On the issues of visually aided feature extraction from 3D range data

Sok, Chhay

2010

Sok, C. (2010). On the issues of visually aided feature extraction from 3D range data.
Master's thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/20826>

<https://doi.org/10.32657/10356/20826>

On the Issues of Visually Aided Feature Extraction from 3D Range Data

SOK CHHAY

School of Electrical & Electronic Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfillment of the requirement for the degree of
Master of Engineering

2010

Acknowledgements

I would not have made this thesis a reality if I were not fortunate enough to have the company of several generous individuals.

My gratitude goes to Associate Professor Martin David Adams for his patience, encouragements, guidance, general advice and fruitful discussions during all ups and downs of my work. I am also deeply grateful to my co-supervisor Associate Professor Sardha Wijesoma for many constructive suggestions on my work.

I would like to thank Mr Tan Chai Soon and other technical staff of the Autonomous Robotics Research Laboratory, who have been very helpful to me during my whole time working in the laboratory. Without their support, my work could not have been going smoothly.

I would like to extend my appreciation further to Dr. John Mullane, Leong Chee Weng, Rao Akshay and Winayhu Aji Bimas for their mental support and technical assistances. I would like also thank the Head of the Division of Control and Instrumentation and Nanyang Technological University for providing me a research environment and all the financial assistance. And I can't forget my TechX Challenge team mates, who helped fill my life with friendship and joy.

Finally, I would like to thank to all my friends and family members who always believe in me and always be there for me when I have a hard time.

Contents

Acknowledgements	i
Summary	vii
List of Figures	xii
1 Introduction	1
1.1 Motivation	1
1.2 Objectives	3
1.3 Contributions	3
1.4 Thesis Organization	4
2 Literature Review	7
2.1 Data Segmentation	7
2.2 Line Extraction	9
2.2.1 Split-and-Merge Algorithm	9
2.2.2 Incremental Algorithm	11
2.2.3 RANSAC Algorithm	12
2.2.4 Hough Transform	12
2.2.5 EM Algorithm	12
2.3 Feature Extraction for 3D Range Data	15

2.4	EKF SLAM	18
2.4.1	Vehicle and Environmental Feature State	19
2.4.2	Relative Observations	19
2.4.3	EKF Prediction Equations	20
2.4.4	The EKF measurement update equations	21
2.4.5	Adding New Landmarks	21
2.5	Fusion of Range and Visual Data	23
3	2D Laser Based Feature Extraction	27
3.1	Laser Data	27
3.2	Point Cluster-based Feature Extraction	28
3.2.1	Spike Landmarks	28
3.2.2	Point Clustering	29
3.2.3	Tree Trunk and Lamp Post Landmarks	31
3.2.4	A SLAM Implementation with Point Cluster Based Features	32
3.2.5	Cluster-Based Point Feature Issue	36
3.3	Line Features	37
3.3.1	RANSAC Line Based Extraction	37
3.3.2	Hough Transform for Line Extraction	39
3.3.3	Comparison of RANSAC and Hough Transform	43
3.3.4	Fitting Lines	44
3.3.5	Implementation of Line-feature based EKF SLAM	45
4	3D Laser Based Feature Extraction	57
4.1	3D Scanning LADAR	57
4.2	3D LADAR Point Cloud	59

4.3	Principal Component Analysis (PCA)	60
4.3.1	PCA Applied to 2D Range Data	61
4.3.2	PCA Method Applied to 3D Range Data	63
5	Visually Aided Feature Extraction	71
5.1	Introduction	71
5.2	Integration of 3D LADAR and Vision	73
5.2.1	Ladybug Panoramic Camera	74
5.2.2	3D LADAR - Camera Calibration	74
5.3	Image Filtering and Segmentation	78
5.3.1	Image Filtering	78
5.3.2	Segmentation	88
5.4	Implementation of PCA on Segmented Range Data	93
5.4.1	Implementation Issues	99
6	Conclusions and Recommendations	103
6.1	Conclusions	103
6.2	Recommendations	105
	Bibliography	107
	Appendix	111

Summary

Sensing environments and data analysis are basic and important abilities that autonomous navigation systems must possess. These provide the systems capability to detect and classify objects. The advantages of working with single sensors are that the structure of the systems is less complicated, and data analysis is also less time consuming and simple. However, the sensors may not provide enough information for complicated environments, or the accuracy of the data is poor to complete certain tasks. These depend on characteristics of individual sensors and the complexity of the given tasks.

This thesis provides an overview of the algorithms and software designed to extract features from 2D range data. A real time algorithm is developed to extract tree trunks, lamp posts and building pillars from the environment by using 2D laser sensors. Similarly, The Hough transform, which is applicable for range data, is developed to extract lines corresponding to building walls. These extracted landmarks are used as input in feature-based Extended Kalman Filter SLAM to improve the precision of localization of the vehicle and mapping of the environment.

Moving from 2D to 3D, to aid in the 3D feature extraction process, an integrated system is used to provide fused data of range and vision. The image given by the camera is filtered and segmented by The Mean Shift algorithm. The range data is then tagged with each image cluster label, based on which the range data was segmented into smaller groups. Finally, Principal Component Analysis (PCA) is

applied to each group of segmented range data to extract plane and line. The method provides alternative approach for 3D line and plane extraction which is less time consuming and comparably robust.

List of Figures

2.1	The relationship between successive range for a planar surface.	7
2.2	Split-and-Merge for Line Extraction	10
2.3	Structure of the Covariance Matrix.	22
3.1	2D Laser Scan.	28
3.2	Spike Landmarks	28
3.3	Point-Distance-Based Method.	30
3.4	Range-Based Method.	30
3.5	Tree Trunk and Lamp Post Extraction.	32
3.6	Google Map of North Spine, NTU.	34
3.7	Superimposition of Vehicle Trajectory on the Google Map.	34
3.8	SLAM and Novatel Outputs fused on the Google Map.	35
3.9	SLAM Output.	36
3.10	Cluster-Based Point Feature Issue.	36
3.11	Flow Chart of RANSAC Algorithm.	38
3.12	Implementation of RANSAC Algorithm.	39
3.13	Hough Representation of A Line.	40
3.14	Hough Space Representation.	40
3.15	Points given in Cartesian Space.	41
3.16	Intersection of sinusoidal lines in Hough Space.	42

3.17 Hough Transform for Line Extraction.	42
3.18 Best Fit Line.	45
3.19 Predicted Observation Model For Case 1.	46
3.20 Predicted Observation Model For Case 2.	48
3.21 Simulation of Line-Feature Based SLAM	49
3.22 Localization Error	50
3.23 Parameters of Line Feature 1	51
3.24 Parameters of Line Feature 2	51
3.25 The Packbot from iRobot.	51
3.26 Hough Transform: Line Extraction.	52
3.27 Line Extraction and Data Association	53
3.28 Localization based on Odometry and IMU.	54
3.29 Localization based on Line Feature-Based SLAM.	55
4.1 Built-in-House 3D LADAR.	58
4.2 3D Representation of a Point.	59
4.3 3D Point Cloud of Outdoor Scans.	60
4.4 Implementation of PCA on 2D data.	62
4.5 Implementation of PCA on 3D data.	64
4.6 Plane Extraction with PCA. (a) Perfect data set (b) Noisy data set (c) Side view of (a) (d) Side view of (b).	65
4.7 PCA on Collinear Planes. Two collinear planes with (a) no trans- lation in z direction (b) 1 unit translation in z direction (c) 10 unit translation in z direction.	66
4.8 PCA on Two Planes. Two Planes with dihedral angle of (a) 0 degrees (b) 2.6 degrees (c) 11.4 degrees (d) 19.5 degrees.	67

4.9	Implementation of PCA on Real Data.	68
4.10	Point Cloud of South Spine.	68
5.1	The Ladybug Camera consisting of 5 horizontally facing cameras, and one vertically facing camera.	72
5.2	LADAR and Vision Integrated System.	73
5.3	School of Communication.	74
5.4	Calibration of The Integrated System.	75
5.5	Range-Color Fusion Module.	77
5.6	Colored 3D Point Cloud of School of Communication.	77
5.7	Colored 3D Point Cloud of ME Workshop.	78
5.8	37-Pixel Circular Mask.	79
5.9	Mean Shift Algorithm	81
5.10	Original Image of School of Communication.	84
5.11	SUSAN Filtering Result of School of Communication.	84
5.12	Bilateral Filtering Result of School of Communication.	84
5.13	Mean Shift Filtering Result of School of Communication.	84
5.14	Original Image of ME Workshop.	85
5.15	SUSAN Filtering Result of ME Workshop.	85
5.16	Bilateral Filtering Result of ME Workshop.	85
5.17	Mean Shift Filtering Result of ME Workshop.	85
5.18	Bilateral vs Mean Shift Filtering: Regions Smoothing.	86
5.19	Bilateral vs Mean Shift Filtering: Edges Preserving.	87
5.20	Visually Aided Range Segmentation.	89
5.21	Initialization of the Connectivity Matrix.	89
5.22	The Final Connectivity Matrix and Labeled Clusters.	89

5.23	Mean Shift Filtering and Segmentation	90
5.24	Image Segmentation.	90
5.25	Segmentation Labeling.	91
5.26	Segmentation of Range Data (Communication School).	92
5.27	Segmentation of Range Data (ME Workshop).	92
5.28	Mean Shift Filtering and Image Segmentation	92
5.29	Result of PCA on Segmented Range Data of Outdoor Scene.	95
5.30	Result of PCA on Segmented Range Data of Indoor Scene.	96
5.31	Disperse Data Set.	97
5.32	Horizontal Plane	98
5.33	Vertical Plane	98
5.34	Calibration Issue	99
5.35	Misclassification Due to Imperfect Calibration.	100
5.36	Application of PCA on A Section of A Building.	101
5.37	Segmentation Issue.	102
5.38	Entrance Wall	102
1	Physical Connections of The System.	111
2	QNX Momentics 6.3.0, Application Builder.	112
3	QNX System Control GUI.	112
4	Source Code Directory.	113
5	3D LADAR Control GUI.	113
6	The Ladybug Camera System	114
7	The Compressor Connection	115
8	The Connection of External Power Supply	115

Chapter 1

Introduction

1.1 Motivation

Automated control systems have been developed and used in many applications to assist humans. Such systems can be found in the form of small scale toys, medium scale household appliances, and large scale military equipment.

Autonomous navigation is also a form of automated control. There has been a significant interest to deploy such systems in several disciplines at ground level, underwater and in outer space. Autonomous navigation systems are necessary to handle tasks in hazardous environments and areas inaccessible to humans.

Abilities to sense the environment, identify objects, and classify scenes are necessary for any autonomous robot vehicle to overcome problems and complete tasks. Various sensors have been used to sense the environment, which consist of active sensors such as laser scanners and millimeter-wave Radars, and passive sensors such as charge-coupled device (CCD) cameras, stereo cameras and Forward looking infrared (FLIR) systems. A laser scanner was employed in [7] and [9] for object and people detection. A vehicle tracking algorithm was developed in [40] by taking data from a Radar as one of the inputs. Object identification, tracking, and pattern recognition algorithms are developed based on vision information [35], [37], [38] and

1.1 Motivation

[40], [50] and [51] developed target tracking systems based on FLIR cameras.

Data processing is an important step to extract meaningful information from data, to reduce the dimensions of the data and make the data appropriate for certain applications. Feature extraction simplifies the amount of resources [5] required to describe a large set of data meaningfully. Analysis with many variables is generally complex and requires large amounts of memory and computation power. Feature extraction may transform the data into other spaces in order to construct a combination of variables describing the data with sufficient accuracy.

2D laser scanners have become very popular and been used in many autonomous systems. In the case of the laser data, the principle steps of feature extraction are segmentation and feature identification based on known models. Successful feature extraction therefore depends on efficient noise reduction [6] and proper segmentation. Methods applicable to range data include the Hough transform [17] and RANSAC [15]. Extracted features such as point clusters, lines or circles are inputs to many autonomous robot applications to perform certain tasks. Simultaneous Localization and Mapping (SLAM) [33] and [34] is an application that takes those features as input to improve the precision of localization of the vehicle and mapping of the environment. Robust feature extraction plays vital role in SLAM algorithms, because it provides a compressed representation of the data and together with data association provides the necessary inputs for feature-based SLAM.

3D representations of the environment recently became a popular research topic (see [19], [52] and [53]), since it represents the real world more precisely and covers the environment in all three dimensions. [22], [23], [25], [28] and [29] proposed algorithms to extract planes from the 3D data. However, the amount of data is extremely huge compared with 2D data, the processing has large memory requirements

1.3 Objectives

and the analysis is complicated for feature extraction.

Data fusion appears to have received a great deal of attention and can be found in [35], [36], [37] [38], [40] and [41]. Extra information or attributes besides the range information might be very useful to segment the data into small clusters, which can be processed independently. Cameras provide color information of the environment, this helps identify objects based on their color. Feature extraction can then be applied to the range data corresponding to that particular object. This is believed to improve the processing time, robustness of feature extraction and overall performance of feature-based applications. This is the philosophy behind the work presented in this thesis.

1.2 Objectives

The objective of this thesis is to study, investigate and analyze:

- 2D point and line feature extraction algorithms with 2D laser data,
- 2D feature-based SLAM algorithms,
- 3D plane and line extraction with the aid of visual data from the cameras.

1.3 Contributions

- A real time algorithm is developed to extract tree trunks, lamp posts and building pillars from the environment by using 2D laser sensors. These extracted landmarks are used as input in point feature-based Extended Kalman Filter SLAM to improve the precision of localization of the vehicle.
- The Hough transform, which is applicable for range data, is developed to extract lines corresponding to building walls. Line feature-based SLAM is

1.4 Thesis Organization

implemented for an indoor test run.

- The integration and calibration of range and vision sensors are carried out to achieve data fusion. 3D plane and line extraction algorithms are developed based on image segmentation (Mean Shift) and PCA methods.
- The impact of calibration and segmentation with respect to plane and line extraction is studied and investigated.

1.4 Thesis Organization

This chapter outlines an overview of the work carried out in the course of the thesis and its structure. The rest of the thesis is organized as the follows:

Chapter 2 presents a literature survey on existing feature extraction methodologies for 2D and 3D laser data. Furthermore, data segmentation methods before feature extraction are reviewed. It also covers the existing work on applications that make use of extracted features as input to achieve other major tasks. Data fusion and its applications are also discussed in the chapter.

Chapter 3 discusses feature extraction algorithms for 2D laser range data such as point clustering, RANSAC, Hough transform. Real time extraction algorithms are developed to extract tree trunks, lamp post, building pillars and walls from 2D laser data. The performance of RANSAC and the Hough transform is compared based on the consistency of the extracted line features. then, a least squares approach is applied to data corresponding to line features to enhance their precision. Extracted features from various methods are used as input for feature-based EKF SLAM to improve the precision of the vehicle localization and the map of the environment. The performance of EKF SLAM is studied and investigated.

1.4 Thesis Organization

Chapter 4 introduces a built-in-house 3D LADAR which is able to produce 3D scans of the environment. It also describes the Principal Component Analysis (PCA) method which is a powerful tool used to compress data, or highlight its similarity and differences. An implementation of the PCA method on simulated 2D and 3D data is carried out to extract lines and planes. Application of the PCA method on real data is examined. Results on real data show insignificant information of the data, since the environment does not comprise just a line or a plane, meaning that laser range point clouds appear to be randomly distributed. Therefore, the range data needs to be trimmed or segmented first before the application of PCA. This will improve the performance of the method.

Chapter 5 describes an integrated system with a 3D LADAR and a camera. The calibration of the system is carried out to find correspondence between the two sensors. The fusion of range information from the laser and color data from the camera is made to produce colored-registered range data. The chapter also illustrates some existing image segmentation algorithms such as Smallest Univalued Segment Assimilation Nucleus (SUSAN), Bilateral filter and Mean Shift algorithms. Their performance on color images is analyzed and compared. In particular, edges are sharp and well preserved by Mean Shift algorithm even though the rest of the image is smoothed. Therefore, the Mean Shift algorithm is applied to segment the image. The fusion of the segmented image and range data is then carried out. The range data is segmented based on the clusters of the image to form small groups of point clouds. Then the PCA method is applied to each group of segmented range data to determine whether the range data is a plane, a line or randomly distributed.

Chapter 6 summarizes and concludes the work presented in the thesis and also provides recommendations for future work.

Chapter 2

Literature Review

2.1 Data Segmentation

Segmentation is the process of clustering a set of data into several groups, each of which possibly associates with different structures of objects. The purpose of data segmentation is to extract useful information or certain models from the data and try to reduce the memory usage by not using the redundant data. Its main objective is trying to speed up the model fitting processes and improve their robustness and efficiency. The segmentation can be done in many ways based on prior criteria.

Figure 2.1: The relationship between successive range for a planar surface.

A popular segmentation method for 2D laser data exploits the relationship between successive range points on a planar surface. Figure 2.1 depicts a 2D laser scan

2.1 Data Segmentation

on a planar surface. Three successive laser points A, B and C have ranges d_i , d_{i+1} and d_{i+2} respectively. Their range relationship was studied in [1], [2] and [3], and expressed in mathematical form as the follows:

$$d_{i+2} = \frac{d_i d_{i+1}}{2d_i \cos \gamma - d_{i+1}} \quad (2.1)$$

where γ is the constant angle between successive samples of the sensor as it rotates about its vertical axis. γ can be 0.25, 0.5, 1.0 degree for 2D SICK laser data. In [3], a filtering process with the Extended Kalman Filter was applied to predict and update the range measurement. This provides optimal estimates as to whether the point is obtained from the same planar surface as its previous neighbor points or it is from a different planar surface. This leads to a possible means of segmenting the range data into planar surfaces.

Besides planar surfaces, some research on the use of other geometrical features has been carried out in [4]. This also addressed the problem of extracting tree trunks from laser scan data where the trees are not perfect circles, and the centre and radius of a circle are estimated by averaging the measurements. The calculation is affected by the number of points representing the trees, since the number of laser points reflected by the same object depends on the distance between the sensor and the object, and the scanning angular resolution.

Object detection is a process which takes place after data segmentation, but both processes are sometimes the same. It can be achieved by feature based approaches described in [8] and [9]. [7] demonstrates a fast line, circle and leg detection from laser scan data. Basically, the algorithm first employs data segmentation method based on point-distance-based method [11]. Range segmentation produces clusters of consecutive scan points, because their range information shows that they probably

2.2 Line Extraction

belong to the same object. Two consecutive points P_i and P_{i+1} belong to the same object as long as their distance is less than a given threshold. Then, the segmented data for each cluster undergoes line, circle and leg detection algorithms to check if the criteria are met. Features are eventually extracted, if the criteria are met.

Segmentation can be achieved not only in the Cartesian space, but in any model space. [10] presents a segmentation method which is applied to the data in the Hough space, and the segmentation is done by accumulating evidences similar to the Hough transform.

2.2 Line Extraction

Line segment is the simplest feature and easily found in any indoor environments. Many algorithms have been proposed for line extraction from 2D range data. The following sections briefly explain various line extraction methods along with their advantages and shortcomings. Those are:

2.2.1 Split-and-Merge Algorithm

Split-and-Merge is a commonly used method for extracting lines from 2D laser data. It has been studied and used in [12]. Figure 2.2 depicts the split and merge process for line extraction. Split-and-merge method can effectively extract lines from data of structured environments. However, the computational time is huge to process data of unstructured environments, especially outdoor environments. Because the data is most likely randomly distributed, splitting phase is dominant in the process.

By making use of the continuity of 2D laser points from a laser scanner, a line is first formed by the two ends of the data set. The distance from each point to the line is calculated, and the maximum distance is then determined. If the distance is

2.2 Line Extraction

greater than the given distance threshold, the data set is split into two subsets at that particular point. The process is repeated until there is no more splitting, then the merging finally takes place. Each step of the algorithm is repeated for every data point and splitting phase.

Figure 2.2: Split-and-Merge for Line Extraction

The algorithm consists of the following steps:

1. Initialize set $s_1 = \{x_1, x_2, \dots, x_n\}$ in a list
2. Fit a line to the set s_i in the list, the two ends of the line are the ends of the data set
3. Determine the point which has maximum distance to the line
4. If the maximum distance is less than a threshold, execute 2
5. Otherwise, split s_i at the point with maximum distance into s_{i1} and s_{i2} , replace s_i by s_{i1} and s_{i2} in the list, execute 2 for the 2 new lines and continue
6. When no more splitting occurs, merge the separated segments

2.2 Line Extraction

2.2.2 Incremental Algorithm

The incremental Algorithm is very simple to implement. The implementation of the method can found in [13]. [14] proposed map building method based on line segments extracted by the incremental algorithm. The algorithm takes advantage of how the environment is scanned by a 2D laser, and performs the following steps:

1. Begin with the 2 first points of the 2D laser data, construct a line
2. Add the next point to the current line model
3. Recompute the line parameters
4. If the number of the inliers are more than the given threshold, line condition is satisfied, continue (go to 2)
5. Otherwise, retrieve the last point and restore the previous line parameters, return the line
6. Continue with the next 2 points, go to 2

The conditions are:

1. the distance of the new point (newly added to the line) to its adjacent point must be smaller than a given threshold
2. the distance of the new point to the current line under extraction must be smaller than a given threshold
3. the angular difference between the new point to its adjacent point must be smaller than a given threshold.

If any of these criteria is not met, it means that the newly added point is assumed not to conform to the current line segment. Therefore, the current line segment stops

2.2 Line Extraction

at this point. If these criteria are met, the parameters of the current line segment need to be updated.

2.2.3 RANSAC Algorithm

The Random Sample Consensus (RANSAC) [15] algorithm is a generic segmentation method and can be used with many types of features once the feature model is defined. It is simple to implement and very popular in computer vision to extract features. The details of the algorithm will be discussed in Section 3.3.1.

The advantages of this algorithm are that it is simple to implement and has good tolerance to noise. However, the method produces random results and sometimes fails to extract the expected lines.

2.2.4 Hough Transform

The method has been used to extract lines and circles from an image [17]. The application of Hough Transform on 2D range data is developed for line extraction in this thesis. The algorithm can extract lines from the data without prior knowledge about the number of the lines. The details of the algorithm will be discussed in Section 3.3.2.

2.2.5 EM Algorithm

Expectation Maximization (EM) is a probabilistic method. The algorithm weights each point's influence on the fit according to its uncertainty, which is derived from sensor noise models. An initial guess of the line's parameters is made, then the weight of each point is calculated from the line model, which is the expectation stage. Finally, a new line model is estimated from the non-uniformly weighted range measurement based on the maximum likelihood approach, this is the maximization

2.2 Line Extraction

stage. EM has been used as a line extraction tool in [16].

The EM algorithm consists of the following steps:

1. Start with a set of N points
2. An initial guess of the line's parameters is carried out by using the Hough Transform algorithm to detect collinear points and approximately estimate the line through these points
3. The distance between each range point and the line are calculated
4. Compute the weight of each range point from the line model based on the covariance of the calculated distance.
5. New line's parameters are estimated from the non-uniformly weighted range points based on the maximum likelihood
6. Repeat 3, 4 and 5 until the maximum number of steps have been reached or convergence of line's parameters has been obtained
7. Repeat steps 2 to 6 until the maximum number of iterations is reached or a line is found
8. If a line is found, store the line, remove the inliers from the data set and go back to 2
9. Otherwise, terminate.

Additional to the definition of each line extraction algorithms described, a comparison between each methods was carried out in [18]. Advantages and drawbacks of each algorithm were highlighted. Several criteria such as speed, complexity and precision were proposed in the comparison process. The experimental results show that

2.2 Line Extraction

the two algorithms Split-and-Merge and Incremental are preferred by their superior speed and correctness, since the experiment was conducted in an indoor structured environment which consisted of mostly walls.

RANSAC and the Hough Transform algorithms were studied in depth and used in the real time implementation. The RANSAC algorithm was chosen because the computational cost is low for the implementation of the algorithm. The algorithm extracts a line by randomly selecting two points with relatively few calculations in one iteration, and the number of iterations can be controlled by adjusting the probability of finding the expected lines according to equation 3.4.

$$k = \frac{\log(1 - p)}{\log(1 - w^n)} \quad (2.2)$$

where k is the optimal number of iterations, w is the probability of the chosen points being inliers, n is the number of points, and p is the probability of success in finding a line.

Thus, the efficiency of the algorithm can be improved with the trade-off between the probability of finding the expected lines and computational cost. The Hough Transform was also chosen because the algorithm is always able to find the expected lines and produces their corresponding ρ and α parameters which can be used directly in the SLAM implementation. The efficiency of the algorithm also depends on the trade-off between the accuracy of the line parameters and the computational cost.

The segmentation of range data is carried out according to the nature of each algorithm. In the RANSAC algorithm, points which are found to lie on a line are removed from the data set, and the process is repeated until the maximum number of iterations is reached. In the Hough Transform, the votes of points which contribute in the bin with the maximum count are removed from other bins. Those points are

2.3 Feature Extraction for 3D Range Data

then clustered into one group.

In the RANSAC algorithm, the line parameters are calculated based on the two points which are randomly selected, and consequently, even though other "inlier" points can be found, the line accuracy depends only on the initially chosen 2 points. This can result in a poor line estimate, and it is better to recompute the line parameters based on all of the inlier data. Similarly, in the Hough Transform, the precision of the parameters is affected by the resolution of the bin. Therefore, in order to improve the accuracy of the line parameters, each group of segmented points undergoes a least squares fit, after the relevant points are determined to belong to a line segment.

2.3 Feature Extraction for 3D Range Data

2D information may not provide enough information for some applications. Uneven outdoor environments, moving objects, and cluttered scenes are the complexity of real world environments. Thus, 3D sensors are necessary to improve certain application performance and overcome some existing problems, because they can provide rich information of the 3D environment.

Methods of building 3D models of the environment are proposed in [19], [20] and [21]. These methods are able to model completely unstructured environments as they can be found in rough terrain or on planetary surfaces, but they are complicated and computationally expensive.

Laser sensors capable of providing 3D scans of the environment are still rare and expensive on the market. Recently, a 2D laser scanner on a rotating support becomes the most popular setup to obtain the 3D scans (see [24], [26] and [25]). A 3D scan can be composed of several hundred 2D scans leading to dense point

2.3 Feature Extraction for 3D Range Data

clouds. Efficient data processing and a compact representation is therefore required to handle it efficiently.

Horn et al. [27] extracted vertical planes from 3D data. Walls and tree trunks are extracted from the 3D data and used as landmarks in [22]. These features could thereafter be used as input for feature-based SLAM. The advantage of these approaches is that they allow the building of a compact map representation and reduction in memory usage.

J. Weingarten and G. Gruener ([24] and [28]) used a segmentation method to decompose the space into regular cells. Cells are chosen to be cubes with a side length of 0.25m. Raw data points are associated to its corresponding cell. The RANSAC algorithm and least-square fitting are implemented to extract planes and improve their precision. In their work, the RANSAC algorithm was chosen due to its simplicity and its robustness to outliers. Eventually, similar planes from neighboring cells are fused together by using a recursive region growing algorithm. For each plane, the objective function to be minimized is

$$S(n_x, n_y, n_z, d) = \sum_{i=1}^n w_i (n_x x_i + n_y y_i + n_z z_i - d)^2 \quad (2.3)$$

where (x_i, y_i, z_i) with $i = 1 \dots N$ are the points generated by the 3D sensor which are lying on a plane. w_i is weighting factor associating to uncertainty information. The normal $\vec{n} = (n_x, n_y, n_z)^T$, which is the important parameter of the plane, can be found by calculating the eigenvector corresponding to the smallest eigenvalue of

2.3 Feature Extraction for 3D Range Data

the following matrix

$$A = \begin{pmatrix} \sum_{i=1}^n w_i x_{t_i}^2 & \sum_{i=1}^n w_i x_{t_i} y_{t_i} & \sum_{i=1}^n w_i x_{t_i} z_{t_i} \\ \sum_{i=1}^n w_i x_{t_i} y_{t_i} & \sum_{i=1}^n w_i y_{t_i}^2 & \sum_{i=1}^n w_i y_{t_i} z_{t_i} \\ \sum_{i=1}^n w_i x_{t_i} z_{t_i} & \sum_{i=1}^n w_i y_{t_i} z_{t_i} & \sum_{i=1}^n w_i z_{t_i}^2 \end{pmatrix} \quad (2.4)$$

Finally, the orthogonal distance between the plane and the origin is found to be $d = \vec{g} \cdot \vec{n}$, where $\vec{g}$ is the center of mass of the laser points which comprise the plane.

Another approach for 3D planar extraction is discussed in [26] and [29]. The approach employs the expectation maximization algorithm to fit a low complexity planar model to 3D range data. In this method, the plane is represented as a tuple $\theta_m = (\vec{n}_m, d_m)$ where $\vec{n}_m$ is unit normal vector and d_m is the Euclidean distance of the plane from the origin. The Euclidean distance from any point z_n to the plane θ_m is given as:

$$d_1(z_n, \theta_m) = z_n \cdot \vec{n}_m - d_m \quad (2.5)$$

In the presence of Gaussain noise with variance ρ in the measurement processes, the probability that z_n belongs to plane θ_m can be expressed as the follows:

$$p(z_n | \theta_m) = \frac{1}{\sqrt{2\pi\rho^2}} \exp \left[-\frac{1}{2} \left(\frac{d_1(z_n, \theta_m)}{\rho} \right)^2 \right] \quad (2.6)$$

The goal of the algorithm is to determine the model (Θ^*, Φ^*) where Θ is a set of planes θ_m and Φ is a set of main directions of these planes, that maximizes the likelihood of the data Z . A mathematical derivation is found to be:

$$(\Theta^{[i+1]}, \Phi^{[i+1]}) = \operatorname{argmax}_{(\Theta, \Phi)} E_{AB} [\log p(A, B, \Phi, \Theta, Z) | \Theta^{[i]}, \Phi^{[i]}] \quad (2.7)$$

On the other hand, 3D data can also be obtained from a stereo cameras. In [23], a Digiclops tri-stereo camera is used to provide 3D range information of the

2.4 EKF SLAM

environment. Similar to planar surface extraction carried out in [24] and [28], a grid-based range data segmentation and plane extraction are implemented with the range data generated by the camera. Since the measurements are noisy and less accurate compared with those given by a laser, the error in the Z, X and Y axes, and the angular error were investigated and studied under various lighting, distance and surface texture conditions.

The feature extraction from 3D data is complicated and computational intensive, since most of the proposed methods are repeating processes and their memory requirement is huge. A non-iterative method will be discussed in chapter 4. In our method, segmentation is first carried out on the image. Then point clouds corresponding to each region of the image are clustered into one group. The PCA method is then applied to each group of point clouds to determine whether the points are randomly distributed over the space, lying on a plane or a line. In this case, the points do not undergo a least square fit, since the PCA method is able to extract a line or a plane with the least squared error based on all of the inlier data.

2.4 EKF SLAM

Once features can be reliably extracted from data, robot navigation is dependent on methods which optimally combine the uncertain information provided by these features with uncertain vehicle motion. A tool which can be used to achieve this is EKF SLAM which is adopted later in this thesis. The Extended Kalman Filter (EKF) [30], [31] and [32] is an estimation theoretic algorithm used as main tool in several navigation and estimation problems including SLAM. The EKF estimator represents the state probability distributions by second order statistics. The optimal states are then determined by minimizing the covariance matrix. The first solution

2.4 EKF SLAM

to the SLAM problem was proposed in [34] by Smith, Self and Cheeseman. They demonstrated the importance of map-vehicle correlations in the SLAM state vector comprising the vehicle state and map-feature (landmark) locations. Dissanayeke et al presented the basic EKF SLAM framework [33] representing both the vehicle and landmark locations by absolute coordinates with reference to a global coordinate frame. The EKF SLAM problem can be formulated probabilistically in a manner described in the following section:

2.4.1 Vehicle and Environmental Feature State

In SLAM, the state vector contains current information of the vehicle's pose and a set of landmarks. The state vector is no longer a constant size vector, instead it grows when new landmarks are discovered and added to the vector. Thus, the state vector can be written as:

$$X_k = [X_{vk} \ M_1 \ M_2 \ \dots \ M_N]^T$$

where X_{vk} is the current pose of the vehicle, and $M_1, M_2, \dots, M_N$ are landmarks 1, 2 to N respectively.

2.4.2 Relative Observations

Observations are the measurements of landmarks or features given by sensors such as a 2D laser scanner, camera or a Radar. A set of landmark measurement with respect to vehicle local frame is given as the following:

$$Z_k = [M_{1Lk} \ M_{2Lk} \ \dots \ M_{N_{Lk}}]^T$$

where subscript Lk indicates the measurement taken in local frame at the time k .

Now, a set of prediction observation can be determined, a filtering process is implemented as the following to find the optimal solution:

2.4 EKF SLAM

2.4.3 EKF Prediction Equations

At time k , the input to the vehicle is given by U_k . Based on a vehicle motion model, the next vehicle pose can be determined as a function of the current state and the input. Thus, the prediction equation of the state vector can be written as:

$$X_{k+1|k} = f(X_{k|k}, U_k) \quad (2.8)$$

where $X_{k+1|k}$ represents the predicted vehicle state at time $k + 1$, given k observations.

The uncertainty of the vehicle also increases due to input noise, and the uncertainty is interpreted by the covariance matrix $P_{k+1|k}$. The matrix can be predicted as follows:

$$P_{k+1|k} = \nabla_{X_v} f P_{k|k} \nabla_{X_v}^T f + \nabla_U f Q_k \nabla_U^T f \quad (2.9)$$

where $\nabla_{X_v} f$ is the Jacobian of the prediction model with respect to the state variables x, y and θ , Q_k is the input process noise at time k and $\nabla_U f$ is the Jacobian of the prediction model with respect to the input variables.

Equation 2.9 calculates the covariance matrix which, in general contains non-zero off-diagonal terms. The terms correspond to the cross-correlations between the state variables and the landmarks which need to be predicted also. The cross-correlations $P_{k+1|k}^{X_v, j}$ between vehicle's pose x, y and θ , and landmark j can be calculated as follows:

$$P_{k+1|k}^{X_v, j} = \nabla_{X_v} f P_{k|k}^{X_v, j} \quad (2.10)$$

and the cross-correlation $P_{k+1|k}^{j, X_v}$ between landmark j and the vehicle state variables is just the transpose of $P_{k+1|k}^{X_v, j}$.

i.e.

$$P_{k+1|k}^{j, X_v} = \left[P_{k+1|k}^{X_v, j} \right]^T \quad (2.11)$$

2.4 EKF SLAM

2.4.4 The EKF measurement update equations

The Kalman Gain W_k is the optimal gain that minimizes the posteriori error covariance P . It is given by:

$$W_k = P_{k+1|k} H_k^T [H_k P_{k+1|k} H_k^T + V_k R_k V_k^T]^{-1} \quad (2.12)$$

where H_{k+1} is the Jacobian of predicted observation model with respect to vehicle's pose, and V_{k+1} the Jacobian of predicted observation model with respect to observation variables.

The optimal linear estimate of the state vector is given by:

$$X_{k+1|k+1} = X_{k+1|k} + W_k [Z_k - h(X_{k+1|k}, 0)] \quad (2.13)$$

where Z_{k+1} is the observation vector given by sensor measurement at time k .

The error covariance is at the minimum with value of W_k calculated, and the covariance matrix is updated as the following:

$$P_{k+1|k+1} = [I - W_k H_k] P_{k+1|k} \quad (2.14)$$

where I the identity matrix.

2.4.5 Adding New Landmarks

It is likely that the vehicle observes new landmarks beside the ones stored in the state vector when it is moving. In this case, the process of adding new landmarks to the vector must take place. New landmark parameters are added into the state vector, and the covariance matrix P propagates by adding cross-correlation terms of new landmarks with respect to vehicle's pose and their initial uncertainty. Thus, the size of state vector grows proportionally to the numbers of landmarks and the covariance matrix P grows with $O(N^2)$ where N is the number of landmarks.

2.4 EKF SLAM

New Landmark Parameters

Let's assume that new landmarks are observed by the sensor at instant k . The sensor provides their information with respect to the local frame. Therefore, the parameters need to be transformed from the local frame to the global frame, then they are added into the state vector:

$$M_G = g(x_k, y_k, \theta_k, M_L)$$

where $g(\cdot)$ is transformation function from the local frame to the global frame, M_L and M_G are landmark parameters with respect to the local frame and to the global frame respectively.

Propagation of Covariance Matrix P:

Similarly, the other properties such as their uncertainty and cross-correlation terms of the new landmarks need to be added to the covariance matrix too.

p^{x_v}	$p^{x_v, 1}$					$p^{x_v, n}$	$p^{x_v, n+1}$
p^{1, x_v}	$p^{1, 1}$					$p^{1, n}$	$p^{1, n+1}$
p^{n, x_v}	$p^{n, 1}$					$p^{n, n}$	$p^{n, n+1}$
p^{n+1, x_v}	$p^{n+1, 1}$					$p^{n+1, n}$	$p^{n+1, n+1}$

Figure 2.3: Structure of the Covariance Matrix.

The structure of covariance matrix P is shown in Figure 2.3. When a new landmark is added, the matrix P increases its columns and rows as shown in the shaded areas in the figure and the elements need to be filled in as the follows:

2.5 Fusion of Range and Visual Data

The uncertainty of a landmark when it is first observed, is the combination of uncertainty of vehicle at that instance and the measurement noise:

$$P^{N+1,N+1} = J_{X_v} P^{X_v} J_{X_v}^T + J_z R J_z^T \quad (2.15)$$

where P^{X_v} is the covariance matrix of vehicle pose, R is the measurement noise covariance matrix, $J_{X_v} = \left(\frac{\partial}{\partial x_k} + \frac{\partial}{\partial y_k} + \frac{\partial}{\partial \theta_k} \right) g(x_k, y_k, \theta_k, M_L)$ and $J_z = \frac{\partial}{\partial M_L} g(x_k, y_k, \theta_k, M_L)$. The cross-correlation between the new landmark and vehicle pose can then be expressed as:

$$P^{X_v, N+1} = P^{X_v} J_{X_v}^T \text{ and } P^{N+1, X_v} = [P^{X_v, N+1}]^T$$

Finally the landmark-landmark covariance can be determined as the following:

$$P^{N+1, i} = J_{X_v} P^{X_v, i} \text{ and } P^{i, N+1} = [P^{N+1, i}]^T$$

2.5 Fusion of Range and Visual Data

Range and vision sensors appear to be the most popular sensors in mobile robot applications. However, each sensor has its own unique characteristic, and one cannot replace another. The main advantage of 2D laser sensors over vision sensors is that they are capable of providing accurate range measurements in large angular fields and at very fast rates. However, the information provided by the 2D laser might not be enough for some specific tasks such as obstacle detection, object identification, etc. Cameras can provide color information of the objects, which is very useful for the applications mentioned. Furthermore, a stereo camera can provide 3D range information of the environment, even though the range precision depends on the distance of the objects. The range information can be good enough for some tasks. Thus, data fusion between both sensors might provide rich information of the environment. A method for fusing range with visual information was proposed in [35].

2.5 Fusion of Range and Visual Data

The proposed method employed a laser scanner and a camera. The system is able to provide 3D structure information.

In the work of [36], A multi-sensor system, composed of a 2D laser and a CCD camera, could provide redundancy and assured reliability and precision of the observed features. Calibration of the relative transformation between the sensors was obtained. Hence, matching between corners and vision edges could be obtained to confirm the robustness of the feature. Integration of new information reduces the uncertainty in the feature location and results in fewer false detections. This improves the quality of data association and enhances the overall performance of feature-based SLAM.

Recently, data fusion between laser and vision sensors has also been implemented to detect moving objects. Due to changing lighting conditions, the interaction or occlusion between targets and the vast computational burdens are the challenges for vision-based tracking systems. Laser-based people tracking systems do not provide any unique information to distinguish one object from another. [37] fuses panoramic vision and laser range data to track people. [38] develops a real time people tracking system with a 2D SICK laser sensor and a video camera using Tsai's model [39]. Once the human body is detected using laser data, corresponding body regions are localized in the video image. A model body region is described by a rectangle in the image plane. Joint likelihood based on laser and vision data is determined. Together with a particle filter, the system can provide consistent people tracking with less time overhead compared with a vision only system.

Similarly, the integration of a 2D laser and a panoramic camera is achieved in [41]. First, the range data from the laser is used to solve the SLAM problem and to extract wall segments. Then, visual information from a panoramic camera is used

2.5 Fusion of Range and Visual Data

to texture walls and floor. A useful 3D model of an indoor office was able to be built by the setup.

In the place of laser sensor, RADAR can be used to provide range information. [40] presents a vehicle detection and recognition system based on the fusion of a RADAR and a visual data. RADAR data can provide consistent distance measurements of vehicles in front, and is also used to narrow down the size of the search window on the video image. With the fused information, the system can provide a robust and real-time vehicle detection.

Similar to the fusion of multi-sensor data in the work described above, the fusion of range and vision will be discussed in chapter 5. The fused information will be used to extract lines and planes from 3D environments.

Chapter 3

2D Laser Based Feature Extraction

3.1 Laser Data

A laser scanner can provide range measurements with less noise and the accuracy can be in the order of centimeters. The most popular laser scanner is the SICK laser which can output range measurements for 180 degree front view or 360 degree full view. It has an angular resolution of 0.25, 0.5 or 1.0 degree, and the maximum range can reach up to 50 meters for a typical SICK laser scanner, and 80 to 150 meters for larger models. With a simple serial communication protocol, the typical scanner outputs a package of range information of a full scan with a frequency of 1Hz. With the latest broadband communication method, the output rate can be up to 50-100 Hz.

Figure 3.1 is a combined scan of two LMS 291 SICK lasers to produce a full 360 degree view with a resolution of 0.5 degrees. The maximum range of the output is up to 81 meters. A maximum is returned if there is nothing within the maximum range at that particular bearing angle.

Typical features which can be extracted from the 2D scan can be lines, edges, and point clusters. The more complicated features can be circle, curves and polygons.

3.2 Point Cluster-based Feature Extraction

Figure 3.1: 2D Laser Scan.

3.2 Point Cluster-based Feature Extraction

3.2.1 Spike Landmarks

Figure 3.2: Spike Landmarks

Spike landmarks (abrupt changes in range) can be extracted as they are local extrema in the scan. They can be identified with range information from the scan, where two range values differ by more than a certain amount, which is the so-called threshold. Basically, the method is trying to identify the jumps in the range data. Because those jumps can be easily identified and extracted from the rest of points, so those can be used as input for certain applications, Simultaneous localization and Mapping (SLAM) is one of them.

3.2 Point Cluster-based Feature Extraction

Figure 3.2 depicts spike landmark extractions from a range versus bearing angle plot. Spike landmark extraction is easy, simple and robust to implement. However, it is risky to trust a single range laser point and difficult to model the noise in the measurements, because:

1. Bearing angle resolution might effect on the consistency of the range data.
2. noise might corrupt the measurements.
3. Wrong measurements are given due to the effect of ghost reflections.

In order to minimize the inconsistency in the range measurements due to the causes mentioned above, a spike landmark which consists of a set of points is preferred and extracted. This leads to point clustering.

3.2.2 Point Clustering

This process aims to group points that have similarity into a cluster, and the data is segmented into small clusters. Normally, distance is used as criteria in the clustering process. The closeness of two adjacent points determines whether the two points are likely to be from the same object or not. Two possible methods have been implemented to measure the closeness of two laser points, those are the following:

Point-Distance-Based method

The point-distance based method [11] is a measure to cluster two or many points in a scan, based on distance between two successive points. Two adjacent points are shown in Figure 3.3 with angle resolution of 0.5 degree. The distance d between the two points, point A and B, is calculated, then a comparison with a given threshold is made; those points belong to the same cluster if d is smaller than the threshold value, otherwise they belong to two different clusters.

3.2 Point Cluster-based Feature Extraction

Figure 3.3: Point-Distance-Based Method.

Range-based method

Figure 3.4: Range-Based Method.

On the other hand, the range-based method is an alternative method to cluster points. This method make use of the relationship of successive scans on a planar surface ([1], [2], [3] and [11]). However, the EKF filtering part is skipped to simplify and speed up the clustering process. Figure 3.4 depicts two successive laser beams with bearing angle difference of 0.5 degree. In this case, point A and B belong to the same cluster if point A is on the segment between A1 and A2, or, in another words, the range of point A is within $\{R - \delta d, R + \delta d\}$, where δd is a given constant. Those two points belong to different groups if point A is outside the segment specified.

The process repeats for every two adjacent points for the whole scan. Then the

3.2 Point Cluster-based Feature Extraction

data is segmented based on the number of clusters found. The centroid of each cluster is calculated, and those centroids are used as feature representation in many applications. d in the first method and δd in the later are appropriately tuned based on applications.

3.2.3 Tree Trunk and Lamp Post Landmarks

Tree trunks and lamp posts are extractable landmarks, because they possess the spike landmark property, and there can be two or more laser points scanned from a tree or lamp post. A similar tree trunk extraction method is carried out in [4]. First, laser points are clustered into groups by using the point-distance based method. then the distance between start point and end point of each cluster is calculated. If the distance is below dth_m which is the maximum diameter, the cluster is considered as the scan from tree trunk or lamp post. This method was also employed in [7] to detect human legs. The method was demonstrated to provide robust detection.

For our tree trunk and lamp post extraction algorithm, the point-distance based method was implemented to cluster the 2D laser points. However, the threshold of distance between two adjacent points is proportional to the range of the laser point. The value of dth_m and the distance threshold were set to 0.8m and 10% of the range value respectively.

Figure 3.5 demonstrates tree trunk and lamp post extraction from a 2D laser data. The right figure plots the laser data with respect to its local reference frame. The laser data is a 360 degree full view from a combined 2 SICK lasers. Tree trunk and lamp post extraction is applied on the data, the extracted features are circled in red. 10 features are extracted from the data, some of them are lamp posts, and other are tree trunks. The left figure is a scene captured by a camera. Time syn-

3.2 Point Cluster-based Feature Extraction

Figure 3.5: Tree Trunk and Lamp Post Extraction.

chronization was achieved between the camera and laser data to check the extracted features (right plot) with those viewable from the camera image (left). The image highlights the ground truth of the trees and lamp posts which are corresponding to the extracted ones from the laser data.

3.2.4 A SLAM Implementation with Point Cluster Based Features

The extracted features can be used as observations in an feature-based EKF SLAM application to enhance the localization of the vehicle. The observations and predicted observations are formulated as follows:

(fx_i, fy_i) are the global location of a tree trunk or a lamp post i and stored in the state vector. The same feature is observed by the sensor and parameterized as ρ and α , the distance to the vehicle and relative bearing angle respectively. Assume that at instant $k + 1$, the vehicle moves from state X_k to X_{k+1} . At state X_{k+1} , the predicted feature location can be calculated with respect to the new predicted pose of the vehicle. For each feature point in the state vector, its new ρ and α can be calculated as follows:

3.2 Point Cluster-based Feature Extraction

$$\begin{bmatrix} \rho_j \\ \alpha_j \end{bmatrix} = \begin{bmatrix} \sqrt{(x_{k+1} - fx_j)^2 + (y_{k+1} - fy_j)^2} \\ \tan^{-1} \left(\frac{fy_j - y_{k+1}}{fx_j - x_{k+1}} \right) - \theta_{k+1} \end{bmatrix} \quad (3.1)$$

By differentiating Equation 3.1 with respect to x_{k+1}, y_{k+1} and θ_{k+1} , the Jacobian of the observation model with respect to pose variable is given as:

$$\nabla_{X_v} h = \begin{bmatrix} \frac{x_{k+1} - fx_j}{\rho} & \frac{y_{k+1} - fy_j}{\rho} & 0 \\ \frac{fy_j - y_{k+1}}{\rho^2} & \frac{fx_j - x_{k+1}}{\rho^2} & -1 \end{bmatrix} \quad (3.2)$$

and the Jacobian of the observation model with respect to fx_j and fy_j is given as the following:

$$\nabla_{X_{fj}} h = \begin{bmatrix} \frac{fx_j - x_{k+1}}{\rho} & \frac{fy_j - y_{k+1}}{\rho} \\ \frac{y_{k+1} - fy_j}{\rho^2} & \frac{x_{k+1} - fx_j}{\rho^2} \end{bmatrix} \quad (3.3)$$

These Jacobians are necessary for predicting the uncertainty of each feature and will be used in EKF framework later. The formulation EKF SLAM is shown in Section 2.4.

A Honda CRV vehicle whose localization is referenced to the global position coordinates (northing, easting and altitude) and provided by a Novatel System (an integrated system that consists of Initial Measurement Unit (IMU), GPS and odometry) was used to perform point cluster feature based SLAM. Two LMS291 SICK lasers were mounted on the top of the vehicle to produce 360 degree scans of the environment.

To examine the performance of SLAM and the robustness of the feature extraction algorithm described in the previous section, a test was conducted on the campus of Nanyang Technological University (NTU). Figure 3.6 is the image of the test site obtained from Google Earth, and the red path is approximately the trajectory of the vehicle. To verify the performance of the feature-based SLAM algorithm, the vehicle was manually driven following the loop shown in red line 3 times. Actual ground truth of vehicle is very difficult to obtain, so the vehicle was started and stopped

3.2 Point Cluster-based Feature Extraction

Figure 3.6: Google Map of North Spine, NTU.

at approximately the same spot. This allows us to easily verify the performance of both the Novatel system and the SLAM algorithm. To simulate the case where there is GPS outage, the GPS signal cable was unplugged during the test, and the localization is based on IMU only.

Figure 3.7: Superimposition of Vehicle Trajectory on the Google Map.

Figure 3.7 depicts the superimposition of the vehicle trajectory based on the Novatel system output as shown as loops in magenta. It can be observed that in the first loop, the localization given by the system was acceptable. However, the position error accumulated and path started to drift always from the ground truth. The 3rd Loop shows significant error in both northing and easting. Based on the coordinates given by the system, the vehicle was hitting the building wall, which

3.2 Point Cluster-based Feature Extraction

does not make any sense.

Figure 3.8: SLAM and Novatel Outputs fused on the Google Map.

The two lasers mounted on the vehicle could repeatedly observe the tree trunks, lamp posts and building pillars. These objects are extracted from the scan and used as features for the SLAM algorithm. The output of SLAM is shown as the blue loops in Figure 3.8. The fact that the vehicle started and stopped almost at the same place makes it easy to verify the correctness of the output from the SLAM algorithm. Each blue loop almost overlaps with each other, which shows that the vehicle was correctly estimated to be driven along the same path. This indicates that SLAM algorithm corrected the path given by the Novatel system, and minimized the error in both northing and easting based on the observations obtained from tree trunks, lamp posts and building pillars.

Figure 3.9 depicts the estimate path of the vehicle and landmarks with respect to its starting point. The estimated landmarks are shown as green stars, while the currently observed ones as blue stars. The figure also shows the uncertainties, as red ellipses, of the estimated vehicle pose and landmarks.

3.2 Point Cluster-based Feature Extraction

Figure 3.9: SLAM Output.

Figure 3.10: Cluster-Based Point Feature Issue.

3.2.5 Cluster-Based Point Feature Issue

There is an issue encountered in SLAM using cluster-based point features. The laser scan is initially clustered into groups. The centroid of each group is then determined and used as features in point-feature based SLAM. However, the centroid of each cluster might vary when the point of view of the sensor changes. In Figure 3.10, a vehicle trajectory is shown moving around a pillar. The vehicle moved from A to B and C. At A, B and C, the vehicle performed a laser scan of the environment. The pillar appeared in the scan, but different portions of the pillar were observed by the vehicle at different locations, this led to different values of cluster's centroid being

3.3 Line Features

estimated. Since there is error in the centroid of the pillar observed by the vehicle, the SLAM algorithm interpreted it as an odometry error and tries to correct the vehicle pose even though the vehicle's pose might be accurate. On the other hand, the calculated centroids of the pillar lying on the inner circle as shown in Figure 3.10. The diameter of the inner circle is proportional to the size of the pillar, this leads to difficulty in modeling the noise of the calculated centroid. It might make the convergence rate of EKF filter slow if the measurement noise is much larger than the actual one, or even diverge if the measurement noise is modeled much smaller.

3.3 Line Features

Lines are alternative features which can be extracted from laser scans. Line extraction from scans, especially in structured and indoor environments, is a fairly easy task. In this report, based on the conclusions of the line based feature extraction algorithms outlined in chapter 2, two line extraction algorithms are discussed and implemented, RANSAC and the Hough Transform:

3.3.1 RANSAC Line Based Extraction

The RANSAC algorithm [15] is now presented along with results which show its advantages and disadvantages. In particular, it will be shown that the RANSAC algorithm produces random results for the same environment scan, which is not suitable for some applications.

Two points are randomly selected from the data set. A line is formed based on these two points. Then the distance of each point in the set to the line is calculated, points belong to an inlier set if the calculated distance is less than a given threshold, else they are called outliers. If the number of inliers is greater than

3.3 Line Features

a certain value given, the line is considered as a good-fit model to the given data set. The algorithm repeats till a good-fit line is determined or a maximum defined iteration number is reached. Figure 3.11 shows the flow chart of the RANSAC

Figure 3.11: Flow Chart of RANSAC Algorithm.

algorithm. The distance and inlier thresholds are given constants, and the number of iterations (k) is determined by Equation 3.4 and it is proven to be optimal number of iteration to find a line [15].

$$k = \frac{\log(1 - p)}{\log(1 - w^n)} \quad (3.4)$$

where w is the probability of all a point being inlier, n is the number of points, and p is the probability that the algorithm produces good result.

Multiple lines can be extracted from the scan by looping the algorithm as many times as the number of lines to be found.

A simulation of the RANSAC algorithm was implemented in Matlab as shown in Figure 3.12. The data set was generated by a ray tracing program with Gaussian noise standard deviation of 0.04m. Lines in yellow are randomly selected and failed

3.3 Line Features

Figure 3.12: Implementation of RANSAC Algorithm.

to be a good-fit line model. The green ones are found to be best fit to the data set given, since the number of their inliers exceed the given inlier threshold (30 points). RANSAC performs fast and well in indoor structured environments, and in Figure 3.12 all four of the dominant lines were correctly extracted.

3.3.2 Hough Transform for Line Extraction

The Hough Transform [17] is a popular algorithm which has been implemented to extract features such as lines and circles from a scan or an image. In particular, the Hough Transform presented in this section is not applied to an image, but to range data. However, the algorithm still holds its properties and effectiveness. The formulation of the Hough Transform for range data will be shown in a later part of this section as well as some results obtained.

Any 2D line can be expressed in Cartesian space as the following:

$$ax + by + c = 0 \quad (3.5)$$

where x, y are variables in Cartesian, with $a, b, c \in \mathbb{R}$.

3.3 Line Features

The problem with this formulation is that vertical lines have infinite parametric values (eg. $a \rightarrow \infty$). Therefore, in Hough space, these lines are represented by their normal (ρ and α) as shown in Figure 3.13.

Figure 3.13: Hough Representation of A Line.

Figure 3.14: Hough Space Representation.

There are an infinite number of lines passing through a given point. For each line, their ρ and α are plotted in the Hough space respectively as shown in Figure 3.14.

3.3 Line Features

In the figure, the given point is $P(5, 2)$, and the relationship of ρ and α of each line passing through P can be written as:

$$\rho = x \cos \alpha + y \sin \alpha \quad (3.6)$$

where ρ is the distance of the origin to the line whose angle between its normal and the horizontal x axis is α , and x, y are the Cartesian coordinates of the given point.

Equation 3.6 can be simplified and expressed in term of a single sine term as follows:

$$\rho = x \cos \alpha + y \sin \alpha$$

$$= \sqrt{x^2 + y^2} \left(\frac{x}{\sqrt{x^2 + y^2}} \cos \alpha + \frac{y}{\sqrt{x^2 + y^2}} \sin \alpha \right)$$

let $\sin \phi = \frac{y}{\sqrt{x^2 + y^2}}$ and $\cos \phi = \frac{x}{\sqrt{x^2 + y^2}}$, we have:

$$\rho = \sqrt{x^2 + y^2} (\sin \phi \cos \alpha + \cos \phi \sin \alpha)$$

$$= \sqrt{x^2 + y^2} \sin(\alpha + \phi) \text{ As seen earlier, lines passing through a given point}$$

Figure 3.15: Points given in Cartesian Space.

can be represented by sinusoidal line in Hough space; thus, if there are n points in Cartesian space (as shown in Figure 3.15), there will be n sinusoidal lines in

3.3 Line Features

the Hough space. Lines passing through points are determined by intersection of their sinusoidal lines in the Hough space. Figure 3.16 depicts the intersection of sinusoidal lines in the Hough space. Finding the intersections of sinusoidal lines is a

Figure 3.16: Intersection of sinusoidal lines in Hough Space.

Figure 3.17: Hough Transform for Line Extraction.

complicated problem. An intuitive way of solving the problem has been implemented in the Hough Transform [17]. Similar to the Hough transform algorithm in image

3.3 Line Features

processing, the observation space is divided into small cells called bins. The bin size is defined by the resolution of the angle α and distance ρ sought to represent a line.

Given a set of points in Cartesian space, voting takes place using Equation 3.6. For each value of the angle α whose resolution is pre-determined, the value of distance ρ is calculated. Then a vote is given to the bin whose α and ρ value are found to be matched with the calculated value of α and ρ . The process is repeated for every point given. Finally, the bin which has the most votes is selected. This implies that a line has been extracted whose normal is ρ and α of the bin which has the most votes. The result of line extraction with the Hough Transform is shown in Figure 3.17.

3.3.3 Comparison of RANSAC and Hough Transform

RANSAC performs much faster than the Hough transform. It is controlled by the number of iterations. The model parameters estimated however are not very precise. The estimate is only correct with a certain probability, since it is a randomized estimator. Furthermore, the algorithm produces random results even though the sample data is the same. While the Hough Transform always produces the same and consistent result with the same data samples, the number of solutions needs not be known a priori and the accuracy is determined by angle and distance resolutions, but the computational time increases by $O(N^2)$ if the number of data increases or the angle and distance resolution are increased.

Since lines extracted by RANSAC are random, and sometimes RANSAC has a chance to fail finding the expected lines, this leads to inconsistency and failure in data association in the EKF SLAM. While the Hough Transform always guarantees to extract line with the most inliers from the scan, which makes the data association

3.3 Line Features

process less vulnerable to false detection and bad association. Therefore, the Hough Transform algorithm is used to extract line features in the EKF SLAM.

3.3.4 Fitting Lines

In the previous section, two algorithms have been introduced to yield line-based feature extraction. However, both algorithms do not take into account if the feature model gives the least square error. Therefore, the data set corresponding to each line extracted by the Hough Transform then undergoes a best fit approach. The approach is to recompute the line parameters which gives the least square error. This improves the accuracy of the line. The approach is as follows:

let r_i^2 be the square distance of point (x_i, y_i) to the line model $(y = ax + b)$, then:

$$r_i^2 = \frac{(y_i - ax_i - b)^2}{a^2 + 1}$$

where a, b are the line's parameters.

The squared distance cost function to be minimized is $\sum_{i=1}^n r_i^2$, where n is the number of points found by the Hough Transform algorithm which fit the line. It can be minimized as follows:

$$\begin{aligned} \frac{\partial \left(\sum_{i=1}^n r_i^2 \right)}{\partial a} &= 0 \\ \frac{\partial \left(\sum_{i=1}^n r_i^2 \right)}{\partial b} &= 0 \end{aligned}$$

By solving the two simultaneous equations, the values of a and b can be found. i.e.

$$a = \frac{n \sum_{i=1}^n x_i y_i - \sum_{i=1}^n x_i \sum_{i=1}^n y_i}{n \sum_{i=1}^n y_i^2 - \left(\sum_{i=1}^n y_i \right)^2}$$

3.3 Line Features

$$b = \frac{\sum_{i=1}^n x_i \sum_{i=1}^n y_i^2 - \sum_{i=1}^n y_i \sum_{i=1}^n x_i y_i}{n \sum_{i=1}^n y_i^2 - \left(\sum_{i=1}^n y_i \right)^2}$$

Figure 3.18: Best Fit Line.

3.3.5 Implementation of Line-feature based EKF SLAM

In this particular case, lines extracted by the Hough Transform are used as features (landmarks). For each line, a set of ρ and α parameters represents the range and bearing of its normal. The transformation of landmarks from the global to local frame appears to have two cases. The formulation of the predicted observation is as follows:

3.3 Line Features

Case 1 - Robot Between Origin and Line Segment

Case 1 corresponds to the current position of the vehicle and the origin of the global coordinate system are lying to one side of the line as shown Figure 3.19.

Figure 3.19: Predicted Observation Model For Case 1.

The global to local transformation of a line feature's range and bearing can be derived as follows:

$$\rho_L = \rho_G - L \cos(\theta - \alpha_G) \quad (3.7)$$

where $L = \sqrt{x_{k+1}^2 + y_{k+1}^2}$ is the Euclidean distance between the global coordinate origin and the robot, subscript L and G refer to the local and global frame coordination respectively.

Simplifying by using sine and cosine law in the right triangle $oo'x_{k+1}$

$$\rho_L = \rho_G - L[\cos\theta\cos\alpha_G + \sin\theta\sin\alpha_G]$$

3.3 Line Features

$$\begin{aligned} &= \rho_G - L \left[\frac{x_{k+1}}{L} \cos \alpha_G + \frac{y_{k+1}}{L} \sin \alpha_G \right] \\ &= \rho_G - x_{k+1} \cos \alpha_G - y_{k+1} \sin \alpha_G \end{aligned}$$

i.e.

$$\rho_L = h_1(x_{k+1}, y_{k+1}, \theta_{k+1}, \rho_G, \alpha_G) \quad (3.8)$$

and since lines $o'A$ and oB are parallel,

$$\alpha_L = \alpha_G - \theta_{k+1} \quad (3.9)$$

so that

$$\alpha_L = h_2(x_{k+1}, y_{k+1}, \theta_{k+1}, \rho_G, \alpha_G) \quad (3.10)$$

The Jacobian of each landmark is derived by differentiating h_1 and h_2 with respect to state variables x, y and θ :

$$\nabla_x h = \begin{bmatrix} \frac{\delta h_1}{x_{k+1}} & \frac{\delta h_1}{y_{k+1}} & \frac{\delta h_1}{\theta_{k+1}} \\ \frac{\delta h_2}{x_{k+1}} & \frac{\delta h_2}{y_{k+1}} & \frac{\delta h_2}{\theta_{k+1}} \end{bmatrix} \quad (3.11)$$

and the Jacobian of each line based landmark with respect to the observation Z (range and bearing) is given by:

$$\nabla_z h = \begin{bmatrix} \frac{\delta h_1}{\rho_G} & \frac{\delta h_1}{\alpha_G} \\ \frac{\delta h_2}{\rho_G} & \frac{\delta h_2}{\alpha_G} \end{bmatrix} \quad (3.12)$$

thus, for a single line based landmark, the Jacobian matrix can be written as:

$$\nabla h = \begin{bmatrix} \frac{\delta h_1}{x_{k+1}} & \frac{\delta h_1}{y_{k+1}} & \frac{\delta h_1}{\theta_{k+1}} & \frac{\delta h_1}{\rho_{k+1}} & \frac{\delta h_1}{\alpha_{k+1}} \\ \frac{\delta h_2}{x_{k+1}} & \frac{\delta h_2}{y_{k+1}} & \frac{\delta h_2}{\theta_{k+1}} & \frac{\delta h_2}{\rho_{k+1}} & \frac{\delta h_2}{\alpha_{k+1}} \end{bmatrix} \quad (3.13)$$

i.e.

$$\nabla h = \begin{bmatrix} -\cos \alpha_G & -\sin \alpha_G & 0 & 1 & x_{k+1} \sin \alpha_G - y_{k+1} \cos \alpha_G \\ 0 & 0 & -1 & 0 & 1 \end{bmatrix} \quad (3.14)$$

3.3 Line Features

Figure 3.20: Predicted Observation Model For Case 2.

Case 2 - Line Segment Between Origin and Robot

In the second case, the Hough line appears between the origin of the global frame and the vehicle as illustrated in Figure 3.20.

The conversion of global to local frame is slightly different from the previous case. In this case:

$$\rho_L = L \cos(\theta - \alpha_G) - \rho_G \quad (3.15)$$

i.e. $\rho_L = x_{k+1} \cos \alpha_G + y_{k+1} \sin \alpha_G - \rho_G$ and

$$\alpha_L = \alpha_G - \theta_{k+1} \quad (3.16)$$

and the Jacobian Matrix is given by:

$$\nabla h = \begin{bmatrix} \cos \alpha_G & \sin \alpha_G & 0 & -1 & y_{k+1} \cos \alpha_G - x_{k+1} \sin \alpha_G \\ 0 & 0 & -1 & 0 & 1 \end{bmatrix} \quad (3.17)$$

3.3 Line Features

Figure 3.21: Simulation of Line-Feature Based SLAM

To demonstrate the reliability and feasibility of the line-feature based EKF SLAM, a simulation program was created in Matlab. Figure 3.21 depicts the results of SLAM in the simulation. In Figure 3.21, two walls were artificially created as shown by the black dashed lines. The vehicle movement was executed in a step by step manner. Sequences of vehicle movements at steps 0, 61, 131, 190, 240 and 300 are shown in Figure 3.21, where the localization of the vehicle based on odometry is shown in cyan, the ground truth shown in green and the estimated localization

3.3 Line Features

based on SLAM shown in red. The vehicle positions are plotted at time intervals of 1 second with a forward speed of 0.5 m/s. In this case, the forward speed and angular rate standard deviations were 0.2 m/s and 4 degree/s respectively. The range and bearing standard deviations of the sensor measurement were 0.4 meters and 3 degrees respectively. In the simulation, the application assumed no failure in data association since the matched features were always artificially generated. It can be observed that the localization estimates of the vehicle reported by the odometry was erroneous and diverged from the ground truth, while the estimated pose values based on SLAM followed the ground truth much closer. This implies that the two observed lines play a very important role in improving the localization of the vehicle under the Extended Kalman Filter.

Figure 3.22: Localization Error

In particular, Figure 3.22 depicts the errors in vehicle heading, and in the x and y directions. The odometry and SLAM outputs have been compared with the ground truth of the vehicle. It can be observed that the SLAM errors are less than 0.1 radians and 0.2m in the heading, x and y directions respectively. While the errors in the odometry are much worse: up to 16m, 25m and 1 radian in x , y and heading respectively.

3.3 Line Features

Figure 3.23: Parameters of Line Feature 1

Figure 3.24: Parameters of Line Feature 2

Figures 3.23 and 3.24 show the plots of the estimated ρ and α of lines 1 and 2 from the SLAM output with respect to their ground truths. It shows that the estimated value of ρ and α of both features converged to their ground truth values.

Figure 3.25: The Packbot from iRobot.

3.3 Line Features

Implementation of line feature-based SLAM was carried on a packbot platform, a photograph of which is shown in Figure 3.25. An IMU and an LDOEM 1000 SICK laser scanner were mounted on the platform. The laser sensor can provide a 360 degree view scan of the environment. Tests were conducted in an indoor car park at the School of Physical and Mathematical Sciences, NTU. As shown in Figure 3.26, a scan of the car park was taken by the laser sensor. The Hough Transform was applied to the data, and extracted lines are shown in green and their normals shown in red. In this case, the threshold on the number of the inliers is set to 30 points.

Figure 3.26: Hough Transform: Line Extraction.

The localization of the packbot was derived based on forward velocity given by the odometry and angular rate change provided by the onboard IMU. The prediction

3.3 Line Features

of robot pose at time k is given as

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ \theta_{k+1} \end{bmatrix} = \begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix} + \begin{bmatrix} V_k \tau \cos(\theta_k + \gamma_k \tau) \\ V_k \tau \sin(\theta_k + \gamma_k \tau) \\ \theta_k + \gamma_k \tau \end{bmatrix} \quad (3.18)$$

where, x, y and θ are the x, y locations and the heading of the robot respectively, V_k and γ_k are the forward velocity and angular rate change, and τ is the time interval.

Figure 3.27: Line Extraction and Data Association

Data association was implemented in the real (non-simulated) SLAM application. The Nearest Neighbour method was adopted in the data association process due to its simple and easy implementation. The predicted and observed features are associated if they yielded the smallest Mahalanobis distance. As shown in Figure 3.27, a scan of the building was taken, and the Hough Transform was applied to the laser points to extract building walls and used as observations in the SLAM algorithm. The estimated building walls and their uncertainties are shown as red

3.3 Line Features

lines and red ellipses. The Mahalanobis distance for each pair of predictions and observations is calculated based on their uncertainty. They are said to be associated if they provide the smallest distance which is smaller than the given threshold. In this case, the threshold is chosen to be 4.6 corresponding to a 0.9 probability of correct association in Chi square tables.

Figure 3.28: Localization based on Odometry and IMU.

The trajectory of the vehicle was plotted and shown in blue line in Figure 3.28. Since the ground truth of the vehicle was difficult to obtain, a different approach was used to verify the performance of the vehicle model. In the figure, plots of the laser scans were superimposed with the pose information at the positions circled. It can be observed that segments corresponding to walls are not a single line but multiple lines. Their thickness is 1 to 1.5m, this indicates that the vehicle positions are erroneous, and the accumulated errors in x and y directions are approximately 1.0 to 1.5m. This is due to slip and uneven ground of floor, it leads to false information of the vehicle localization.

The Hough Transform was then implemented to extract lines corresponding to

3.3 Line Features

building walls, and they were used as observations in the line feature-based SLAM algorithm. Figure 3.29 depicts result of line feature-based SLAM algorithm. Vehicle positions were corrected by the SLAM algorithm and shown in red. Again, the scans at the indicated positions were plotted in magenta to verify the accuracy of the vehicle localization provided by the SLAM algorithm. It shows that thickness of the boundary is drastically, this indicates improvement of vehicle localization. The estimated lines corresponding to walls are shown green line, which are the final output from SLAM algorithm.

Figure 3.29: Localization based on Line Feature-Based SLAM.

Chapter 4

3D Laser Based Feature Extraction

The chapter introduces a 3D LADAR, which is an in-house built laser sensor capable of scanning 3D environments. The setup of the sensor is demonstrated and 3D data provided by the sensor is examined. Due to computational complexity of the existing feature extraction methodologies in [22], [23], [24] and [25], Principal Component Analysis (PCA) [42] and [43] is implemented to extract planes and lines from the point clouds. However, in order to enhance the extraction procedure based on the PCA method, it is necessary to segment or trim the data first.

4.1 3D Scanning LADAR

LADAR stands for LAser Detection And Ranging. As shown in Figure 4.1, it is an in-house built laser scanning system. The setup for this 3D LADAR is quite different from the one used in [24], [26] and [25]. The system consists of a Riegl laser scanner at the base covered by metallic housing, and a rotating frame on the top. A mirror is attached to the rotating frame, which can rotate about its horizontal axis, which make the system able to scan the environment vertically. The rotating frame itself is controlled by a servo motor, this enables the 360-degree horizontal scan of the

4.1 3D Scanning LADAR

Figure 4.1: Built-in-House 3D LADAR.

system. Both motors come with a built-in encoder. By feeding back the encoder signal, the bearing and elevation angles of the mirror can be calculated.

By trial and error, the number of encoder counts per revolution can be determined. In this case, N_{cpr} represents the number of counts per revolution of the bearing frame, and N_{cc} is the number of the current count, thus the bearing angle can be determined as the following:

$$\alpha = \frac{360N_{cc}}{N_{cpr}} \quad (4.1)$$

In addition, there is an internal binary switch to report the initial position of the rotation frame. This binary switch signal is used to trigger the reset of the encoder count, so that the count error is not accumulated.

The servo motor controlling the mirror also reports its elevation angle. Due to the limitations of the mechanical design, the mirror can rotate within -30° to $+30^\circ$ with respect to its horizontal axis. The range given by the laser can reach up to

4.2 3D LADAR Point Cloud

150m.

Therefore, the system can perform 3D scans with desired angular resolution by controlling the rotational speed of both motors. Any point $P(x, y, z)$ in 3D space can be represented as $P(\rho, \alpha, \beta)$, where ρ, α, β are the distance from the sensor's origin, bearing angle and elevation angle respectively, as shown in Figure 4.2, and ρ, α and β are provided by the 3D LADAR.

Figure 4.2: 3D Representation of a Point.

4.2 3D LADAR Point Cloud

An illustration of a scan using the 3D LADAR is shown in Figures 4.3a and 4.3b. A laser beam is emitted and reflected from the environment by the mirror. A receiver is then able to detect the returning beam and calculate the distance of the target based on its time of flight. By facing the mirror at different bearing and elevation angles, the environment can be scanned and it can be represented by point cloud

4.3 Principal Component Analysis (PCA)

data.

For each point, the Riegl laser scanner gives its range. The bearing and elevation angle of the mirror are given by the encoders. For each elevation angle, a full horizontal 360 degree scan is performed. The environment is then represented by batches of horizontal planes at each elevation angle. Figures 4.3a and 4.3b depict point clouds of NTU's School of Communication and a car park. The resolution of the scan of the environment depends very much on both bearing and elevation angle resolutions. However, in order to attain a high resolution, the rotation speed of the frame needs to be slow and more scans need to be recorded, thus the scanning time increases which is not desired for a real time system.

Figure 4.3: 3D Point Cloud of Outdoor Scans.

4.3 Principal Component Analysis (PCA)

PCA is a tool used to identify patterns in the given data, and express it in such a way as to highlight their similarities and differences [43]. Since 3D structures in the environment are complicated and not identical, finding structural patterns is a challenging and complicated task. PCA is a powerful tool and allows for the

4.3 Principal Component Analysis (PCA)

possibility of handling such tasks. Moreover, based on the information from PCA on the data, the number of dimensions can be reduced without much loss of information [43]. The formulation as well as some simulated examples of PCA are shown in the following section.

4.3.1 PCA Applied to 2D Range Data

Assuming a 2D set of data $X = \{x_i, y_i\}$ with $i = 1, 2, \dots, n$, the means x_m and y_m of the data set of each coordinate value, are given as:

$$x_m = \frac{\sum_{i=1}^n x_i}{n}$$

$$y_m = \frac{\sum_{i=1}^n y_i}{n}$$

where n is the number of points per 2D scan.

By subtracting the relevant mean from each data, point variance and covariance terms can be calculated as:

$$\text{cov}xx = \frac{\sum_{i=1}^n (x_i - x_m)(x_i - x_m)}{n-1}$$

$$\text{cov}xy = \frac{\sum_{i=1}^n (x_i - x_m)(y_i - y_m)}{n-1}$$

$$\text{cov}yx = \frac{\sum_{i=1}^n (y_i - y_m)(x_i - x_m)}{n-1}$$

$$\text{cov}yy = \frac{\sum_{i=1}^n (y_i - y_m)(y_i - y_m)}{n-1}$$

Where $\text{cov}xx$ and $\text{cov}yy$ are the variance of x and y variables respectively. $\text{cov}xy$ and $\text{cov}yx$ are the cross covariance between x and y variables.

4.3 Principal Component Analysis (PCA)

Then, the covariance matrix of the data can be written as the following:

$$C = \begin{bmatrix} \text{cov}xx & \text{cov}xy \\ \text{cov}yx & \text{cov}yy \end{bmatrix}$$

Finally, the eigenvalues of the covariance matrix can be determined by solving the following equation:

$$CY = \lambda Y \quad (4.2)$$

Since $Y = 0$ is the trivial solution, it thus requires:

$$\det(C - \lambda I) = 0$$

where I is identity matrix and λ are the eigenvalues

The eigenvectors can be calculated by substituting the eigenvalues (λ_1 and λ_2 in this case) into Equation 4.2: Figure 4.4a shows an implementation of PCA on

Figure 4.4: Implementation of PCA on 2D data.

a simulated data set. There are 100 points in this data set. x_i increments from 1 to 100m with a step of 1m, and $y_i = 2 * x_i + 10 + 20(\text{rand}() - 0.5)$, where $\text{rand}()$ generates random numbers between 0 to 1 and $i = 1, 2, \dots, 100$. This function is set up to simulate data which is approximately co-linear, perturbed by random noise.

4.3 Principal Component Analysis (PCA)

The means of the data are found to be $x_m = 50.50$ and $y_m = 110.64$. The eigenvalues are $\lambda_1 = 6.6$ and $\lambda_2 = 4218.5$. With orientations given by the eigenvectors $v_1 = [-0.89 \ 0.44]^T$ and $v_2 = [0.44 \ 0.89]^T$, the covariance ellipse can be determined and plotted as shown in the figure. The semi-axes of the covariance ellipse are obtained by taking the square root of the eigenvalues. In this case, the ratio of major axis length over minor axis length is 25.27, this means that points are spreading along the v_2 direction, and the ratio is big enough to assume that points are forming an approximate straight line, and the data can be transformed into a 1D space.

On the other hand, points from a data set may completely scatter over the given space. Figure 4.4b depicts the case where points are completely random in a 2D space. There are 100 points in the set, each point is randomly generated with $-50m < x, y < 50m$. The means are found to be $x_m = -0.55$ and $y_m = 0.37$. The minor axis length and major axis length are 25.74 and 27.03 respectively. Thus, their ratio is almost 1, this implies that the points completely scatter the whole space given, and no underlying pattern of data structure (feature) can be extracted.

4.3.2 PCA Method Applied to 3D Range Data

Similarly to the 2D case, the procedure of calculating the variances and covariance matrices is the same in the 3D case. However, an additional variable z_i is introduced into the equation, and covariance matrix is now a 3-by-3 square matrix with elements shown as follows:

$$C = \begin{bmatrix} \text{cov}xx & \text{cov}xy & \text{cov}xz \\ \text{cov}yx & \text{cov}yy & \text{cov}yz \\ \text{cov}zx & \text{cov}zy & \text{cov}zz \end{bmatrix} \quad (4.3)$$

For demonstration purposes, Figure 4.5a shows a helical plot of a data set. The axis lengths of the data, which are the square root of the eigenvalues of C , are found to be 9.09, 35.35 and 35.43. The ratios between the axis lengths are not significantly

4.3 Principal Component Analysis (PCA)

Figure 4.5: Implementation of PCA on 3D data.

large. Thus, it implies that the points are spreading randomly over the given space. This is because the helix is not confined to any particular lines of planes in 3D space. In figure 4.5b, the data is arranged as a sine wave propagating on the plane $z = 2.5x$. The means of the data are found to be $x_m = 50.5m$, $y_m = 0.85m$ and $y_m = 15.25m$, and 0.0, 14.0 and 32.58 for the axis lengths of the covariance ellipsoid. This implies that the 3D data set can be reduced to 2D without loss of any information since one of the axis lengths is 0, much smaller than the other two. The other implication of the data set is that the points be on a plane whose normal given by the eigenvector corresponding to the smallest eigenvalue.

Figure 4.6 depicts the application of the PCA method on a noisy data set. First, a perfect plane is shown in Figure 4.6(a), which was arbitrarily generated with the plane equation $2x - y - z + 5 = 0$, where $0 < x < 20$ and $0 < y < 20$. Then, random noise with zero mean and variance of 1 square unit were generated and added onto the data set, and the result is shown in Figure 4.6(b). Figures 4.6(c) and 4.6(d) illustrate the side view of cases (a) and (b). This shows the effect of the random

4.3 Principal Component Analysis (PCA)

Figure 4.6: Plane Extraction with PCA. (a) Perfect data set (b) Noisy data set (c) Side view of (a) (d) Side view of (b).

noise on the data set, which makes some points no longer lie on the plane. The PCA method was then applied on the perfectly coplanar and noisy data sets. In case of perfectly co-planar data, one of the eigenvalues is 0, which implies that the data set is perfectly distributed on a plane. On the other hand, the smallest eigenvalue in the noisy data case is no longer 0 due to the random noise added to the data. However, the ratios of the eigenvalues are 1%, 26.5% and 72.5% respectively, so those points can still be considered as lying on the same plane. This implies that the noise sensitivity of PCA is less significant, and the method is still able to extract planes from data under limited random noise situations.

The case of coplanar, plane segments has also been evaluated as shown in Figure 4.7. In all 3 figures, there are two plane segments and each segment contains

4.3 Principal Component Analysis (PCA)

Figure 4.7: PCA on Collinear Planes. Two collinear planes with (a) no translation in z direction (b) 1 unit translation in z direction (c) 10 unit translation in z direction.

50 points, and the equation of the right hand segment is $x - y + z = 0$, where $0 < x < 10$ and $0 < y < 10$. In Figure 4.7(a), the equation of the planar segment on the left is $x - y + z = 0$, where $-10 < x < 0$ and $-10 < y < 0$. After the application of PCA on the data set, one of the eigenvalues is found to be 0. Thus, those 100 points are distributed on a plane. In Figure 4.7(b), the planar segment on the left is now translated in the z direction by -1 units. The eigenvalues, in this case, are determined to be 0.2, 4.9 and 8.3 respectively, and the ratio corresponding to the smallest eigenvalue is 1.5%. According to the criteria on Page 78, Section 5.4, those 100 points are considered to be randomly distributed over the space. This implies that if the segment was translated slightly less than 1 unit in the z direction, the two segments would be extracted as one single plane. On the other hand, the left one is translated by -10 units in Figure 4.7(c). The eigenvalues are 1.5, 4.2 and 12.7 respectively, and the smallest ratio is 8.1%. The result shows that the two planes cannot be merged together as one single plane, since the distance between them is significantly large.

The application of PCA on two non-coplanar segments is further examined as shown in Figure 4.8. The segment on the top left is given by $x + y - z = 0$, where $0 < x < 10$ and $0 < y < 10$, while the equation of the other segment is

4.3 Principal Component Analysis (PCA)

Figure 4.8: PCA on Two Planes. Two Planes with dihedral angle of (a) 0 degrees (b) 2.6 degrees (c) 11.4 degrees (d) 19.5 degrees.

$x + my - z = 0$, where $0 < x < 10$, $-10 < y < 0$ and m is a parameter to be adjusted experimentally. The parameter m was chosen to be 1.0, 1.1, 1.5 and 2.0 in Figures 4.8(a), 4.8(b), 4.8(c) and 4.8(d) respectively. The dihedral angles which is the angle between the normals of the two segments are found to be 0, 2.6, 11.4 and 19.5 degrees respectively. According to the PCA method, the smallest ratios in the 4 cases are 0%, 0.67%, 2.8% and 4.7% respectively. Thus, a plane corresponding to both segments would be extracted in the first two cases. This also implies that the PCA method fails to extract any planes in the cases with the presence of multiple segments which are not approximately coplanar.

Figures 4.9a, 4.9b and 4.10 show the implementation of PCA on real data given by the 3D LADAR. The axes are the three axes of the ellipsoid of the covariance

4.3 Principal Component Analysis (PCA)

Figure 4.9: Implementation of PCA on Real Data.

Figure 4.10: Point Cloud of South Spine.

matrix from Equation 4.3. In the first figure, the axis lengths are 0.905, 1.07 and 2.37 respectively. So the point cloud is approximately evenly spread over the three direction of the axes. Similarly, the point cloud in the second figure also spreads

4.3 Principal Component Analysis (PCA)

approximately evenly over the space. However, if an appropriate portion of the point cloud is removed from the second figure, this results in Figure 4.10, where the axis lengths are 2.0, 7.01 and 17.01 respectively. One of the axis lengths is comparably small to the other two, which indicates a 2 dominant directions in the point cloud data. This implies that 3D point cloud can be transformed as a 2D plane.

This indicates that the PCA method can examine the data set in a very efficient way, since it does not perform any iteration process. Furthermore, the PCA method also provides information of the normals of a planes, which can be important features for robot navigation. However, the drawback is the method does not provide much useful information when applied to raw data sets, since most of the environments consist of complex structures not just single planes or lines. Therefore, segmentation of the data is necessary before applying PCA method on the data. This will enhance the performance of the method. This is the subject of chapter 5, in which color information from a camera is fused with the 3D point cloud data, to first segment it into more suitable regions in which PCA can be applied with greater success.

Chapter 5

Visually Aided Feature Extraction

5.1 Introduction

In the previous chapter, it was shown that if PCA is applied to typical complete 3D point cloud data sets, then dominant data directions are unlikely to be found, as many planes, lines etc may be present within the point cloud. In order to improve feature extraction with the PCA method, the data needs to be segmented or trimmed first. Therefore, a vision system has been integrated on top of the existing system. In addition to the range information provided by the laser, the color information can be obtained from the camera. The camera system is shown in Figure 5.1. After fusion of both sets of sensor data, the system is able to produce range and color information for each point of the point cloud data set.

Visual information is obtained from stitched images produced by panoramic camera through visual processing. The image is first captured by the camera, which is then segmented to divide the image into smaller regions. Much research work has been carried out towards image segmentation, such as: Smallest Univalued Segmentation Assimilating Nucleus (SUSAN) [44] which produces result for gray scale image. However, the color image needs to be converted to gray scale and some information is lost during the conversion. Anisotropic diffusion [47] is another image smoothing

5.2 Introduction

Figure 5.1: The Ladybug Camera consisting of 5 horizontally facing cameras, and one vertically facing camera.

method and prevents smoothing across edges by performing the averaging in the orthogonal direction to the gradient of the luminance. This is important to preserve the accuracy of the segment edges. Bilateral filtering [45] is a non iterative scheme for edge preserving smoothing. The Mean shift algorithm [48] and [49], which is a non-parametric estimator of the density gradient in the spatial-range domain, is a useful tool for edge preserving, filtering and segmentation on both grayscale and color images. The Mean shift algorithm is a clustering technique which does not require prior knowledge of the number of clusters.

SUSAN, bilateral filter and mean shift algorithms were selected to filter and smooth the images captured by the camera. The performance of each methods will be shown, as well as a comparison of the algorithms in term of smoothing and edge preserving aspects. It will be shown that the Mean shift algorithm, in this case, performs better than the other two algorithms in term of edge preserving. Since edges and boundaries play an important role in image segmentation, the Mean Shift algorithm is used as the segmentation tool in our work.

5.2 Integration of 3D LADAR and Vision

5.2 Integration of 3D LADAR and Vision

Multi-sensor fusion has become very popular, because different sensors possess different characteristics, abilities, advantages and disadvantages. Thus, fusion of data from different sensors adds more value or information to the data. Different types of sensors have been combined together to provide appropriate data fusion for certain applications. A laser and a camera have been integrated together in [35], [37] and [38], in which range and visual data provided by the system is used for 3D SLAM applications, and object and people detection. [40] replaced the laser with a Radar, and the integrated system is used for vehicle recognition purposes.

Figure 5.2: LADAR and Vision Integrated System.

Our system, on the other hand, comprises a 3D LADAR and a Ladybug camera. The Ladybug is mounted on top of the 3D LADAR as shown in Figure 5.2, the vertical axes of both sensors are overlapping with each other. The camera is able to produce a stitched image to give a 360 degree field of view of the environment. In order to use information from both sensors, the problem of finding correspondents needs to be solved, so a calibration between the two sensors is necessary.

5.2 Integration of 3D LADAR and Vision

5.2.1 Ladybug Panoramic Camera

The Ladybug, as shown in Figure 5.1 is a spherical digital video camera system which is capable of recording images and videos. It consists of 5 cameras at the side of the cylinder covering 360 degrees, and another camera on the top. 6 images are produced simultaneously. An image stitching algorithm has been developed to merge the 6 images together and generate a stitched image to provide 360 degree view. The process is to find the intrinsic and extrinsic parameters of the camera, so that the overlapping region of each camera can be determined and aligned to stitch the six images together. Figure 5.3 depicts an image of School of Communication (NTU) after the stitching process.

Figure 5.3: School of Communication.

5.2.2 3D LADAR - Camera Calibration

With reference to the calibration model described in [39], the calibration of our integrated system has been carried out as follows:

As shown in Figure 5.4, a set up scene was built to calibrate the integrated system. Card board sheets and standing poles were used in the set up as points of interest, which makes it easy to find the correspondents between camera image pixels and laser point cloud data from the same object. The idea is to find the image pixel that corresponds to a given 3D point cloud, which both range and visual information

5.2 Integration of 3D LADAR and Vision

Figure 5.4: Calibration of The Integrated System.

are obtained from the same point in the real world.

5 points which are corresponding to the 4 corners and the center of an image are suggested by the camera manufacturer to take as reference points in the calibration process. Due to constraints on the elevation angle of the laser beam, an adjustment of the reference points is made accordingly. Then, a transformation matrix of the 3D LADAR point cloud to image pixel is to be determined. Due to the spherical projection property of the stitched image, two important parameters for each point need to be known. Those are the bearing and elevation angles.

Bearing Angle

Since the vertical axes of both sensors is overlapping with each other, the relationship between their bearing angles is just an offset. The offset between both sensors is determined by trial and error.

$$\alpha_{sp} = \alpha + \alpha_c \quad (5.1)$$

where α_{sp} is the bearing angle in the spherical projection, and α_c is constant

5.2 Integration of 3D LADAR and Vision

Elevation Angle

The derivation of elevation angle relationship between spherical projection of the image and laser beam is shown in section 5.4.1. The important results of this derivation are used below to solve the transformation matrix between image pixel coordinates and laser range, bearing and elevation coordinates. Based on a small angle approximation assumption, the relationship between the two is derived as follows:

$$\beta_{sp} = \pi/2 - \beta + d/\rho \quad (5.2)$$

where β_{sp} is the elevation angle in the spherical projection system, d is the spacing distance between the LADAR and the camera, and ρ is the range information provided by the LADAR.

By applying equations 5.1 and 5.2 according to the size of the image, the pixel index of a given point from the LADAR obeys the following function:

$$\begin{bmatrix} x_i \\ y_i \end{bmatrix} = f(\rho, \alpha, \beta) \quad (5.3)$$

where x_i, y_i are x and y index values of the pixel, ρ, α and β are the range, the bearing and elevation angle of the given 3D point.

By substituting the values of x_i and y_i found in the image, the pixel value, the color intensities in this case, is obtained,

$$\begin{bmatrix} r_i \\ g_i \\ b_i \end{bmatrix} = I(x_i, y_i) \quad (5.4)$$

where r_i, g_i and b_i are red, green and blue color value of the pixel.

The fusion module can be illustrated as in Figure 5.5. The process is repeated for each range data sample. By fusing the color values of each pixel with the 3D point cloud, colored 3D scans are obtained as shown in Figures 5.6 and 5.7. Since the

5.2 Integration of 3D LADAR and Vision

Figure 5.5: Range-Color Fusion Module.

calibration is not perfect, some points are indexed with the incorrect color, this is due to errors in the transformation matrix. However, in term of overall performance, the fusion gives a better visualization of the environment. Trees, walls, posters, objects and buildings possess textures and colors as their appearance in the real world. Moreover, due to color information which is an addition attribute of the objects, feature identification can be easily and consistently achieved.

Figure 5.6: Colored 3D Point Cloud of School of Communication.

5.3 Image Filtering and Segmentation

Figure 5.7: Colored 3D Point Cloud of ME Workshop.

5.3 Image Filtering and Segmentation

5.3.1 Image Filtering

As shown in Section 4.3.2, segmentation is paramount importance before the application of PCA on the range data. Here, segmentation corresponds to smoothing the image.

Image smoothing blurs the image to make the color intensity of a single object as uniform as possible. This improves the quality of image segmentation process. Pixels which belong to the same object have almost the same color, so they are clustered into the same group. Segmentation can be done without image smoothing, but an image of a single object will then possibly be segmented into many regions due to unavoidable slight color variations due to lighting or non-uniform texture. Thus, the extracted planes would either be incomplete or erroneous. With better image segmentation, the point cloud clustering performs better. The higher the number of range points associated with a segment, the more accurate the plane or line extracted with the PCA method will be.

Three filtering methods have been implemented on the stitched image produced

5.3 Image Filtering and Segmentation

by the Ladybug, and the performance of each method has been examined. The filtering algorithms are discussed below:

The SUSAN smoothing algorithm

Smallest Univalue Segmentation Assimilating Nucleus SUSAN [44] is an effective but easy smoothing technique which has been implemented in many applications. A circular mask with constant size is selected as a window, and the brightness of each pixel is compared with the Nucleus (the center pixel of the mask). The comparison is carried out within all pixels of the image, and regions are smoothed with the value of the Nucleus but it preserves the image structures (edges).

Figure 5.8: 37-Pixel Circular Mask.

A mask of 37 pixels as shown in Figure 5.8 was chosen. All the numeric figures used in this algorithm and mask size are based on work of [44]. The center of the mask shifts from one pixel to another till covering the whole image. For each placement of the mask, the following parameters are calculated based on the local pixel RGB values:

- $c(x, x_o) = e^{-\left(\frac{I(x) - I(x_o)}{t}\right)^6}$, where x is current pixel, x_o is center pixel of the

5.3 Image Filtering and Segmentation

mask, function $I(.)$ returns the intensity value of the pixel and threshold t determines the minimum contrast of edges which will be picked up, and was found to be 27 for both images, which gave minimum contrast of edges.

- $g = \sum c(x, x_o)$, which is the area USAN
- $nmax$ is the threshold for the area USAN and equal to $3*37/4$, which is three-quarter of the mask
- $R(x_o) = \begin{cases} g - nmax & \text{if } g > nmax \\ 0 & \text{otherwise} \end{cases}$

$R(x_o)$ is the edge response of the pixel where the center of the mask is. By repeating the above steps for each image pixel, an image of edge response is constructed.

The Bilateral Filter

The Bilateral Filter[45] is also a smoothing and edge preserving technique which is non iterative and simple. The idea is to combine range and domain filtering. The CIE-Lab color space [46] is the perceptual metric used in the filter, because it endows the space of the colors with highly perceptual measure of color similarity. Two functions need to be determined in the algorithm. Those are closeness function $c(\delta, x)$ and similarity function $s(\delta, x)$, which are Gaussian functions of the Euclidean distance between their input arguments. Function c is considered as Gaussain domain weight and defined as follows:

$$c(\delta, x) = e^{-\frac{d^2(\delta, x)}{2\sigma_d^2}} \quad (5.5)$$

where $d^2(\delta, x)$ is the Euclidean distance between two points, and σ_d is the geometric spread.

5.3 Image Filtering and Segmentation

Similarly, the function s is given below:

$$s(\delta, x) = e^{-\frac{\phi^2(\delta, x)}{2\sigma_r^2}} \quad (5.6)$$

where $\phi(\delta, x)$ is the difference of the pixel light intensity, and σ_r is the spread, which is the measurement spread of the intensity of light.

Then the product of the two functions is calculated. It is known as the response of the filter. The dc component of the low-pass signals is the sum of the responses of the filter, which is defined as follows:

$$k(x) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} c(\delta, x) s(\delta, x) d\delta \quad (5.7)$$

And finally, the output of the filter can be calculated by using the following equation

$$h(x) = k^{-1}(x) \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(\delta) c(\delta, x) s(\delta, x) d\delta \quad (5.8)$$

where $h(x)$ is filtered image and $f(x)$ is the input image.

The Mean Shift Algorithm

Figure 5.9: Mean Shift Algorithm

The Mean Shift algorithm [48] and [49] makes use of a Kernel density estimation technique known as the Parzen window technique, which is the most popular density

5.3 Image Filtering and Segmentation

estimation method, to determine the convergent centroid of the window. The idea is to find the local maxima of a probability density. Figure 5.9 illustrates the procedure of the algorithm. The algorithmic steps can be summarized as follows:

- Choose a radius of the search window
- Start with a random region of interest
- Determine the centroid of the data
- Move the search window to the location of the new centroid
- Repeat until convergence

Since the image is the combination of spatial and range (color) spaces, the Mean Shift method takes both into account. The algorithm manages two low-level vision tasks, smoothing with preserving the edges and image segmentation. Furthermore, gray level or color images are acceptable as inputs. To apply the Mean Shift algorithm to an image, for each $j = 1 \dots n$, where n is the number of the image pixels:

- Initialize $k = 1$
- Initialize a new variables $y_k = x_j$, for each image pixel x_j , where x_j is a vector containing the spatial and range (color) information of pixel j .
- Compute $y_{k+1} = \frac{1}{n_k} \sum_{x_i \in S_1(y_k)} x_i$, where $S_1(y_k)$ is a window with radius 1 and centered on y_k , and there are n_k points are in the window.
- Increment k , compute y_{k+1} till convergence by checking $\|y_{k+1} - y_k\| < \exp^{-3m}$ with m is the greater value between h_r and h_s .

5.3 Image Filtering and Segmentation

- Assign $z_i = (x_i^s, y_{conv}^r)$.

The last assignment implies that image pixel at x_j has the range (color) components of the point of convergence y_{conv}^r .

After the filtering process, a clustering process takes place which tries to group the pixels in a small region if the difference of pixel value between two pixels is below the given threshold h_r and their spatial difference is below the value of h_s . Thus, the flow of the clustering procedure can be described as the following:

- Filtered image is produced by mean shift filtering algorithm
- Determine the clusters $\{C_p\}_p = 1, \dots, m$ by grouping all z_i , which are closer than h_s in the spatial domain and h_s in the color space
- Assign labels to each clusters
- Clusters which have less number of pixels might be eliminated

Result and Discussion

In the implementation of the image filtering process, the parameters of each filter were chosen according to their characteristics. The filter parameters of the SUSAN filter described in Section 5.3.1 were used. In the Bilateral filter, a window size of 5x5 was selected, spatial and range (color intensity as a floating number between 0 to 1.0) threshold were set to 3 and 0.1 respectively. In the Mean Shift algorithm, the spatial and range (color intensity as an integer between 0 to 255) were 8 and 7 respectively, and M (the minimum cluster size) was set to 100 pixels. Since with these settings, each filter produced comparably good result with moderate computational time.

5.3 Image Filtering and Segmentation

Figure 5.10: Original Image of School of Communication.

Figure 5.11: SUSAN Filtering Result of School of Communication.

Figure 5.12: Bilateral Filtering Result of School of Communication.

Figure 5.13: Mean Shift Filtering Result of School of Communication.

5.3 Image Filtering and Segmentation

Figure 5.14: Original Image of ME Workshop.

Figure 5.15: SUSAN Filtering Result of ME Workshop.

Figure 5.16: Bilateral Filtering Result of ME Workshop.

Figure 5.17: Mean Shift Filtering Result of ME Workshop.

5.3 Image Filtering and Segmentation

In order to examine how the lighting factor can affect the filtering process of the three methods implemented, indoor and outdoor environments were examined separately. The first batch of results (see Figures 5.11, 5.12 and 5.13) are from the outdoor scene. The system was setup in front of the NTU communication school. It was recorded at noon and under a clear sky. The other scene was inside the NTU Autonomous Robotics Research Laboratory (see Figures 5.15, 5.16 and 5.17). The room was subjected to artificial lighting.

The results, in Figures 5.11 and 5.15 show that the SUSAN filter performed well in smoothing the regions and preserved most part of the edges. However, some edges were smoothed and they became broken lines. Furthermore, the SUSAN method doesn't take into account the color information of the image, since the image was first converted into gray scale before the filtering process. This leads to loss of information and the merging of some regions which are not from the same objects.

Results produced by the Bilateral filter and the Mean Shift algorithms are comparably similar, but each filter possesses its own unique characteristic and difference, which can be observed in the results.

Figure 5.18: Bilateral vs Mean Shift Filtering: Regions Smoothing.

In terms of smoothing, the Bilateral Filter performed better than the Mean Shift filter. Figure 5.18 shows a filtered region from the image for both algorithms. It can be observed that the smoothed region given by Bilateral filter is almost uniform,

5.3 Image Filtering and Segmentation

while the Mean Shift method produced a less uniform smoothed region. However, the result from the Mean Shift algorithm can be improved by adjusting the threshold of the spatial and the color spaces, with a cost of an increase in computational time.

Figure 5.19: Bilateral vs Mean Shift Filtering: Edges Preserving.

However, in terms of their edge-preserving aspects, the edges are very sharp and can be observed clearly in the filtered image from the Mean Shift algorithm, while the edges from the Bilateral filter are blurred.

Unlike many other algorithms, the Mean Shift takes both spatial and color information into account. The method determines the gradient of color change within a window whose size is determined by the spatial and color resolutions. The idea is to find the local maxima of a probability density and to determine the convergent centroid of the window. The convergence is obtained if the norm of the difference between the spatial and color values of the same pixel at steps n and $n - 1$ is less than a threshold which is an exponential function of the spatial and color resolutions. This yields good results in the edge preservation of the image. As a result, object boundaries in the image, filtered with the Mean Shift algorithm, are very sharp and well preserved.

Since the edges play a very vital role in the segmentation process, so the Mean Shift algorithm was chosen to perform the image filtering and segmentation.

5.3 Image Filtering and Segmentation

5.3.2 Segmentation

The segmentation of range data is a challenging and computationally intensive procedure because 3D point clouds can contain millions of points. An image is a projection of the 3D environment onto 2D plane. Thus, performing segmentation in 2D space can be computationally simpler than in 3D space. Segmentation carried out in the image space is relatively simple since colors provide extra valuable information. The image is segmented into regions based on the similarity of the color intensity, and one block of a segmented image has high probability to represent one object in the environment.

After image filtering, a clustering process takes place. The aim is to segment the image into small regions according to the closeness of spatial and color range information after the filtering process. This is so that the corresponding laser point cloud data can be extracted for PCA to be applied. An algorithm is applied to find the connectivity between neighborhood pixels. Two pixels are considered as connecting to each other if their difference in spatial and range (color) value is less than the given threshold (h_s and h_r). A label index is assigned to the cluster to differentiate one from another. Then, based on this cluster index, the range data is segmented and PCA is applied to determine if the data forms a line, a plane or the data just randomly scatters over the given space. Thus, the process can be illustrated as the flow chart shown in Figure 5.20.

The clustering process was carried out with implementing *bwlabel()* in MatLab. *bwlabel()* is a function to cluster the connected pixels together and label them. First, a connectivity matrix is initialized as shown in Figure 5.21. It is a binary matrix. If m and n are the width and height of the given image, the connectivity matrix is initialized with the size of $(2m + 1) \times (2n + 1)$. Binary 1 is assigned to locations

5.3 Image Filtering and Segmentation

Figure 5.20: Visually Aided Range Segmentation.

Figure 5.21: Initialization of the Connectivity Matrix.

$\{2i, 2j\}$ for all $i = 1, 2, \dots, m$ and $j = 1, 2, \dots, n$. This represents every alternate pixel of the image. Binary 0 is assigned to the remaining cells of the matrix. The left, right, top and bottom cells of each binary 1 will contain the information between the image pixel with its left, right, top and bottom neighbors respectively. Binary 1 is assigned to the cell if the difference for both spatial and range value between the two cells is below the given threshold (h_s and h_r are 8 and 7 respectively), and binary 0 otherwise. Figure 5.22 is an example of the final output of the connectivity

Figure 5.22: The Final Connectivity Matrix and Labeled Clusters.

5.3 Image Filtering and Segmentation

matrix. After all binary 1s and 0s are filled in the matrix based on the comparison made between each pair of adjacent pixel values, the connectivity test takes place to group all the pixels which connect to each other. In Figure 5.22, three clusters are formed. In order to facilitate range data segmentation later, the color value of each pixel is replaced by the cluster label.

In the Mean Shift algorithm, the clustering is constrained by another parameter M . Clusters whose number of pixels fall below the value of M are eliminated. In this case, the value of M is set to 100 pixels. Label value 0 is assigned to pixels which do not belong to any cluster.

Figure 5.23: Mean Shift Filtering and Segmentation

Figure 5.23 depicts the result of Mean Shift filtering and segmentation. It highlights one section of the image undergoing filtering and segmentation. Different clusters are visualized by using different colors as shown in Figure 5.24.

Figure 5.24: Image Segmentation.

It can be observed that some regions (shown in the original image color) didn't

5.3 Image Filtering and Segmentation

belong to any clusters. This is because these regions were assigned a number of pixels less than the value of M . In Figure 5.23, it is clear that there are two clusters and a stripe (horizontally in the middle of the highlighted boxes in Figure 5.23) which doesn't belong to any of the two clusters. Figure 5.25 depicts the cluster map corresponding to the cut region of the image. Three values appears in the map. Numerical figures 38 and 40 represent two clusters respectively, where 0 represents those pixels not belonging to any clusters in the map.

Figure 5.25: Segmentation Labeling.

Fusion between range and color then takes place again. The image after this segmentation process is used to segment range data which belong to the same cluster. To make the process easier, cluster labels are attached to each range data to tell which object that particular range data belongs to. Figures 5.26 and 5.27 demonstrates laser point cloud data taken in outdoor and indoor scenes, and their respective color segmentation. Points which have the same color are in the same cluster. The different colors highlight estimated different clusters or objects in the environment.

Figure 5.28 depicts an example of imperfect segmentation with the Mean Shift algorithm. The glass panels were exposed to direct lighting, and there is a sharp intensity change on them. As a result, The Mean Shift segmented a single panel

5.3 Image Filtering and Segmentation

Figure 5.26: Segmentation of Range Data (Communication School).

Figure 5.27: Segmentation of Range Data (ME Workshop).

Figure 5.28: Mean Shift Filtering and Image Segmentation

into few regions due to huge differences in the intensity values of neighboring pixels. Therefore, point clouds corresponding to a particular panel were also segmented into several clusters. This should produce several planar segments which are co-planar.

5.4 Implementation of PCA on Segmented Range Data

However, if the pixel number of each region is less than the given threshold, those pixels are not clustered as one region. This implies that some panels are missed in the extraction process. Furthermore, the accuracy of the plane (in the final output of the extraction process) can be affected, because there are less points corresponding to each segmented region and they are subjected to sensor noise.

5.4 Implementation of PCA on Segmented Range Data

After range segmentation has been achieved, the final step is to carry out the application of PCA on the segmented point cloud data as shown in Figure 5.20. Figures 5.26 and 5.30 depict the result of the implementation of PCA on the segmented range data for the outdoor and indoor environments respectively. The range data is segmented according to the number of the clusters in the image. The cluster index number tag is attached to each range data point to identify which cluster that particular range data belongs to. Moreover, the range data can now be sorted based on the cluster index number. For each cluster, the range data undergoes PCA to determine whether the data belongs to a plane, a line or is randomly distributed over the, now reduced, segmented space.

For each point cloud cluster, their mean and covariance are calculated. Then, the eigenvalues of the covariance matrix are obtained, since eigenvalues play very important role to highlight the vital information of the data set. In Figures 5.29 and 5.30, the bold red lines in each segmented range image represent the axes of the ellipsoid of the covariance matrix. There are three axes, because we are dealing with 3D data set. In this particular case, a ratio test is carried out for each set of Eigenvalues. Assuming λ_1 , λ_2 and λ_3 be the Eigenvalues of the data set, their ratios

5.4 Implementation of PCA on Segmented Range Data

are defined as the following:

$$T_i = \frac{\lambda_i}{\lambda_1 + \lambda_2 + \lambda_3} \quad (5.9)$$

where $i = 1, 2, 3$

The ratios determine the categories which the data set belongs to. If one of the T_i is smaller than the given threshold T , it indicates that the data set is estimated to be on a 2D plane. The data set is assumed to be approximately co-linear if there are two T_i smaller than T . The data is assumed to be randomly scattered over the given space if non of the T_i are smaller than T , meaning that no features are extracted.

For the outdoor scene, 12 sets of segmented range data corresponding to 12 clusters of the image are found, and 9 segments are found in the indoor scene. Each set of point cloud data was examined by PCA method. The threshold T is set to 1%, and the results of outdoor clusters 1 and 10, and indoor cluster 5 (Label index is shown on the corner of each segmented range images) are as shown as follows:

5.4 Implementation of PCA on Segmented Range Data

Figure 5.29: Result of PCA on Segmented Range Data of Outdoor Scene.

5.4 Implementation of PCA on Segmented Range Data

Figure 5.30: Result of PCA on Segmented Range Data of Indoor Scene.

5.4 Implementation of PCA on Segmented Range Data

Figure 5.31 Outdoor Cluster 1

Figure 5.31: Disperse Data Set.

The Eigenvalues λ_1, λ_2 and λ_3 of the covariance matrix are 8.205, 80.40 and 176.198 respectively.

By applying equation 5.9, the ratios are $T_1 = 3.09\%$, $T_2 = 30.29\%$ and $T_3 = 68.72\%$. Since all the ratios are greater than threshold $T = 1\%$, this leads to the conclusion that the data is neither lying on a plane nor on a line, it is just random in the given space. This makes sense, since that section of the image shows trees, their shadow on the floor and the nearby building, which cannot be represented by lines or planes.

Figure 5.32: Outdoor Cluster 10

The Eigenvalues of this data set are 0.027, 7.408 and 23.423, and their ratios are 0.09%, 24.01% and 75.9% respectively. T_1 , in this case, is smaller than 1%, thus the data is assumed to be approximately co-planar. The normal of the plane can be determined, which is the Eigenvector corresponding to the smallest Eigenvalue. It is $\mathbf{u} = -0.0146\mathbf{i} - 0.0043\mathbf{j} + 0.9999\mathbf{k}$. The result is consistent since that portion of scan is corresponding to the floor, which is a horizontal plane and has a normal in the z -axis direction.

5.4 Implementation of PCA on Segmented Range Data

Figure 5.32: Horizontal Plane

Figure 5.33: Indoor Cluster 5

Figure 5.33: Vertical Plane

For this segmented range data, the Eigenvalues are found to be 0.0005, 0.1136 and 0.1252, and their ratios are 0.21%, 47.47% and 52.32% respectively. T_1 is once again smaller than the threshold, and the Eigenvector corresponding to λ_1 is $\mathbf{u} = 0.1552\mathbf{i} + 0.9879\mathbf{j} + 0.0024\mathbf{k}$. The plane, in this case, is perpendicular to y-axis. This makes sense since it is a section of the vertical wall of the lab.

5.4 Implementation of PCA on Segmented Range Data

Figure 5.34: Calibration Issue

5.4.1 Implementation Issues

Calibration

Figure 5.34 depicts the error which is caused in the calibration of the integrated system. This is due to the assumption of small angle approximation made in finding the relationship between the elevation angle of the laser beam and that of the light ray going into the camera from the same object. By geometry,

$$\beta' = \pi/2 - \beta + \phi$$

Applying the cosine rule for the triangle ABC, ϕ can be calculated as follows:

$$d^2 = \rho'^2 + \rho^2 - 2\rho'\rho \cos \phi$$

where ρ and ρ' are the distances of point A to the laser origin and camera origin respectively, β and β' are the elevation angles of the laser beam and light ray coming from point A respectively, and d is the distance between camera and laser origins.

5.4 Implementation of PCA on Segmented Range Data

For large value of ρ , ρ' is almost equal to ρ and the value of ϕ is significantly small. thus, we have:

$$d^2 = 2\rho^2(1 - \cos \phi)$$

$$d^2 = 4\rho^2 \sin^2 0.5\phi$$

$$d = 2\rho \sin 0.5\phi$$

$$d = \rho\phi$$

thus, $\phi = d/\rho$ and $\beta' = \pi/2 - \beta + d/\rho$.

This implies that the elevation of the image for the near objects is erroneous, and it is a better approximation for far objects. A erroneous elevation angle of the image leads to the misclassification of the object, therefore the segmented range data corresponding to that particular object is also erroneous, and the PCA method might fail to extract any expected feature from the given data set.

Figure 5.35: Misclassification Due to Imperfect Calibration.

Figure 5.35 depicts some misclassification areas due to imperfect calibration. It can be observed that part of the object near to the sensor might be classified in the same cluster as an object at far away, due to the small difference between their elevation angle. On the other hand, part of the far object might be clustered in the same group as the near object too. This is highlighted with the yellow regions in Figure 5.35.

Misclassification can cause PCA method to fail to extract any plane or line

5.4 Implementation of PCA on Segmented Range Data

from the data set. Figure 5.36 shows the scan of a section of a building. At the elevation angle observing the building, there are bushes. Some part of the bushes were classified as the same group as the building. After application of PCA on the data set, the Eigenvalues were found out to be 2.633, 12.131 and 53.695. Their ratio is greater than the threshold 1%. This implies that the bushes make PCA fail to find the plane which corresponds to that section of the building.

Figure 5.36: Application of PCA on A Section of A Building.

Segmentation

Image segmentation plays a important role in classifying the data into small groups. Misclassification can occur in an image due to several factors such as:

- Different objects having the same color are clustered into the same group,
- Single objects having several colors are split into groups of clusters,
- Single objects with different light intensities are clustered into many groups,
- Different lighting conditions make the algorithm produce different results

Figure 5.37 highlights some segmentation results which are considered to be inconsistent. It can be observed that the walls inside the lab (left image in Figure 5.37)

5.4 Implementation of PCA on Segmented Range Data

Figure 5.37: Segmentation Issue.

produce different RGB values in the image. After being filtered and clustered by Mean Shift Algorithm, the result shows that some points of the wall are disassociated with other parts. On the other hand, the right hand image in Figure 5.37 demonstrates that part of building was misclassified as belonging to the same group of pixels as the entrance wall.

Figure 5.38: Entrance Wall

Figure 5.38 demonstrates the plot of the misclassified data set. The Eigenvalues were calculated to be 1.728, 6.582 and 108.001. The smallest ratio T_1 is 1.486% greater than the threshold. Again, this shows that the misclassification of the building makes the PCA fail to find the plane corresponding to the entrance wall.

Chapter 6

Conclusions and Recommendations

6.1 Conclusions

Feature extraction has been a basic and vital component in autonomous systems, because it is of paramount importance in object identification, tracking and feature-based robot navigation. In particular, the performance of feature-based SLAM applications are found to be dependent on feature extraction. Robust feature extraction simplifies data association in SLAM and improve overall its performance. 2D SLAM is fragile and diverges in long term, due to vehicle pitching and rolling on uneven ground. In recent years, 3D SLAM has received considerable interest. However, there are several issues to be addressed in 3D SLAM, such as large memory usage, computational complexity and time and reliable feature extraction. The importance of these issues was comprehensively dealt with in Chapters 1 and 2.

Chapter 3 illustrated feature extraction for 2D range data. Point clustering, RANSAC and Hough transform methodologies were discussed in the chapter. The extraction of tree trunks, lamp posts and building pillars was demonstrated for outdoor environments. The EKF SLAM framework was implemented using those extracted features as landmarks. The SLAM results showed that these landmarks

6.1 Conclusions

helped improve vehicle localization. The robot view point issue of cluster-based point feature implementation was also explained. With different view points of the sensor, the number of points representing the same cluster can be different, thus their calculated centroid can be different as well. This led to inconsistent feature noise modeling and affected the performance of SLAM. Similarly, line-feature based SLAM was also examined. Walls of a building were extracted as lines and used as landmarks for SLAM. The result depicted an improvement of vehicle localization with feature-based EKF SLAM for both types of features.

Moving from 2D to 3D, chapter 4 introduced a built-in-house 3D LADAR, which provided range, bearing and elevation information of each point in the environment. The system could produce low and high resolution 3D point cloud data by controlling the rotating speeds of the frame and mirror. Plots of point clouds were illustrated in 3D. The chapter also introduced the PCA method and its mathematical formulation, which is used to compress data and detect dominant data directions. In terms of computational speed, the PCA method performed significantly fast to categorize whether the data sets belong to a plane, a line or they are randomly distributed. However, its performance on complete real data sets was poor. PCA always showed that real data was randomly distributed since data from many artifacts is always present. Thus, range data needs to be trimmed or segmented first.

To aid in the range segmentation process, an integrated system was built and described in chapter 5. After calibration, the system was able to provide fused data of range and vision. 3D point cloud data with RGB color information was demonstrated. To achieve plane and line extraction, the following algorithm was performed on the fused data. First the image given by the camera was filtered and segmented. Three filtering methods were examined. The Mean Shift performed well

6.2 Recommendations

in terms of edge preserving, it was therefore chosen to be used in our work. After the filtering process, the image were segmented. Then fusion between range data and segmented image was carried out. The range data was tagged with each image cluster label, based on which the range data was segmented into smaller groups. Finally, PCA was applied to each group of segmented range data to categorize the data whether it was a plane, a line or random. The result showed significant improvement in plane and line extraction over the result shown in chapter 4. Vertical and horizontal planes were extracted from the range data, and their normals were determined. The results also illustrated that some planes could not be extracted due to calibration errors of the integrated system and imperfect segmentation of the image.

In conclusion the work of this thesis addresses plane and line extraction for 3D data with the aid of visual data. The combination of range and color information enhances the reliability of PCA method for 3D plane and line extraction.

6.2 Recommendations

- The extracted planes and lines can be used as input for certain applications. Particularly, they can be landmarks in feature-based 3D SLAM applications. PCA provides the normal vector of the planes extracted which can be used to parameterize the features in 3D. By augmenting the dimension of the existing 2D feature-based SLAM application, implementation of 3D feature-based SLAM is possible.
- A perfect calibration of the integrated system is impossible to achieve. However, an error feedback system could be possible to minimize the calibration error. A cluster from an image is linked to a certain subset of the range data,

6.2 Recommendations

through the calibration. If the corresponding range point subset then fails to yield PCA based dominant directions, then the calibration could be erroneous. The range data subset could then be trimmed or process to remove possible outlier data, which could then be used to automatically adjust the calibration parameters.

Bibliography

- [1] M. D. Adams, Sensor modeling, Design and Data Processing for Autonomous Navigation. World Scientific, 1999.
- [2] M. D. Adams and A. Kerstens, Tracking naturally occurring indoor features in 2D and 3D with lidar range/amplitude data, The International Journal of Robotics Research, 17 (1998) 907-923.
- [3] Sen Zhang, Lihua Xie, Martin Adams, Feature Extraction for Outdoor Mobile Robot Navigation Based on a Modified Gauss Newton Optimisation Approach, Intl Journal of Robotics and Autonomous Systems, Elsevier, Vol 54, Issue 4, pages 277-287, April 2006.
- [4] J. E. Guivant, F. R. Masson and E. M. Nebot, Simultaneous localization and map building using natural features and absolute information, Robotics and Autonomous Systems, 40 (2002) 79-90.
- [5] Rojkova, V. and Kantardzic, M. 2007. Feature Extraction Using Random Matrix Theory Approach. In Proceedings of the Sixth international Conference on Machine Learning and Applications (December 13 - 15, 2007).
- [6] M. Adams, F. Tang and W. S. Wijesoma. Convergent Smoothing and Segmentation of Noisy Range Data in Multiscale Space. IEEE Trans. on Robotics, pp. 746-753, June 2008.
- [7] J. Xavier, M. Pacheco, D. Castro, A. Ruano, and U. Nunes, Fast line, arc/circle and leg detection from laser scan data in a player driver, in IEEE International Conference on Robotics and
- [8] D. Schulz, W. Burgard, D. Fox, and A. Cremers, "Tracking multiple moving targets with a mobile robot using particle filters and statistical data association," in IEEE Int. Conf. on Robotics and Automation, pp.1665-1670, Seoul 2001.
- [9] B. Kluge, C. Khler, and E. Prassler, "Fast and robust tracking of multiple moving objects with a laser range finder," in IEEE Int. Conf. on Robotics and Automation, pp.1683-1688, Seoul, 2001.
- [10] Arras K.O., Siegwart R.Y., Feature Extraction and Scene Interpretation for Map-Based Navigation and Map Building, Proceedings of SPIE, vol. 3210, Mobile Robotics XII, p. 42-53, Pittsburgh, USA, 1997

Bibliography

- [11] C. Premebida and U. Nunes. Segmentation and Geometric Primitives Extraction from 2D Laser Range Data for Mobile Robot Applications. In: Proc. 5rd National Festival of Robotics Scientific Meeting (ROBOTICA), pp- 17-25, Coimbra, Portugal.
- [12] G. A. Borge and M. -J. Aldon. Line Extraction in 2D Range Images for Mobile Robotics. *Journal of Intelligent and Robotic Systems*, 40:267-297, 2004
- [13] R. Taylor and P. Probert. Range Finding and Feature Extraction by segmentation of Image for Mobile Robot Navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation, ICRA*, 1996
- [14] J. Vandorpe, H. V. Brussel, and H. Xu. Exact Dynamic Map Building for a Mobile Robot using Geometrical Primitives Produced by a 2D Range Finder. In *Proceedings of the IEEE International Conference on Robotics and Automation, ICRA*, pages 901-908, 1996.
- [15] M. Fischler and R. Bolles. Random sample consensus: A paradigm for model fitting with application to image analysis and automated cartography. *Communications of the ACM*, 24(6):381-395, 1981.
- [16] S. T. Pfister, S. I. Roumeliotis, and J. W. Burdick. Weighted Line Fitting Algorithms for Mobile Robot Map Building and Efficient Data Representation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2003.
- [17] R.O. Duda, P.E. Hart, "Use of the Hough Transform to Detect Lines and Curves in Pictures", *Com. of ACM*, Vol.18, No.9, pp.509-517, 1975.
- [18] V. Nguyen, A. Martinelli, N. Tomatis, R. Siegwart. A Comparison of Line Extraction Algorithms using 2D Laser Rangefinder for Indoor Mobile Robotics. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, 2005.
- [19] D.F. Huber, and M. Hebert, A New Approach to 3-D Terrain Mapping, *International Conference on Intelligent Robots and Systems (IROS)*, Kyongju, Korea, October 1999.
- [20] S. Lacroix, A. Mallet, D. Bonnafoous, G. Bauzil; S. Fleury, M. Herrb, and R. Chatila, Autonomous Rover Navigation on Unknown Terrains: Functions and Integration, *International Journal of Robotics Research*, vol. 21, issue 10, October 2002.
- [21] H. P. Moravec, Robot Spatial Perception by Stereoscopic Vision and 3D Evidence Grids, Tech Report CMU-RI-TR-96-34, Pittsburgh, USA, September 1996.
- [22] R. Unnikrishnan, and M. Hebert, Robust Extraction of Multiple Structures from Non-uniformly Sampled Data, *International Conference on Intelligent Robots and Systems (IROS)*, Las Vegas, USA, October 2003.
- [23] D. Viejo and M. Cazorla. Plane Extraction and Error Modeling of 3D data. *Intl Symposium on Robotics and Automation*, August 2006.

Bibliography

- [24] Weingarten, J. and Siegwart, R. EKF-based 3D SLAM for Structured Environment Reconstruction. In Proceedings of IROS, Edmonton, Canada, August 2-6, 2005. (IROS2005)
- [25] P. Kohlhepp, P. Pozzo, M. Walter, R. Dillmann, Sequential 3D-SLAM for mobile action planning, Proceedings of IROS, 2004, Sendai.
- [26] R. Triebel, W. Burgard and F. Dellaert , Using Hierarchical EM to Extract Planes from 3D Range Scans. In Proc. of the IEEE International Conference on Robotics and Automation (ICRA), 2005
- [27] J. Horn, G. Schmidt, Continuous localization of a mobile robot based on 3D-laser-range-data, predicted sensor images, and dead-reckoning, Robotics and Autonomous Systems, 1995, v. 14, pp. 99-118
- [28] J. Weingarten, G. Gruener, R. Siegwart, A Fast and Robust 3D Feature Extraction Algorithm for Structured Environment Reconstruction, Proceedings of ICAR, 2003, Coimbra.
- [29] Y. Liu, R. Emery, D. Chakrabarti, W. Burgard, and S. Thrun, Using EM to learn 3D models with mobile robots, in Proceedings of the International Conference on Machine Learning (ICML), 2001.
- [30] H. W Sorenson, Kalman Filtering, Theory and Application, IEEE press 1985
- [31] A. Glebb, Applied Optimal Estimation, MIT press 1974.
- [32] P.S. Mayback, Stochastic Models Estimation and Control- Volume 1, Academic Press 1982
- [33] M. W. M. G. Dissanayake, P. Newman, S.Clark, H. F. Durrant-Whyte, and M. Csorba, "A solution to the simultaneous localization and map building (SLAM) problem", IEEE Transactions on Robotics and Automation, vol. 17, no. 3, pp. 229-241, June 2001.
- [34] R. SMith, M. Self, and P.Cheeseman, "A stochastic map for uncertain spatial relationships", Fourth International Symposium of Robotics Research, 1987, pp. 467-474.
- [35] H. Baltzakis, A. Argyros, and P. Trahanias. Fusion of range and visual data for the extraction of scene structure information. In Proc. International Conference on Pattern Recognition (ICPR), pages 40007-40012, Quebec, Canada, August 2002.
- [36] J. A. Castellanos, J. M. Martinez, J. Neira and J. D. Tardos. Simultaneous Map Building and Localization for Mobile Robots: A Multisensor Fusion Approach. Proceedings of the International Conference on Robotics and Automation, Belgium, May 1998.
- [37] P. Chakravarty, R. Jarvis, Panoramic Vision and Laser Range Finder Fusion for Multiple Person Tracking, in Proc. IEEE Int. Conf. Intelligent Robots and Systems, pp. 2949-2954, 2006.

Bibliography

- [38] X. Song, J. Cui, H. Zhao and H. Zha. Bayesian Fusion of Laser and Vision for Multiple People Detection and Tracking. SICE Annual Conference, August 2008.
- [39] Y. Tsai, A versatile camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using Off-the-Shelf TV Cameras and Lenses, IEEE Journal of Robotics and Automation, pp. 323-344, 1987.
- [40] U. Kadow, G. Schneider and A. Vukotich. Radar-Vision Based Vehicle Recognition with Evolutionary Optimized and Boosted Features. IEEE Intelligent Vehicles Symposium, pp749-754, June 2007
- [41] P. Biber, H. Andreasson, T. Duckett and A. Scilling. 3D Modeling of Indoor Environments by a Mobile Robot with a Laser Scanner and Panoramic Camera
- [42] I. Jolliffe. Principal Component Analysis. New York: Springer-Verlag, 1986.
- [43] Linsey Smith. A tutorial on Principal Component Analysis, 2002
- [44] S. M. Smith and J. M. Brady, "SUSAN-A New Approach to Low Level Image Processing", Intl Journal of Computer Vision, 23(1), 45-78, May 1997.
- [45] C. Tomasi and R. Manduchi, "Bilateral Filtering for Gray and Color Images", proc. of Intl Conference on Computer Vision, IEEE (1998) 839-846.
- [46] G. Wyszecki and W.S. Styles. "Color Science: Concepts and Methods, Quantitative Data and Formulae. Wiley, New York, NY, 1982.
- [47] G. Sapiro and D. L. Ringach. Anisotropic diffusion of color images. In Proc. SPIE, 2657:471-382, 1981
- [48] Y. Cheng. Mean shift, mode seeking, and clustering. IEEE Trans. Pattern Anal. Machine Intell., vol. 17, 790-799, 1995.
- [49] D. Comaniciu, P. Meer: Mean Shift Analysis and Applications, IEEE Int. Conf. Computer Vision (ICCV'99), Kerkyra, Greece, 1197-1203, 1999
- [50] P. Maybeck and S. Rogers. Adaptive tracking of multiple hot-spot target IR images. IEEE Trans. Automatic Control, pp. 937-943, Oct 1983.
- [51] J. S. Li, D. Shi and W. Yang. Infared Target Extraction in FLIR Imagery Based on Spatio Temporal Using Fuzzy Clustering. Intelligent Information Techonology Application, pp. 848-851, Dec 2008.
- [52] K. Kim, J. Summet, T. Starner, D. Ashbrook, M. Kapade, I. Essa. Localization and 3D reconstruction of urban scenes using GPS. Iswc, pp.11-14, 2008 12th IEEE International Symposium on Wearable Computers, 2008
- [53] H. Chen and Z. Xu. 3D Map Building Based on Stereo Vision. Proc. IEEE Intl. Conf. pp. 969-973 August, 2006.

User Manual for 3D LADAR

I. Physical Connections of The System

Figure 1: Physical Connections of The System.

- The connections of PC, control box and 3D LADAR are as shown in Figure 1
- The control box is supplied by 24V DC power and supplies 16V to the 3D LADAR
- The control box is connected to the 3D LADAR with serial interfaces
- The control box is connected to The PC by using ME 3000 PCI card

II. QNX Platform

1. "root" and "rootpass" are the username and password respectively to log on the QNX system
2. Click on "Launch", "QNX Momentics 6.3.0", then "Application Building" to start the development software
3. The GUI development program window is popped up
4. To load the existing program, click "File" then "Open"
5. Change the file directory to "/root/workspace/carryall17", select file "carryall17", then click "Open"
6. The program GUI is as shown in Figure 2, it consists of vehicle control module, module displaying output of onboard sensors and 3D LADAR control module.
7. The following actions are necessary for software development:
 - To make clean, click "Build" menu, then "Make Clean" or press "Shift + F3"
 - To generate UI, click "Build" menu, then "Generate UI" or press "Shift + F7"
 - To build and run, click "Build" menu, then "Build and Run" or press "F6"

Figure 2: QNX Momentics 6.3.0, Application Builder.

Figure 3: QNX System Control GUI.

Figure 4: Source Code Directory.

- To run, click "Build" menu, then "Run" or press "Shift + F6". The application window is popped up as shown in Figure 3
8. To edit source codes, click "Text Editor" on the right panel, a file browser window pops up, change the file directory to "/root/workspace/carryall17/src" as shown in Figure 4, select the file and click "Open".
 9. To perform scanning with 3D LADAR, some user inputs are necessary to fill in the control module as shown in Figure 5. The inputs are mirror elevation angles and its resolution, and speed of rotating frame, which is controlled by an input voltage. Once the inputs are filled in, click "Start" to perform the scan, click "Stop" to stop it and "Quit" to exit the program.

Figure 5: 3D LADAR Control GUI.

User Manual for Laser Camera Calibration in Linux and QNX

Figure 6: The Ladybug Camera System

1. Install the IEEE-1394b PCI Card

- Place the IEEE-1394b PCI card in an open PCI slot.
- Connect the external power supply connector (from the picture above) to the power supply only if there is no power line to the PCI card (Ladybug2 camera is powered by the IEEE1394 PCI card in our case)
- Turn the computer on and log into Linux.
- Ensure that the PCI card is properly installed as an IEEE 1394 Bus host controller.

2. Connect the Head to the Compressor

- Plug the orange fiber-optic cable that is provided with the KIT into the Optical.
- Link connectors of the Head and Compressor.
- Plug the 10m 3-pin power link cable into the Power Link connectors of the Head and Compressor.

Figure 7: The Compressor Connection

Figure 8: The Connection of External Power Supply

- NOTE: avoid tight bends in the cable, which can affect data transmission.

3. Connect the External Power Supply to the Compressor Plug the 24V 2.5A external power supply into the Compressor External Power Supply Connector. The Status LEDs on the 1394b bus side of the Compressor and the Head should both turn on (see Status LEDs section above).

4. Connect the 1394b PCI Card and Cable to the Compressor

- Plug the 4.5 meter, 9-pin to 9-pin, IEEE- 1394b cable into the 1394b PCI card and the Compressor 1394b Connector.
- In most cases, the MS Windows Found New Hardware Wizard will appear and prompt you to install the driver. Proceed to Step 5.

Linux Platform

5. Install The Ladybug Driver in Linux

- a. Install Library dc1394.

Go to the directory where your kernel source code is present. Normally it is in /usr/src. It is suggested that a fresh copy of the kernel source code should be downloaded from www.kernel.org.

Thus, `cd /usr/src/kernel directory` (`kernel-2.6.10` is the name of the source code directory. It can vary from system to system.)

1. `make menuconfig`
2. The menu will come up. Select Device Drivers
3. The Device Drivers menu will come up. Select IEEE1394 (Firewire) Support
4. Mark the following as `M` in the menu that is shown:
 - OHCI-1394 support
 - OHCI-1394 Video Support
 - OHCI-1394 DVI/O Support
 - RAW IEEE1394 I/O Support
5. Press ESC to come out of the menu
6. Save the configuration

The Libraw and Libdc library files were downloaded from sourceforge.net and installed in the machine. The libdc1394 library had example files which were modified and executed to perform different operations like capturing Bayer pattern images and converting to color images.

The following steps are required to install libraw files.

1. `su root`
2. `tar -xvzf libraw1394-1.2.0.tar.gz`
3. `cd libraw1394-1.2.0`
4. `./configure`
5. `make`
6. `make install`

The following steps are required to install libdc files.

1. `su root` (if you haven't already)
2. `tar -xvzf libdc1394-1.1.0.tar.gz`
3. `cd libdc1394-1.1.0`
4. `./configure`
5. `cd libdc1394`
6. `make`
7. `make install`

Next step is to install the modules.

1. modprobe ohci1394
2. modprobe video 1394
3. modprobe ieee1394
4. modprobe raw1394
5. lsmod to view that all the modules (ieee1394, raw1394, ohci1394 and video1394) are now installed.

The modules have some dependencies that need to be considered.

1. ohci1394 to video1394
2. ieee1394 to video1394, ohci1394, and raw1394

Proper 1394 devices need to be created. The following commands are required to create the devices.

1. cd /dev
2. mknod raw1394 c 171 0
3. chmod 666 /dev/raw1394
4. mkdir video1394
5. cd video1394
6. mknod 0 c 171 16
7. mknod 1 c 171 17
8. chmod 666 /dev/video1394/*

Every time the system is restarted, the nodes are destroyed and need to be created again. This completes the installation phase. In order to make sure the installation was correct, the camera was connected to the IEEE1394 card. The testlibraw command was used to test the presence of the host card, camera and other related parameters.

b. Install jpeg Library.

Files: /usr/local/lib/{libjpeg.a, libjpeg.so, libjpeg.so.6, libjpeg.so.6.a}
 /usr/local/include/{jconfig.h, jmorecfg.h, jerror.h, jpeglib.h}
 /usr/local/bin/{cjpeg, djpeg, jpegtran, rdjpgcom, wrjpgcom}
 /usr/local/man/man1/{cjpeg.1.gz, jpeg.1.gz, djpeg.1.gz, jpegtran.1.gz, rdjpgcom.1.gz, wrjpgcom.1.gz}

Sources obtainable at:

ftp://ftp.cs.columbia.edu/jpeg/src/jpegsrc.v6a.tar.gz
 http://sunsite.unc.edu
 ftp://ftp.x.org

First of all, remove any old installation in /usr/bin.

If you only want the static library do the following. Unpack the source, then run `./configure` and `make`. Install with `make install` and `make install-lib`, strip the binaries, and compress the manual pages.

If you want the static and the shared library, then do the following. Unpack the archive `jpegsrc.v6a.tar.gz` and `cd` into the newly created subdirectory, run `./configure`. Then edit the Makefile, so you create the shared library as well. `CFLAGS = -O3 -fomit-frame-pointer -fPIC -I$(srcdir)`

all: libjpeg.so.6 libjpeg.a cjpeg djpeg jpegtran rdjpgcom wrjpgcom

libjpeg.so.6: libjpeg.so.6a

ln -sf libjpeg.so.6a libjpeg.so.6

libjpeg.so.6a: \$(LIBOBJECTS)

\$(CC) -shared -Wl,-soname,libjpeg.so.6 -o libjpeg.so.6a \$(LIBOBJECTS)

cjpeg: \$(COBJECTS) libjpeg.so.6

\$(LN) \$(LDFLAGS) -o cjpeg \$(COBJECTS) libjpeg.so.6 \$(LDLIBS)

djpeg: \$(DOBJECTS) libjpeg.so.6

\$(LN) \$(LDFLAGS) -o djpeg \$(DOBJECTS) libjpeg.so.6 \$(LDLIBS)

jpegtran: \$(TROBJECTS) libjpeg.so.6

\$(LN) \$(LDFLAGS) -o jpegtran \$(TROBJECTS) libjpeg.so.6 \$(LDLIBS)

Run `make` and `make install`; `make install-lib`. You'll have to install the shared library by hand. Then run `ldconfig`, strip the installed binaries, and compress the man pages.

c. Install OpenCV.

In order to install OpenCV library on Linux, the student compiled the source file. The sequence of compilation was followed as stated below [14]. The latest version of the library is OpenCV-1.0.0 at the time of writing.

1. + GTK+ 2.x or higher including headers
2. + pkgconfig
3. + libpng, zlib, libjpeg, libtiff, libjasper with development files.
4. + Python 2.3, 2.4 or 2.5 with headers installed (developer package)
5. + libavcodec, etc. from ffmpeg 0.4.9-pre1 or later + headers.

The source file is extracted. A new directory named 'opencv-1.0.0' was created. In the new directory, following commands were executed.

1. `tar xvzf OpenCV-1.0.0.tar.gz`
2. `cd opencv-1.0.0`
3. `./configure --prefix=/opencv_library_install_path/opencv-1.0.0`
4. `make`; `make install`

In order to use the libraries in Linux, their path should be specified. Library path was specified in `/etc/ld.so.conf.d/` by creating a file called `'opencv.conf'` which contained the opencv library path (Default configuration is `/usr/local/lib`). Once the file is created, `'ldconfig -v'` command was executed as root to make new set library paths effective.

Following commands were executed to ensure OpenCV was running successfully:

1. `cd /.../opencv-1.0.0/`
2. `make check`

6. System Operation at Start-Up

Open a terminal and type the following commands (can copy all these commands from the HOWTO.txt file in `/home/bimas/Ladybug/src` folder:

```
modprobe ohci1394
modprobe ieee1394
modprobe video1394
modprobe raw1394
cd /dev
mknod raw1394 c 171 0
chmod 666 /dev/raw1394
mkdir video1394
cd video1394/
mknod 0 c 171 16
mknod 1 c 171 17
chmod 666 /dev/video1394/*
```

Image Acquisition - Format7 Mode0 image

1. type `cd /home/bimas/Ladybug/src`
2. `gcc -g -O2 -Wall -Wunused -c -o pgrutils.o pgrutils.c`
3. `g++ -g -O2 -Wall -Wunused -o ladybug70 ladybug_Mode7_0.c -ldc1394 -lm -lcv -lhighgui -lcvaux`
4. `./ ladybug70`
5. Images stored in folder `/home/bimas/Ladybug/test_result` as filenames `000000.ppm` to `000005.ppm` for each of the six cameras

Image Stitching

1. Run the `ladybugPanoStitchExample` program in the Ladybug SDK
2. Port the `cammask.bim`(mapping of pixel values on to the stitched image) file to linux Paste the `cammask.bim` file into the `/home/bimad/Ladybug/src` folder
3. type `cd /home/bimas/Ladybug/src`

4. `gcc -g -O2 -Wall -Wunused -o ladybug70Stitch ladybug_Mode7_0_Stitch_Integration.c pgrutils.o -ldc1394 -lm -lcv libjpeg.a`
5. `./ladybug70Stitch`
6. Image stored in folder `/home/bimas/Ladybug/test_result` as filename `test-00000-panImage.ppm`

Video Stitching

1. type `cd /home/bimas/Ladybug/src`
2. `gcc -g -O2 -o ladybug70StitchVideo ladybug_Mode7_0_Stitch_Integration_Video.c pgrutils.o -ldc1394 -lm -lcv libjpeg.a`
3. `./ ladybug70StitchVideo`
4. Image stored in folder `/home/bimas/Ladybug/test_result` as filename `video1.avi`

Finding first set of Extrinsic parameters (Camera mounted on top of laser)

Capture distorted image from Camera0 and note pixel values of a 5 points (4 corners and centre of the image). Input these pixel values to `ladybugrectifypixel` method in the ladybug SDK in windows. Now port these corrected pixel values to linux. Enter the rectified pixel values of image features and their corresponding 3D laser data in a text file (`points.coord.txt` in `/home/bimas/Ladybug/src` folder). Use the OpenCV method `FindExtrinsicCameraParams2` to find the extrinsic parameters for the pattern.

1. type `cd /home/bimas/Ladybug/src`
2. `gcc -g -O2 -o LasCamCalTest LasCamCalTest.c -lm -lcv`
3. `./ LasCamCalTest`

Take note of the extrinsic parameters generated in the terminal window. Enter these values into `projectLaserOnPanIma.c`

Projecting the laser points onto the image

1. Input the X,Y and Z coordinates from the laser scan
2. `gcc -g -O2 -o projectLaserOnPanIma projectLaserOnPanIma.c -lm -lcv ./ project-LaserOnPanIma`
3. Laser point is projected onto the new image `rst_test-00000-panImage.ppm` in the `test_result` folder.

Projecting series of points onto the image

1. `gcc -g -O2 -o projectLaserOnPanIma2 projectLaserOnPanIma2.c -lm -lcv`
2. `./ projectLaserOnPanIma2`
3. The code reads laser points from `laser_data.txt` file and generates the pixel values.

4. Laser line is projected onto the new image rst2_test-00000-panImage.ppm in the test_result folder.

Generating the RGB values for the laser points for a still image

1. Save a copy of panoramic image generated as pano.ppm.
2. `gcc -g -O2 -o projectLaserOnPanIma3 projectLaserOnPanIma3.c -lm -lcv`
3. `./ projectLaserOnPanIma3`
4. The code reads laser points from ladarc.txt file and generates the pixel values.
5. Results saved in RGB.txt file.

Generating the RGB values for the laser points for a stitched video

1. Save a copy of panoramic video generated as video1_03.avi.
2. Read data from ladarc3.txt
3. `gcc -g -O2 -o projectLaserOnVideoPanIma projectLaserOnVideoPanIma.c pgrutils.o -ldc1394 -lm -lcv libjpeg.a`
4. `./ projectLaserOnVideoPanIma`
5. Results saved in RGBVideo.txt.

Generating the Laser Scans in OpenGL

1. Obtain the RGB values from RGBVideo.txt file and give it as input parameters for the program in OpenGL.
2. The laser scans are generated with the executable file.

QNX Platform

The file to generate the stitched video is ported from linux to QNX. Camera driver is installed using the SedNet PCI card and LLA package provided.

1. Login to the QNX machine
2. Register the product by completing the registration form included in the installation package of the SedNet 1394 Stack v4.1.1 for QNX Momentics.
3. License key is generated through the mindready support team.
4. For Activation of SEDNET 1394 Driver online., follow the intructions below:
 - Navigate to www.mindready.com
 - Click Login link. Click on register here if you do not have a web account with mindready. Otherwise, enter your username and password.

- Complete the account registration form and click submit.
 - Click the Online License Generator link to continue with licensing your SedNet 1394 driver.
 - Now complete the license generator form with your Mindready certificate number, the UID number from your OHCI communication adapter and your company ID code.
 - Click Generate button.
 - The Generated key is displayed in the Automated License Generator window.
 - The SedNet driver is activated.
5. Any previous installation(s) of the SedNet 1394 stack must be removed prior to installing this package.
 6. Install the x86 version using the following command
 - Open terminal and type `cd /root/ladybug/stack`
 - `qnxinstall -r licgui-1.0.1-x86-MNY.qpr`
 7. Install the x86 version using the following command
 - Open terminal and type `cd /root/ladybug/stack`
 - `qnxinstall -r STK_ohci_x86-4.1.1-x86-MNY.qpr`
 8. Install the x86 version using the following command
 - Open terminal and type `cd /root/ladybug/iidc`
 - `qnxinstall -r IIDC_x86-1.1.1-x86-MNY.qpr`
 9. Open terminal and type `cd /root/ladybug`. Type `licgui`. Press Enter.
 10. Start the camera driver. Type the following commands:
 - `cd /root/workspace/iidcdrv/x86/o`
 - `./iidcdrv`
 11. Go to the directory containing the iidc camera application to stream stitched video with the Ladybug2 camera. Type the following:
 - `Cd /root/workspace/iidcApiExample/x86/o`
 - `./iidcApiExample`
 - The stitched video is captured into `video_01/avi`