RGB-D visual odometry and ICP method showed 10-11 cm error. The drawback of this work is that no extra update step is considered if the visual odometry algorithm loses the features and the tracking is failed. Another problem is whenever the depth sensor is out of range they do not have 3D key-points for RANSAC matching even though they were detectable in 2D. This can be rather common with their single forward facing sensor configuration. For example, when the sensor is looking in a forward direction, the majority of the center pixels will likely be out of range.

The focus of this thesis is on the use of RGB-D and Velodyne HDL-32 LiDAR for indoor mapping in two separate mapping scenarios (small room and large corridor). In the first scenario which is small room single RGB-D sensor is used to illustrate the capability of this sensor in generating the dense point cloud and 3D dense reconstruction of an indoor environment. In the second scenario, which is a large corridor Velodyne HDL-32 is used as a main sensor aided with RGB-D sensor.
Figure 2.8 illustrates the result of the point cloud obtained from different sensors including SICK LMS200 laser scanner, RGB-D sensor, 3D laser scanner RIEGL VZ-400, Velodyne HDL-32 and thermal camera.

The Velodyne HDL-32 produces 3D range scans by rotating an array of 32 beams around its vertical axis at 10Hz and producing close to around 700,000 points per second or close to 2,200 points per laser in the range of one to 70 meters. In the horizontal direction, the array provides an angular resolution of approximately 0.16 degree with 360-degree field of view (FOV).

Vertically, the pitch angles range from -30.67 to +10.67 degree with an angular resolution of 1.33 degree. Its range measurement accuracy typically is within 2 cm.

One of the drawbacks of using Velodyne HDL-32E is the spacing between consecutive laser scan lines, which increases sparsity (adjacent points become more apart) of measured point clouds as the distance from the sensor increases. Another point is low vertical angular resolution, which causes poor result from traditional point cloud registration algorithms. Hence, other pre-techniques should be considered, such as coarse registration using aiding sensors to enhance the results.
Figure 2.8. Point cloud generated from different sensors (SICK LMS (2D laser range finder), RGB-D camera, LiDAR, Velodyne HDL-32, Thermal camera)

2.7 Overview of Map Representation

Once a 3D point cloud dataset has been acquired using one of the methods presented in the previous section, it needs to go through a series of geometric processing steps in order to extract meaningful information. It is the role of a mapping system to process and convert the raw input
point cloud data into different representations and formats based on the requirements imposed by each individual application.

There are several methods for surface reconstruction, which can be roughly categorized into three classes: model-based, point-based, or volume-based methods.

The first group tries to fit geometric primitives such as planes to the points (T. Liu, 2010). Even though this method works reasonably well for artificial objects, they are not useful for the high structural variance in natural scenes. Point-based algorithms create a surface from the points using triangulation and interpolation to generate a mesh (M. Gopi, 2002). A disadvantage of both model and point-based methods, is that while they may cope with a reasonable amount of noise, they are sensitive to outliers, such as those originating from dynamic objects and mis-registrations.

Volume-based techniques rasterize the space into a regular voxel grid or an octree (B. Curless, 1996). Each measurement is rendered into the volumetric grid as a ray or beam of free space emanating from the sensor to the measured point, thereby erasing non-static points as rays from other viewpoints pass through the voxels that were occupied in previous scans. A surface model can then be extracted from the volume.

2.7.1 Octree based mapping

To understand the geometry around a query point, most geometric processing steps need to discover a collection of neighbouring points that represent the underlying scanned surface through sampling approximations. The mapping system therefore needs to employ mechanisms for enabling the search of point neighbours in fast ways, without re-computing distances between each other every time. This is done by spatial decomposition techniques such as kd-trees or octrees (Hornung, 2013), and partitioning the point cloud data into chunks, such that queries with respect to the location of the points can be answered fast. The previous decomposition techniques can
construct a volumetric representation for a cloud, by enclosing all points in boxes (also called “voxels”) with different widths. An example of such representations is given in Figure 2.9 for octree data structures of Stanford Bunny from dataset provided in point cloud library (Rusu, 2011). An octree is a hierarchical data structure for spatial subdivision in 3D space. Each node in an octree represents the space contained in a cubic volume, usually called a voxel. This volume is recursively subdivided into eight sub-volumes until a given minimum voxel size is reached. The minimum voxel size determines the resolution of the octree. Since an octree is a hierarchical data structure, the tree can be cut at any level to obtain a coarser subdivision if the inner nodes are maintained accordingly.

Figure 2.9. Octree representation of Stanford Bunny point cloud, with a leaf size of 1.5cm

Besides providing fast access to point locations and their corresponding neighbours, octree representations are also popular in the context of collision detection applications, where instead of
estimating distances to each point, it is possible to perform raycasting to the voxels encompassing to discover the portions of space which are free or occluded.

Octrees avoid one of the main shortcomings of fixed grid structures: They delay the initialization of map volumes until measurements need to be integrated. In this way, the extent of the mapped environment does not need to be known beforehand and the map only contains volumes that have been measured.

The other spatial decomposition technique is bd-tree (James, 2004), which is a variant of a kd-tree structure and optimized to provide a greater robustness for highly cluttered point cloud datasets. However, in contrast to octree structures, bd-trees or kd-trees are more difficult to update, and, thus, their usage is mostly limited to static scenes for applications working with individual point cloud datasets.

2.7.2 Triangular mesh based mapping

Another representation of the point cloud is based on classical triangular surface reconstruction (Hoppe, 1992) (Schroeder, 1992) approach. A triangle mesh is a type of polygon mesh in computer graphics. It comprises a set of triangles (typically in three dimensions) that are connected by their common edges or corners. Figure 2.10 shows the triangular mesh representation of a Stanford Bunny from point cloud library (Watanabe, 2001) dataset.
One class of triangulation algorithms computes a mathematical model prior to triangulation to ensure a smooth mesh while being robust to noise (Kazhdan, 2006). This type of algorithm assumes surfaces are continuous without holes, which is usually not the case in open scene scans or maps acquired with typical robotic sensors. Another class of algorithms connects points directly, mostly being optimized for high-quality point clouds with low noise and uniform density. While these algorithms retain fine details in objects (Bernardini, 1999), they are again less applicable to noisy datasets captured with an RGB-D or LIDAR sensor, where occlusions create large discontinuities.

For the real-world condition the Greedy Projection Triangulation (GPT) approach has been developed (Gopi, 2002). The algorithm generates triangles in an incremental mesh-growing manner, resulting accurate and fast triangulations. However, the GPT approach keeps all available points to preserve geometry, which is not usually essential for point clouds including surfaces that are easily approximated by geometric primitives. To solve this problem a hybrid triangulation
method was introduced by (Ma, 2013), where point clouds are segmented into planar and non-planar areas for separate triangulation. Another triangulate mesh based algorithm is known as QuadTree-Based (QTB) algorithm, which was created to remove planar segments prior to triangulation. The QTB algorithm substantially reduces the amount of redundant points, although a number of limitations reduce its performance. For example, the algorithm does not assure that final planar points will lie inside the original planar region, which can lead to considerable shape distortion. The algorithm also produces replicate vertices, overlapping triangles and artificial holes along the boundary.

Recently (Ma, 2013) introduced robust and accurate approach for planar segment decimation and triangulation. In comparison to the QTB algorithm, this algorithm assures geometrical accuracy during simplification with fewer triangles, artificial holes or overlapping faces and without repeated points. The algorithm receives a point cloud as input and creates a triangular mesh as a result. The process begins with plane detection to divide non-planar segments. Curvature-based region growing approach was used for plane segmentation (Ma, 2013). Decimation is also applied to develop a more robust solution for triangulation.

Another approach is based on using mesh and assigning a semantic labelling to for representing meaningful information. To estimate a meshed representation of the scene one can use a sequence of depth information from proper sensor like RGB-D for indoor and stereovision for outdoor scenes. Depth estimation is incrementally fused into a single 3D reconstruction using the volumetric Truncated Signed Distance Function (TSDF) (Bylow, 2013) representation. A signed distance function assigns to each voxel a value equal to the signed distance to the closest surface interface (zero crossing), with positive increasing values corresponding to free space and
negative decreasing values corresponding to points beneath the surface. The representation allows for the efficient combination of multiple noisy surface measurements, obtained from different depth maps by averaging signed distance measures from every depth map at each voxel.

The Marching Cubes method presented by (Lorensen et al., 1987) is another algorithm for generating triangle mesh. This algorithm sub-divides the scanned volume into cubic cells. For every cell the intersections between the cell edges and the surface are computed. Pre-computed surface patterns are then used to create local triangle mesh estimation. To interpolate the intersections, surface representations like planes or splines are fitted to the local data using least squares fits. A property of the Marching Cubes algorithm is that it generates far more triangles than are required to represent an object.

One disadvantage of triangular mesh representations is that they are costly to update if the scene changes. Taking for example the dataset in Figure 2.10, if for example an object is attached to the rabbit and is removed from it, the entire surface of the rabbit might need to be regenerated, depending on how many of the surface components are actually connected with each other.

2.7.3 Convex planar based mapping
A solution for triangular mesh problem addressed in previous section (attached object removed from the original object require regeneration of the surface) can be addressed by modeling the surface using convex planar polygons (Hähnel, 2003). In this case, the system models the point cloud first using an octree implementation with a small leaf width and then proceeds at estimating sets of planar patches that best approximate the underlying surface model (Figure 2.11). The resultant models can be more inaccurate than triangular surface meshes, but it provides important advantages in terms of the update speed for applications with tight computational constraints.
2.7.4 Segmentation based mapping

Another solution is to segment the point cloud into 3D shapes such as cylinders, planes, spheres, cones, etc. in order to provide smoother data approximations. For each object cluster the system first attempts to fit a primitive geometric shape, and then use triangular meshes to model the remaining points. The rest of the data can be left as points and labeled as such, thus providing a true hybrid representation of the scene, using a multitude of formats presented so far: points, geometric models, and triangular meshes.

Figure 2.11. Convex planar patches approximating of Stanford Bunny
Chapter Three: Coarse Registration

3.1 Introduction

In GPS-denied areas (e.g., indoor environment) very poor positioning information is typically available. Inertial measurements units (IMUs) are acceptable over short period of time for estimating position, however; their error drifts over longer durations. Therefore, IMUs are typically integrated with other sensors to provide satisfactory results. Vision sensors on the other hand are more reliable for pose estimation in an indoor environment. Using visual sensors, the pose of the system can be obtained by using Visual odometry (VO) or Relative orientation techniques. Visual odometry is the process used for obtaining relative orientation parameters between consecutive camera frames. The term Relative orientation (RO) is defined as the process of estimating the Exterior Orientation Parameters (EOPs) of one camera with respect to the camera coordinate system of another camera.

In this chapter, two stage registrations are proposed where the result of the first step (coarse registration) is used for the next step, which is fine registration. The main idea behind the two stage registration approach is to enhance the process of alignment between consecutive frames especially in the case where the noise in sensor measurement is quite high (e.g., RGB-D point cloud). This chapter is also explains the use of stereo and RGB-D camera for VO and 3D reconstruction.

3.2 Feature based registration (Visual Odometry)

Recently, many approaches have been used for pose estimation using visual odometry technique including state estimation methods such as Kalman filters (Howard, 2008) and Particle filters (Eade, 2006), as well as Structure-From-Motion (SFM) (Konolige et al., 2007) , (Nister et al.,
SFM techniques are typically more robust and scalable, specifically when using stereo as it can reduce the scale ambiguity. In the SFM based stereo VO, there are a number of design approaches that affect overall system performance. A main subset of these approaches consists of the process of: detecting local features, establishing feature correspondences between frames, estimating an initial pose, and refining that pose to reduce error accumulation (Alismail, 2010).

Basically visual odometry pose estimation can be classified into two categories namely sparse and dense visual odometry. In sparse visual odometry techniques only a part of the available image data, e.g. small patches at certain points in the image are used for pose estimation. However, in dense visual odometry technique the whole image data (all the pixels) will be used for pose estimation. In dense visual odometry approaches the camera motion is calculated by registering consecutive images through photometrical error minimization between images or geometric error between 3D surfaces (Kerl, 2013). One of the first dense visual odometry methods was proposed by Comport et al. (2010) based on stereo image pairs. Steinbrucker et al. (2011) and Tykkala et al. (2011) proposed similar dense methods using the data from RGB-D cameras.

Figure 3.1 illustrates the steps required to find the transformation from set of image frames using a sparse visual odometry approach.

The typical structure in a sparse visual odometry system begins with feature extraction from the new image using proper feature detector method (in this thesis Kanade-Lucas-Tomasi (KLT), Speeded Up Robust Features (SURF) are used to detect the features). Afterwards, correspondences between the new features and features from the previous frame are established. This can be obtained by comparing small patches around the feature points. A match is assumed if the error between two patches is minimal. Instead of patches feature, descriptors can also be used. These descriptors are calculated from the surrounding pixels of a feature point and represented as vectors.
Although these descriptors are more robust to mismatches than image patches, they are more expensive to compute.

Figure 3.1. Flowchart of transformation estimation from set of image frames

Finally, the pose or transformation between two images is computed by minimizing the re-projection error between every pair of matched feature points.

Various approaches (2D-to-2D, 3D-to-2D and 3D-to-3D) can be implemented to ensure correct feature association for improving the accuracy of the motion estimate. As an additional or final step, the detected features and camera poses can be integrated into a global map. Using global optimization approach, the map and the camera trajectory can be further tuned and corrected to obtain position estimates with higher precision and compensate the drift over time.

3.3 Feature detection

Feature extraction to find a sparse set of corresponding locations across different images is the first step in many approaches for camera pose estimation. The main advantage of key-points is that they permit matching even in the presence of clutter, occlusion, and large scale and orientation changes (Szeliski, 2010).

There are two main approaches to extract the feature points and their correspondences. The first is to find features in one image and track the feature across other images using a local search technique, such as correlation or least squares. The second is to independently detect features in
all the images under consideration, and then match the features based on their local appearance. The former approach is more suitable when images are captured from close range or in rapid sequence (e.g., video sequences). In cases where the camera has large amount of motion or appearance is changed the latter approach is more suitable. Example of previous technique is in stitching together panoramas (Ma et al., 2007), establishing correspondences in wide baseline stereo (Schaffalitzky and Zisserman 2002), or performing object recognition (Fergus et al., 2007).

The feature point detection and matching steps can be divided into four separate stages. During the feature detection (extraction) stage, each image is searched for locations that are likely to match well in other images. At the feature description stage, each region around detected key-point locations is transformed into a more compact and stable (invariant) descriptor that can be matched against other descriptors. The feature matching step, efficiently searches for correct matching candidates in other images. The feature tracking step is an alternative solution to the third step that only searches a small neighbourhood across each detected key-points and is therefore more reliable for video processing.

Once the feature detection is done, a decision must be made on which features come from corresponding locations in different images in order to match them. In some cases, e.g., for video sequences (Shi and Tomasi 1994) or for stereo pairs that have been rectified (Scharstein and Szeliski 2002), the local motion around each feature point may be mostly translational. In this case, simple error metrics, such as the sum of squared differences (SSD) or normalized cross-correlation (NCC) (Zhao, 2006), can be used to directly compare the intensity variation in small patches around each feature point (Tuytelaars, 2008). Since feature points may not be accurately extracted, a more accurate matching score can be calculated by performing incremental motion refinement, but this can be computationally expensive and may even decrease performance (Brown et al.,
2005). In some cases, however, the local appearance of features will change in orientation and scale, and sometimes may undergo affine deformations, (Manmatha, 1993). Hence, it is better to first extract a local scale, orientation, or affine parameters (Domokos, 2010) and in the next stage use these parameters to resample the patches before creating the feature descriptors.

3.4 Feature matching

After features extraction and correspondence search across consecutive images, some preliminary feature matches between these images should be established. In this section, the feature matching process is divided into two separate steps. The first is to select a matching technique, to identify which correspondences are passed on to the next stage for further processing (Szeliski, 2010). The next stage is to use efficient data structures and algorithms to apply this matching as quickly as possible.

3.5 Matching strategy and error rates

Deciding on which features are reliable for matching depends on the context in which the matching is being performed. Typically, the Euclidean (vector magnitude) distances in feature space can be directly used for deciding on potential match points.

having Euclidean distance metric, the simplest matching technique is to set a pre define threshold (maximum distance) and to select all matches from other images within this threshold. If the threshold is set to a high value, there will be too many false positives, and incorrect matches. However, if the threshold is set to a low value, there will be too many false negatives, or many correct matches being missed.
In Figure 3.2 digits 1 and 2 are features being matched against features in other images. At the current threshold setting (the solid circles), the green 1 is a true positive (good match), the blue 1 is a false negative (failure to match), and the red 3 is a false positive (incorrect match). If the threshold is higher than the dashed circles, the blue 1 becomes a true positive but the brown 4 becomes an additional false positive.

Because of difficulty to set the distance threshold; the useful range of thresholds can vary a lot by moving to different parts of the feature space (Mikolajczyk and Schmid, 2005). In Figure 3.3 at a fixed distance threshold (dashed circles), descriptor fails to match (e.g. incorrectly matches $D_C$ and $D_E$). A preferred technique in such cases is to easily match the nearest neighbour in feature space. Since some features may have no corresponding matches (e.g., they may be part of background clutter or they may be occluded in the other image), a threshold is still used to cut down the number of false positives.
Figure 3.3. Fixed threshold, nearest neighbour, and Nearest Neighbour Distance Ratio (NNDR) matching.

For example if the nearest neighbour is picked, $D_A$ correctly matches $D_B$ but $D_D$ incorrectly matches $D_C$.

A useful strategy for searching can be to examine the nearest neighbour distance to that of the second nearest neighbour, preferably taken from an image that is known not to match the target (e.g., a different object in the database) (Brown, 2002). It is possible to define this as nearest neighbour distance ratio (Mikolajczyk and Schmid 2005) given in an equation (3.1)

$$NNDR = \frac{d_1}{d_2} = \frac{\|D_A - D_B\|}{\|D_A - D_C\|}$$  \hspace{1cm} (3.1)

where $d_1$ and $d_2$ are the nearest and second nearest neighbor distances, $D_A$ is the target descriptor, and $D_B$ and $D_C$ are its closest two neighbours.

According to Figure 3.3 using Nearest Neighbour Distance Ratio (NNDR) matching, the small NNDR $d_1/d_2$ correctly matches $D_A$ with $D_B$, and the large NNDR correctly rejects matches for $D_D$. 

42
3.5.1 Efficient matching

Once the decision is made on the matching technique, a search should be done to robustly select potential candidates. The easiest method to find all corresponding feature points is to compare all features against all other features in each pair of potentially matching images. Because the previous process is quadratic in the number of extracted features, it is impractical for most applications.

A better technique is to use an indexing structure, such as a multi-dimensional search tree or a hash table, to quickly search for features close to a given feature. Such indexing structures can either be built for each image separately (which is suitable if only certain matches are considered, e.g., searching for a particular object) or it can be built globally for all the frames in a given database. The later approach is faster, since it removes the need to loop over each image. In case of having large databases (millions of images or more), even more efficient structures based such as vocabulary trees, (Nister and Stewenius 2006)) can be used. A simpler approach is multi-dimensional hashing, which copy descriptors into fixed size buckets or cells based on some function used to each descriptor vector. At matching time, each new feature is map into a bucket, and a search of nearby buckets is used to return potential candidates, which can then be sorted or graded to determine which are valid matches.

Other methods are k-d trees (Samet, 1989), slicing (Nene and Nayar, 1997), metric tree (Nister and Stewenius, 2006). Muja and Lowe (2009) evaluate a number of these approaches, and introduce a new one based on priority search on hierarchical k-means trees. According to previous approach multiple randomized k-d trees often provide the best performance.

3.5.2 Feature match verification

Once corresponding matches have been found across the consecutive images, it is possible to use geometric alignment to verify which matches are inliers and which ones are outliers. For instance,
if the image expected to be rotated or translated in the matching view, it is possible to consider a global geometric transform and store only matching features that are sufficiently close to this estimated transformation. A procedure for choosing a subset of matches and then identify a larger set is often called RANdom SAmple Consensus (RANSAC) (Fischler and Bolles, 1981). Once an initial set of correspondences has been obtained, some systems search for additional matches, e.g., by looking for additional correspondences along epipolar lines or in the neighboring of estimated locations based on the global transforms.

**3.6 Feature-based pose estimation**

Feature-based alignment is the problem of estimating the motion between two or more sets of matched 2D or 3D points.

Figure 3.4 shows three common methods (Scaramuzza, 2011) for obtaining the pose of the camera from a set of images:

- 2D-to-2D image matches correspondence using epipolar constraints(recover motion up to scale)
- 3D-to-2D correspondence known as Perspective-n-Point (PnP) in computer vision or Resection in photogrammetry
- 3D-to-3D point registration (Iterative Closest Point (ICP))
3.6.1 2D-to-2D feature based pose estimation

The majority of conventional computational procedures in photogrammetry rely on the correspondence between image point primitives where there is no attempt to recover any explicit three-dimensional (3D) geometry of the scene. The geometric relationship between projection centers of two cameras, a point in 3D space, and its potential location in both images is defined by Epipolar geometry. A benefit of knowing the epipolar geometry is that, given any point in either image, epipolar geometry defines the possible locations of the corresponding point (if visible) in the other image meaning that the search space is reduced from 2D to 1D for finding the correspondence in other image frame.
Figure 3.5. Epipolar geometry

Figure 3.5 (a) illustrates how an epipolar plane is obtained by both camera centres and any 3D point captured by cameras. In Figure 3.5 (b) the object point will lie on the ray passing the image point and its centre of projection, $C$ and $C'$. Any 3D point $X$, captured by both cameras, forms a plane $\Pi$ along with both centres of projection. This plane will intersect both image planes forming lines $l$ and $l'$, called epipolar lines. Therefore, if image points $x$ and $x'$ correspond to a single physical point $X$ then $x, x', C, C'$, and $X$ must all lie in a single plane. This is known as the coplanarity constraint. An epipole is the image of the centres of projection in the other image plane (denoted in Figure 3.5 by $e$ and $e'$). By choosing different 3D points different planes are created all of which pass through the epipoles (i.e. centres of projection). This is illustrated in Figure 3.6. All planes pass through the epipoles intersect the image planes at epipolar lines. Searching correspondence along the epipolar result in better matching pairs in both cameras.
Figure 3.6: For every planes passing through epipole, there is a corresponding epipolar line $l$ and $l'$, in the respective image planes.

Using perspective projection, the relationship between corresponding points which are captured in stereo-pair (2D to 2D), can be described by the co-planarity condition. This condition mathematically describes the fact that corresponding points in the reference and input images belong to the corresponding epipolar plane. The coplanarity constraint can be defined by constraining the normal to the epipolar plane to be perpendicular to the base vector (Equation 3.2).

This condition is defined as follows.

\[
(\overrightarrow{C}x \times \overrightarrow{C'}x) \cdot \overrightarrow{b} = 0
\]  

(3.2)

Where

$\overrightarrow{b}$: The vector between the two perspective centers of the stereo pair, referred to as the image base

47
\[ \bar{b} = C - C' = \begin{bmatrix} C_x - C'_x \\ C_y - C'_y \\ C_z - C'_z \end{bmatrix} \]  \hspace{1cm} (3.3)

\( \bar{C}x, \bar{C}'x \) : The vectors from the perspective center to conjugate points in the left and right images, respectively.

\[ \bar{C}x = R(\omega, \varphi, \kappa)_l \begin{bmatrix} x_l - x_p \\ y_l - y_p \\ -c \end{bmatrix}, \quad \bar{C}'x = R(\omega, \varphi, \kappa)_r \begin{bmatrix} x_r - x_p \\ y_r - y_p \\ -c \end{bmatrix} \]  \hspace{1cm} (3.4)

Where

\( x_p, y_p, c \): Interior orientation parameters (calibrated principal point position and principal distance of the camera with respect to image coordinate system).

\( x_l, y_l, x_r, y_r \): Image point coordinates in left and right camera.

\( \omega, \varphi \) and \( \kappa \): Representing the rotation angles between the ground and image coordinate systems.

### 3.6.2 3D-to-2D feature based pose estimation

The second class for estimating the transformation parameters is based on 3D-to-2D points. Example of this method is a Resection (known as PnP in computer vision) technique, where EOP of an image can be recovered by having at least a set of three non-collinear Ground Control Points (GCP).

In this technique, the collinearity model is used to relate points in image coordinate with points in object coordinate space, and this relation is expressed as a function of the exterior orientation parameters.

Traditionally, the parameters are estimated by way of a mathematical adjustment involving manually identifying and measuring control points in the image. Because each measured point forms two collinearity equations, at least three control points are required to estimate the six
exterior orientation parameters. The introduction of more than three points increases the redundancy and strengthens the solution of the parameters.

\[
\begin{bmatrix}
    x_i - x_p \\
    y_i - y_p \\
    -c
\end{bmatrix} = sR^T(\omega, \varphi, \kappa) \begin{bmatrix}
    X_i - X_o \\
    Y_i - Y_o \\
    Z_i - Z_o
\end{bmatrix}
\] (3.5)

Where

- \( s \) is the scale;
- \( x_i, y_i, \) are the image coordinate of the point;
- \( X_i, Y_i, Z_i \) are the object coordinates of the \( i^{th} \) point;
- \( X_o, Y_o, Z_o \) are the object coordinates of the perspective center;
- \( \omega, \varphi, \kappa \) are the rotations between the image and ground coordinate systems; and
- \( c \) is the calibrated focal length.

### 3.6.3 3D-to-3D feature based pose estimation

#### 3.6.3.1 Stereo based visual odometry

In recent years many algorithms for visual odometry have been developed, which can roughly be divided into two categories, monocular cameras based methods (Yamaguchi, 2006) and methods using stereo camera. These approaches can be subdivided into methods which either use feature matching (Johnson, 2008) between consecutive images or feature tracking over a sequence of images.

Using two cameras (stereo vision) with enough baseline between them, it is possible to estimates the depth and motion between consecutive stereo camera frames. In the following section pose estimation algorithm using stereo vision camera is explained. The algorithm consists of two stages where in the first stage the system is initialized for creating a reference point, and in the next stag the system enters a loop to estimate the pose of the camera.
The initialization consists of the following steps (Figure 3.7):

1. Reading in an image pair.
2. Detecting features in the image pair.
3. Matching the features between the left and right pair. Delete all features that have not match, or are considered outliers.
4. Triangulating the valid matches and get their 3D real world coordinates.
5. Saving the matched features, their properties, and the resulting 3D point cloud. This will be the key-frame reference. Until a new key-frame is taken, all movement is calculated with reference to this frame.

Figure 3.7 Schematic of the pose estimation algorithm. Initialization to get a key-frame and store the properties.
After obtaining the reference data, the main loop will run, which will provide us the pose estimated with respect to the reference frame. It should be mentioned that the reference frame would change if no correspondences found in the following frames. The following steps describe the main loop procedure (Figure 3.8):

1. Reading in a new pair of images.
2. Extracting features.
3. Matching features between the current left image and the saved key-points of the key-frame left image. Delete all key-points in the current image that have not been matched to a key-point in the key-frame or are considered outliers.
4. Matching the remaining key-points in the current left image to the current right image. Again delete all key-points that have not been matched successfully or are considered outliers.
5. Triangulating the valid matches in the current frame. Save the resulting 3D coordinates to the current point cloud.
6. Using the current point cloud and the saved key-frame point cloud in the pose estimation routine to get the rotation matrix R and the translation vector T which describe the current position w.r.t. the key-frame.

After extracting features in the images, correspondences between two sets of such features have to be detected. This has to be done between the left and right image of one frame, as well as between the current frame and the key-frame.
Figure 3.8 Schematic of the pose estimation algorithm. Loop to read images, find key-points, find correspondences in the key-frame, then find correspondences between left and right, triangulate and finally estimate the pose.

There are two main categories to find correspondences in consecutive camera frames: Those relying on descriptors (matching), and those that do not use any descriptors (tracking).

Regardless of which method is used, finding correspondences usually consists of two steps (Hurzeler, 2010):
1. Correspondences between the left image of the present frame and the left image of the key-frame should be found. This part of the algorithm is called the tracking step. The reason is that, here features are tracked since they stay in the field of vision. The correspondences found in this stage are later used in the pose evaluation step, where the estimator needs to know which triangulated 3D point of the present frame relates to which 3D point of the key-frame. It would also be beneficial to do it for both the left and right images, but the gain in performance would not worth extra computational effort. After matching, all the features in the current left image that have not been successfully matched to a feature in the key-frame are removed in order to accelerate the rest of the algorithm.

2. The remaining features in the left image are now paired with the features in the right image. Here, using rectified images can be exploited to either accelerate the matching process, or, at least to remove outliers. Using rectified images (assuming a perfect rectification) means that features in the left image can only correspond to features in the right image that have the same y-value. This is because rectification warps the image such that the epipolar lines are parallel and row aligned. Because the image rectification is not ideal, therefore usually not only points directly on the epipolar line, but points within a marginal spanned tolerance in y-direction are also considered (Figure 3.9).

![Figure 3.9. Left-right matching using rectified images: allow a small tolerance to compensate for imperfect rectification.](image-url)
Once the correspondences between image frames are obtained we have to do the matching part. Basically matching can be done using two different approaches:

1. Descriptor based matching: For every feature a descriptor is computed. This happens in both images independently, which requires a feature detector with good repeatability. The advantage of using descriptors is that choosing a smart descriptor can have a positive effect on speed and accuracy of the matching process (Figure 3.10). The drawback of using descriptors is the computation cost which is mainly due to matching process.

![Figure 3.10. Feature matching (2 frames, moving camera), colors encode disparities](image)

2. Without Descriptors (tracking): Detecting correspondences without the use of descriptors has the advantage, that features have to be extracted less often, and the computational effort to compute descriptors can be saved. Figure 3.11 show the schematic of descriptor-less matching: features are only extracted in one image, and correspondences are then searched in the other image without computing any features or descriptors.
The later approach (tracking) is mostly done in computer vision using optical flow method. Optical flow computes the displacement of pixels from one image to the other based on intensity gradients. As described before, only initial features have to be detected. For one such feature in the first image, the displacement computed by optical flow yields the corresponding feature in the second image (Figure 3.12 and Figure 3.13).

Figure 3.11. Schematic of matching without using descriptors

Figure 3.12. Feature tracking and matching (5 frames, stereo camera), colors encode track orientation.
Because of the well-known aperture problem, the basic idea of computing optical flow usually relies on two assumptions:

1. The pixel intensities remain constant from one image to the next. This is the so-called brightness constraint.
2. The optical flow is smooth, there are no sudden jumps in its magnitude or sudden changes of directions. This is the so-called smoothness constraint.

The main properties of the optical flow matcher are

- Features only need to be computed in the key-frame, and no descriptors are needed.
- Relatively robust to motion blur.
- Subpixel accuracy (using multiple pyramid levels).
• Changes in illumination can violate the brightness consistency constraint.

Once the features are extracted and the correspondences are matched, triangulation should be performed to find the 3D coordinate corresponding to the image points and use these 3D points to find the pose of the camera. The pose is computed using 3D-to-3D matching technique in a least square optimal way (minimizing the sum of square distance between 3D-to-3D points) (Figure 3.14).

Figure 3.14. Stereo visual odometry, Results on the Karlsruhe data set (Kitt, 2010)

Once the matched feature points are obtained we can use the triangulation to obtain the 3D reconstruction of the environment (Figure 3.15).
Figure 3.15. Reconstruction of the scene from triangulation (Kitt, 2010)

The use of stereo vision is not the only possible choice to estimate the pose of the system. In general, it is possible to estimate motion with monocular vision. The advantage of such a setup is primarily the potential to save weight and power by having only one camera. Using monocular vision, it is possible to compute the essential or fundamental matrix by using a 7- or 8-point algorithm on consecutive images taken by the camera. Decomposing the matrix leads to the recovery of the translation and rotation, though the translation can only be found up to a scaling factor. In general, a unique solution can be found, but in order to get a robust recovery algorithm, the baseline between consecutive images has to be sufficiently large and the images have to be taken with two different view points.

It is also possible to use RGB-D sensor for pose estimation which will be discuss in the next section.
3.6.3.2 RGB-D-based Visual Odometry

Many relevant applications in robotics and computer vision require the ability to quickly acquire 3D models of the environment and to estimate the camera pose with respect to this model. A robot, for example, needs to know its location in the world to navigate between places. This problem is a classical and challenging chicken-and-egg problem because localizing the camera in the world requires the 3D model of the world, and building the 3D model in turn requires the pose of the camera. Therefore, both the camera trajectory and the 3D model need to be estimated at the same time. With the introduction of the Microsoft Kinect camera, a new sensor has appeared on the market that provides both color images and dense depth maps at full video frame rate. This allows us to create 3D model of indoor environment using depth information and visual features.

In this thesis two different methods for visual odometry is proposed which are based on 3D-to-3D matching using RGB-D sensor. In the first method, different 2D feature extraction methods namely SURF (Bay, 2008) and Lucas–Kanade (Lucas and Kanade, 1981) are used for extracting 2D image points. RANSAC is used in the next step to remove the outliers. In the following step the inlier features are project in 3D space using depth information from RGB-D sensor to extract the pose through 3D-to-3D matching. It should be mentioned for correct matching it should be always check if the corresponding point in 2D image has depth information. This is very important as the pose estimation may diverge due to the incorrect correspondence between 2D and 3D points. Figure 3.16 illustrates the process for extracting the features for pose estimation.
Another consideration that should be taken into account when using RGB-D sensor is distance, as the RGB-D sensor measures up to almost 5 meters, hence additional constrain should be taken in to account to reject the point with higher than 4 meters depth information, so that they won’t affect the 3D-to-3D matching part solution.

Working with 2D image points for 3D-to-3D matching and pose estimation, it is necessary to check on every frame that the extracted 2D features in the images have corresponding depth values and discard those points without depth information.
The later issue motivated us for applying second method for visual odometry based on 3D features points namely Normal Aligned Radial Feature (NARF) (Steder et al., 2011). Since previous 3D feature detection technique relies on knowledge of normal and curvature properties of the surface, these parameters should be explained first.

3.6.3.3 Surface Normals and Curvature Estimates

The raw point cloud does not provide any semantic information about the type of the scanned surfaces (i.e., planar, linear surfaces). Therefore, the segmentation methodology starts by utilizing PCA to determine the geometric properties of the local neighbourhood of laser points. To check whether or not a certain laser point (query point) belongs to planar, linear, or cylindrical surface, the following steps are taken. First, a local neighbourhood \((p_i)\) is defined to enclose the \((n)\) nearest neighbors to the query point. Then, a covariance matrix \((C)\) is formed based on the dispersion of the points from their centroid \((\bar{p})\) as given by equation 3.6

\[
\bar{p} = \frac{1}{k} \sum_{i=1}^{k} p_i
\]

The solution for normal vector \(\vec{n}\) is given by analyzing the eigenvalues and eigenvectors of the covariance matrix \(C \in R^{3 \times 3}\) of points, express as:

\[
C = \frac{1}{k} \sum_{i=1}^{k} w_i (p_i - \bar{p}) \cdot (p_i - \bar{p})^T
\]

In equation 3.7 The term \(w_i\) represent a possible weight for \(p_i\). \(C\) is symmetric and positive semi-definite, and its eigenvalues are real numbers \(\lambda_j \in R\).

The eigenvectors and eigenvalues are quite useful in determining the geometric nature of the established neighbourhood. The eigenvectors represent the orientation of the neighbourhood in the 3D space, while the eigenvalues define the directions of corresponding eigenvectors (Pauly et al.,...
For a planar neighbourhood, one of the estimated eigenvalues will be quite small when compared to the other two. The eigenvector that corresponds to the smallest eigenvalue will represent the approximate direction of the normal vector to the plane enclosed within this neighbourhood. In general, the quality of the estimated information through the eigenvalue analysis depends on the number of the enclosed points within the neighbourhood and the existence of outliers.

This ratio between the minimum eigenvalue and the sum of the eigenvalues approximates the change of curvature in a neighbourhood of points centered around a point (Equation (3.8)).

$$\sigma_p = \frac{\lambda_0}{\lambda_0 + \lambda_1 + \lambda_2}$$

(3.8)

Small values of $\sigma_p$ indicate that all points are on the plane tangent to the surface.

3.6.3.4 Normal Aligned Radial Feature (NARF)

The Normal Aligned Radial Feature (NARF) key-point detector has two major characteristics: Firstly, NARF extracts key-points in areas where the underlying surface is stable and the neighbourhood contains major surface changes. This causes NARF key-points to be located in the local environment of significant geometric structures and not directly on them. Secondly, NARF takes object borders into account, which arise from view dependent non-continuous transitions from the foreground to the background. Thus, the silhouette of an object has a direct influence on the resulting key-points.

Figure 3.17 illustrates the process for extracting NARF key points.
Stable interest points need significant changes of the surface in a local neighbourhood to be robustly detected in the same place even if observed from different perspectives. The following procedure are taken to extract stable NARF interest points:

- Finding borders in the range image (non-continuous traversals from foreground to background), this is done by looking for substantial increases in the 3D distances between neighbouring image points.
- Checking the local neighbourhood of every image points and determine how much the surface changes at that position, this will give more information about the borders.
• Finding the dominant directions in the surrounding of each image point and calculating an interest value that the points must be in positions, which support stable areas for normal estimation or the descriptor calculation in general.

3.7 Calibration of visual sensor

Calibration is an important pre-step process required in order to extract internal and external camera parameters for further processing. Generally, calibration techniques are classified into two categories: photogrammetric calibration and self-calibration.

Photogrammetric calibration:

• In this method camera calibration is performed by looking at a calibration object whose geometry in 3-D space is known with good precision. The calibration object usually consists of two or three planes, which are orthogonal to each other. Calibration apparatus and carefully arrange setup are required for this approach.

Self-calibration

• In this technique no calibration objects is required. The camera internal parameters are obtained with image information while changing the position of the camera in a static scene. Therefore, if images are captured by the same camera with fixed internal parameters, correspondences between three images are sufficient to recover both the internal and external parameters. While this approach is very flexible, it requires many parameters to be estimated, and the results might not be reliable.

In this work calibration is done on RGB-D sensor as it will be used in the mapping process. The RGB-D device has two cameras (RGB and depth) and one laser-based IR projector as shown in Figure 3.18
In this work the calibrations was done based on chessboard patterns, using OpenCV’s calibration routines. Here, the calibration procedure and the output parameters are briefly described.

This calibration technique only requires the camera to observe a planar pattern shown in Figure 3.19 at a few (at least two) different orientations. The pattern can be printed on a laser printer and attached to a reasonable planar surface (e.g., a hard book cover). Either the camera or the planar pattern can be moved by hand.

For the calibration, the assumption is the camera 3D coordinate coincides with the world coordinate system. In the homogeneous representation, a 3D point in the world coordinate system is denoted by \( M = [X, Y, Z, 1]^T \), and its corresponding 2D projection in the color image is \( m = [u, v, 1]^T \). The camera is model by the usual pinhole model according to equation (3.9)

\[
m_{3\times1} = sK_{3\times3}[R \ t]_{3\times4}M_{4\times1}
\]

Where \( s \) is a scale factor and \((R, t)\), are extrinsic parameters relating the world coordinate system to camera coordinate system. \( K \) is the intrinsic matrix, which is given by equation(3.10)
\[ K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \] \hspace{1cm} (3.10)

In equation (3.10), \( f_x, f_y \) are the focal lengths in x- and y-direction and \((c_x, c_y)\) are the coordinates of the principal point.

In total, there are \( n \) image pairs (color and depth) captured by the RGB-D camera. The positions of the calibration board in the \( n \) images are different, as shown in Figure 3.19. The set up consists of local 3D coordinate system \((X_i, Y_i, Z_i)\) for each position of the calibration model plane, such that the \( Z_i = 0 \) meaning that the plane coincides with the model plane. This has shown in equation (3.11)

\[ Z=0 \text{world coordinate and calibration plane are coincide} \rightarrow \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = s \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & t_x \\ r_{21} & r_{22} & t_y \\ r_{31} & r_{32} & t_z \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \] \hspace{1cm} (3.11)

In addition, here it is also assume that the model plane has a set of \( m \) feature points. Usually features are the corners of a checkerboard pattern.

Given the set of feature points, the goal is to recover the intrinsic matrix \( K \), and extrinsic parameters (rotations and translations \( R_i, t_i \)), and the transformation between the color and the depth cameras \( R \) and \( t \).
In order to calculate the extrinsic calibration parameters, the concept of homography was used as the same planar surfaces were observed by both sensors (RGB and depth).

A planar projective transformation or homography is transformation on homogeneous 3-vectors represented by a non-singular $3 \times 3$ matrix $H$ in equation (3.12):

Using equation (3.11) and homography approach, it is possible to find the rotation and translation parameter according to equation (3.12)

$$H = \begin{bmatrix} h_1 & h_2 & h_3 \end{bmatrix} = s \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & t_x \\ r_{21} & r_{22} & t_y \\ r_{31} & r_{32} & t_z \end{bmatrix} = sK[R_1 \ R_2 \ t]$$

$$\rightarrow R_1 = \frac{1}{s}K^{-1}h_1 \quad R_2 = \frac{1}{s}K^{-1}h_2 \quad t = \frac{1}{s}K^{-1}h_3$$

There is no guarantee that the estimated $R_1$ and $R_2$ from equation (3.12) will be orthogonal. Hence, the orthogonality of rotation matrices should be checked according to equation (3.13).
\[(1) \rightarrow R_1^T R_2 = \left( \frac{1}{s} K^{-1} h_1 \right)^T \left( \frac{1}{s} K^{-1} h_2 \right) = 0 \] (3.13)

\[\rightarrow h_1^T (K^{-1})^T K^{-1} h_2 = 0\]

\[(2) \rightarrow \|R_1\| = \|R_2\| \rightarrow R_1^T R_1 = R_2^T R_2 \]

\[\rightarrow h_1^T (K^{-1})^T K^{-1} h_1 = h_2^T (K^{-1})^T K^{-1} h_2 \]

\[B = (K^{-1})^T K^{-1} = \begin{bmatrix}
\frac{1}{f_x^2} & 0 & -c_x \\
0 & \frac{1}{f_y^2} & -c_y \\
-c_x & -c_y & \frac{1}{f_x^2} + \frac{1}{f_y^2} + 1
\end{bmatrix}\]

Camera intrinsic parameters can be extracted from \(B\) where \(B\) is symmetric and defined by a 6D vector

\[B = [B_{11}, B_{12}, B_{13}, B_{22}, B_{23}, B_{33}]^T\]

Using the symmetric form of \(B\) it is possible to rewrite equation (3.13) according to equation (3.14)

\[h_1^T B h_j = v_{ij}^T b = 0\]

\[h_1^T B h_1 - h_2^T B h_2 = v_{11}^T b - v_{22}^T b = 0\]

Where

\[v_{ij} = [h_{i1} h_{j1}, h_{i1} h_{j2} + h_{i2} h_{j1}, h_{i2} h_{j2}, h_{i3} h_{j1} + h_{i1} h_{j3}, h_{i3} h_{j2} + h_{i2} h_{j3}, h_{i3} h_{j3}]^T\]

Therefore, the two fundamental constraints given in equation (3.14) from a given homography, can be rewritten as 2 homogeneous equations as shown in equation (3.15):

\[\begin{bmatrix}
v_{12}^T \\
(v_{11}^T - v_{22}^T)
\end{bmatrix} b = 0 \] (3.15)

The solution to equation (3.15) is through eigenvector analysis of \(V^T V\) associated with the smallest eigenvalue where \(V\) is a \(2n\times6\) matrix. If \(n \geq 3\) a unique solution for \(b\) can be obtained. Once \(b\) is
estimated, camera intrinsic matrix $K$ and extrinsic parameters can be computed based on equation (3.12).

Distortion model is considered according to equation (3.16)

$$x_i' = x_r'(1 + k_1r^2 + k_2r^4 + k_3r^6) + 2p_1x_r'y_r' + p_2(r^2 + 2x_r'^2)$$

$$y_i' = y_r'(1 + k_1r^2 + k_2r^4 + k_3r^6) + 2p_2x_r'y_r' + p_1(r^2 + 2y_r'^2)$$

(3.16)

In equation (3.16)

$x_i', x_r'$: Corresponding image coordinates

$r = \sqrt{x_r'^2 + y_r'^2}$

$k_1, k_2, k_3$: Radial distortion coefficients

$p_1, p_2$: Decentric lens distortion parameters.

Considering $Q = [k, p]$, if at least three corresponding points $x_i', x_r'$ are observable then equation (3.16) can be solved according to

$$\arg \min \left[ \sum_{i=0}^{n} (x_{ri}' - f(Q, x_{ri}')(x_{ri}' - f(Q, x_{ri}'))^T \right]$$

(3.17)

In equation (3.17), $n$ is number of image correspondences. $f(Q, x_{ri}')$ is the right side of equation (3.16). Equation (3.7) can be solved using singular value decomposition.

Figure 3.20 illustrates the software interface for calibration of RGB-D sensor. The input to the software is set of images captured from chessboard pattern at different locations. In order to calculate the internal and external camera parameters, the software requires number of internal corners of chessboard pattern in horizontal and vertical direction. Also, it is necessary to define the size of the chessboard cube, so that the corner detection algorithm works better.
Figure 3.20. Software interface for camera calibration
Chapter Four: **Fine Registration**

### 4.1 Introduction

Current 3D indoor mapping systems can be categorized based on their method of registration. (Pomerleau et al., 2011, used a total station to localize a set of static laser scanner positions with millimeter precision. To measure the position and orientation of the scanner, they mounted three reflective prisms around the platform, allowing the total station to make three separate distance measurements. However, this system is inefficient when the position and mounting angles of the scanner could not be measured from the position of the total station. When this occurred, the total station had to be moved and re-localized, which greatly increase the mapping time.

A faster and more reliable method is to distribute reflective targets in the scene and use them to align the scans in post-processing mode. Zimmermann and Eßer (2008) used this technique to survey a catacomb, since it provides some flexibility in positioning the scanner. They reported an average error of 5–6mm when aligning scan positions, given a minimum of five reflective targets viewable in common. However, Brenner et al. (2008) noted that this technique has several drawbacks. They reported that distributing manually the targets, collection and manual intervention during registration takes five times longer than the scanning time.

A hybrid technique is often used for larger environment, where the cumulative errors of target-based registration can become very large (Zimmermann, 2008). In these cases, a set of ground control points are used by conducting a traverse with a total station and these are used as a baseline for target alignment.

Automatic 3D point cloud alignment is a main issue in mobile mapping applications as it is faster and does not require any external information. The most commonly used solution for automatic
registration is the well-known ICP (Iterative Closest Point) algorithm. This approach performs a fine registration of two overlapping point clouds by iteratively minimizing the sum of squared distances between corresponding points to estimate the transformation parameters.

In this chapter four different registration techniques namely point-to-point, point-to-plane and plane-to-plane and edge based ICP will be discussed in detail. In order to minimize the cost function and obtain the transformation parameters, heading orientation only considered for the platform.

The main drawback of iterative registration is that the registration process may get caught in local minima if the assumptions do not hold, for example if the clouds only partially overlap and/or if the initial alignment is further away from true reference. In this case, false correspondences can negatively affect the alignment solution. However, various methods exist to sort out false correspondences (rejection step) and improve convergence. Moreover, if the clouds are already roughly aligned (coarse registration), iterative registration provides efficient and robust means to refine that initial guess and optimally align the point clouds.

4.2 Registration

Basically, a point cloud \( p \) is a data structure that is used in order to represent a collection of multi-dimensional points \( p \in \mathbb{R}^n \). In a 3D point cloud, the elements usually represent by \( X, Y, \text{and} \ Z \) geometric coordinates of an underlying sampled surface. When more information is available, such as color information or information about local surface normal \( n \) or curvature, the points \( p \in P \) are represented by a longer vector. Given a source point cloud \( p \) and target point cloud \( q \), the problem of registration relies on finding correspondences between \( p \) and \( q \), and estimating a transformation \( T \) that, when applied to \( p \), aligns all pairs of corresponding points. One of the main
problems of alignment is that these correspondences are usually not known and need to be determined by the registration algorithm. Having correct correspondences, there are different ways of computing the optimal transformation that align the point clouds. In general, the main objective of registration is to align individual point clouds and fuse them to a single point cloud, so that subsequent processing steps like segmentation and reconstruction can be applied. Figure 4.1 shows the process of registration using the clouds obtained from sensors. The idea behind pre-processing step is to down-sampling the point cloud for uniform distribution and reducing the computational cost. Two stage registration is proposed in this thesis: first is based on coarse registration described in previous chapter using information from RGB-D sensor and the other is based on the fine registration using iterative closest point.

Figure 4.1 Overview of the registering process using a pair of point clouds
4.3 General Registration Pipeline

The fine registration can be divided into six stages, which will be described in more details in the next sub-sections:

- Selecting set of points in one or both clouds.
- Matching source cloud points to the target cloud points.
- Weighting the corresponding pairs appropriately (high weight to points with close distance and less weight to points with far distance in ICP).
- Rejecting certain pairs based on looking at each pair individually or considering the entire set of pairs.
- Assigning an error metric based on the point pairs and minimizing the error metric.

4.3.1 Selection of Points

Based on the application and sensor used for measurement, point clouds can become quite large. Consequently, aligning large point clouds is computationally expensive than registering clouds with smaller size. Moreover, sometimes the data is redundant or unnecessarily detailed for the task of registration. Hence, registering only subsets of the original point clouds is enough.

The following strategies can be used for the proper selection of the points:

- Using all available points (computationally costly as the search for registration should be done between all the available points) (Besl, 1992).
- Uniform sub-sampling of the available points (Turk, 1994).
- Random sampling (with a different sample of points at each iteration) (Masuda, 1996).
- Selection of points with high intensity gradient, if color or intensity is available (Weik, 1997).
- Choosing points such that the distribution of normal among selected points is as large as possible (Rusinkiewicz, 2001)
According to (Rusinkiewicz, 2001) for a scene with a good distribution of normals the exact sampling strategy is not critical and the convergence performance is similar. However this is not the case if the distribution of the points varies from place to place.

4.3.2 Matching Points

Correspondence finding and matching is the next stage in registration process. Basically the correspondence finding is the process of searching for the closest points in source and destination clouds. Different methods exist to enhance the matching procedure:

- Finding the closest point in the other point cloud (Besl, 1992). This computation can be accelerated using a \( k-d \) tree and/or closest point caching, octrees or voxels(down-sampling of the points based on nearest neighbours)
- Projecting the source point onto the destination cloud, and then performing a search in the destination cloud to minimize the point-to-point distance (Benjemaa, 1997)
- Compatibility metrics based on color (Godin, 1994) and angle between normals (Pulli, 1999) of the source and destination point clouds.

4.3.3 Weighting of Pairs

Assigning different weights to the corresponding point pairs found by the previous two steps (selection of the points and matching points) can enhance the alignment process. The weighting of the point pairs can be considered as a method for correspondence rejection, adjusting the influence of noisy corresponding points in the minimization process.

The following are different weighting conditions

- Constant weight
- Assigning lower weights to pairs with greater point-to-point distances (equation (4.1)). (Godin,1994)
\[ \text{weight} = 1 - \frac{\text{Dist}(p_1, p_2)}{\text{Dist}_{\text{max}}} \]  \hfill (4.1)

- Weighting based on compatibility of normals (equation (4.2)):

\[ \text{weight} = n_1 \cdot n_2 \]  \hfill (4.2)

### 4.3.4 Rejecting Pairs

In most cases, the set of points in the source point cloud do not correspond and relate exactly to the points in the destination or target cloud. In these cases, the outlier rejection can be used to remove the incorrect pairing of points. The following outlines a number of ways that can be used in rejecting the outliers:

- Rejection of corresponding points that have more than user specified distance.
- Rejection of pairs whose point-to-point distance is larger than some multiple of the standard deviation of distances.
- Rejection of pairs containing points on other cloud boundaries (Turk, 1994). Usually, this situation appears when one of the two clouds is incompletely extracted, due to occlusions, and may lead to incorrect alignments. A possible solution is to identify the subsets of points from one cloud that have the same correspondent point on the other cloud and keeping only the pair with the minimum distance.

### 4.3.5 Error Metric and Minimization

The final step of the registration is the algorithm for minimizing the error metric. Below are some common techniques for minimization of the cost function used to generate the transformation parameters:

- Sum of squared distances between corresponding points. For an error metric of this form, there exists closed form solutions for determining the rigid-body transformation that
minimizes the error. Solution methods based on singular value decomposition (SVD)
(Arun, 1987), quaternions (Horn, 1987), orthonormal matrices (Horn, 1988), and dual
quaternions (Walker, 1991) have been proposed. Eggert et. al. have evaluated the
numerical accuracy and stability of each of these (Eggert, 1997), concluding that the
differences among them are small.

• The above “point-to-point” metric, taking into account both the distance between points
and the difference in colors (Johnson, 1997).

• Sum of squared distances from each source point to the plane containing the destination
point cloud (Chen, 1991). In this “point-to-plane” case, no closed-form solutions are
available. The least-squares equations may be solved using a generic nonlinear method
(e.g. Levenberg-Marquardt), or by simply linearizing the problem (i.e., assuming
incremental rotations are small.

• Sum of squared distances between the fitted plane in source cloud and the fitted plane in
destination cloud which is called “plane-to-plane” minimization technique.

4.4 ICP point-to-point minimization

Given two independently acquired sets of 3D points, \( p_i \) and \( q_j \), minimization should be done
according to equation (4.3) in order to find the transformation \((R, T)\) consisting of a rotation matrix
\( R \) and a translation vector \( t \):

\[
E(R, T) = \sum_{i=1}^{N_p} \sum_{j=1}^{N_q} w_{i,j} \left\| R p_i + T - q_j \right\|^2
\]  

(4.3)
Where \( N_p \) and \( N_q \), are the number of points in the source and destination cloud. \( w_{i,j} \) is a weighting parameter and assigned 1 if the \( i-th \) point of \( p \) describes the same point in space as the \( j-th \) point of \( q \), otherwise \( w_{i,j} \) is 0. Two things have to be calculated: First, the corresponding points, and second, the transformation \((R, T)\) that minimizes \( E(R, T) \) on the base of the corresponding points. The double summation in equation (4.3) can be rewritten in a simplified form:

\[
E(R, T) = \frac{1}{N} \sum_{j=1}^{N} \left\| Rp_i + T - q_j \right\|^2, \quad N = \sum_{i=1}^{N_p} \sum_{j=1}^{N_q} sgn w_{i,j}
\]  

(4.4)

The ICP algorithm iteratively calculates the point correspondences. In each iteration step, the algorithm selects the closest points as correspondences and calculates the transformation \((R, T)\) for minimizing equation (4.3). The assumption is that in the last iteration step the point correspondences are correct.

### 4.4.1 Direct Solutions of the ICP Error Function

We can categorize the minimization algorithms into direct and indirect methods. Computations based on Gradient descent, the Gausss Newton method and the Levenberg-Marquardt algorithms belong to the category of indirect methods. Indirect methods have the disadvantage that they need to perform several evaluations of the equation (4.3). Therefore, they need inevitably more time than direct methods. Direct methods are also called closed form solutions.

### 4.4.2 Closed Form Solution

As previously mentioned, different methods exist to calculate the transformation parameters for ICP algorithm. Here, the SVD-based solution is discussed. To calculate the transformation using SVD-based method algorithms, first, computation of rotation from translation should be separated by using the centroid of the points according to equation 4.5:
Replacing equation (4.5) in the error function equation (4.4), the cost function becomes:

\[ E(R, T) = \frac{1}{N} \sum_{i=1}^{N} \left\| Rp_i' - q_i' + (Rp_m + T - q_m) = t \right\|^2 \]

\[ = \frac{1}{N} \sum_{i=1}^{N} \|Rp_i' - q_i'\|^2 + \frac{2}{N} t \sum_{i=1}^{N} (Rp_i' - q_i') + \frac{1}{N} \sum_{i=1}^{N} \|t\|^2 \]

To minimize the above cost function, all terms have to be minimized. In equation (4.6) the second sum is already zero, since all values refer to the centroid. The third part has its minimum for \( t = 0 \) or

\[ t = q_m - Rp_m \]

Therefore the algorithm has to minimize only the first term, and the error function is expressed in terms of the rotation only(equation 4.8):

\[ E(R, T) \propto \sum_{i=1}^{N} \|Rp_i' - q_i'\|^2 \]

Using SVD-based approach, the rotation \( R \) is represented as an orthonormal \( 3 \times 3 \) matrix. The optimal rotation is calculated by \( R = V U^T \) where \( V \) and \( U \) are derived by the singular value decomposition \( H = UAV^T \) of a correlation matrix \( H \). This \( 3 \times 3 \) matrix \( H \) is given by equation 4.9.
\[ H = \sum_{i=1}^{N} p_i^T q'_i = \begin{pmatrix} S_{xx} & S_{xy} & S_{xz} \\ S_{yx} & S_{yy} & S_{yz} \\ S_{zx} & S_{zy} & S_{zz} \end{pmatrix} \]  

\[ S_{xx} = \sum_{i=1}^{N} p'_{ix} q'_{ix}, S_{xy} = \sum_{i=1}^{N} p'_{ix} q'_{iy}, \ldots \]

### 4.4.3 Linearized Solution of the ICP Error Function

Given a rotation matrix based on the Euler angles and considering only heading (equation 4.10) for the platform

\[ R_\theta = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \approx \begin{bmatrix} 1 & -\theta & 0 \\ \theta & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \]  

The cost function can be written according to equation (4.11)

\[ E(R, T) = \sum_{i=1}^{N} \| R p_i + T - q_i \|^2 \]

\[ E(R, T) = \sum_{i=1}^{N} \left\| \begin{bmatrix} 1 & -\theta & 0 \\ \theta & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_{ix} \\ p_{iy} \\ p_{iz} \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} - \begin{bmatrix} q_{ix} \\ q_{iy} \\ q_{iz} \end{bmatrix} \right\|^2 \]

Unknown coefficient \( \theta, t_x, t_y, t_z \) can be estimated by taking partial derivative according to

\[ \frac{\delta E(R, T)}{\delta \theta} = 2 \sum_{i=1}^{N} \left[ \theta (p_{ix}^2 + p_{iy}^2) + p_{ix} t_y - p_{ix} q_{ix} \right] = 0 \]  

\[ \frac{\delta E(R, T)}{\delta t_x} = 2 \sum_{i=1}^{N} (t_x + p_{ix} - \theta p_{iy} - q_{ix}) = 0 \]

\[ \frac{\delta E(R, T)}{\delta t_y} = 2 \sum_{i=1}^{N} (t_y + p_{iy} + \theta p_{ix} - q_{iy}) = 0 \]
\[
\frac{\delta E(R, T)}{\delta t_z} = 2 \sum_{i=1}^{N} (t_z + q_{i,z}) = 0
\]

Therefore, the coefficient \( \theta, t_x, t_y, t_z \) can be obtained as follows:

\[
\theta = \frac{1}{N \sum_{i=1}^{N} (p_{i,x}^2 + p_{i,y}^2) + (\sum_{i=1}^{N} p_{i,x})^2 - (\sum_{i=1}^{N} p_{i,y})^2}
\]

\[
\sum_{i=1}^{N} p_{i,x} \sum_{i=1}^{N} q_{i,y} - \sum_{i=1}^{N} p_{i,x} \sum_{i=1}^{N} p_{i,y}
\]

\[
+N \left[ \sum_{i=1}^{N} p_{i,y} p_{i,x} - \sum_{i=1}^{N} p_{i,x} q_{i,y} \right]
\]

\[
t_x = \frac{1}{N} \left( \sum_{i=1}^{N} q_{i,x} - \sum_{i=1}^{N} p_{i,x} + \theta \sum_{i=1}^{N} p_{i,y} \right)
\]

\[
t_y = \frac{1}{N} \left( \sum_{i=1}^{N} q_{i,y} - \sum_{i=1}^{N} p_{i,y} - \theta \sum_{i=1}^{N} p_{i,x} \right)
\]

\[
t_z = \frac{1}{N} \left( \sum_{i=1}^{N} q_{i,z} \right)
\]

4.5 ICP Point-to-Plane Minimization

Another method for registration is based on point-to-plane error metric (Low, 2004) where the objective is to minimize the sum of the squared distance between each source point and the tangent plane at its corresponding destination point (Figure 4.2). Unlike the point-to-point metric, it does not have a closed-form solution; therefore, the minimization is done under the assumption of small rotation angles \( \sin \theta \sim \theta \) and \( \cos \theta \sim 1 \).
The point-to-plane ICP improves performance by taking advantage of surface normal information (minimizes error along the surface normal). The insight of the point-to-plane algorithm is that the point cloud has more structure than an arbitrary set of points in 3D space. Equation (4.14) illustrates the point to plane cost function that should be minimized to extract the transformation \((R, T)\) parameters consisting of a rotation matrix \(R\) and a translation vector \(T\).

\[
E(R, T) = \sum_{i=1}^{N} \| (R \mathbf{p}_i + \mathbf{T} - \mathbf{q}_i) \cdot \mathbf{n}_i \|^2
\]

\[
E(R, T) = \sum_{i=1}^{N} \| \begin{bmatrix} 1 & -\theta & 0 \\ \theta & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{p}_{i,x} \\ \mathbf{p}_{i,y} \\ \mathbf{p}_{i,z} \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} - \begin{bmatrix} \mathbf{q}_{i,x} \\ \mathbf{q}_{i,y} \\ \mathbf{q}_{i,z} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{n}_{i,x} \\ \mathbf{n}_{i,y} \\ \mathbf{n}_{i,z} \end{bmatrix} \|^2
\]
In equation (4.14), \( n_i \) is a normal vector at the surface \( q_i \). Equation (4.14) can be solved using partial derivative and with the knowledge of normal vector extracted from planar patch segmentation algorithm, which will be described in the next section.

4.6 ICP Plane-to-Plane Minimization

Another registration technique is based on plane-to-plane registration, which take into account surface information from both source and target point clouds. The transformation parameters are obtained by fitting planes to both source and destination clouds and minimizing the distance between both planes. In this thesis, the Generalized-ICP (GICP) algorithm developed by (Segal, 2009), for plane-to-plane registration is used. The GICP combines the Iterative Closest Point (ICP) and point-to-plane ICP algorithms into a single probabilistic framework.

The minimization of cost function for GICP is calculated according to equation (4.15)

\[
E(R, t) = \sum_{i=1}^{N} [d_i^{(T)^T} (C_i^p + T C_i^q T^T)^{-1} d_i^{(T)}]
\]

(4.15)

In equation (4.15), \( T \) is the transformation parameter and \( d_i^{(T)} \) is difference between the source point cloud and the transform target point cloud according to the following equation

\[
d_i^{(T)} = p_i - T q_i
\]

(4.16)

The assumption for the probabilistic term is to consider a set of points \( P = \{p_i\}, Q = \{q_i\} \) with independent Gaussians distribution according to \( p_i \sim N(p_i, C_i^p) \) and \( q_i \sim N(q_i, C_i^q) \). \( \{C_i^p\} \) and \( \{C_i^q\} \) are covariance matrices associated with the measured points.

Computing the surface covariance matrices requires a surface normal associated with every point in both scans. In order to calculate the covariance matrix, each sampled point is assumed to be distributed with high covariance along its local plane, and very low covariance in the surface
normal direction. For example, if the surface normal associated to the measured point which is along the $x$ direction the covariance is calculated according to equation 4.7.

$$
C_i^p \text{ or } C_i^p = \begin{bmatrix}
\epsilon & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix}
$$

(4.17)

Where $\epsilon$ is a small constant representing covariance along the normal. In cases where the angle difference between the normal vectors are more than a predefine threshold, the algorithm will automatically discount the matches as they may belong to different surfaces.

It should be mentioned here that using only planar surface patches to perform registrations may be a limitation in environments that do not contain such features. However, since indoor environment contain several planar structures, finding planar patches is easy and robust.

4.7 ICP Edge based Minimization

In many environments, important details of the scene can be captured in edge points, enabling us to use only these edge points for registration rather than a full point cloud or a uniformly down sampled point cloud. Edge detection can be viewed as a means of intelligently selecting a small sample of points that will be informative for registration. This approach can be both faster and more accurate than alternative approaches. Figure 4.3 shows the result of edge detection using canny edge detector.
Figure 4.3. Edge extraction from RGB image using canny edge detection

Having depth information from RGB-D camera, it is possible to detect 2D edges from RGB data and back-project these edge points to the 3D point cloud. The alignment process begins with Canny edge algorithm to finds first-order image gradients (directional change in the intensity or color in an image), $G_x$ in $x$ and $G_y$ in $y$ directions, by applying Sobel gradient operator to the input image. In the following step the gradient vector orientation for a given edge pixel can be calculated according to equation 4.18.

$$\theta = \arctan \frac{G_y(u,v)}{G_x(u,v)}$$

(4.18)

each point $P$ is defined in the point cloud as:

$$P = [p_x, p_y, p_z, p_\theta]$$

Where $P_\theta$ is the gradient angle of the corresponding edge pixel.

A new metric $d'$ is define by equation 4.19.

$$d'(p, q) = \begin{cases} d(p, q) & \text{if } |p_\theta - q_\theta| < t_p \\ \infty & \text{if } |p_\theta - q_\theta| \geq t_p \end{cases}$$

(4.19)
\[ d(a, b) = \sqrt{(p_x - q_x)^2 + (p_y - q_y)^2 + (p_z - q_z)^2} \]

Informally, the distance between two edge features is the same as the Euclidean distance if their gradient angles are within a certain threshold, and infinity otherwise. Thus, when looking for the nearest neighbour to an edge feature point \( P \), only edges with similar gradient angles are considered as possible candidates. The magnitude of the angular threshold \( t_\alpha \) is not of great importance; a value of 30 or 45 degree is sufficient to prune out edges, which are pointing in completely different directions.
Chapter Five: Global Optimization

5.1 Introduction

The goal of this chapter is to optimize the steps taken in previous chapters for the alignment of the point clouds. This is particularly important as the previous methods still suffer from the drift over long period (e.g. Large corridor); therefore, it will result in inconsistency in the mapping solution. The main contribution of this chapter is to use segmentation as an optimization technique to extract planar patches and normal vector associated with the patches to update the pose globally. The last step is loop closure which minimizes the error between first and last acquired scans.

5.2 Segmentation

In robotics and computer graphics field different methods have been developed over the last decade for segmenting a scene into geometric shape primitives (Vosselman et al., 2004). Many of these approaches developed for extracting planar segments from point cloud data. This thesis is mainly focus on the planar segmentation, hence the possible planar segmentation methods are discuss in details.

Various methods have been proposed to detect planar surfaces from point cloud data. These methods aim to build homogenous region within the point cloud data based on some geometric criteria. The segmentation methods proposed in various literatures falls mainly into the following four groups:

• Segmentation based on clustering of features
• Segmentation based on surface growing
• Segmentation by model fitting
• Hybrid segmentation technique
5.2.1 Segmentation based on clustering of features

In these methods, features are explained first for each point based on the geometrical and radiometrical characteristics. The features commonly establish the position of each point, local surface normal, and best fitting surface residual. An n-dimensional feature space is established to map the n-features of each point. Hence, clusters are created in the feature space. The performance of this method depends on the selected features and their derivation method as well as the methods used in partitioning the feature space. As features of individual points are generally described using points in local neighbourhood, this technique of segmentation is also sensitive to the noise in data and is affected by the definition of the neighbourhood.

5.2.2 Segmentation based on surface growing

In this technique, the algorithm starts from a point and grows around neighbouring points based on certain similarity criteria (Figure 5.1). Vosselman et al. (2004) describe this technique of segmentation, which basically involves steps of identification and growing of seed surface.

5.2.2.1 Identification of seed surface:

A seed surface consists of a group number of neighbouring points that fits well in plane. In order to select the seed surface, a group of adjacent points are identified and tested to identify if they fit well to the planar surface or not. If a selected plane is found to fit within some predefined threshold, it is identify as seed surface; otherwise another point is tested.

5.2.2.2 Growing of seed surface

Once the seed surface is selected, each point in the seed surface is tested to find the neighbouring points that may fit to the plane. This process is mainly intended to extend the surface towards its neighbourhood. The points are added in the growing surface if they meet the predefined criteria.
After adding a point, the plane equation is updated. This decision of accepting a point to the plane can be based on one or more of the following criteria:

- Proximity of point: the condition for adding points is a certain distance threshold of points from current seed surface.
- Global planarity: a candidate point is added in a segment if the orthogonal distance of points to the plane is within some threshold.
- Surface smoothness: according to this criteria, local surface normal for each point in the point clouds is estimated. A candidate point and the normal of the growing surface should below some threshold value.

Figure 5.1. Planar patch segmentation on RGB-D point clouds using region growing followed by convexhull
5.2.3 Segmentation by model fitting

This approach is based on the idea that many objects can be decomposed into geometric primitives (Al-Durgham and Lari, 2014) like planes, cylinders or sphere (Schnabel et al., 2007). This technique tries to fit primitive shapes such as plane in point cloud data and identify the points that conform to the mathematical representation of the primitive shapes, which are frequently encountered in point clouds.

Hough transform and Random Sample Consensus (RANSAC) are the two most well-known techniques used in this category. Below is a brief description of RANSAC as it will be used in the segmentation algorithm.

5.2.3.1 RANSAC

The Random Sample Consensus (RANSAC) paradigm is used for robust fitting of parametric model to the measurement that may have high degree of noise and outlier. RANSAC creates a large number of hypotheses of primitive shapes by randomly picking minimum subset of sample points that each separately determines the parameter of primitive (Schnabel et al., 2007). The scoring mechanism is applied to detect the best primitive.

The process begins with minimum number of points eg. three points are randomly selected to estimate parameters of candidate plane. In the following step the remaining points are examined and if they are within the threshold, the score will be given to the selected plane. If the points are not within the threshold, the above procedure will be done again, and plane with the highest score are stored. After given number of iteration, either the plane with the most score is identify as the detected plane or as the failure.
Schanabel et al. (2007), notes that the complexity of RANSAC is mainly because of two major factors: the minimal distance candidates and the cost of quantifying the score for every candidate plane.

5.2.4 Hybrid segmentation technique

In this technique, various methods are combined to detect planar segments. In general, region growing method is combined with other plane detection methods as it takes into account the spatial proximity of the points in more natural way. Roggero (2002) combined hierarchical region growing and Principal Component Analysis (PCA) to segment point cloud. PCA is used to define the aggregation criteria and to describe the geometrical properties of the surfaces.

5.3 Transformation from segmentation

One of the additional methods that proposed in this thesis is to obtain the orientation from geometrical properties of the patches extracted from the point clouds data. This is highly important as it can update the pose of the systems in cases where not enough features exist in the environment and no loop closure is detected. The later issue occurs more often in un-texture areas such as corridors. In these cases, the pose update is totally lost from and the error accumulates over time. Hence, another source of information is required to update the pose of the system in these situations. Generally, two non-coplanar lines which can be identified in two different scans allow us to estimate the relative scale, three shifts, and three rotation angles between the two scans. Here, the normal vector is used as a coplanar line associated with planar patches in each epoch and the orientation can be obtained by calculating the angle between the normal vectors.
Figure 5.2. Finding the angle between the normal vectors in two consecutive frames and compensating for accumulated error

Figure 5.2 illustrates the normal vectors associated with planar patches extracted from segmentation algorithm. The angular deviation between two 3D lines (normal vectors) can be derived through the dot product of their direction vectors according to equation (5.1).

$$\theta = \sin^{-1} \left( \frac{\|\vec{n}_1 \times \vec{n}_2\|}{\|\vec{n}_1 \cdot \vec{n}_2\|} \right)$$  (5.1)

5.4 Loop closure

Another solution for global optimization is the loop closure detection, which is a typical problem in simultaneous localization and mapping (SLAM) that requires a mobile robot to recognize previous places accurately and correctly when they are revisited. Correct loop closure detection can help the robot to reduce the accumulated errors during the mapping process, while incorrect detection of loop closure can introduce redundancy or incorrect update to the map.

Various techniques exist for loop closure based on type of sensors used in the mapping procedure. Appearance based methods using visual sensors (Newman et al. 2006, Dellaert, 2006) are common technique to detect the loop closure. The idea behind appearance based technique is to find
similarity between the current view and the previously visited view to detect the loop closure. However, selecting places on the basis of similarity is too naive since different places may look very similar. Here, two loop closure techniques namely bag-of-words and Fast Appearance Based Mapping (FAB-MAP), that addressed the previous problem in a probabilistic framework, are explained.

5.4.1 Bag-of-Words (BoW)

The term “bag of words” refers to document classification techniques where documents are considered as random sets of words.

The words used in image processing are local image features. The local features can be histograms of gradient orientations or image patches, or even color histograms. As these features are sensitive to noise, they are not used directly as words. Hence, another step namely vector quantization (clustering) technique is required to categorized the features. The final solution of this discretization is the dictionary. According to the words, a classifier is trained to identify the categories.

Figure 5.3 shows the process of loop closure using appearance based (BoW) approach. The following steps are generally performed to detect loop closure using BoW:

- Extracting the features of the images using the feature extraction algorithms.
- Quantizing the descriptors into clusters using clustering methods.
- Constructing the visual dictionary with the visual words.
- Applying a weight for each image, a higher weight means the visited image sharing more visual words with the current image, hence it represent the same place.
Because the mobile robot obtains images continuously during its motion, the adjacent images always look more similar to each other especially if the environment contain similar pattern. Hence, it is important to compute the similarity between a new acquired image and the last image. Higher similarity value means that the two images are similar, so the current image will not be used for loop closure detection. However, if similarity is under the fixed threshold then the current image represents a new location. The similarity threshold should be set according to the resolution of the image and the image acquired rate. If the threshold is too small the number of deleted images is higher and the robot might not able to find the closed loop when it comes back to the starting point of the cycle.

Figure 5.3 illustrates the BoW technique for detecting the loop closure.
Several authors have proposed extensions to this basic approach. Filliat (2007) described a system where the visual vocabulary is learned online. Schindler et al. (2007) have described how to encrypt visual vocabulary generation so as to get more discriminative visual words, and discuss the application of the technique to large scale environment localization with a database of 10 million images.

5.4.2 FAB-MAP

Fast Appearance Based Mapping (FAB-MAP), is a technique for place recognition and mapping developed for mobile robotics applications (Cummins, 2008). It tries to solve some of the key aspects of the navigation problem, which is a core task for autonomous robots. The main problem that FAB-MAP tries to solve is “perceptual aliasing” (the fact that different parts of the workspace may appear the same to the robot’s sensors) in a probabilistic framework. This is a very difficult problem for a number of reasons. Firstly, the world is dynamic; two images of the same location may look different due to changes in lighting, or entering and leaving of people from the scene, etc. Secondly, and more challengingly, the world is visually repetitive. Common textures such as un-texture walls are present in different places.

For each new observation that the sensor collects, FAB-MAP calculates the probability that the observation came from one of the previous known places or a new place, and then updates the belief about the appearance of each place.

5.4.3 Explicit Loop Closing Heuristic (ELCH)

When mapping large-scale outdoor or indoor environments, the scenes may consist of hundreds of 3D laser scans. In these cases the global optimization (graph optimization) technique is a good solution to minimize the error in alignment process. Generally, the graph consists of set of nodes
representing the robot poses (scanner position) and set of edges to identify pairs of nodes that were already matched using alignment technique.

When detecting a closed loop, registration is applied to transform the last acquired scan. This transformation dissociates the last node from the current graph and yields a transformation vector that consists of a rotation and translation to transform the last node to a position with minimal error with respect to the first node of the loop. In the following step, the transformation vector has to be distributed over the graph, (i.e., over the previously encountered poses).

The input to this algorithm consists of unprocessed scan data, and correspondences between poses. Loop closing is performed by adding additional edges, if the robot encounters a position close to another position where it had been before (Sprickerhof, 2009).

5.4.4 Loop Closing based on manual registration

Given the first and last scan of a loop the algorithm detects a closing loop by aligning the first and last acquired 3D scan (in case of RGB-D aligning occurs between first and last image). The transformation computed from this registration is divided by the number of 3D scans in the loop and distributed with additional weighting factor over all scans. This will cause the additional constrain for compensating error over multiple scans.

Figure 5.4 shows the loop closure detection process. It can be seen from the left side of the figure that the pose error increase over time (blue circle) and with the loop closure this error should be minimized. The red circle shows the first and last frame which are used to detect the loop and update the pose of the system.
5.5 Obstacle avoidance

To autonomously navigate in an indoor environment, a mobile robot requires the capability to decide whether an obstacle should be neglected or considered. Previously, the most common sensors for obstacle avoidance were ultrasonic rangefinders. These sensors have many negative side effects such as very wide beam angle, frequent false and multiple reflections of the sound wave from objects (Apostolopoulos, 2000). However, they have the advantage of identifying the presence of glass objects. Hence, they can be used as an additional sensor for obstacle detection. Most of the existing approaches for obstacle avoidance use stereovision (Davison, 2007, Fregene, 2002), which is sensitive to environmental condition (e.g., ambient illumination). As an alternative or supplement, recently 3-D Laser Rangefinders (LRFs) have been used (Kweon, 1990). However, 3-D LRFs are very costly, bulky, and heavy to be used only for obstacle avoidance purpose. A more reasonable solution for lower-cost robots is a 2-D LRF. Henrick and Krotkov (1997) employed 2-D LRF to aid stereovision for obstacle and hazard detection. The scanner was pointed to the ground at 45°. Due to the small range of laser scanner, the scanner was used for safety purposes only, i.e., when an obstacle is detected, an emergency command is used to stop the robot.
Apostolopoulos (2000) also used a 2D LRF to complement stereovision in a terrain mapping application. In this work, a Sick LMS 220 was placed on “Nomad” robot so as to “look” diagonally downward and forward. As the robot moves forward, the laser beam capture the ground ahead of the robot and produced range data of the terrain. Based on this data, the Nomad generate a so called “goodness map,” which was computed based on the current scan data and the previous ground level. The “goodness map” was then combined with the map built by stereovision for further processing and path planning.

In many cases, it is enough that the environment is only examined by the laser scanner. This will save the amount of computational operations, because the basic pre-processing of such data requires only simple filters, which may exclude erroneous measurements. Many current navigational methods assume motion of the robot in one plane and likewise for their implementation only planar input data from the sensors are sufficient.

The point cloud data obtained from the sensor systems can be modeled using a 2D, 2.5D, or 3D map. A 2D grid map, also known as an occupancy grid, uses the binary values of 1 or 0 as grid cells that occupied by obstacles or free. These systems are of highly efficient for path planning and navigation since a robot is very interested in the location of obstacles. The 3D grid map is made up of voxels that take up considerably large amount of memory and complexity but are very useful for path planning in air and under water applications (Kweon, 1990). Because of the complexity and time required to analyze 3D data, a very common method of extraction is to model the environment in a 2D grid with additional information and is referred to as 2.5D, which will hold much more meaningful information of the cell, more than just a binary value of a 2D occupancy map.
In this work, to make the system autonomous, a 2D laser range finder (SICK LMS100) is placed in front of Seekur Jr Robot. This laser range finder produces 2D range scans by rotating its beams around its horizontal axis at 50Hz and has a range of 20 meters. In the horizontal direction, the array provides an angular resolution of 0.5 degree with 270 degree field of view (FOV). In order to detect the obstacle the algorithm keep checking 3 areas in front of the robot including left, front and right. Once the obstacle is detected the algorithm force the robot to rotate against the area where the obstacle was detected in order to detect obstacles and make the system autonomous a threshold of 1 meter is considered for the laser range finder. Figure 5.5 shows the simulation interface for obstacle avoidance. The interface is designed in a way that it can get the map with predefine obstacle and the start and end point for the robot. The robot then detects the obstacle and navigates to the destination.

A real-time software based on Aria library (MobileRobot, 2014) is developed to lock the robot every time the obstacle is detected and force the robot to rotate and change its direction based on the area that obstacle is detected.
Figure 5.5. Simulation for obstacle detection using laser range scanning (blue circle is the position that is define for the robot to move there, red lines are laser measurements).
Chapter Six: **Experimental Results**

### 6.1 Introduction

This chapter will present the experimental results of the proposed indoor mobile mapping algorithm. Two scenarios for indoor mobile mapping are examined. The first scenario shows the ability of single RGB-D sensor to map a small room while the second scenario shows the results for large corridor. It has been shown that RGB-D aided Velodyne HDL-32 lead to better result than single RGB-D sensor. Moreover, two-stage registration approach, namely coarse registration and fine registration, has been described. Finally, adding loop closure result in consistency in the final mapping solution.

### 6.2 Hardware Description

The main sensors used in this thesis are RGB-D camera, Velodyne HDL-32 and SICK LMS-111 2D laser range finder.

#### 6.2.1 RGB-D camera

The RGB-D camera consists of three main components: a projector that projects a pattern, IR camera that detects the returning pattern and RGB camera that provides colour information. The field of view of RGB-D camera is 58 degree horizontally and 40 degree vertically, and it produces 640 × 480 pixels depth images at 30 frames per second. The range of operation is between 0.5 m and ~5.0 m (Khoshelham, 2012).

At close range (0.5–2 m), the accuracy of depth range is between 1 and 6 mm with a spatial XY-resolution of 3 mm at 2 m. Table 6-1 summarizes the specifications of this RGB-D camera as provided by the manufacturer.
Table 6-1. The specifications of the RGB-D camera (Kinect)

<table>
<thead>
<tr>
<th>Angular Field-of-View</th>
<th>horizontal, vertical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frame rate</td>
<td>30 Hz</td>
</tr>
<tr>
<td>Resolution RGB and IR depth camera</td>
<td>640 x 480 pixel</td>
</tr>
<tr>
<td>Depth range</td>
<td>0.8 m - 5 m</td>
</tr>
</tbody>
</table>

6.2.2 Velodyne HDL-32

Velodyne HDL-32 includes 32 lasers aligned over a 41.34° vertical field of view (from +10.67° to -30.67°). Table 6-2 summarizes the specifications of this laser scanning system as provided by the manufacturer.

Table 6-2. The specifications of the Velodyne HDL-32E laser scanner

<table>
<thead>
<tr>
<th>Time of flight distance measurement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Measurement range 70 m (1 m to 70 m)</td>
</tr>
<tr>
<td>32 laser/detector pairs</td>
</tr>
<tr>
<td>+10.67 to -30.67 degrees field of view (vertical)</td>
</tr>
<tr>
<td>360 degree field of view (horizontal)</td>
</tr>
<tr>
<td>10 Hz frame rate</td>
</tr>
<tr>
<td>Accuracy: &lt;2 cm (one sigma at 25 m)</td>
</tr>
<tr>
<td>Angular resolution (vertical) ~1.33°</td>
</tr>
<tr>
<td>Angular resolution (horizontal) ~0.16° at 600 rpm</td>
</tr>
</tbody>
</table>

**Output:**
- Approximately 700,000 points/second
- 100 Mbps Ethernet connection
- UDP packets
- distance
- rotation angle
6.2.3 SICK LMS-111 Rangefinder

SICK LMS-111 is a 2D laser range finder that has 270 degree field of view in the horizontal plane. Table 6-3 summarizes the specifications of this laser scanning system as provided by the manufacturer. The laser range finder is used in this thesis for obstacle avoidance.

Table 6-3 The specifications of the SICK LMS-111 laser range finder

<table>
<thead>
<tr>
<th>Field of view</th>
<th>270° max (may be restricted based on mounting location on robot)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scanning frequency</td>
<td>25 or 50 Hz</td>
</tr>
<tr>
<td>Operating range</td>
<td>0.5-20m (depending on resolution settings)</td>
</tr>
<tr>
<td>Resolution</td>
<td>1mm (1cm, or 10cm also available)</td>
</tr>
<tr>
<td>Angular resolution</td>
<td>0.5° or 1°</td>
</tr>
<tr>
<td>Systematic error</td>
<td>±30mm</td>
</tr>
<tr>
<td>Statistical error</td>
<td>±12mm</td>
</tr>
<tr>
<td>Interface</td>
<td>Ethernet</td>
</tr>
<tr>
<td>Data transmission rate</td>
<td>100Mbit Ethernet</td>
</tr>
</tbody>
</table>

6.3 Test I: (RGB-D mapping result for a small room)

6.3.1 Test Description

The conference room (Figure 6.1) of the Geomatic Department, at the University of Calgary was chosen for the first experiment. The room is about 8x8 meters with large meeting table and book cabinets. Single RGB-D camera is used for our first experiment, since the room contains enough texture for feature extraction. In order to map a room, the test was done by rotating the RGB-D sensor slowly 360 degrees around the room on a nearly static position. The number of frame captured by RGB-D sensors was 590 frames (RGB 590 frames and depth frame 590 frames). The capturing time was 20 seconds.
6.3.2 Mapping Procedures

The procedure of the mapping algorithm using single RGB-D sensor is illustrated in Figure 6.2.

The mapping procedure using RGB-D camera is divided to the following steps

1. Pre-processing of the RGB image (Gaussian pyramid)

2. Visual odometry
   a. Feature extraction
   b. Feature matching
   c. Inlier detection
   d. Motion estimation (First stage of registration)
3. Registration of the point cloud (ICP)(Second stage of registration)

4. Loop closure

5. Mapping

Figure 6.2. Flowchart of 3D mapping using single RGB-D sensor

The procedure begins with pre-processing on RGB image acquired from RGB-D camera. The main idea behind the image processing is to smooth the image and remove the noise or other undesired fluctuation in image. To achieve that the RGB image first converted to gray scale and then down-sample using Gaussian pyramid. Down-sampling (higher pyramid levels) act as a low pass filter
on the original image. Figure 6.3 shows the Gaussian pyramid for just one level. Features in higher levels represent salient points in lower frequencies whereas features in lower pyramidal levels represent salient points in high frequencies.

![Gaussian pyramid](image)

Figure 6.3 Down-sampling the image using Gaussian pyramid, Left(level 0 no down-sampling), right(level 1, one level down-sampling), red circle is a radius search for feature detection.

It is also possible to add more levels to the pyramid to get more stable features. However, this will cause more computation and low update-rate. The other advantage of the Gaussian pyramid is to remove the features that are similar and close to each other by reducing the search area (radius) in the image. Also features that are in higher level of the pyramid are more stable to image blur due to the fast movement of the camera.

In the next step features are extracted at each level of the Gaussian pyramid using Lucas-Kanade-Tomasi (KLT) method which is an algorithm for corner detection. It should be noted for every feature in the RGB image the corresponding pixel in the depth image is check and if no depth information is available the feature will be discarded. The features that have the highest score meaning that they appear in all the pyramid level are picked as stable features.
In the next step, features are matched across frames by comparing their feature descriptor values. In the following step RANSAC is used to remove the incorrect feature matches between frames (outliers).

Since the first stage registration is based on 3D-to-3D matching technique (chapter 3), the 2D image inliers points are projected in to 3D space using depth information. Figure 6.4 shows the Lucas-Kanade-Tomasi (KLT) feature detection (left figure) and the projection of these features in 3D space using RGB-D depth information (right figure).

![Image](image_url)

Figure 6.4. Left: the locations of features in the image. Right: the corresponding location of the 2D features in 3D space using depth information

In the next step, the transformation parameters are obtained using 3D-to-3D matching (Coarse registration) which minimizes the Euclidean distances between the inlier feature matches. The transformation obtained from coarse registration step is used as an initial guess for fine registration using ICP techniques. The pose (rotation and translation) estimated from previous steps (two registration steps) still suffer from drift. Hence, in the next step, the drift is removed by employing loop closure (global optimization).
The algorithm detects a closing loop by aligning the first and last acquired image frames. Since the test was done by rotating the RGB-D sensor 360 degree around the room, the last frame is the place where the mapping process is started. The transformation computed from alignment between first and last frame is divided by the number of frames captured in the loop and distributed with additional weighting factor over all frames. This will introduce an additional constrain for compensating error over multiple scans.

Figure 6.5 shows the software architecture for mapping using RGB-D camera. The main software developed for the modelling consists of three main threads one handling the acquisition of the RGB-D data and the second handling the process of visual odometry and ICP and the last thread is for loop closure. The threading allows the system to work in parallel without losing any data.

![Figure 6.5 Software architecture for mapping using RGB-D camera](image-url)
6.3.3 Test I Results

Figure 6.6 shows the result of the modelling before the loop closure. It can be seen from this figure that the walls were deviated from the true pose because of the error in visual odometry caused by the error in features extraction and matching process. In order to remove the error in the estimated pose, the loop closure was added where the relative pose between first and last frame was used as feedback to the algorithm and to update the global pose. Figure 6.7 shows the result after loop closure. The compensation error after loop closure is ~12 degree in yaw angle which is measured by calculating the orientation between the last frame and the first frame.

Figure 6.6. 3D modelling before loop closure
To know roughly the accuracy of the final mapping solution the point to point distance of door frame width was measured in CloudCompare (Figure 6.8) and the result was compared to true value of door frame with a tap. The result illustrates that the error is 2 cm.
To analyse the performance of RGB-D visual odometry and ICP the comparison is done on the result of RMS error between two consecutive frames from visual odometry and visual odometry and ICP together. The expectation is to get more accurate result by combining coarse registration and fine registration steps. According to Figure 6.9 the RMS error and number of iteration for convergence of visual odometry and ICP is less than visual odometry alone. This fact is more clear in the final solution model Figure 6.7 as the deviation of walls are removed and the point cloud are more aligned.

Figure 6.9. RMS error of visual odometry and ICP and visual odometry alone

As described before, the test was done by rotating the RGB-D sensor 360 degree; hence the expected translation should be close to zero. Figure 6.10 illustrates the result of the translation in $x$, $y$ and $z$ direction. There are number of unexpected jump in the translation, which is caused by
unsmooth rotation of the sensor during the test. This will directly affect the mapping solution as it generates noise in the point cloud registration. To compensate for this error a threshold is being set to remove the frames that contain more than 20 cm error in their translation.

Figure 6.11 illustrates the result of rotation from visual odometry. As it is expected the rotation occurs around the z axis (heading) for about 360 degree. The roll and pitch angles are almost zero.

![Graph showing VO translation estimate (tx,ty,tz)](image)

More than 20 cm jump in translation (ignore the measurements)

Figure 6.10. VO translation estimate (tx,ty,tz)
The current algorithm processing time is less than a minute for processing 558 frames and augmenting point clouds of size 21,476,262.

6.4 Test II: (RGB-D aided Velodyne HDL-32 mapping result for large corridor)

6.4.1 Test Description

In the second experiment a large corridor located in the University of Calgary CCIT building 3rd floor was chosen as an example of mapping for large indoor environment, Figure 6.12 shows some pictures of the test environment.
The performance of the mapping algorithm described in this section was evaluated by performing an experiment in a corridor of size 33x11 meters and by capturing 667 RGB-D frames and 14 Velodyne HDL-32 LiDAR scans, each with 81,000 data points. The LiDAR scans were captured in a “stop and scan” mode. Hence the robot was forced to stop at 14 positions, as shown in Figure 6.13, to capture the scans.

Figure 6.12. University of Calgary CCIT building third floor
The goal of this thesis was to first show the capability of RGB-D for mapping. Hence the first experiment was purely based on RGB-D sensor to map a room with enough texture. However, because different indoor environment such as corridor contain less features and texture, it is not possible to rely on the RGB-D sensor alone in such environment. Basically, if the pose estimation is not accurate especially over long run then the map generated by the sensor measurement is diverging from true map. Figure 6.14 shows the pose drift of the robot over long run. It can be seen that because of the drift in robot pose estimation, the map generated over long trajectory contain error (stars in Figure 6.14 are feature or map measurement from environment, the circle shows the uncertainty of measurements).
Figure 6.14. Drift in robot pose estimation (blue) due to the drift in visual odometry (magenta)

Figure 6.15 clarify the reason of not using single RGB-D sensor for large environment. This test was done using single RGB-D sensor with the algorithm described in the experiment one. It can be seen that the output of the algorithm totally diverge after short period of time.

Figure 6.15. Drift of single RGB-D sensor for mapping of large corridor
One of the reasons of not having good result with single RGB-D sensor over long trajectory is because of not having enough depth information for corresponding features in the image. As it can be seen in Figure 6.16, while the RGB image is complete, parts of the depth images might be empty. This occurs any time the RGB-D camera is not able to determine the distance to the given point (range of RGB-D camera < 4 m).

![Figure 6.16. (Left) RGB image and (right) depth information captured by an RGB-D camera](image)

Hence, in this experiment RGB-D camera is used as an aiding sensor to Velodyne HDL-32 Lidar for the mapping of the corridor. It should be mentioned that to integrate the RGB-D frames with LiDAR point cloud for fine registration; all corresponding RGB-D frames should be recorded and stored at the time of each LiDAR scans. This is done by triggering the RGB-D sensor capturing software with an input given by an operator.

### 6.4.2 Mapping procedures

The procedures for mapping of RGB-D aided Velodyne camera is illustrated in Figure 6.17.
The mapping procedures begin with coarse registration using visual odometry algorithm. For coarse registration two different feature extraction techniques, one based on SURF and NARF are considered. Figure 6.18 shows the feature extraction using SURF from RGB image and NARF features from point cloud. The first reason for selecting these two feature extraction techniques was to see whether enough feature points can be extracted in case of low texture in environment. Another reason was that, since 3D-to-3D matching technique was used for pose estimation, the 2D features extracted (SURF) from image should be projected into 3D space and this will cause additional noise. Hence, NARF is used to directly extract features from 3D point clouds. NARF is
applies to range images and thus cannot be computed directly on laser scans. Hence in this thesis it is used to extract key-points from RGB-D point clouds.

Figure 6.18. RGB-D 2D and 3D matching Left: 2D-2D matching using SURF from two consecutive frames. Right: NARF key-point

In the next step, Sample Consensus Initial Alignment (SAC-IA), (Rusu et al., 2009), which is an adaptation of Random Sample Consensus (RANSAC) (Fischler and Bolles, 1981) is used to remove the outliers. This procedure begins with selecting random points in the source cloud and matched them with the most potentially similar corresponding points in the target cloud. Rotation and translation can be obtained using minimal three correspondences. This process will repeat until transformation that yields the smallest alignment error on the clouds is obtained.

Basically, the SAC-IA algorithm performs another coarse registration, before applying ICP algorithm which can improves the probability of ICP’s convergence. It should be mentioned SAC-IA algorithm is used after NARF feature extraction on the key-points extracted from the point clouds.
The pose estimated from previous steps was used for initialization of iterative closest points (ICP) algorithm (Fine registration), which is a well-established technique for aligning multiple scans. Algorithm 1 describes the pseudo code of ICP (Table 6-4).

<table>
<thead>
<tr>
<th>Table 6-4. Iterative Closest Point (ICP) algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Algorithm 1: Iterative Closest Point (ICP)</strong></td>
</tr>
<tr>
<td><strong>Input</strong>: Two set of point clouds: ⎧⎨⎩ ⎫⎬⎭( {a_i}, {b_i} ) and Initial transformation ((R_0, T_0)) from visual odometry</td>
</tr>
<tr>
<td><strong>Output</strong>: The refine transformation ((R, T)), that align two point clouds</td>
</tr>
<tr>
<td>1: while not converged do</td>
</tr>
<tr>
<td>2: For ( i \leftarrow 1 ) to ( N ) do</td>
</tr>
<tr>
<td>3: If ((R_{w,i} \leq r_{max}) &amp; &amp;(t_{0,i} \leq d_{max})) then</td>
</tr>
<tr>
<td>4: ( w_i \leftarrow 1 )</td>
</tr>
<tr>
<td>5: else</td>
</tr>
<tr>
<td>6: ( w_i \leftarrow 0 )</td>
</tr>
<tr>
<td>7: ( (R, T) = (R_{pre}, T_{pre}) )</td>
</tr>
<tr>
<td>8: ( E(R, T) = \sum_{i=1}^{N_a} \sum_{j=1}^{N_b} w_{i,j} | Ra_i + T - b_j |^2 )</td>
</tr>
<tr>
<td>9: end</td>
</tr>
<tr>
<td>10: end</td>
</tr>
</tbody>
</table>

It should be mentioned, \( d_{max} \) (translational threshold) and \( R_{max} \) (rotational threshold), play an important role in the performance of ICP. Setting a high value for these parameters increases the chance of convergence (far distance point are also considered), however; it results in more incorrect correspondences and low accuracy. On the other hand, choosing a low value for these parameters increases the accuracy of estimated transformation as it includes more close points. In this work, the value for \( d_{max} = 1 \) meter and \( R_{max} = 20 \) degree are considered for rotation and translation between RGB-D frames (robot moves slowly between capturing frames) was smooth. For the fine registration three different approaches namely point-to-point, point-to-plane
and plane-to-plane were used. This is mainly done in order to compare the performance of each method in terms of accuracy and convergence.

The next step is to check for loop closure and segmentation. Basically the loop closure is done by manually registering the last and first frame of RGB-D camera similar to the first experiment since the RGB-D sensor is the main sensor for pose estimation and the drift caused by this sensor affect the mapping solution, thus the drift should be minimized. Moreover, since the loop closure alone cannot fix the drift, segmentation is used as an additional step to suppress the drift and fixed the problem of re-localization in case where the tracking is lost (low texture).

The primary interest for segmentation is the extraction of planar features including walls, and floors and the normal vector associated with each patches. Figure 6.19 shows the procedure for segmentation of RGB-D point cloud.

![Segmentation procedure](image)

Figure 6.19 Segmentation procedure

Once the point cloud from RGB-D sensor is acquired, the Principal Component Analysis (PCA) is performed to classify the point clouds based on the geometric properties of their local neighbourhood. In the following step the classified planar points are used as a seed points to starts region growing process for extraction and merging of the region of interest (planar patches). Using
the eigen-value analysis it is possible to decompose the covariance matrix in order to determine
the geometric nature of the neighbouring points. Basically, the point that has minimum curvature
value is placed on flat area and can be selected as a seed point. Once the seed point is obtained we
can begin the process of region-growing. The region starts its growth by testing every neighbour
of the seed points.

This is based on certain similarity criteria: Generally, two steps are required to add new points to
an existing segment and expand the region: First, distance to the closest point in the segment is
less than certain threshold; second, the local normal calculated at these points are at an angle less
than certain threshold in compare to the current seed point.

6.4.3 Test Results

For the processing of the RGB-D camera and LiDAR scanner point clouds, down-sampling is
performed to the original point clouds via voxel grid filtering. Figure 6.20 shows the point cloud
obtained from velodyne HDL-32 (left side) and down sample version of the point cloud using
voxel grid (right side). The leaf sizes (cube size) of 0.01m, was considered for the voxel grid
because of the density of the point cloud. All the points present in each voxel (i.e., 3D box), will
be approximated (i.e., down-sampled) with their centroid (mean value).
Figure 6.20. Velodyne HDL-32E scan, in the right, the velodyne scan after being filtered by a voxel grid with a leaf size of 0.01

A voxel grid filter tends to make the spatial distribution of a point cloud uniform meaning that point cloud data with non-uniform density can be homogenized through the 3D voxel grid filtering. Additionally, using voxel grid alleviates computation cost for aligning the point cloud. More specifically, voxeling the point cloud is also helpful in segmentation process as the workspace is discretized into a 3D grid, and then the points that fall into each voxel are examined to determine how planar or cylindrical that voxel is. Matches between planar features are then constrained by the offset between the centroids in the direction of their surface normals and the angle between their surface normals.

In the next step the alignment is done using coarse and fine registration method. Figure 6.21 illustrates the result of LiDAR fine registration using ICP after initialization with RGB-D coarse registration (left side Figure 6.21). It can be seen that the coarse registration is not enough to register the two consecutive laser scans (right side Figure 6.21).
It should be mentioned that the comparison is done on different ICP variant to find the optimal registration method. Compared to ICP where all points are used in the registration process, in point to plane approach only points whose normal vector intersect the second view are consider in the matching and registration. Hence, it is more likely to include correct corresponding points in the registration process. However, we cannot guarantee hundred percent that point-to-plane method has always provided better result to point-to-point method. This is specially the case in conditions where the region of overlap contain contours, or cylinder, however, in these cases plane-to-plane still perform better than other two methods. This is illustrated in Figure 6.22 by comparing the result of Velodyne RMS error (root square of the distance between consecutive scans) for standard ICP, point-to-plane ICP and plane-to-plane ICP (GICP).
Figure 6.22. RMS error of standard ICP, point-to-plane ICP and plane-to-plane ICP with $d_{max}=1$

It can be seen from the histogram that the plane-to-plane method (GICP) outperform the other two methods (point-to-point and point-to-plane) as the corridor is almost consist of planar environment.

Table 6-5 represents the RMSE of translation and rotation, between coarse registration and fine registration (plane-to-plane ICP) using SURF and NARF feature extraction technique over the 14 LiDAR scans. The RMSE is calculated by considering the result of translation and rotation of ICP from LiDAR registration as a reference and calculating the RMSE of RGB-D coarse registration for both SURF and NARF with respect LiDAR registration (rotation and translation).

Table 6-5: Performance of RGB-D coarse registration over 14 sequences comparing to ICP point-to-plane of LiDAR scans
This comparison illustrates that although 2D-to-2D matching require extra step (projection) for 3D-to-3D matching, however; it still gives more features points in compare to the NARF key point detector which is directly do the matching in 3D point clouds. According to Table 6-5, in some frames (#scan 6-7, #scan 7-8, #scan 10-11, #scan 12-13, #scan 13-14) we couldn’t find enough features and as a result the system lost its tracking. Therefore NARF is not suitable option for feature extraction in places where not enough texture exists. Also in terms of comparison of RMSE error the result of SURF is much better that NARF method.

The segmentation plays an important role in the final result especially in places where not enough features exists. In these moments the system can be re-localized by extracting rotation between normal vectors belonging to the walls and floor extracted by the segmentation algorithm described in previous section. One issue that should be considered is the number of planar patches extracted from each frames (Figure 6.23) and the normal vectors corresponding to them. Typically, the expected number of planar patches should be three belonging to two parallel walls and one floor in each frame of RGB-D camera except those places where include doors and other features.

<table>
<thead>
<tr>
<th>Sequence LiDAR scans</th>
<th>SURF-based visual odometry</th>
<th>NARF-based visual odometry</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Transl RMSE</td>
<td>Rot.RMSE</td>
</tr>
<tr>
<td>#scan 1-2</td>
<td>0.093 m</td>
<td>3.43 deg</td>
</tr>
<tr>
<td>#scan 2-3</td>
<td>0.081 m</td>
<td>3.21 deg</td>
</tr>
<tr>
<td>#scan 3-4</td>
<td>0.045 m</td>
<td>2.53 deg</td>
</tr>
<tr>
<td>#scan 4-5</td>
<td>0.544 m</td>
<td>4.33 deg</td>
</tr>
<tr>
<td>#scan 5-6</td>
<td>0.984 m</td>
<td>3.65 deg</td>
</tr>
<tr>
<td>#scan 6-7</td>
<td>0.991 m</td>
<td>5.62 deg</td>
</tr>
<tr>
<td>#scan 7-8</td>
<td>1.045 m</td>
<td>5.96 deg</td>
</tr>
<tr>
<td>#scan 8-9</td>
<td>1.325 m</td>
<td>4.17 deg</td>
</tr>
<tr>
<td>#scan 9-10</td>
<td>1.543 m</td>
<td>4.78 deg</td>
</tr>
<tr>
<td>#scan 10-11</td>
<td>1.677 m</td>
<td>4.55 deg</td>
</tr>
<tr>
<td>#scan 11-12</td>
<td>1.512 m</td>
<td>3.67 deg</td>
</tr>
<tr>
<td>#scan 12-13</td>
<td>1.785 m</td>
<td>5.87 deg</td>
</tr>
<tr>
<td>#scan 13-14</td>
<td>1.665 m</td>
<td>5.01 deg</td>
</tr>
</tbody>
</table>
However, because of the noise in the sensor, an average of 10 planar patches are obtained in each frame that includes only 2 walls and floor.

Figure 6.23. Planar segmentation (10 planar patch segments) from two consecutive RGB-D frames

In order to select optimal 3 normal vectors (belonging two parallel wall and one floor) additional constrains is set to check for the orthogonality (normal vector of walls and floor) and parallel condition (zero angle deference between normal vector of two walls).

The principal component analysis (PCA) of the covariance matrix of a three-dimensional point cloud (which is in this specific case in fact equivalent to the eigen decomposition) yields three eigenvectors with corresponding eigenvalues. The latter indicate the variance of the point cloud along the corresponding axes, and can be used to determine the shape of the point cloud.

In order to remove the error in the estimated pose loop closure was added, where the relative pose between first and last frame was used as feedback to the algorithm to update the global pose. The
compensation error after loop closure is ~15 degree in yaw angle which is calculated from the registration (Rotational difference) of first and last LiDAR scan measurement.

To test the mapping accuracy of the system, a number of measurements are done on the original map of the corridor using a Google map ruler. Figure 6.24 shows the original size of the corridor (length and width). The comparison between the mapping solution (Figure 6.25) obtained by the proposed algorithm and the true value obtained from the measured schematic (Figure 6.24); shows the error of 2 meters in length, 0.05 meter in width and 0.96 meter and 0.42 on sides.

Figure 6.24. Schematic of university of Calgary CCIT building third floor with true values for length and width
Figure 6.25. 3D Mapping using LiDAR after plane-to-plane registration and loop closure
Chapter Seven: **Conclusion and future work**

7.1 Conclusion

The main idea of this work was to generate accurate and consistent autonomous mapping solution for both small and large indoor environment.

In order to achieve this goal 3D point clouds of an indoor environment should be created, while simultaneously moving the robot. This requires a high level of accuracy from positioning systems such as the inertial measurement unit (IMU), odometer or vision systems. By knowing the pose (relative orientation and translation) of the robot at any time, the process of registration can be started, where the aim is to align multiple local scans into a common coordinate system. The registration of local scan might not lead to final consistent mapping solution due to the noise in the sensor (RGB-D camera) measurement which will be propagated over time especially in case of large indoor environment. Hence, additional optimization steps including segmentation and loop closure should be taken into account and considered to compensate for this error.

**Investigating the use of single RGB-D sensor for generating consistent mapping solution**

The use of single RGB-D sensor was examined for generating consistent mapping solution for small and large indoor environment. It is shown that single RGB-D sensor is enough to map small room with enough texture, however; for the large corridor the use of single RGB-D sensor result in divergence of visual odometry and losing the features especially in cases where the camera faces non-texture areas for example flat walls. The previous problem motivates us to use RGB-D as an aiding sensor and velodyne HDL-32 as a main sensor for mapping large corridor. The main challenge of this work in both scenarios was to enhance the process of registration and alignment
of multiple local scan to build a final map solution. Several steps including: coarse registration, fine registration, loop closure and segmentation were performed to enhance the final map solution.

**Investigating the use of Stereo vision for localization and mapping**

The use of stereo vision for pose estimation and 3D reconstruction is examined for outdoor dataset. Optical flow is used for feature tracking and poses estimation. The concept of key-frame is introduced to reduce visual odometry drift over long run. According to key-frame concept, the motion is estimated by comparing the newest frame against a reference frame (initial frame). If the camera motion relative to the initial frame is successfully computed with a sufficient number of inlier features, then the reference frame is not changed. Otherwise, the current frame replaces the reference frame after the estimation is finished. If motion estimation against the reference frame fails, then the motion estimation is tried again with the second most recent frame.

The main reason for using the stereo is to show its capability for estimating the pose of the system over long trajectory.

**Two-stage registration approach (Coarse and Fine registration)**

Two-stage registration approach was proposed in this thesis where the result of the first stage namely coarse registration is used for enhancing the second alignment stage which is fine registration.

In the coarse registration process single RGB-D sensor was used to pre-register the point cloud. The main idea for using coarse registration is to provide good initialization for fine registration step. In the fine registration step three different minimization techniques namely point-to-point, point-to-plane and plane-to-plane were tested in order to have optimal alignment for the point cloud belonging to different local scans. Previous methods still contain error over long run; hence,
additional steps including segmentation and loop closure were used to reduce the accumulated errors.

**Re-localization in case where no features are detected in the environment**

One of the faced problems during the test was the sensitivity of the visual odometry algorithm to the lack of features in the surrounding environment, which caused extract pose from visual odometry algorithm to deviate from true pose. This problem was address using different methods. The first method was switching between ICP and visual odometry and using the ICP alone. This approach allows the system to have consistent solution in short term when no enough features exist in the surrounding environment. However, for long period of time the pose (only rotation) is obtained through segmentation part. In the segmentation part, the normal vectors associated with the planar patches were extracted (two walls and floor) in each frame. The normal vectors associated with consecutive frames are then compared for the angle difference (we expect the normal vector associated with walls and floors are parallel and the normal vector associate with the floor and walls are orthogonal to each other). With the previous knowledge the pose (only rotation) of the system is updated, in cases where no features were detected in environment. Also, to limit the error of visual odometry in the long run, loop closure technique was used, which globally update the pose of the system and provide consistent solution.

**Loop closure to remove the drift and enhance the registration process**

Loop closures part was done manually by estimating the pose between first and last frames in the loop. The transformation computed from alignment between first and last frame was divided by the number of frames captured in the loop and distributed with additional weighting factor over all frames. Adding loop closure result in consistency in the final mapping solution.
7.2 Future Work

Following studies should be approached in future work:

- **Removing errors caused by the existence of moving objects in the mapping process**

  Usually, the presence of moving objects causes errors and reduces the overall quality of the map. This is a considerable problem since many robot applications are in non-static environments. Mapping of a dynamic environment can be obtained by classifying the objects in a dynamic environment. There are two types of moving objects: those that are permanently in motion and objects that are in static mode for period of time (sometimes most of the time) and move temporally. The first group are objects that have changed position each time they are observed by the robot’s sensors. The second group are objects that may or may not have changed their position since they were observed previously. They are however known to have moved object at least once. One robust solution to the dynamic mapping is to use a combination of SLAM and DTMO (Detection and Tracking of Moving Objects) to detect and track moving objects.

- **Fixing Holes in Depth Image**

  One of the problem using RGB-D depth image is in cases where the depth data is missing. To fix this problem and improve the quality of the depth map one solution is to use median filter with an adaptive window size to correct for the holes. However, this approach violates the constraint of keeping the depth edges sharp. Another solution is to use both filtering methods and classification technique to preserve the edges.

- **Using new version of RGB-D camera(Time of flight) for ambient lighting issue(the scene is too bright)**
Structured light RGB-D camera (old version) relies upon a projected intensity pattern to obtain depth. The pattern can become corrupted by ambient light and ambient shadows. The new RGB-D camera uses time of flight to obtain depth. This technique is resistant to ambient light.

- **Optimizing the registration process**

To optimize the process of registration, three-dimensional Normal Distributions Transform (3D-NDT) can be added as a refine step for registration of the point cloud. The 3D-NDT algorithm subdivides the point cloud into a 3D grid of cells (known as voxels) and computes a Probability Density Function (PDF) for each cell. The algorithm then finds the transformation that maximises the likelihood that the points of another point cloud lie on this reference surface.

- **Reducing the search area to detect the loop closure**

Another future work is to reduce the search area to find the loop closure. This can be done by adding statistical filtering and check for the value of the covariance matrix.

- **Path planning algorithm for autonomous robot**

In order to achieve an autonomous mapping and navigation system, mobile robot must be able to do path planning between a start and end point in the environment. This is particularly important especially for large environment where lot of potential direction may exists from start point to the destination. The path planning involves two different problems: first, to provide a global path in terms of intermediate sub-goals and, second, to travel between consecutive sub-goals while avoiding unexpected and moving obstacles along the way. The latter problem can be solved in completely reactive approach and to planned trajectories that take into consideration different constraints such as non-holonomic kinematics dynamics. Also if a graph-based model of the environment is available, this problem is commonly solved using optimal graph search by using a certain cost function.
• Replacing RGB-D sensor or integrating it with other sensors

Instead of using RGB-D visual odometry, it is possible to rely on stereo visual odometry system for estimating the camera pose from successive stereo image pairs. For the pose estimation from stereo visual odometry another methods such as 3D-to-2D point correspondences can be used.

Aiding sensors such as wheel odometer or IMU can be integrated with visual sensor in order to remove the drift of visual odometry and compensate for error in places where no enough features can be extracted from environment.

• Global optimization

For the loop optimization part windowing bundle adjustment (Sliding Window SBA) and pose graph optimization such as g2o (Kummerle et al., 2011) can be added to locally optimize the camera trajectory and reduces both translation and rotation drift, especially over long distance travel.

The overall motion obtained by simple concatenation or ‘chaining’ (motion estimation between consecutive camera frames) generates small error, which will be accumulated quickly over time. To avoid the problems of simple chaining, sliding window SBA approach can be used. Instead of optimizing for two consecutive images only, we can choose a subset of images, to perform SBA upon. The pose and structure parameters estimated in this way are used as initial estimates in the next run, where the window slides further one frame. With that basic sliding window approach, bundle adjustment is performed every consecutive n-window in the sequence of the obtained images, even if the robot has not or only very little moved while the images of the window were taken. Therefore another idea is to only include those images into the window, which pose are more than a certain threshold away from the pose of their
respective predecessor in the window. Other approach is to use full bundle adjustment and optimizes the whole bundle of obtained images at once. It determines camera poses and structure parameters for all recorded frames in one big optimization loop. However the later approach is costly and less accurate in compare of SBA.
References


137


Nuchter, A., Elseberg, J., and Borrmann, D. Irma3D—an intelligent robot for mapping applications. In *Proceedings of the IFAC Symposium on Telematics Applications (TA)* (Vol. 6).


Appendix 1

Analyzing Velodyne User Datagram Protocol (UDP) packets using Boost socket programming

Velodyne provides an output of UDP packets, which requires an extra processing step in order to be converted into a readable point cloud format (Velodyne, 2014) (Velodyne Manual, 2014). This section describes the structure of Velodyne UDP packets as well as the development of socket programming software based on Boost library in order to provide connectionless-oriented communication between the Velodyne and the host computer. In addition, the calibration procedure is performed on the Velodyne raw data in order to obtain point cloud data which is very essential for the proposed mobile mapping platform. Moreover, extra analysis was performed to investigate any loss or delay of UDP packets during the transmission process of sending the Velodyne datagrams to host computer (Tarko, 2014).

UDP provides a connectionless session between two end systems, which makes it simple (no connection required), flexible (messages can be sent to a different recipient with each message-sending attempts), efficient (no extra overhead of packets to the network) and fast. The simplicity of the UDP technology makes it inappropriate for some applications but perfect for more sophisticated applications that are able to provide their own connection-oriented functionality. Alternatively, there are applications whose data has an extremely high time value. One example would be a video conference session, such applications would prefer to use UDP because data delivered late or out of sequence is just discarded.

However, there are some disadvantages associated with UDP protocol. These disadvantages include unreliability (there is no guarantee that the host recipient will ever receive the message), multiple datagrams that are out of sequenced (if multiple datagrams send to host receiver), and
message size limitations (large UDP packets should be broken up into several smaller IP fragments and later re-assembled at the receiving end).

Velodyne HDL-32 data packets contain a Header, 12x Blocks with 32x Angle, Distance and Intensity info, and a Timestamp. The header contains 42 bytes start with and ends with FFFF. There are 12 blocks of data structures and each block contains 2 bytes angle and 32 channels including 2 bytes distance and 1 byte intensity data. Moreover, there are 4 bytes for Timestamp and 2 bytes for factory use. In total the packet structure contain 1200 bytes excluding the header, time stamp and factory use bytes (Figure b.1).
Recent progress and support for extended functionality in the form of libraries provides software developers to have the ability to create efficient and flexible applications. With the emergence of Boost libraries, much of the lower-level plumbing is reduced.

Boost provides a portable library known as “Asio” for network programming (including sockets) with support for TCP/UDP protocols. The following diagram (Figure b.2) shows how the synchronous socket connection operates, which includes the following steps:

- Initiates the connection operation by calling the I/O object.
- The I/O object forwards the request to the io_service.
- The io_service calls on the operating system to perform the connect operation.
- The operating system returns the result of the operation to the io_service.

Figure b.2. Synchronous Operations
• The io_service translates any error resulting from the operation into a boost::system::error_code. An error_code may be compared with specific values, or tested as a boolean (where a false result means that no error occurred). The result is then forwarded back up to the I/O object.

• The I/O object throws an exception of type boost::system::system_error if the operation failed.
Appendix 2

Advanced Robotics Interface for Applications (ARIA)

Advanced Robotics Interface for Applications (ARIA) developed by Mobile Robots Inc, is a popular, reliable and powerful C++ Robotics library commonly used at various research labs around the world. This is the main library that is used in this thesis for developing algorithm on the Seekur Jr. In this section we briefly explain the capability of this library for autonomous navigation.

Connecting to Seekur Jr is the first step to start programming. As the Seekur Jr has an onboard computer one can directly program the on-board computer. However, the on-board computers are small and are fixed to the robot. A remote computer is the best way to work with the robots. There are two ways to approach communication with the on-board computer. The first approach is to use a program such as SSH (Secure Shell) for Linux and Remote Desktop Connection using Windows. Secure Shell (SSH) or Remote Desktop Connection is a network protocol used to communicate between two networked machines. If this approach is followed, all of these software packages need to be installed on the robot’s on-board computer. There is no need to install these packages on the remote machine. The second approach is to use the remote machine and use a client server relationship to communicate the two machines. In this case, all software packages must be installed on both machines. A client program needs to be written for the onboard machine which sends requests to a server that is running on the robot’s onboard computer. The “ArNetworking” library is used to write the client and server programs. Because of the complexity of the second approach we stick using the first approach.
**Robot architecture**

The Seekur Jr robots act as a server in a client-server environment. The low level details of mobile robotics are managed by servers embodied in the operating system (SeekurOS) software of the robot’s micro-controller. The client software that provides the high level control of the robot must run on a computer connected to the micro-controller. This can either be the on-board PC that communicates with it directly through a serial connection, or via a remote networked PC, which requires a server program to be running on the robot PC, providing the communication link between the remote PC and the micro-controller.

The onboard software communicates with SeekurOS via a simple packet-based protocol (described below) via an RS-232 serial connection between the robot and an onboard computer. The protocol is a bidirectional byte stream, in which sequences of bytes called packets represent individual commands (when sent from client software to SeekurOS), or Server Information Packets, which are commonly known as SIPs or simply packets, (when sent from SeekurOS back to the client software). Packets consist of five main elements: a two-byte header, a one-byte count of the number of subsequent packet bytes, a one-byte command or packet type identifier followed by packet data, and finally a two-byte checksum. Packet data is divided into one or more value fields. The meaning and sequence of fields are specific to each packet or command type. Each field has a data type, which determines the size of that field (in bytes). Figure c.1 illustrates the handshaking connection of the client-server between host computer and robot OS.
Figure c.1. Robot architecture (SeekurOS as server controlling the low level parts and onboard computer as client responsible for high level tasks)

*Robot Synchronization Cycle*

According to (Figure c.2) Server Information Packets (SIPs) are sent by the robot server containing information updates about the robot and its accessories. The standard SIP is sent by the robot to a connected client automatically every 100 milliseconds (this frequency may be configured in the firmware parameters). SIPs packets contain the robot's current position and current translational and rotational speeds. The standard SIP is sent on a constant cycle, and reception of this SIP triggers a new iteration of ArRobot's synchronized task processing cycle. This cycle consists of a series of synchronized tasks, including SIP packet handling, invocation of sensor interpretation
tasks, action handling and resolution, state reflection, and invocation of user tasks, in that order. Since the task cycle is (normally) triggered by the reception of each SIP (unless the robot platform begins to fails to send SIPs or the task cycle is explicitly disassociated from the robot connection), each task will be invoked in a predictable order, have the most recent data to act upon. No task will miss an opportunity to use a SIP, and as long as the tasks do not take too much time to execute, each SIP is handled, as soon as possible after the robot sends it.

Figure c.2. Diagram of software architecture for synchronous robot operation