Real Time 3D Mapping for Small Wall Climbing Robots

Blair David Sidney Howarth

A thesis submitted in fulfilment
of the requirements for the degree of
Doctor of Philosophy

School of Mechanical and Manufacturing Engineering
The University of New South Wales

February 2012

THE UNIVERSITY OF NEW SOUTH WALES
Thesis/Dissertation Sheet

Surname or Family name: **HOWARTH**

First name: **BLAIR** Other name/s: **DAVID SIDNEY**

Abbreviation for degree as given in the University calendar: **MTRN (1662)**

School: **MECHANICAL AND MANUFACTURING** Faculty: **ENGINEERING**

Title: **REAL TIME 3D MAPPING FOR SMALL WALL CLIMBING ROBOTS**

Abstract 350 words maximum:

Small wall climbing robots are useful because they can access difficult environments which preclude the use of more traditional mobile robot configurations. This could include an industrial plant or collapsed building which contains numerous obstacles and enclosed spaces. These robots are very agile and they can move fully through three dimensional (3D) space by attaching to nearby surfaces. For autonomous operation, they need the ability to map their environment to allow navigation and motion planning between footholds. This surface mapping must be performed onboard as line-of-sight and wireless communication may not always be available.

As most of the methods used for robotic mapping and navigation were developed for two dimensional usage, they do not scale well or generalise for 3D operation. Wall climbing robots require a 3D map of nearby surfaces to facilitate navigation between footholds. However, no suitable mapping method currently exists. A 3D surface mapping methodology is presented in this thesis to meet this need.

The presented 3D mapping method is based on the fusion of range and vision information in a novel fashion. Sparse scans from a laser range finder and a low resolution camera are used, along with feature extraction, to significantly reduce the computational cost. These features are then grouped together to act as a basis for the surface fitting. Planar surfaces, with full uncertainty, are generated from the grouped range features with the image features being used to generate planar polygon boundaries. These surfaces are then merged together to build a 3D map surrounding a particular foothold position.

Both experimental and simulated datasets are used to validate the presented surface mapping method. The surface fitting error is satisfactory and within the required tolerances of a wall climbing robot prototype. An analysis of the computational cost, along with experimental runtime results, indicates that onboard real time operation is also achievable. The presented surface mapping methodology will therefore allow small wall climbing robots to generate real time 3D environmental maps. This is an important step towards achieving autonomous operation.

Declaration relating to disposition of project thesis/dissertation

I hereby grant to the University of New South Wales or its agents the right to archive and to make available my thesis or dissertation in whole or in part in the University libraries in all forms of media, now or here after known, subject to the provisions of the Copyright Act 1968. I retain all property rights, such as patent rights. I also retain the right to use in future works (such as articles or books) all or part of this thesis or dissertation.

I also authorise University Microfilms to use the 350 word abstract of my thesis in Dissertation Abstracts International (this is applicable to doctoral theses only).

11 Feb 2012

…… …… ……

Signature Witness Date

The University recognises that there may be exceptional circumstances requiring restrictions on copying or conditions on use. Requests for restriction for a period of up to 2 years must be made in writing. Requests for a longer period of restriction may be considered in exceptional circumstances and require the approval of the Dean of Graduate Research.

FOR OFFICE USE ONLY Date of completion of requirements for Award:

THIS SHEET IS TO BE GLUED TO THE INSIDE FRONT COVER OF THE THESIS

Abstract

Small wall climbing robots are useful because they can access difficult environments which preclude the use of more traditional mobile robot configurations. This could include an industrial plant or collapsed building which contains numerous obstacles and enclosed spaces. These robots are very agile and they can move fully through three dimensional (3D) space by attaching to nearby surfaces. For autonomous operation, they need the ability to map their environment to allow navigation and motion planning between footholds. This surface mapping must be performed onboard as line-of-sight and wireless communication may not always be available.

As most of the methods used for robotic mapping and navigation were developed for two dimensional usage, they do not scale well or generalise for 3D operation. Wall climbing robots require a 3D map of nearby surfaces to facilitate navigation between footholds. However, no suitable mapping method currently exists. A 3D surface mapping methodology is presented in this thesis to meet this need.

The presented 3D mapping method is based on the fusion of range and vision information in a novel fashion. Sparse scans from a laser range finder and a low resolution camera are used, along with feature extraction, to significantly reduce the computational cost. These features are then grouped together to act as a basis for the surface fitting. Planar surfaces, with full uncertainty, are generated from the grouped range features with the image features being used to generate planar polygon boundaries. These surfaces are then merged together to build a 3D map surrounding a particular foothold position.

Both experimental and simulated datasets are used to validate the presented surface mapping method. The surface fitting error is satisfactory and within the required tolerances of a wall climbing robot prototype. An analysis of the computational cost, along with experimental runtime results, indicates that onboard real time operation is also achievable. The presented surface mapping methodology will therefore allow small wall climbing robots to generate real time 3D environmental maps. This is an important step towards achieving autonomous operation.

Acknowledgements

There are many people that I would like to thank for helping me through the past four years of my PhD studies. The process has been long, challenging and at times difficult but overall it has been a rewarding one, especially now that the end is in sight!

To my PhD supervisor, Associate Professor Jayantha Katupitiya, you have been not only a mentor over the past four years, but a friend too. Thank you for your guidance and support. You have been able to provide a good balance between giving me the freedom to work independently whilst always being available to assist when required. Thanks also to my co-supervisor, Dr Jose Guivant, for your valuable insights. Your ability to provide a fresh perspective has been very valuable. I would also like to acknowledge the financial support of the ARC Centre of Excellence programme, along with the APA scholarship programme, who have helped fund my studies.

A massive thank you must go to my parents, Graeme and Shirley, and my two sisters, Louise and Emily. You have given me so much love and support over the years. You are always interested in what I am doing and how I am progressing and I am so grateful to have such a wonderful and loving family. Being able to talk or Skype regularly has made studying away from home that much easier.

To Avi, my incredible girlfriend, I want to say a big thanks for all your support and encouragement. You always seem to be able to push me to be better and you have the amazing ability to make me smile.

Throughout my PhD studies I have worked alongside many fine students, both postgraduate and undergraduate. Thanks especially to my fellow PhD candidates, James and Mark. I have enjoyed our discussions on all manner of issues and you have always been able to provide fresh ideas and feedback.

Basser College was my home for the majority of my studies here at UNSW. After moving from New Zealand I didn't know many people in Sydney. Moving into Basser as a Tutor/Resident Fellow was one of the best decisions I ever made. I felt so welcomed and met so many wonderful people in what has been, and always will be, the best residential college at UNSW. I have made many new friends and I have immensely enjoyed the good natured trans-tasman rivalry. A big shout out to my

fellow Basser tutors, especially to Evan, El, Sarah and Lauren.

A special mention must go to Dr Geoff Treloar. Your guidance and support as head of Basser College was most valuable but you were also a good friend. Basser is now poorer without your stewardship and your many words of wisdom. Thanks also to my physio Mark for putting up with me over the last four years and fixing up my neck, back and myriad other injuries.

A final thanks to Geoff, Mark and Louise for proof reading and providing valuable feedback on my thesis. Your help has been invaluable in polishing the final version of this thesis and I am very grateful for your time and effort.

Declaration

"I hereby declare that this submission is my own work and to the best of my knowledge it contains no materials previously published or written by another person, or substantial proportions of material which have been accepted for the award of any other degree or diploma at UNSW or any other educational institution, except where due acknowledgment is made in the thesis. Any contribution made to the research by others, with whom I have worked at UNSW or elsewhere, is explicitly acknowledged in the thesis. I also declare that the intellectual content of this thesis is the product of my own work, except to the extent that assistance from others in the project's design and conception or in style, presentation and linguistic expression is acknowledged."

Signed: 

Date: 11th February 2012

Copyright

Authenticity

"I certify that the Library deposit digital copy is a direct equivalent of the final officially approved version of my thesis. No emendation of content has occurred and if there are any minor variations in formatting, they are the result of the conversion to digital format."

Signed: 

Date: 11th February 2012

Contents

Abstract ii

Acknowledgements iii

Table of Contents viii

List of Abbreviations xiii

1 Introduction 1

1.1 Aim and Scope of Research . . . 3
1.2 Thesis Structure . . . 4
1.3 Publications . . . 5

2 Literature Review 7

2.1 Introduction . . . 7
2.2 Wall Climbing Robots . . . 7
2.2.1 Adhesion Techniques . . . 8
2.2.2 Locomotion Configurations . . . 10
2.2.3 Mapping Capabilities . . . 11
2.3 Simultaneous Localisation and Mapping . . . 13
2.3.1 2D SLAM . . . 14
2.3.2 3D SLAM . . . 16
2.4 3D Mapping . . . 20
2.4.1 Point Clouds . . . 21
2.4.2 Grid Based 3D Maps . . . 24
2.4.3 Planar 3D Maps . . . 24
2.5 Summary . . . 27

3 A Robotic System Requiring 3D Mapping 29

3.1 Introduction . . . 29
3.2 Wall Climbing Robot Prototype . . . 30
3.2.1 Design Implications . . . 32

3.3 Data Acquisition . . . 33
3.3.1 Experimental Data Acquisition Setup . . . 33
3.3.2 Simulated Data Acquisition Setup . . . 37
3.4 Image Sensor . . . 38
3.4.1 Choice of Camera . . . 38
3.5 Range Sensor . . . 39
3.5.1 Choice of Laser Range Finder . . . 40
3.6 Extrinsic Sensor Calibration . . . 41
3.6.1 Sensor Fusion . . . 42
3.6.2 Calibration Parameters . . . 43
3.7 Summary . . . 44

4 Extraction and Grouping of Surface Features 46
4.1 Introduction . . . 46
4.2 Image Feature Extraction . . . 47
4.2.1 Extraction of Image Lines and Corners . . . 47
4.2.2 Graphical Representation of Image Features . . . 51
4.2.3 Transformation of Image Points . . . 52
4.3 Range Feature Extraction . . . 57
4.3.1 2D Line Segmentation Literature . . . 57
4.3.2 Proposed 2D Line Segmentation Method . . . 60
4.3.3 Results of 2D Line Segmentation . . . 69
4.3.4 Linearity Verification of Line Segments . . . 72
4.3.5 Representation of 2D Line Segments . . . 75
4.3.6 Transformation of Range Points . . . 76
4.4 Feature Grouping . . . 81
4.4.1 Feature Grouping Literature . . . 81
4.4.2 Proposed Feature Grouping Method . . . 83
4.4.3 Laser-Image Feature Grouping . . . 88
4.4.4 Image-Image Feature Grouping . . . 92
4.4.5 Laser-Laser Feature Grouping . . . 95
4.4.6 Results of Feature Grouping . . . 98
4.5 Discussion . . . 101
4.5.1 Image Feature Extraction . . . 101
4.5.2 Range Feature Extraction . . . 103
4.5.3 Feature Grouping . . . 104
4.6 Summary . . . 106

5 Fitting Planar Surfaces to Grouped Features 108
5.1 Introduction . . . 108
5.2 Surface Fitting Literature . . . 109
5.3 Overview of Proposed Surface Fitting Method . . . 110
5.4 Infinite Plane Fitting . . . 112
5.4.1 Calculation of Plane Parameters . . . 113
5.4.2 Derivation of Plane Parameter Covariance . . . 118
5.5 Generation of Planar Polygon Boundaries . . . 134
5.5.1 Corner and Line Registration . . . 135
5.5.2 Registration Combinations . . . 137
5.5.3 Merging of Registered Corners . . . 141
5.5.4 Post Processing of Polygon Boundaries . . . 143
5.6 Results of Planar Surface Fitting . . . 148
5.6.1 Infinite Plane Fitting Results . . . 149
5.6.2 Polygon Boundary Generation Results . . . 151
5.6.3 3D Scan Reconstructions . . . 152
5.7 Further Results and Validation . . . 152
5.7.1 Validation Scenario 1 - Clutter . . . 153
5.7.2 Validation Scenario 2 - Walls with Shadow . . . 155
5.7.3 Validation Scenario 3 - Outdoors . . . 156
5.8 Discussion . . . 158
5.8.1 Plane Parameter Fitting . . . 158
5.8.2 Plane Parameter Accuracy . . . 162
5.8.3 Plane Covariance Calculations . . . 163
5.8.4 Polygon Boundary Generation . . . 166
5.8.5 Further Results and Validation . . . 168
5.9 Summary . . . 175

6 Merging of Surfaces to Generate 3D Maps 176
6.1 Introduction . . . 176
6.2 Scan Alignment and Surface Registration . . . 177
6.2.1 Pose Estimation . . . 178
6.2.2 Planar Surface Registration . . . 180
6.3 Infinite Plane Merging . . . 181
6.3.1 Directional Consistency of the Normal Vector . . . 182
6.3.2 Ensuring Normal Vector Remains Normalised . . . 183
6.4 Polygon Boundary Merging . . . 185
6.4.1 Projection of Polygon Boundaries onto Merged Plane . . . 185
6.4.2 Merging of Projected Polygon Boundaries . . . 188

6.5 Surface Merging Results . . . 188
6.5.1 Results for Experimental Dataset . . . 188
6.5.2 Results for Simulated Dataset . . . 190
6.6 Discussion . . . 192
6.6.1 Merging of Planar Surfaces . . . 193
6.6.2 Usage of Planar Surfaces for SLAM . . . 195
6.7 Summary . . . 197

7 Performance Analysis **199**
7.1 Introduction . . . 199
7.2 Computational Performance . . . 199
7.2.1 Computational Complexity Analysis . . . 200
7.2.2 Runtime Results . . . 201
7.2.3 Discussion of the Computational Performance . . . 207
7.3 Discussion of the Level of Sparseness . . . 209
7.3.1 Sensor Sparseness . . . 209
7.3.2 Tilting Angle Sparseness . . . 211
7.3.3 Adaptive Tilting Angle . . . 212
7.4 Summary . . . 213

8 Conclusions and Future Work **215**
8.1 Thesis Contributions . . . 215
8.1.1 Planar Surface Fitting . . . 216
8.1.2 Surface Feature Extraction . . . 217
8.1.3 Feature Grouping . . . 217
8.1.4 Accuracy and Uncertainty of Planar Surfaces . . . 218
8.1.5 Real Time Operation . . . 219
8.1.6 3D Map Construction for Navigation . . . 219
8.2 Future Work . . . 220
8.2.1 Full Implementation on CRACbot Prototype . . . 220
8.2.2 Estimation of Surface Properties . . . 220
8.2.3 Integration of Nonplanar Surfaces . . . 221
8.2.4 Optimisation of Sparseness . . . 221

References **222**

List of Figures **235**

List of Tables **237**

List of Algorithms **238**

Appendix **240**

A Hardware Specifications **240**

A.1 CRACbot Specifications 240

A.2 Image Sensor 245

A.3 Range Sensor 246

List of Abbreviations

2D	Two Dimensional
3D	Three Dimensional
CF	Coordinate Frame
CRACbot	Compact Remote Articulated Climbing Robot
DOF	Degrees of Freedom
EIF	Extended Information Filter
EKF	Extended Kalman Filter
EM	Expectation Maximisation
FoV	Field of View
ICP	Iterative Closest Points
KF	Kalman Filter
LRF	Laser Range Finder
MRDS	Microsoft Robotics Developer Studio
MUMC	Minimally Uncertainty Maximum Consensus
NDT	Normal Distributions Transform
OLS	Ordinary Least Squares
PC	Principal Component
PCA	Principal Components Analysis
RANSAC	Random Sample Consensus
RMSE	Root Mean Square Error
SIFT	Scene Invariant Feature Descriptor
SLAM	Simultaneous Localisation and Mapping
SPmap	Symmetries and Perturbations map
TLS	Total Least Squares
VSE	Visual Simulation Environment
WCR	Wall Climbing Robot

Chapter 1

Introduction

The field of mobile robotics promises to enrich our lives by automating mundane or dangerous tasks. Despite significant advances in recent years, some environments remain difficult to access for most mobile robots using traditional locomotion. Scenarios such as the maintenance of an industrial plant or urban search and rescue in a collapsed building pose many challenges for traditional robotic configurations. Ground based and aerial robots are usually unable to operate in these difficult environments because they often consist of enclosed spaces and numerous obstacles.

A Wall Climbing Robot (WCR), however, can climb onto, over or through most obstacles in its way. These robots are typically small in size but are highly agile and they can move fully through 3D space by attaching themselves to nearby surfaces. This ability, whilst complicating the navigation and path planning processes, allows these robots to undertake a variety of useful tasks in environments which are typically difficult or dangerous for a human or wheeled/aerial robot to access. Therefore, a WCR that could successfully navigate these environments would have obvious appeal. Yan *et al.* [1] echoed these sentiments, stating that '*the potential is enormous for wall-climbing robots which can work in extremely hazardous environments*'.

As these WCRs can move in 3D, they obviously need some kind of 3D mapping system to facilitate navigation and motion. They are unable to use the 2D mapping approaches which are sufficient for most ground based robots. Therefore, they require a surface mapping approach that is able to provide suitable 3D surface in-

formation in real time. Without such a mapping capability, these robots are unable to operate autonomously. This is a requirement for use in many of the enclosed or confined environments where WCRs are beneficial. Generally, the lack of line-of-sight makes remote control operation unreliable or impossible to achieve in these scenarios.

The field of WCRs has seen a number of robots developed in recent years and many different sizes and configurations have been proposed. Small WCRs are of more interest than their larger cousins as they are generally more agile and are able to access smaller spaces. However, because of the small size of these robots their mapping capabilities are generally limited or nonexistent. Most require external assistance for mapping and navigation due to their lack of onboard sensing or computational hardware. Because of this, no suitable method has been developed that fully meets the onboard 3D mapping needs of a typical small WCR. Other robotic mapping methods are either limited to 2D use, are not real time using onboard hardware or they consist of an unsuitable map representation.

The required 3D map should accurately detail a WCR's immediate environment. One of the main tasks for a WCR is to plan motion between footholds and so the generated 3D map must contain enough information to facilitate this process. Most nonplanar surfaces are, however, unsuitable for use as footholds. Therefore the focus of the 3D mapping approach should be the fitting of good quality planar surfaces that accurately represent the surrounding environment.

Small WCRs are also typically payload limited because they must support their entire weight against gravity. This limits the availability of onboard processing and sensing hardware. This restriction is further exacerbated by the need for the surface map to be created in real time. A WCR robot is unable to move, and thus continue to map and navigate, until it has a reasonable environment map available.

This thesis will meet this mapping need by presenting a 3D mapping solution using the smart fusion of sparse data from a camera and a laser range finder. This will provide a real time 3D mapping capability for a WCR. It will focus on the use

of planar surfaces and will be designed to operate under the payload and hardware limitations inherent to small WCRs.

1.1 Aim and Scope of Research

The aim of the research presented in this thesis is **the development of a methodology to provide a 3D surface mapping capability for a wall climbing robot.**

Objectives

In achieving this aim, the following objectives should be met:

1. The 3D mapping method should generate planar surfaces to represent a wall climbing robot's immediate environment.

2. These planar surfaces should be accurate enough to allow a wall climbing robot to use them successfully as potential footholds. The uncertainty of each surface must also be estimated.

3. The 3D mapping method should be computationally inexpensive to allow real time operation using limited computational and sensing resources.

4. The generated 3D map should facilitate the successful navigation of a wall climbing robot through its environment.

Scope

The scope of this thesis is limited to the operation of a small wall climbing robot in a semi-structured environment consisting predominantly of planar surfaces. The custom built wall climbing robot outlined in this thesis uses suction cups to attach itself to surfaces. As nonplanar surfaces are unsuitable as footholds, the 3D mapping method should be primarily focussed on mapping the planar surfaces in the

environment. The representation of any nonplanar surfaces will be discussed in this thesis. However, they will not be implemented in this work.

While the generated 3D maps could be later used within some Simultaneous Localisation and Mapping (SLAM) framework, that is not the focus of this work. Each 3D map should be a local map only and it should accurately describe a wall climbing robot's local environment. Its primary task should be to enable some navigation process to move the robot towards the next desired foothold. The actual implementation of any navigation, motion planning or SLAM framework is outside the scope of this work.

The mapping method also needs to be designed for operation using the limited computational resources available to a small wall climbing robot. Any existing approaches that involve post-processing or computationally intensive calculations are unsuitable. Real time mapping is a requirement and this must be performed using a computationally inexpensive approach. Real time is defined in this thesis as requiring the 3D mapping to be completed in the time taken for the robot to acquire each 3D scan and then move to the next position.

1.2 Thesis Structure

Chapter 2 lays out the background and motivation for this research. The relevant literature is discussed with a focus on wall climbing robots, simultaneous localisation and mapping, 3D mapping and surface representations.

Chapter 3 presents the relevant hardware used in this thesis. The custom built wall climbing robot is introduced along with its primary sensors which are a camera and a laser range finder. This will provide further background and context to the choices motivating the theory developed in the later chapters.

Chapter 4 details the extraction of the range and image features. These are used as the basis for the proposed surface mapping approach. A method is then developed to group these features together that is based on the physical surfaces

which generated them.

Chapter 5 details the approach used to fit planar surfaces, with full uncertainty, to the grouped range features found in Chapter 4. A method to generate polygon boundaries for each surface is then developed using the planar projection of the image features.

Chapter 6 details the merging of individual 3D scans which consist of the planar surfaces extracted in Chapter 5. This allows the construction of a local 3D map about the wall climbing robot's current foothold.

Chapter 7 investigates the computational performance of the proposed surface mapping method. The computational cost and running times are analysed and the level of sparseness of the input data is discussed.

Chapter 8 concludes this thesis by summarising the research performed. The main contributions of this thesis are outlined and future work is proposed.

Appendix A details the wall climbing robot used as the motivation for this work, as introduced in Chapter 3.

1.3 Publications

The following publications have been produced during the period of this research:

- **B. Howarth**, M. Whitty, J. Katupitiya, J. Guivant. "Extraction and Grouping of Surface Features for 3D Mapping." To appear in *Australasian Conference on Robotics and Automation (ACRA), 2011* Dec. 2011

- **B. Howarth**, J. Katupitiya, A. Szwec, J. Guivant. "Novel Robotic 3D Surface Mapping using Range and Vision Fusion." In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on* , pp.1539-1544, 18-22 Oct. 2010

- **B. Howarth**, J. Katupitiya, R. Eaton, S. Kodagoda. "A Machine Learning Approach to Crop Localisation using Spatial Information." In *International Journal of Computer Applications in Technology*, 2010, volume 39, pages 101-108

- R. Eaton, J. Katupitiya, K. W. Siew, and **B. Howarth**. "Autonomous Farming: Modeling and Control of Agricultural Machinery in a Unified Framework." In *International Journal of Intelligent Systems Technologies and Applications*, 2010, volume 8, pages 444-457.

Chapter 2

Literature Review

2.1 Introduction

This chapter presents the existing robotics research in the fields of wall climbing robots and 3D Mapping. It will demonstrate the need for the contributions of this thesis as well as providing the required context and background information. Where appropriate in later chapters, further review of the relevant literature will be provided.

The specific role of wall climbing robots will be introduced and defined within the more general area of mobile robotics. The common robotic mapping approaches will be reviewed, particularly those in the area of 3D mapping and localisation. The ability for these approaches to be adapted for use by a wall climbing robot will be discussed and the need for a new 3D mapping approach will be shown to exist.

2.2 Wall Climbing Robots

Robots have been long used in the areas of manufacturing and automation and in the past few decades they have become increasingly mobile. With the rapid advancements in computational power and robotic hardware of late, mobile robots have become smaller and more autonomous than ever before. Such small mobile robots have become a part of an increasingly popular area of research.

One interesting area of research within the field of mobile robots is that of Wall Climbing Robots (WCRs). Yan *et al.* [1] commented that '*The potential is enormous for wall-climbing robots which can work in extremely hazardous environments*' while Xiao *et al.* [2] noted that '*It has been a long-time dream to develop a miniature climbing robotic system with the ability to climb walls, ... thus transforming the present 2D world of mobile rovers into a new 3D universe.*' Remarks such as these give an indication of the potential benefits available from advancing the state of the art in this area.

These robots are not constrained by gravity. Instead they attach themselves to surfaces, such as walls, and are able to then move around and across any usable surface within reach. This mode of locomotion is quite different to a wheeled or tracked mobile robot, which are constrained by gravity to the ground plane. Wall climbing robots are useful in many situations that preclude the use of a typical ground based robot because they can climb onto and over most obstacles in their way. This useful ability, whilst complicating the navigation and path planning processes, allows these robots to complete useful tasks in environments that are usually unfriendly to robots.

Potential applications include maintenance, surveillance, inspection, exploration and search and rescue operations. Wall climbing robots are especially useful in industrial or cluttered environments and applications include the inspection and maintenance of nuclear and petrochemical storage tanks [1], the unrestricted exploration of urban structures [3] and the welding of large aluminium tanks [4]. These environments are difficult or impossible to access with a ground based robot. Other problematic environments could include a collapsed building, ventilation ducting, the interior of an aircraft wing or some other complicated enclosed structure.

2.2.1 Adhesion Techniques

Many wall climbing robot designs have been proposed in the literature. The primary differences in design involve the robot configuration and the adhesion technology

used. To an extent these design considerations are often influenced by the intended purpose and environment of the robot.

The use of magnetic adhesion is a simple process. However, its use is limited to ferrous surfaces. This has restricted its use to robots specifically designed for a particular industrial site. Yan *et al.* [1] described the development of two wall climbing robots, one of which used permanent magnets to attach to oil tanks for the purpose of sandblasting and spray painting.

The use of vacuum seals is a common approach to adhesion for wall climbing robots. This can take the form of static or dynamic vacuum approaches. Static vacuum involves the use of a vacuum pump to create a static vacuum seal which then requires only infrequent use of the pump to compensate for leakage. This leads to lower power consumption. Dynamic vacuum maintains a constant pressure and is in continual operation. The advantage is that the suction cup or skirt can slide and so the robot can drive around using wheels or tracks but at the cost of higher power consumption. This was used by Tang *et al.* [3] for their city climber robots.

A hybrid vacuum approach was proposed by Taylor *et al.* [4]. Their TigBot used static suction in combination with a low friction seal on each suction cup. This allowed the cup to be dragged over the surface in a smooth manner using wheeled locomotion. This robot was used for the welding of tanks and so needed to be able to drive at a smooth and constant rate. This hybrid approach was suitable for this task as the tanks were a known and constantly flat surface. This approach does not generalise well as it is unable to transition between surfaces. This is also a weakness of the dynamic vacuum methods.

A different method is inspired by geckos which are natural climbers and this approach was used by Menon *et al.* [5]. They have designed a climbing robot whose body, gait and adhesion is based on the gecko. Their dry adhesion pads used micro fibres to produce synthetic gecko feet. The adhesion is based on van der Waals forces and so should work on a variety of surfaces in theory, but more development is needed to verify this potential.

A promising recent approach called electroadhesion has been proposed by Prahlad *et al.* [6]. This technique induces electrostatic forces between the wall and its electroadhesive pads. This allows for high adhesion forces and works for a variety of surface materials. It does require a high voltage, but uses relatively little power.

2.2.2 Locomotion Configurations

The chosen method for locomotion differs significantly between robots in the literature. However, most can be grouped into either legged, wheeled or tracked locomotion. In many cases this choice is a function of the method of adhesion used.

The use of wheels or tracks with a movable adhesion method allows for common mobile robot designs to be quickly modified to create a wall climbing robot. It also allows for simplified motion planning, control and navigation, as general methods can often be easily adapted. The downside is that the robot is often less agile than a legged robot and can have difficulty in transitioning between surfaces with different orientations.

Dynamic vacuum lends itself to continual motion and so most approaches use wheels or tracks. Examples of this design can be found in the work of Yan *et al.* [1] where the first of two robots they propose has wheels inside its vacuum skirt for compactness. The CityClimber robots of Xiao *et al.* [2] and Tang *et al.* [3] also use this design. The TigBot of Taylor *et al.* [4] follows a similar approach but uses a static vacuum seal with very low friction and external wheels. A comparable locomotion approach is seen in the work of Prahlad *et al.* [6] where electroadhesive pads and flaps are attached to tracks.

Legged locomotion, whether bipedal or using more legs, is a more complicated configuration but provides for more flexibility and agility in motion. The simplest approaches have distinct bodies and multiple legs. An example is the four legged robot of Menon *et al.* [5] built to mimic a gecko. These robots are generally larger and heavier than other wall climbing robots. This is due to the extra complexity of having many legs and the extra actuation and control hardware needed to support

them. This approach also suffers from the same limitations in terms of agility as wheeled wall climbing robots.

A more agile configuration, such as a bipedal legged robot, is ideal for general use in enclosed and complicated environments. Minor *et al.* [7] proposed a bipedal design based on four joints. Its configuration involved two legs joined at the centre with a revolute hip joint. They showed that a revolute hip joint provided the most agility in being able to cross the maximum number of potential surface orientations. In confined spaces a prismatic hip joint provided better mobility. Their robot only had four joints and was under actuated to save weight. This limited its gait to a flipping or inching motion.

The prototype climbing robot developed by Ward *et al.* [8] extends this bipedal concept to six joints allowing greater flexibility in motion and gait. However, this does further complicate the process of navigation and motion planning due to the high number of degrees of freedom.

2.2.3 Mapping Capabilities

For a WCR, or indeed any robot, to be of some practical use it requires the ability to perceive and interact with its environment. Interaction generally involves the locomotion of the robot or the use of some implement or tool such as a robotic arm. The focus of this thesis is on the perception capabilities of WCRs and so this interaction will not be further investigated in this work.

The perceptual capability of a robot involves the sensing of its environment and the acquisition and interpretation of pertinent information about it. The major areas of robotic perception involve the mapping of an environment, keeping track of the robot's location within that map and the acquisition of task specific data.

In most WCRs such perception is minimal or nonexistent, as for many the primary focus has been the physical robot design and construction. The integration of sensors and perceptual capabilities to allow navigation is often ignored or proposed as future work [7]. The most basic level has involved operation via remote control

with a human operator using a video camera feed or through direct line-of-sight [1]. For many useful activities this is a severe limitation.

The lack of autonomy eliminates one of the main advantages of robotics in automating monotonous tasks and freeing up human labour for more productive pursuits. Many of the specific environments in which a wall climbing robot would be useful precludes such line-of-sight human operation. Video and other wireless telemetry may also broadcast poorly due to interference from the structure. This was seen in the search and rescue robots deployed during the World Trade Center disaster, as detailed by Casper *et al.* [9].

Some WCRs have introduced small amounts of autonomous activity, such as with the TigBot [4]. While they used remote video for control, they did introduce a line following capability to track the weld seam on the tank being welded. Meiting *et al.* [10] used a camera on their wall cleaning robot, but its use was limited to obstacle avoidance. Tang [3] used stereo vision to fit planar surfaces for mapping. This procedure was done offline and would take further work to integrate it onto their CityClimber robot, especially as stereo vision is generally computationally expensive. However, their robot is larger than most WCRs and it may be able to house more processing hardware. This extra size comes at the expense of mobility and agility.

For most small WCRs, a typical computer is too big and heavy to be used onboard due to payload restrictions. A miniature computer module must be used instead. The Gumstix [11] Verdex Pro board is used in this thesis and has a clock speed of 600MHz. It is detailed in Appendix A and it is at the top end of miniature computers currently available.

An alternative mapping approach is to use ground based robots to aid in the map building process. They can either fully map the scene or fuse maps and sensor data with those from the wall climbing robot. Xiao *et al.* [12] uses this approach with the CityClimber robot as an alternative to the mapping method of Tang [3]. However, this method is not applicable to many useful scenarios because the WCR

may need to operate alone or be out of line-of-sight of the ground based robots.

The use of reactive navigation, as seen in the work of Balch [13] and Arkin [14], is not a feasible alternative to environmental mapping for WCRs. While it may be viable for some 2D robotic navigation scenarios, the nature of the WCR 3D navigation problem precludes its use. These robots often need to locate and carefully plan their 3D motion to discrete foothold locations. This is difficult to perform using a reactive approach as it can be inefficient, which can waste limited onboard power, or possibly lead to a catastrophic collision or fall.

In general the mapping and perceptual methods currently used by small wall climbing robots are limited at best. This is especially true where the robot is payload limited yet needs to navigate in real time, which is the case for the robot used in this thesis. No methodology currently exists to build a real time 3D map for a WCR using limited computational resources. A 3D map is necessary to allow a wall climbing robot to plan and navigate between potential footholds, which in some environments may be few and far between.

2.3 Simultaneous Localisation and Mapping

The ability of a mobile robot to perceive its environment autonomously, and navigate within it, raises two issues. Firstly, some representation or map of the environment needs to be constructed. Secondly, the robot must keep track of its current location. These are the mapping and localisation problems respectively. These two issues are connected and must be considered in parallel in order to obtain a consistent solution for either. This combination led to the development of Simultaneous Localization and Mapping (SLAM) methods.

The SLAM problem was first tackled over two decades ago with seminal papers by Smith *et al.* [15][16] and Durrant-Whyte [17]. They outlined the relationship between the mapping and localisation problems and also laid out the basis for the SLAM solutions so widely used today. The mapping half builds a map estimate based on some estimated pose. The localisation then determines the new pose

estimate based on the map estimate. This duality means that errors in one half will feed through to the other.

2.3.1 2D SLAM

The majority of SLAM research has been limited to the 2D case. Most robots are constrained to the ground plane and therefore 2D SLAM is sufficient assuming the ground is approximately level. It is also significantly less complex to map and localise in 2D, compared to the 3D case. Whilst recent research has moved SLAM into the third dimension, the 2D SLAM case is still instructive for an overview of the SLAM problem, despite it being insufficient for a wall climbing robot. 3D SLAM is discussed later in Section 2.3.2.

The general SLAM approach involves a two step cycle of map building based on the current robot pose estimate and then a pose update to localise the robot based on the new map estimate. The most common approach is feature based SLAM where the map consists of a set of features or landmarks. These landmarks might be points, lines or other interesting features. They should be able to be easily re-observed from multiple viewpoints and be sufficiently unique to be easily distinguished from one another.

The landmarks are tracked and their position estimate is updated using a Bayesian filter such as the commonly used Extended Kalman Filter (EKF). The robot pose is also estimated as it is correlated to each landmark through the mapping process. A full 2D SLAM solution of this form is provided by Dissanayake *et al.* [18]. For a more detailed overview of the background and theoretical basis of SLAM, refer to the tutorial articles by Durrant-Whyte *et al.* [19] and Thrun [20].

As Durrant-Whyte *et al.* [19] notes, in a theoretical sense 2D SLAM is considered a solved problem. There are, however, many practical issues and problems that are the focus of further research. The computational complexity of the basic EKF approach scales poorly as the number of landmarks increases. Many optimisations have been proposed to improve the efficiency to allow practical real time implemen-

tations. Guivant [21] and Leonard [22] proposed improvements such as suboptimal local submaps and more efficient reformulations of the EKF equations.

Others have proposed alternative probabilistic filters to replace the standard EKF for SLAM. These included the Unscented Kalman Filter [23], the Extended Information Filter (EIF) [24] and the Particle Filter [25], which is based on Monte Carlo methods. SLAM using a particle filter, or FastSLAM, is the most popular alternative to EKF based SLAM. It represents the probability distributions by a number of particles which allows the probability distributions to be more accurately represented, if enough particles are used. This allows for the tracking of multiple hypotheses, unlike the EKF which is limited to a unimodal Gaussian distribution. FastSLAM also scales much better than EKF SLAM but can diverge from the correct solution in some cases if care is not taken.

An alternative approach to landmark based SLAM is called scan matching. This approach estimates the pose of the robot by attempting to align a new scan with the current map to determine the change in rotation and translation since the last scan. This localises the robot pose and allows the new scan to be merged with the current map. Scan Matching is commonly performed using the Iterative Closest Points (ICP) algorithm [26] which attempts to find the rotation and translation by minimising the distance between matched points in each scan. This is an iterative solution and can be computationally expensive if the point cloud is large, especially in 3D SLAM where it is more commonly applied.

Two dimensional SLAM does suffer from some limitations, mainly that it is limited to application in a 2D world only. This is fine for many applications, particularly wheeled or tracked mobile robots that operate on level ground. However, if the ground is sloping, uneven or, if the robot is moving fully in 3D, then 2D SLAM is insufficient.

The maps generated and maintained by most 2D SLAM approaches consist of a sparse set of landmarks and the local environmental knowledge provided by this representation is limited. It may be necessary to overlay a separate 2D or 3D

mapping implementation to allow for some higher level tasks such as navigation in complex environments. The denseSLAM algorithm was introduced by Nieto *et al.* [27] which incorporated a dense local map representation within the 2D SLAM framework. However, this approach does not appear to be well suited for extension into 3D SLAM.

2.3.2 3D SLAM

To localise and map a robot that moves in 3D, it is obvious that a 3D SLAM implementation is needed. Recent research has shown the successful development and implementation of such methods. Like 2D SLAM, there are two main approaches. They are feature based methods and scan matching methods. Both can be considered to be extensions of their 2D counterparts as introduced previously.

Feature Based 3D SLAM

Three dimensional feature based SLAM involves the extraction of 3D landmarks which are tracked and estimated using a probabilistic filter such as the EKF, similar to that of 2D feature based SLAM. In 3D the feature extraction and registration is much more complex than in the 2D case. The features used are often visual in nature as they are easier to match than range point features. Approaches using only range data generally seek to extract planes, lines or other features for the same reasons.

Early work using visual features was done by Se *et al.* [28] who proposed an approach to 3D SLAM that used visual point features from a stereo vision system. The point features used Scale-Invariant Feature Transform (SIFT) descriptors [29] that allowed them to be matched from multiple poses. These are expensive to compute on top of the stereo vision computation and so using visual feature matching is not a viable option for this work because of this.

3D single camera SLAM has been demonstrated by Davison *et al.* [30] which tracks features via their bearing angle and requires multiple views to estimate the

feature position. The features used were image patches and showed good results for a moving camera. This approach was demonstrated on a humanoid robot by Stasse *et al.* [31]. It works best with a smooth sensor motion and a high frame-rate. It is, nevertheless, poorly suited to the discrete stop-scan-go approach that is commonly used by many mobile robots. The scans are too infrequent and often have minimal overlap.

An alternative feature based approach proposed by Ellekilde *et al.* [24] used a 3D range camera for range estimates of visual point features extracted from a separate camera. This eliminated the need for expensive stereo vision calculations. However, these range cameras are generally noisier than laser range finders. They used an EIF with SIFT features in a similar manner to Se *et al.* [28]. The EIF is mathematically similar to the EKF but is reformulated using the inverse covariance matrix.

Rather than use point features, Kohlhepp *et al.* [32] proposed and demonstrated a method for using planes in 3D SLAM. They did not, however, fully integrate the planes within the EKF as features and chose to use a separate 3D mapping system instead. Weingarten *et al.* [33] extended this approach by fully incorporating the planar surfaces into the EKF as landmarks for 3D SLAM. This was combined with the Symmetries and Perturbations Map (SPmap) framework of Castellanos *et al.* [34] to represent the planes in the EKF. This approach proved successful for a ground based robot but was most suited to structured environments with a few large planes. It was based on infinite planes as the features but they later introduced planar polygon segments [35] which aided in the data association.

A similar approach was also used by De la Puente *et al.* [36] and Zureiki *et al.* [37] with similar results. They each proposed different methods of segmentation for the extraction and fitting of the planar features. Zureiki *et al.* [37] also proposed the idea of using infinite 2D lines from a camera image to produce additional features. However, this was not implemented in their work. This idea will be explored further in this thesis.

The approach of using planar features has the potential to greatly reduce the

data dimensionality compared with the original point cloud. It can also smooth the noisy raw data and is generally easier to visualise. It is useful for autonomous perception tasks such as navigation and footstep planning for wall climbing robots. However, this data reduction comes at the cost of reducing the information available, as the plane fitting is an approximation process. This approach is especially dependent on having a robust plane extraction method and is most suited to structured environments such as urban scenes. Nonplanar surfaces such as vegetation and other irregular objects can cause difficulties, as they cannot be easily handled by plane fitting.

The planar segmentation techniques currently used are generally computationally expensive and therefore would struggle to run in real time on some small robots. If a computationally inexpensive segmentation approach could be developed then the use of planar features as a map representation would be an ideal map representation for any robot that has computational constraints.

Scan Matching 3D SLAM

The second main approach used for 3D SLAM is scan matching. This is an extension of the 2D scan matching approach into 3D. It is the predominant approach when the sensor used provides 3D range data in the form of a point cloud. Scan matching involves acquiring a new 3D scan and then attempting to determine the rotation and translation that has occurred since the previous scan. The new scan can then be aligned and with the previous points and merged into the map to update it. It is often preferred over 3D feature based SLAM due to the difficulty in extracting and matching good 3D features. Scan matching, however, can operate with just the raw point cloud.

Surmann *et al.* [38] demonstrated an early implementation of this approach in an indoor environment while Nuchter *et al.* [39] used the same approach for the mapping of an abandoned mine. They used a variation of ICP to allow real time operation. The underground mine environment was feature poor, and so feature

based SLAM would likely fail. Scan matching SLAM approaches are, however, very useful in such environments.

This method was further developed by Nuchter [40] by also looking into optimisations to reduce the computation time. This is especially important as the major disadvantage of scan matching is that it can be computationally expensive in 3D, especially for large point clouds. However, despite such optimisations, the computation required is still prohibitive for a computationally constrained robot, such as the one used in this work.

Another weakness of scan matching is that it generally requires significant overlap between successive scans which does not always occur. Borrmann *et al.* [41] focussed on ways of ensuring the map remains globally consistent. This tries to avoid a common problem for scan matching, where the global error grows with each sequential match, as the new scan is only compared with the latest estimate.

An alternative method for scan matching is the Normal Distribution Transform (NDT) [42]. It is used by Magnusson *et al.* [43] in a mine mapping 3D SLAM implementation and they compare their results using NDT with ICP. They showed that NDT was faster and slightly more reliable than ICP, despite it being less widely used for scan matching.

Some researchers have used planes rather than points as the basis for scan matching. The aim was to greatly reduce the computational burden as a few planes are easier to match and align than a large point cloud. The plane extraction process must be robust and accurate for this to work. Viejo *et al.* [44] modified the ICP algorithm to work with planes extracted from the raw range point cloud. They first determined the rotation between plane sets based on the difference in plane normal vectors. The translation between scans is then calculated separately. However, the pose error was significant and their method did not incorporate all six degrees of freedom (DOF) required for true 3D SLAM.

Harati *et al.* [45] proposed a 3D SLAM method based on orthogonal corners which showed promise in highly structured urban environments but would be un-

suitable in general. They encoded corners based on their orientation. Due to the orthogonality assumption the number of possible encodings was small and therefore the scan matching is greatly simplified. The rotation was determined by matching and comparing the corner encodings and all that remained was the translation offset. Pathak *et al.* [46] found the optimal rotation by maximising a value function based on the angle between normal vectors of the planes. These methods all followed a similar path of using only the infinite plane parameters for scan matching while ignoring other useful information such as polygon edges, corners and other surface descriptors.

3D SLAM Limitations

In many of the example implementations of 3D SLAM the map produced is insufficient for further high level use in tasks such as navigation. This is especially true for a wall climbing robot that requires knowledge of available planar surfaces to act as footholds. The methods based on planar rather than point features appear to be the way forward in this regard. SLAM methods using point features could, however, be used to track the pose and allow a planar 3D map to be built on top.

Another issue involves the computational requirements of these approaches, particularly those based on visual features, stereo range imaging and scan matching for SLAM. These methods are all computationally expensive. This is fine in many cases where the robot is large and ground constrained and so weight and computational power are not a constraint. However, if onboard computational resources are limited, these methods are unusable because real time operation would not be possible.

2.4 3D Mapping

It is often necessary to implement some form of useful 3D map on top of the SLAM algorithm due to the limitations of the map produced. The exception to this is the planar 3D SLAM approach. The 3D maps are commonly used for higher level tasks

such as navigation, visualisation, object recognition or footstep planning for a wall climbing robot. For these tasks, SLAM maps consisting of topological features or point clouds are insufficient.

2.4.1 Point Clouds

The simplest approach to producing a 3D map is to display the raw 3D points as a point cloud. The 3D data in a point cloud is the starting point for most 3D mapping approaches such as grid, mesh or planar based methods. The raw 3D data is supplied by a 3D sensor and several different configurations have been suggested. A 2D laser range finder facing vertically can acquire 3D information as the robot moves around [47][48]. This requires a separate 2D navigation system and is only applicable to ground based robots. This configuration is also unable to map surfaces directly in front of the robot without deliberately turning to scan it.

More recent approaches have used sensors that provide individual 3D scans which can be used in a stop-scan-go fashion. Early 3D laser range finders were custom built and so performance was varied, an example of which was used by Kweon *et al.* [49]. These were built with the pan and tilt integrated within the scanner. However, they often had a limited field of view and, due to their custom nature, they were not fully developed. More commonly a tilting 2D laser range finder is used in a nodding fashion to generate the 3D point cloud [39][33]. This allows well characterised 2D laser range finders to be used and provides a large field of view. The weakness of this approach is the length of time it takes to generate the 3D scan as many 2D scans are needed.

Stereo Vision [50] can be used to generate a point cloud in 3D from a single frame per camera pair. This does however, require a lot of extra computational effort. Recently developed 3D time-of-flight cameras provide a consistent 3D scan per frame [51]. These camera based approaches suffer from higher noise and a smaller field of view than the tilting laser range finder configurations.

Once the point cloud from a scan has been acquired it needs to be registered

with the rest of the 3D data. The 3D SLAM approach to this problem commonly involves scan matching [39] to track the pose and register the new data directly. In this case the map used is the point cloud itself which is suitable for basic tasks and human visualisation. Other approaches, such as that of Mahon *et al.* [48], use the pose from EKF based 2D SLAM to register their 3D point cloud data and then build a more useful 3D map from the raw points.

While a point cloud provides a useful visualisation of the data on its own, it can be enhanced in several ways. Underwood *et al.* [52] built a coloured 3D point cloud by assigning colour to each range point. They aligned a camera image with the range data from a laser range finder for the purpose of easing the visualisation of the point cloud. A simple extension to point clouds is to build a triangular mesh by joining the raw data points together. This also eases the visualisation of the raw data by giving it a more solid appearance. It does nothing to reduce its dimensionality and unlike plane fitting it does not smooth the sensor noise which can lead to a jagged appearance.

This approach was used by Sequeira *et al.* [53] to model indoor environments. They used these meshes to register successive scan. However, the accuracy of this method was not thoroughly investigated. Another weakness of using meshes is that it cannot easily deal with holes such as doors or windows in the point cloud. Mesh and point based approaches are best suited to mapping and modeling irregular shapes rather than structured planar environments.

Pervolz *et al.* [54] created textured 3D maps by building a mesh from the point cloud and then used colour information from cameras to map texture onto the mesh. This provided good visualisation of an irregular indoor scene for a human viewer. Frueh *et al.* [55] used this same approach but for a large outdoor urban scene. They used a vertical laser range finder attached to a car rather than the nodding laser range finder used by Pervolz. Bok *et al.* [56] applied texture mapping to their 3D reconstruction on an outdoor urban scene but their work was mostly focussed on using this for motion estimation.

This same idea of using camera image information to fill in the gaps of the laser range finder point cloud can be generalised to increase the resolution of the entire point cloud, rather than just in some areas. Diebel *et al.* [57] used Markov random fields to use high resolution camera images to greatly improve the resolution of the range point cloud. This approach essentially interpolated between range points based on the smoothness of the image colour data. High resolution coloured point clouds were produced that showed good visual quality. No ground truth information was used to determine the accuracy of this approach.

Andreasson *et al.* [58] took a broader look at using vision to interpolate 3D laser range finder point clouds. They quantitatively compared five different interpolation methods and the method used by Deibel *et al.* [57]. These approaches greatly increase the dimensionality of the data and are more suited to visualisation tasks rather than robotic navigation, especially for smaller robots with limited resources. The effects of pose error and scan registration between point clouds was not discussed as each scan was taken from a known fixed point.

A passive triangulation approach was used by Dias *et al.* [59] to add extra points into a laser range finder point cloud. They used this technique to provide extra detail to improve the overall model in areas that were of high interest or were occluded. Jiang *et al.* [60] generated two point clouds from a stereo vision system and a laser range finder respectively. They then used an energy minimisation framework to fuse the two point clouds. As with the interpolation methods of Diebel *et al.* [57] and Andreasson *et al.* [58], these methods were applied to modeling tasks and the accuracy compared to ground truth is unknown as only qualitative visual comparisons were made.

The use of multiple sensors, particularly a camera and a Laser Range Finder (LRF), has shown promise in improving the mapping capabilities of a robot. The use of this sensor fusion approach has been limited to use in improving a model's visualisation by texture mapping and resolution improvements. These are primarily useful for 3D modeling applications rather than navigation. The use of such sensor

fusion is relatively under used in 3D mapping methods aimed at real time navigation.

2.4.2 Grid Based 3D Maps

An early approach for robotic mapping was the occupancy grid. It was described by Moravec for the 2D case [61] and a later extension to the 3D mapping case [62] was demonstrated. This approach involves dividing the world into a grid of discrete cells and then calculating the probability that a cell is occupied or free based on sensor measurements. The 3D case is of more interest but has extremely high memory requirements and is inefficient as most grids are free space. It suffers from accuracy issues depending on the grid resolution and, similar to a point cloud, is difficult to use for higher level 3D tasks. Multi-Resolution grid approaches have been proposed [63] to reduce the computational requirements. However, the other limitations of the grid approach remain.

Elevation maps are a more useful grid based mapping approach whereby each cell contains the height of the environment at that point. This is a 2.5D mapping approach and cannot handle vertical surfaces or overhanging surfaces such as bridges. Extensions have been proposed by Pfaff *et al.* [64] to label the cells and treat vertical and overhanging surfaces differently while Triebel *et al.* [65] allowed each cell to have multiple patches. All these grid based approaches are not true 3D mapping methods and are only suitable for use by a robot driving on the ground. The limited resolution and high dimensionality also limits their use by a wall climbing robot, especially as the grids only work when the robot is parallel to the ground plane.

2.4.3 Planar 3D Maps

The use of surface features such as planes, rather than a 3D point cloud, shows a lot of potential in providing richer 3D maps. Birk *et al.* [66] argued that a paradigm shift is required to move from using point clouds based maps to surface based representations stating '*...large surface patches are ideally suited for 3D scan registration as a core element for 3D SLAM.*'. They discuss important advantages such as a high

level of compactness as well as good suitability for use with computational geometry and higher level problems such as navigation and motion planning.

Some recent 3D mapping approaches have attempted to extract features or objects from the data to both smooth and simplify the map for the ease of higher level perception. As mentioned previously, this approach has been successfully integrated into the SLAM problem. Other examples of planar map building exist based on known or 2D SLAM pose information. The most common feature based 3D mapping approach involves attempting to fit planar surfaces to the raw range points. This is essentially a segmentation problem, with the plane fitting itself being relatively trivial.

The Expectation-Maximization (EM) Algorithm has been adapted to the mapping problem by Liu *et al.* [67]. EM iteratively registers data points from the 3D point cloud to the most likely planar surface in the map and then recalculates those surfaces with a better fit to the registered points. This was used as a pure mapping process as the robot pose was assumed known. This method produced a good planar map, however, it had several weaknesses. Some assumptions were required about the number of surfaces used. If this initial model is wrong, then EM produces a poor segmentation. The risk of local convergence of the iterative algorithm compounds this issue.

As a solution to these problems, Liu *et al.* [67] proposed to use an outer EM loop that will split and merge the surfaces to iteratively settle on the correct number of scene surfaces. This was further developed by Lakaemper *et al.* [68] who used patches instead of infinite planes within the EM framework. These approaches were computationally very expensive and could only realistically be achieved as a post-processing operation. This weakness makes EM unsuitable for many common robotics tasks that require some form of real time navigation or perception.

An extension to EM based segmentation using both a laser range finder and a camera was proposed by Andreasson *et al.* [69]. They assigned colour to each range point and used this extra information to aid in the association of correspondences

when assigning points to planes. This is based on the assumption that points generated from a surface will share a similar colour which in many cases holds true. This assumption is especially useful for distinct planes that may lie close together with similar orientations and it should provide more robust matching. However, it still suffers from the same computational limits as other EM approaches.

A region growing approach to segmentation was used by Hahnel *et al.* [47] to simplify their original 3D map which consisted of a mesh of points which had been built using scan matching. Their region growing approach was based on random seed points which then added additional neighbouring points that appeared to lie on the same plane. The best planes from this random process were kept and the rest were rejected. This was followed by a merging process.

Their segmentation was performed on the entire 3D point cloud, which consisted of hundreds of thousands of points. This was a function of the scanning configuration used where a vertical facing laser range finder provided the 3D data. It had to be run offline due to its high computational cost. This reduced the complexity of the model but it still consisted of several thousand planar surfaces as well as the points that were not matched to planes, which were kept unaltered. This resulted in the map looking visually simpler but remaining still very complex. A similar mapping approach was also used by Mahon *et al.* [48] who also used the same laser range finder setup.

Poppinga *et al.* [70] extended this work and reformulated the algorithm to run incrementally. They used a 3D range camera which produced 3D range and intensity images. This allowed the algorithm to run more efficiently by processing one 3D scan at a time and taking better advantage of the sequential nature of the data. Other region growing approaches have been used for planar segmentation including the work of Zureiki *et al.* [37]. They segmented each individual 3D scan by estimating local normals for each point based on its immediate neighbours and then associated points with similar normals.

The approach of de la Puente *et al.* [71] was based on computer vision tech-

niques. Planes were fitted to each point and its neighbours. The residual of the fit was stored to create a new range residual scan where edges could be detected as local maximums. A flood-fill algorithm was then used to assign points within these boundaries to the same surface. They extended their approach to handle spheres and cylinders as well as planes which allows for a more accurate representation of some objects. This was not actually implemented for surfaces other than planes and they did not mention how they would be distinguished during the fitting stage.

Weingarten *et al.* [72] developed a segmentation methods based on dividing the point cloud into a grid and then fitting one plane to the points within each grid. A Random Sample Consensus (RANSAC) [73] approach was used to find the dominant plane per grid and then a merging operation combined similar planes in neighbouring grids. This was reasonably fast, however, the accuracy relied on the grid resolution being suitable. It was also limited to one plane per cell which could lead to smaller surfaces being completely overlooked.

2.5 Summary

The literature shows many successful methods for building a 3D map for robotics use. For the most part, however, they are all unsuitable for use by a small wall climbing robot. Such a robot requires a true 3D mapping method that can be used from any pose. It should also be computationally light so that it can run in real time with the limited onboard processing capabilities of such a robot. All the wall climbing robots described in the literature had limited mapping capabilities and therefore a new method needs to be developed to build a useful 3D map.

All the mapping approaches based on post processing of the data for map building are unusable. A wall climbing robot needs to be able to map in real time so that it can move and continue to build its map. Those methods that did run in real time were operating on powerful processors and are still computationally very expensive. They would not be suitable where processing resources are at a premium. This includes methods based on stereo vision data and expensive segmentation methods.

As the wall climbing robot used in this thesis requires a map of flat surfaces to evaluate potential footholds, all the methods based on point clouds and meshes are inadequate. It appears that using planar features is the best way forward and successful planar SLAM approaches have been demonstrated by Weingarten *et al.* [33] and Pathak *et al.* [74]. The main issue with their methods lie with the cost of their segmentation and planar feature extraction processes. There exists no computationally cheap method for performing this segmentation.

The use of fused laser range finder data and camera image features shows promise in improving mapping techniques. However, in the literature its use was mainly limited to improving the robustness or visualisation of the data and mapping. There does appear to be potential for this information to be fused in other ways to produce lightweight segmentation and mapping methods and this is an under explored area of research. The use of these sensors, and the fusion of their data, will be introduced in the next chapter. Further background on the design of a wall climbing robot will also be provided and this will allow the theoretical contributions of later chapters to be placed into context.

Chapter 3

A Robotic System Requiring 3D Mapping

3.1 Introduction

The previous chapter detailed the literature relevant to Wall Climbing Robots (WCRs) and the generation of 3D surface maps for usage in robotics. This chapter will detail the required background information regarding the robotic hardware used in this thesis. This will allow the theoretical and practical contributions of later chapters to be placed in the proper context.

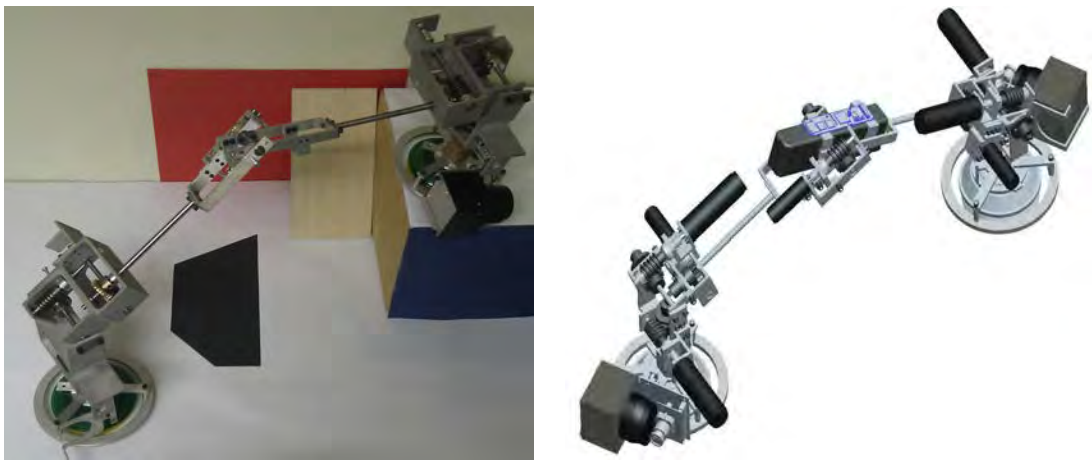
The target robotic system will be introduced, along with its design choices, and the resulting implications on the mapping requirements will be discussed. An experimental test setup is outlined which simulates the desired robot scanning motion and the data acquisition process is described. A simulator is also introduced to complement the real world experimental datasets. The two main sensors used, a camera and a laser range finder, are described and the reasons for choosing each particular sensor are discussed. Finally, the extrinsic calibration between the two sensor coordinate frames is described to allow their data to be fused together.

3.2 Wall Climbing Robot Prototype

A series of custom WCRs have been designed and built for operation in indoor environments. These have served as the motivation for this thesis which has imposed several design constraints on the proposed mapping approach. These are detailed below in Section 3.2.1. A brief overview of the robot is presented below and further information and specifications can be found in Appendix A.1.

Wall Climbing Robots are not constrained to the ground like most common mobile robots and have many useful capabilities as discussed previously in Section 2.2. Their 3D motion does introduce new problems and these will be discussed in this section to provide a context for the 3D mapping solution that is proposed in later chapters.

The current generation prototype robot can be seen below in Fig. 3.1. It is the second generation prototype and has evolved from the first generation robot designed by Ward [75]. The main improvements developed for the new prototype include the addition of onboard electronics and sensors as well as improvements to the joint mechanisms, gearboxes and feet designs. The majority of this redesign was performed by King [76] and Szwec [77] in collaboration with the author and so the robot design itself is not considered as a contribution of this thesis.



(a) CRACbot. (b) CAD model.

Figure 3.1: Wall climbing robot - Current prototype.

The current generation robot has been named the Compact Remote Articulated

Climbing Robot (CRACbot) and is shown in Fig. 3.1(a). It is under construction and is missing several pieces of hardware in this photo. A CAD model is also shown in Fig. 3.1(b). The CRACbot has seven revolute joints to provide six DOF of motion, as seen in Fig. A.2 of Appendix A.1, and it is an entirely self contained bipedal robot. It has two feet, each containing a suction cup, that are used to climb walls or to walk on any other suitable flat surface. This configuration provides a high level of agility and flexibility to achieve any desired pose within reach. This robot has been designed to maximize the ability to access confined spaces. This will allow it to negotiate difficult environments such as air ducts, pipelines and industrial plant that might contain inclined and vertical surfaces.

The CRACbot is able to move in 3D space. Therefore, a 3D mapping implementation is needed. This will have to provide the locations of suitable potential foothold surfaces to enable navigation and motion. In some cases it is possible to provide such a robot with a prebuilt environmental map but this is often impractical. Such an approach would greatly limit the range of potential applications and significantly increase the time and cost to deploy the robot in each new environment. It would also require that the environment match the supplied map exactly and any changes, however small, would require a new map to be constructed for the robot. The ability to construct its own 3D map in real time is thus a significant advantage.

To generate useful environmental data the CRACbot is equipped with one colour camera and one laser range finder mounted on each foot. During operation it can secure itself via one foot, with the other foot free to make observations of the environment. The free foot, with the camera and laser range finder mounted, can rotate to achieve any desired scanning position within its workspace to scan the environment to allow the construction of a 3D surface map.

For further information and detailed specifications of the CRACbot, refer to Appendix A.1.

3.2.1 Design Implications

The design of the CRACbot has implications for the required 3D mapping methodology. This impacts the choice of the robotic hardware, particularly the sensors, and also any mapping approach used. The robot can scan surfaces while it is attached to them so the ranges of interest are typically less than 600mm. As the robot walks on these surfaces, rather than just mapping their location, the need for a highly accurate 3D surface mapping system becomes apparent.

Like most wall climbing robots, the CRACbot is small and payload limited. Its entire weight must be supported against gravity by the suction cups and thus the sensing and processing hardware mass must be minimised. This leads to a major design consideration that the algorithms used must be lightweight and aimed at operating using the limited onboard computational resources available. This would also allow such mapping methods to be generalized for use in other small robots or autonomous vehicles with similar constraints.

Relation to Thesis Objectives

The payload and computational constraints of the CRACbot led to the choice of using sparse, low resolution data acquired from both a laser range finder and a colour camera. As mentioned in Section 1.1, the aim of this thesis is to create a mapping method to produce 3D surface maps of a sufficient quality to enable the CRACbot to navigate successfully between footholds. This method must be able to operate under these computational constraints by intelligently fusing these two sensor modalities.

The generated 3D surface map needs to contain the majority of planar surfaces in the environment. Nonplanar surfaces are unsuitable for usage as footholds because the suction cups are unable to successfully attach to them in most cases. The planar surfaces need to be mapped to an accuracy of $\pm 3^\circ$ in orientation and ± 20mm in distance. These values are a function of the tolerances built into the CRACbot foot mechanism as a small amount of error can be tolerated during footstep placement.

3.3 Data Acquisition

While the wall climbing robot described above is the motivation for the surface mapping approach presented in this thesis, it was also necessary to acquire simulated environmental data. An experimental setup and a simulator were developed for initial testing and validation of the proposed surface mapping approach. This allowed the data to be acquired under carefully controlled conditions with a known ground truth for each surface. The pose of the sensors could also be determined accurately which allowed the mapping theory to be developed independently of any particular localisation method used. The intended scanning motion of the climbing robot was to use the final wrist joint to tilt the sensors through one angular DOF θ and provide a 3D scan, as shown in Fig. 3.2. This was the action that has been emulated by the experimental and simulated setups as described below.

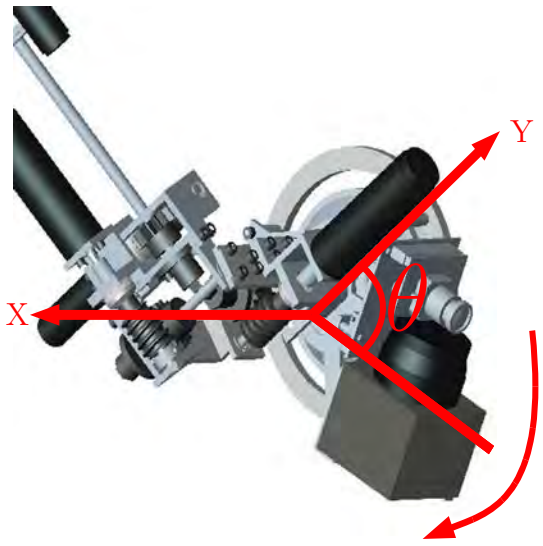


Figure 3.2: Scanning motion for data acquisition. Sensors tilt through angle θ about the X axis.

3.3.1 Experimental Data Acquisition Setup

For the purposes of gathering experimental data during the construction of the climbing robot prototype, an experimental test setup was built to simulate the scanning motion of the wall climbing robot, as shown in Fig. 3.2. This setup

consisted of a base plate, a sensor suite and a number of objects comprising the target environment to scan.

Base Plate

The base plate was a 700mm square x 12mm thick aluminium plate with tapped holes every 50mm in a grid pattern. This allowed for sensors and target objects to be mounted at accurately known ground truth locations. This base plate can be seen in Fig. 3.3(a) below. During data acquisition much of this aluminium surface was covered with white paper to create a more amenable floor surface material. Due to the often large incidence angle with the ground plane, caused by the sensor suite being restricted to ground level, the aluminium creates specular reflections that are not easily accounted for in the current sensor model. This white paper also covers up the unused mounting holes which tend to confuse the feature extraction process.

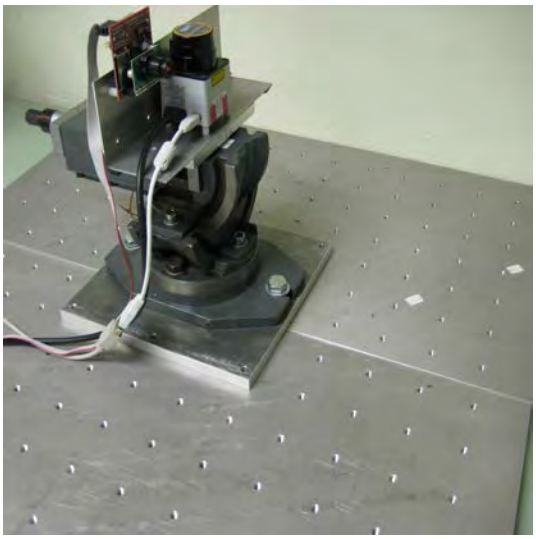
Sensor Suite

The sensor suite consisted of one LRF and one camera. The particular sensors used will be discussed later in this chapter. The camera was mounted to the right hand side of the LRF whereas, in the current WCR prototype, the camera is mounted directly above the LRF. This is not a significant difference and it will be shown in Section 4.3 that the proposed mapping methods can handle any offset that is not too large.

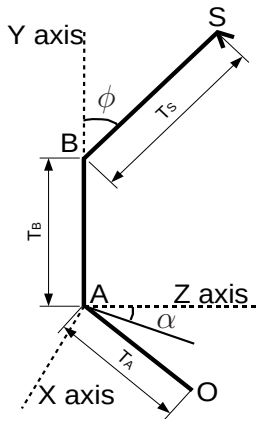
The sensors were mounted on a three DOF swiveling machinist vice. This allowed the sensor pose to achieve any desired angle about any axis within the workspace of the vice. This vice was in turn mounted on a small 12mm aluminium plate which could be accurately positioned anywhere on the base plate. The base plate and sensor suite can be seen in Fig. 3.3(a).

The kinematics for the swivel vice is shown in Fig. 3.3(b). Joint B, which is offset from the base at A by T_B, provided the tilting motion by rotating through the angle ϕ to achieve the 3D scans. Joint A, at the base of the robot, allowed it to

rotate about the vertical Y axis through the angle α. This joint was offset from the scan origin at point O by T_A. The sensors are located at position S which is offset from Joint B by T_S.



(a) Swivel vice and sensor suite mounted on base plate. Origin O is under the bolt at the vice base.



(b) Kinematics of swivel vice simulating the robot scanning motion.

Figure 3.3: Experimental setup for data acquisition.

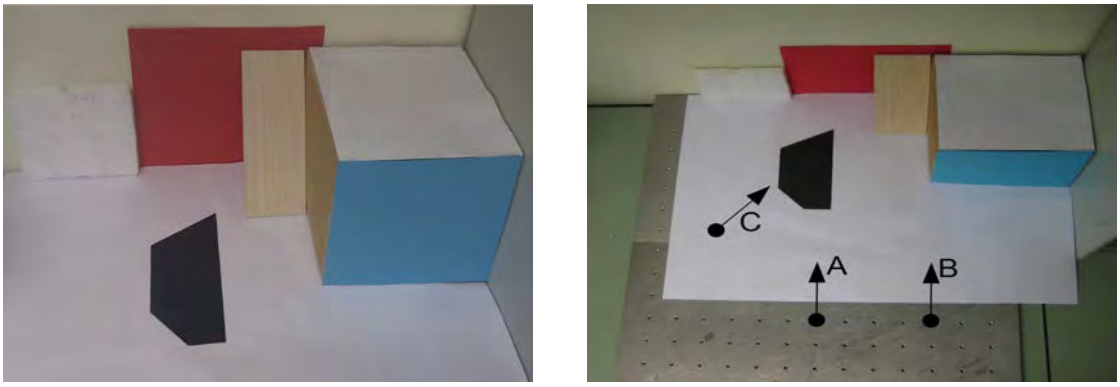
After positioning the vice and sensor suite in the desired pose for a 3D scan, the vice was tilted forwards to allow the sensor suite to exhibit a nodding motion. This simulates the intended scanning motion of the robot which will tilt its final foot joint in a similar manner, as in Fig. 3.2. Custom written C++ code was used to interface with the sensors for data acquisition and MATLAB was used to interpret the raw data for the mapping methods detailed in later chapters.

The mobility and agility of the vice limits the usable workspace of the sensor suite, so that it is not possible to test all possible pose configurations. This is not seen as a major disadvantage as most useful poses are achievable. A simulator was also implemented which does not suffer from this limitation and this will be detailed shortly.

Experimental Test Scenario Environment

The main test environment used to acquire test data for the mapping methods proposed in this thesis is shown below in Fig. 3.4(a). Three 3D scans were taken at different poses and these are shown in Fig. 3.4(b). For the exact pose of each scan A-C, refer to Table 3.1. Each 3D scan consisted of nine 2D scans taken at a tilt angle of 0° to 40° below the horizontal at a step of 5°. The data acquired will be used to validate the surface mapping methods introduced in the following chapters. For some experiments different environments will be used to test particular aspects of the proposed methods and they will be introduced and detailed at that point.

Throughout the thesis this dataset will be referred to as the experimental dataset. In some cases it may instead be referred to as the real dataset to contrast it with the simulated dataset detailed below.



(a) Test scenario environment.

(b) Top view of test environment showing sensor poses.

Figure 3.4: Experimental (Real) scenario used for data acquisition.

Table 3.1: 3D scan poses for experimental dataset.

3D Scan	X	Y	Z	θ_A
A	0m	0m	0m	0°
B	0.2m	0m	0m	0°
C	-0.2m	0m	0.2m	-45°

This test environment consists of ten planar surfaces with variation in material, colour, shape and orientation. This environment does not contain any nonplanar surfaces and this is further discussed in Section 4.3.4.

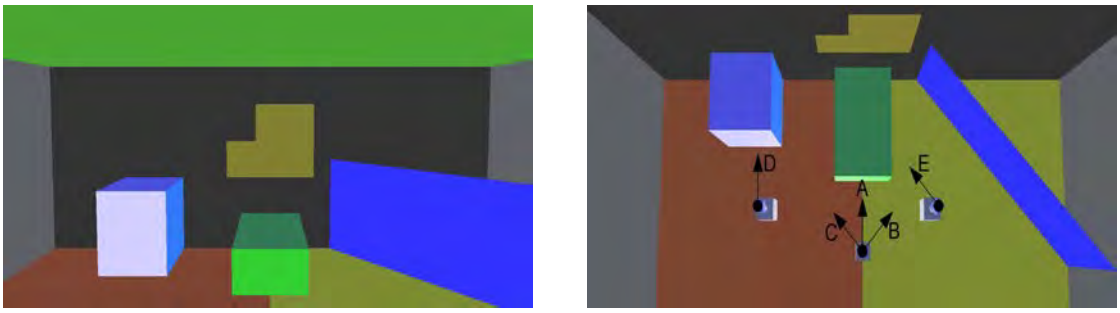
3.3.2 Simulated Data Acquisition Setup

To complement the experimental test setup described above, a simulator was developed using Microsoft Robotics Developer Studio (MRDS) and the associated Visual Simulation Environment (VSE). They were used to simulate the robot and provide known and controlled data acquisition to further develop and verify the proposed mapping approach. The VSE allows for many scenarios to be quickly and accurately simulated whereas the real experimental setup allows for the real world influences, such as noise, to be tested.

The simulated robot was also simplified, so that it contained two joints in the same manner as the experimental test setup shown in Fig. 3.3(b). The same kinematic model was used although the translations T_A, T_B, T_O differed.

Simulated Test Scenario Environment

The simulated test environment used in the remainder of this thesis is shown below in Fig. 3.5(a). Five 3D scans were taken at different poses and these are shown in Fig. 3.5(b). Each 3D scan consisted of eleven 2D scans taken at a tilt angle of 50° above to 50° below the horizontal at a step of 10°. The data acquired will be used to validate the surface mapping methods introduced in the following chapters.



(a) Simulated test scenario environment.

(b) Top view of simulated test environment showing sensor poses.

Figure 3.5: Simulated scenario used for data acquisition.

Table 3.2: 3D scan poses for simulated dataset.

3D Scan	X	Y	Z	θ_A
A	0m	0m	0m	0°
B	0m	0m	0m	-45°
C	0m	0m	0m	45°
D	-0.6m	+0.4m	-0.2m	0°
E	+0.4m	+0.4m	-0.2m	45°

3.4 Image Sensor

A colour image sensor, or camera, is a valuable source of information for use by a robot or autonomous vehicle for perception and navigation purposes. A camera produces a 2D image that is a projection of the 3D scene it is looking at. This image can provide colour, texture and shape information that is independent from the range of the objects in the scene. Cameras are generally inexpensive so vision has become popular for use in mobile robotics. The major disadvantage of vision is the computational expense of extracting useful information from images which can be significant for advanced algorithms such as stereo vision. This is especially true for high resolution images as for many computer vision algorithms the complexity scales poorly. The design considerations for the wall climbing robots discussed in Section 3.2.1 ruled out the possibility of any computationally expensive techniques. A low resolution colour camera is therefore highly desirable and any mapping methods developed will need to be able to compensate for this.

3.4.1 Choice of Camera

With the ideal camera having low resolution and providing colour images, the camera chosen was the CMUcam3 [78]. This camera fit the requirements as well as providing ease of use for initial development. There are however, many other similar colour cameras available which could be used in its place as long as they too are low resolution. The CMUcam3 also provides an added benefit of having its own dedicated onboard processor. This could be used to ease the computational burden on the main robotic processor which will have limited computational power as men-

tioned previously. Further details of the specifications and camera calibration can be found in Appendix A.2.

3.5 Range Sensor

The use of Laser Range Finders (LRFs), or lidars, in the field of mobile robots has become almost the norm in recent years. They provide environmental information at an accuracy and resolution that is unmatched by most other sensors. They form the backbone for many major algorithms for localisation and navigation [19][79]. LRFs have been preferred to other range sensors such as sonar or infrared due to their higher accuracy and angular resolution and, stereo vision aside, are the predominant range sensor choice in mobile robotics of late. A review of the benefits of LRFs compared to other range sensors, particularly stereo vision, was performed by Hebert [80].

A LRF consists of a laser emitter and receiver pair as well as the required electronics to infer the range measurement from the received data. Two types of laser measurement techniques are used. Firstly, amplitude modulated continuous wave LRFs emit a laser beam at a precisely known frequency. The returned beam will be phase shifted by a distance proportional to the range of the surface object. The second method is called time-of-flight which emits a pulse and records the time taken to travel to the object and return. Each technique has advantages and disadvantages but for most applications the difference is not significant. For further information on the specific differences, particularly in reference to the error models and calibration, refer to Adams [81].

The laser module is mounted on a motorised platform that typically rotates at a rate of 10-75Hz in common LRFs. This allows the LRF to scan and provide range measurements in a 2D plane. Some LRFs also scan in 3D but such scanners tend to be expensive, slow and bulky and so are unsuitable for many mobile robotics applications. A more common approach to generating 3D scans in robotics is to use a cheaper and more reliable 2D LRF and provide a tilting [82][39] or translating [48]

motion external to the LRF module.

3.5.1 Choice of Laser Range Finder

There many different LRF brands and models available for use in robotics and autonomous vehicles. These vary in size, accuracy, cost and many other factors and so the choice depends on the application requirements, as each LRF has its own strengths and weaknesses.

LRFs are increasingly being used in fields such as surveying and geographic information systems. These LRFs provide highly accurate range measurements with large operating ranges in the order of 100m or more. Examples and the usage of this type of LRF are discussed by Boehler *et al.* [83]. However, these LRFs are unsuitable for use in mobile robotics as they are too expensive for most robotic applications and their very high accuracy is generally not necessary anyway.

Two common LRFs used in robotic applications are the SICK and Hokuyo models. The SICK LMS-200 has been particularly popular for use in robotics and has been used by Nuchter *et al.* [39], Weingarten *et al.* [33], Harati *et al.* [45], Ye *et al.* [82] and many others. This LRF has been well characterized by Ye *et al.* [84] and has shown robust performance in mobile robotic applications. All the previous examples, however, were from ground constrained wheeled robots. Due its relatively large size (185mm×156mm×210mm) and weight (4.5kg), the SICK LMS-200 is unsuitable for use by a small wall climbing robot where the entire payload must be supported against gravity. This LRF also performs poorly at ranges less than 500mm in some situations.

A better option for smaller robots is the Hokuyo URG-04LX [85] which has found favour in recent years due to its much smaller size (50mm×50mm×70mm) and low mass (0.16kg). It has been extensively characterised by Kneip *et al.* [86] and Okubo *et al.* [87] and both Lee *et al.* [88] and Pascoal *et al.* [89] found it to have similar performance characteristics to the widely used SICK LMS-200, but in a much smaller package.

A third option is to use a 3D range camera such as the recently developed Swiss-Ranger series as used by May *et al.* [51] and Surmann *et al.* [90]. These cameras are also very small and lightweight and use the principal of time-of-flight to produce a 3D image with range and intensity information at each pixel. It has a field of view of roughly 45° horizontally and vertically and can supply images at 30Hz. This is a promising sensor for use in robotics application but has several limitations. It has been found to have a lower accuracy and more complicated noise model and calibration requirements when compared with the SICK and Hokuyo LRFs, and a relatively small field of view.

Another similar sensor is the recently released Microsoft Kinect sensor which was designed for use with the Xbox gaming console. It provides coloured 3D range images, similar to the time-of-flight cameras, to provide a RGB-D image. The main limitation for its use in this work is its minimum range, which is currently 1200mm. The required operating range for the CRACbot is 0-1000mm and so this sensor is unsuitable until its minimum range can be significantly reduced.

The Hokuyo URG-04LX LRF has been chosen for this work. This was primarily due to its small size and weight and its well characterised performance. The SICK LMS-200 is too bulky for a small WCR to feasibly carry and the Swiss Ranger time-of-flight cameras still require further development and characterisation. Further details of the Hokuyo URG-04LX specifications and calibration can be found in Appendix A.3.

3.6 Extrinsic Sensor Calibration

The kinematics of the experimental robot setup was detailed in Fig. 3.3(b) above, but this only provided the transformations up to the sensor Coordinate Frame (CF) origin, denoted as S. The exact rotation and translation offsets for each sensor must be known in order to transform data points from the local LRF and camera CFs to the sensor base CF. They can then be transformed into the world CF.

Another use for this extrinsic calibration is to transform laser range points onto

the camera image and vice versa. This allows these two sensor modalities to be fused together in a smart fashion as opposed to relying on each sensor independently. This section will introduce common approaches to sensor fusion and detail the extrinsic calibration parameters.

3.6.1 Sensor Fusion

Most SLAM and mapping systems use only one sensor type, usually either laser range data [33][48] or vision information [30][91]. In recent years it has become more common to add multiple sensors to robots. This is partly due to reduced cost and size of sensors and also the increase in computational resources. This has allowed the analysis and fusion of more data to provide more complex information about the environment.

One very common sensor set to fuse has been LRF range data with image data from a camera. These two sensors have been found to provide complementary data. A good comparison can be found in [59]. For example LRFs provide accurate range data but with limited resolution. Cameras can provide higher resolution data and provide texture information but is also dependent on lighting conditions.

Most efforts to combine image and range data have been for the purpose of visualization for a human operator. The approach involves creating a geometric map using only range data and then projecting the image data onto the 3D point cloud [92] or modeled surfaces [67] to colour the produced map. Sometimes colour or intensity information has been used as extra information to aid in correspondences and localization for SLAM problems [93][94], in addition to range information. Again colour is mapped onto range data to create more descriptive 3D points.

Another line of fusion has been to extract line features from range images and camera images and attempt to match them up in order to register the two images and extract planar surfaces [95]. This approach is also used as a method for calibrating the transformation between camera and laser CFs [96].

During fusion between multiple sensors it is crucial to ensure that the measure-

ments are correctly time stamped so that they can be fused together correctly. This was investigated in detail by Carballo *et al.* [97] to ensure correct time stamping for the Hokuyo URG-04LX. This problem is more apparent during high speed scanning or when a robot does not stop moving while acquiring each scan. If the scanning approach of Stop-Scan-Go is used, where the robot is stationary during data acquisition, then a basic time stamping approach for sensor fusion is sufficient.

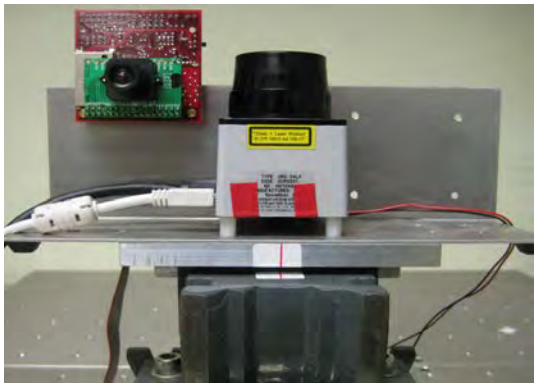
In general, much of the information available from a pairing of a LRF and a camera is wasted in most approaches. There is potential to use this information in a smart manner during sensor fusion for mapping.

3.6.2 Calibration Parameters

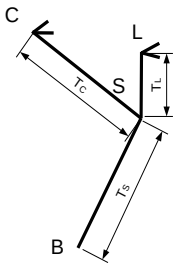
The parameters required for the calibration between the camera and LRF coordinate frames are introduced below for both the experimental and simulated setups.

Experimental Calibration Parameters

The external calibration was performed by combining the manufacturers specifications with precision measurements of the sensors themselves to manually find the required offsets. The sensor suite can be seen below in Fig. 3.6 where the translations from the sensor CF, at point S, to the LRF and camera CFs are denoted T_L and T_C respectively. There was no rotation as the sensor mountings were designed to avoid this.



(a) Sensor suite mounted on swivel vice.



(b) Kinematics of sensor suite.

Figure 3.6: Extrinsic calibration parameters for sensor suite.

Automatic methods for calibration between LRFs and cameras have been proposed by Zhang *et al.* [98], Li *et al.* [99] and Scaramuzza *et al.* [100] amongst others. The general approach involves extracting features such as image lines LRF edges from each sensor and attempting to find the transformation that aligns them. The parameters found using the manual approach were verified and found to be sufficient to produce a quality calibration. Thus the automatic techniques were unnecessary in this case.

Simulated Calibration Parameters

The extrinsic calibration of the simulated sensor CF transformations is known exactly from the simulator parameters specified. Both the camera and LRF were placed exactly at the sensor base origin with no rotation so that no transformations were necessary, which leads to the trivial values for $T_l = T_c = [0 \;\; 0 \;\; 0]^T$. This allowed the aperture of each sensor to be in the exact same position which is obviously an impossible situation for any real system. This configuration does allow for simplified kinematics and minimises any occlusions caused by offsets between the two sensors.

3.7 Summary

This chapter has detailed the robotic hardware used in this thesis along with the reasons for the design choices made. These choices have implications for the development of a suitable mapping methodology. In particular, the limited onboard computational resources and the need for real time operation require a fast and lightweight mapping approach. It must also accurately map the planar surfaces in the robot's immediate environment because they are required as footholds for a WCR.

A camera and LRF were chosen as the primary sensors. The CMUcam3 camera was chosen for its low resolution. The Hokuyo URG-04LX was chosen as the LRF due to its small size and weight. It has been extensively characterised and provides good performance at the desired operating range of 0-1000mm. Chapter 4 will

shortly detail the extraction of surface features from the raw data of these two sensors and their subsequent grouping. These feature groups will then be used for planar surface fitting in Chapter 5. The experimental and simulated test setups described in this chapter will be used to validate these methods.

Chapter 4

Extraction and Grouping of Surface Features

4.1 Introduction

The target wall climbing robot was described in Chapter 3. The use of both a laser range finder and a colour camera was discussed and the idea of using the fusion of these two sensing modalities for mapping was introduced. Chapter 2 also found that for many plane fitting methods, the planar segmentation of the raw 3D data was slow and computationally expensive. This chapter proposes an alternative approach based on the extraction of features from the range and image data. A method is then proposed to group these surface features together to provide the basis for the planar surface fitting method proposed in Chapter 5.

The extraction of image lines and corners from each camera image as well as the extraction of 2D line segments from the laser range finder scans will be described in Sections 4.2 and 4.3. The representation of each feature type is then explained and transformations are detailed to convert from the sensor coordinate frames into the world coordinate frame. After the feature extraction stage is completed, the features need to be associated together according to the physical surfaces that generated them. This process, detailed in Section 4.4, is referred to as feature grouping in

this work and a method to perform this task is proposed and evaluated. This is an important step to ensure that the planar surface fitting process is accurate and robust.

4.2 Image Feature Extraction

Image lines and corners are extracted from each image and these are used to provide boundary information for the surface fitting process. This section describes the extraction process used and the representation of these features in a graph structure. The coordinate transformation equations are detailed to transform an image pixel into the world CF and then project it onto the planar surface as a polygon boundary corner.

4.2.1 Extraction of Image Lines and Corners

The extraction of image features is a common task in computer vision as it allows for great reductions in the amount of data to be stored and operated on. The aim is to find representative features of the important information present in the image. The most common features used are line and point features. Line features predominantly consist of image edges while point features consist of interesting pixels or small pixel groups. The point features are either corners of physical surfaces or could be visually unique or interesting points. In this work physical corners and edges are desired, as the aim of this research is to extract physical structure.

Image Feature Extraction Method

The method used to extract the image lines and corners involves a two part process. A standard Canny edge detector [101] was used to extract edge curves from the image. Corners were then found within each curve using the approach of He *et al.* [102], which used local curvature. These are fairly standard algorithms. While any other edge detector could be used, Canny edge detection generally gives good

results.

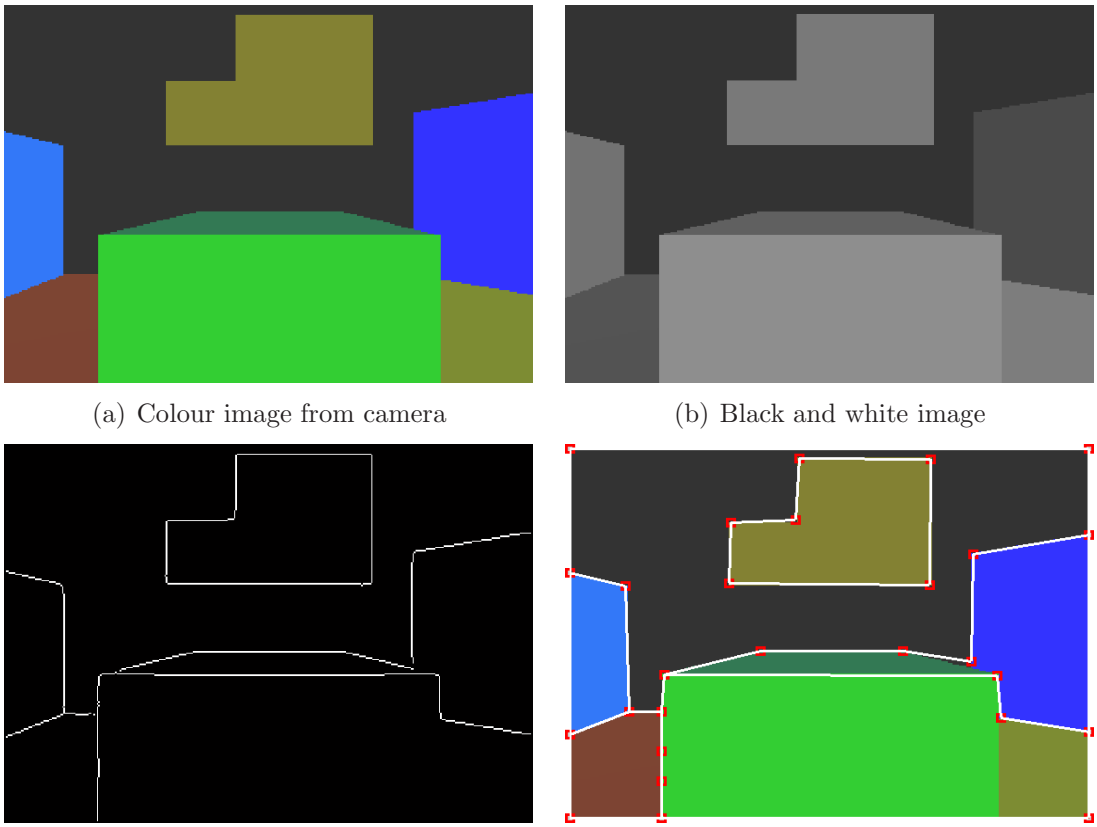
The curves were split at each corner into individual edges and those passing certain criteria were selected as image lines. The two criteria used were a minimum number of supporting points and a linearity test. This did have the effect of eliminating valid short lines as well as nonlinear curves but improved the robustness of the extraction, especially against false texture edges.

Linear image edges were chosen as they can be represented compactly by two end points. This greatly reduces the data size as well as simplifying the 3D coordinate transformations. Nonlinear edges could be represented by curves or approximated by piecewise linear segments but this is not used at this point. This limitation is not as severe as it first seems. The surface mapping approach presented in this thesis is focussed primarily on the mapping of planar surfaces. The target environment of operation for the CRACbot is any structured indoor environment. These generally contain a high proportion of planar surfaces which commonly exhibit many linear edges in such an environment.

The final image feature extraction step involves the merging of redundant corners to create junctions where more than two lines met. This extra step is required as the corner extraction process operates on individual curves. This is most commonly used at T junctions as the main and side edges can correspond to separate extracted edge curves. Some corners were generated by the intersection of an image line with the outer image frame boundary and these corners were joined together at this stage to provide closed surfaces to be used later. These image corners and lines will be referred to as virtual features as they do not correspond to physical surface features. This will be clarified further in Section 5.5.1.

An example of this extraction process is shown in Fig. 4.1 below. Figure 4.1(a) is the raw colour image which is then converted into black and white for use by the Canny edge detection algorithm, as in Fig. 4.1(b). The output edge curves are shown in 4.1(c). Following the corner detection process, the final extracted image features can be seen overlaid on the original colour image in Fig. 4.1(d). Note that

two image edges are not extracted. The first is missed as it is too short and the second is missed as the black and white intensity difference between the two surfaces is below the threshold.



(a) Colour image from camera

(b) Black and white image

(c) Extracted edges from Canny edge detector

(d) Final extracted image lines and corners

Figure 4.1: Image feature extraction process.

Image Feature Extraction Results

The accuracy of the image feature extraction process was analysed using the experimental and simulated datasets to determine the accuracy of the extracted features and the success rate of the extraction. The results are shown below in Table 4.1.

The metrics used below are explained as follows.

True + (%)

A true image feature (corner or line) that was successfully extracted.

False + (%)

An extracted feature (corner or line) that did not correspond to any physical feature or showed a gross location error.

RMSE (pixels)

The Root Mean Square Error of the extracted image corner positions. The Euclidean distance in pixels is used.

Table 4.1: Image feature extraction results

DataSet	Images	Corners	RMSE (pixels)	True +	False +	Lines	True +	False +
Real	9	188	1.19	83.3%	11.7%	158	81.6%	6.3%
Simulated	11	189	0.84	95.2%	4.8%	164	89.0%	2.4%

The corner accuracy, measured by the Root Mean Square Error (RMSE), was 1.19 pixels (experimental data) and 0.84 pixels (simulated data). This error was used as an indication of the uncertainty of the corner extraction used in Section 4.2.3 and should also include an additional quantization error of 0.5 pixels. Another consideration is the sensitivity of this error to the particular environment being scanned as well as the tuning of the edge detection parameters. An extra factor of 50% was introduced to allow for this.

The final errors used for the uncertainty of the corner extraction were 2.5 pixels (experimental data) and 2.0 pixels (simulated data). The corner extraction had a success rate of 83.3% (experimental data) and 95.2% (simulated data). As the corners are detected from within extracted image edges, the majority of failed corner detections were due to a failure to detect that edge and this will be discussed shortly.

The corner accuracy calculation excluded corners that were clearly incorrect and these were instead recorded as false positive corners. These were caused by several factors. Firstly, image noise caused some image edges to be over segmented along a linear edge. This is not of major concern as it will be handled during the polygon boundary merging in Section 5.5. Secondly, highly textured regions can cause virtual edges and corners to be generated. However, this was not such an issue in the scenarios used. Proper tuning of the edge detection parameters can help to negate this but retuning may be required when the environment changes significantly. Lastly, errors in the junction merging algorithm occasionally caused some corners to be merged incorrectly resulting in errors of five or more pixels. This

was primarily an issue with short edges or edges being merged at very acute angles.

The image line detection rate was 89.0% (simulated data) and 81.6% (experimental data). The main reasons for detection failure were adjacent surfaces having very similar material/texture or the image edge did not contain a sufficient number of supporting pixels. The false positive rate was insignificant at 2.4% (simulated data) and 6.3% (experimental data) and most of these get filtered out in the grouping stage of Section 4.4. These were caused by similar reasons to the false positive corners, namely textures and shadows as well as poor junction merging.

4.2.2 Graphical Representation of Image Features

Once the image lines and corners have been extracted they require some kind of representation for storage and feature grouping. A graphical structure is used for this purpose. The image corners become the graph nodes and the image lines are the edges connecting the nodes. An example structure is shown below in Fig. 4.2. Note that, in this work, every extracted line has both its endpoints marked as corners by definition.

A graphical representation has been used for grouping of image features by Horaud *et al.* [103] and McLauchlan *et al.* [104] amongst others. This representation is compact and allows for easy searching of connected features for the feature grouping process. Further details of the feature grouping method used in this work can be found in Section 4.4.

One major difference compared to the graph structures of other approaches is that in this work each image line is used twice during the grouping stage. Every physical edge has two sides corresponding to the two physical surfaces it separates. This must be tracked to ensure that each image side is grouped uniquely to one and only one surface. It is the image line sides that are grouped and not the image line itself.

Each image line is thus represented as a directed edge running from corner A to corner B. It does not matter which image line endpoint (corner) is chosen to

be corner A. The important consideration is that the choice is consistent. When looking along the image line towards corner B, Side 1 is defined as the left-hand side and Side 2 is the right-hand side of the line. This notation allows the sides to be tracked and referenced independently of the orientation of each image line. This is shown in Fig. 4.2 for line L_e.

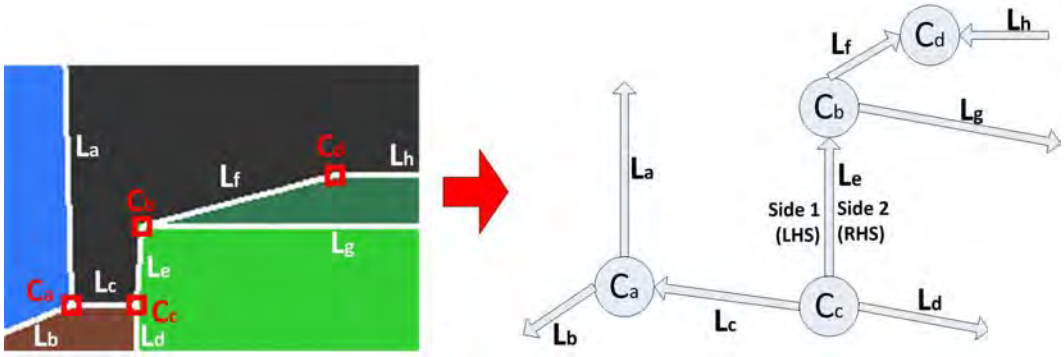


Figure 4.2: Graphical representation of image features. The image corners are nodes and the image lines are directed edges.

4.2.3 Transformation of Image Points

After the image features have been extracted and converted to the graphical structure introduced above, each image feature needs to be transformed from the camera CF to the world CF before it can be used for surface fitting. This is done point wise as each image line is represented by its two endpoints. This 2D to 3D transformation can only be performed up to a scale factor Z which is the unknown depth parameter.

Extra constraints are required to determine the exact 3D location of each image point. The constraint used here requires the image point to lie on a known planar surface and this allows Z to be calculated. The determination of this planar surface and its parameters are detailed later in Section 5.4.

Camera CF to World CF Transformation

Given a pixel location $\mathbf{p_i} = [u\ v\ 1]^T$ from an image, that is U pixels wide and V pixels high, it is possible to determine the location of the point in the world CF.

This can be done only up to a scale factor Z. To find the 3D point first requires a conversion from image coordinates to normalized camera coordinates. This gives the normalised image point $\mathbf{p_{cn}}$ using (4.1). This uses the camera calibration matrix K from (A.1).

$$\mathbf{p_{cn}} = \begin{bmatrix} x_n \\ y_n \\ 1 \end{bmatrix} = K^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K^{-1}\mathbf{p_i} \tag{4.1}$$

To estimate the uncertainty in the normalized x_n and y_n components, K is expanded and then (4.1) can be rearranged to give (4.2).

$$\begin{aligned} x_n &= \frac{u - u_0}{f_u} \\ y_n &= \frac{v - v_0}{f_v} \end{aligned} \tag{4.2}$$

These are used to derive the variances $\sigma^2_{x_n}$ and $\sigma^2_{y_n}$ using standard uncertainty propagation [105]. As the two components of (4.1) are independent their variances can be calculated using (4.3) where F is the generic function and x_i are the variables of interest. The variances used are found in Tables A.2 and 4.1.

$$\sigma^2_F = \sum_i \left(\frac{\partial F}{\partial x_i}\right)^2 \sigma^2_{x_i} \tag{4.3}$$

This is used to find $\sigma^2_{x_n}$ in (4.4). The equation for $\sigma^2_{y_n}$ is found in a similar manner. The covariance matrix $\Sigma_{p_{cn}}$ is also shown in (4.5).

$$\sigma^2_{x_n} = \frac{\sigma^2_u}{f^2_u} + \frac{\sigma^2_{u_0}}{f^2_u} + \frac{(u - u_0)^2 \sigma^2_{f_u}}{f^4_u} \tag{4.4}$$

$$\Sigma_{p_{cn}} = \begin{bmatrix} \sigma^2_{x_n} & 0 & 0 \\ 0 & \sigma^2_{y_n} & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{4.5}$$

After finding $\mathbf{p_{cn}}$, it is then required to convert this normalized point from the camera CF to the world CF. Essentially $\mathbf{p_{cn}}$ represents a line in 3D space passing through the camera centre $\mathbf{p_c} = [0\ 0\ 0]^T$ and with a direction vector given by $\mathbf{p_{cn}} = [p_{cn} - p_c]$. To convert this line to world coordinates, the two components need to be separately transformed to give the line equation shown in (4.6). The two components are a direction vector $\mathbf{v}$ passing through a point $\mathbf{p_0}$. Note that the vector $\mathbf{v}$ should not be confused with the scalar v from (4.1) which represents an image pixel location.

$$\mathbf{p} = \mathbf{p_0} + Z\mathbf{v} \qquad Z \in \mathbb{R} \tag{4.6}$$

The transformations into the world CF of $\mathbf{v}$ and $\mathbf{p_0}$ use the kinematics of the robot scanning motion. This is simplified in this work to using only one revolute joint and this was detailed in Section 3.3. The translation from the camera CF origin to Joint B, seen in Figs. 3.3 and 3.6, is denoted $T_c = T_C + T_S$. Joint B tilts through the angle ϕ to provide the 3D scanning motion. T_B is the translation from joint B back to the scan origin at joint A. Note that the rotation about joint A and the translation back to the world CF are not relevant at this point so are not mentioned further. They will be fully detailed later in Section 6.2.1.

The rotation about joint B is described by the rotation matrix R_ϕ shown in (4.7). These parameters are used to find $\mathbf{v}$ using (4.8) and $\mathbf{p_0}$ using (4.9).

$$R_\phi = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi) & \cos(\phi) \end{bmatrix} \tag{4.7}$$

$$\mathbf{v} = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} = B(x_i, y_i, z_i, \phi) = R_\phi \mathbf{p_{cn}} \tag{4.8}$$

$$\mathbf{p_0} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}_w = C(x_i, y_i, z_i, \phi) = R_\phi(T_c + \mathbf{p_c}) + T_B$$
$$= R_\phi(T_c) + T_B \tag{4.9}$$

The covariance of $\mathbf{p}$ from (4.6) is a function of the covariances of $\mathbf{p_0}$, Z and $\mathbf{v}$. The rotation matrix R_ϕ shown in (4.7) is used to find the covariance Σ_v as in (4.10) using the covariance $\Sigma_{p_{cn}}$ and variance σ_ϕ^2. Together they form the block diagonal input covariance matrix Σ. The Jacobian J_B of B from (4.8) is used to propagate the error. As the tilt angle ϕ is independent of the image point $\mathbf{p_{cn}}$, the Jacobian can be split and it can be shown that the remaining Jacobian terms for x, y, z are equivalent to the rotation matrix R_ϕ.

$$\begin{aligned}\Sigma_v &= J_B \Sigma J_B^T \\ &= \begin{bmatrix} \frac{\partial B}{\partial x} & \frac{\partial B}{\partial y} & \frac{\partial B}{\partial z} \end{bmatrix} \Sigma_{p_{cn}} \begin{bmatrix} \frac{\partial B}{\partial x} & \frac{\partial B}{\partial y} & \frac{\partial B}{\partial z} \end{bmatrix}^T + \begin{bmatrix} \frac{\partial B}{\partial \phi} \end{bmatrix} \sigma_\phi^2 \begin{bmatrix} \frac{\partial B}{\partial \phi} \end{bmatrix}^T \\ &= R_\phi \Sigma_{p_{cn}} R_\phi^T + \begin{bmatrix} \frac{\partial B}{\partial \phi} \end{bmatrix} \sigma_\phi^2 \begin{bmatrix} \frac{\partial B}{\partial \phi} \end{bmatrix}^T \end{aligned} \tag{4.10}$$

The covariance Σ_{p_0} is also found in the same manner, as shown in (4.11).

$$\Sigma_{p_0} = R_\phi \Sigma_{T_c} R_\phi^T + \begin{bmatrix} \frac{\partial c}{\partial \phi} \end{bmatrix} \sigma_\phi^2 \begin{bmatrix} \frac{\partial c}{\partial \phi} \end{bmatrix}^T + \Sigma_{T_B} \tag{4.11}$$

Line-Plane Intersection of Image Point

As the converted camera pixel represents a line in 3D space, further information is needed to determine the scale factor Z and thus the unique point in 3D that was imaged by the camera. If the parameters of the surface that contains the imaged

point are known, then it is possible to calculate the exact 3D location of that point. A plane in 3D can be represented in a number of ways. One of the most common is the Hessian normal form whereby the set of planar points $\mathbf{p}$ conform to (4.12). The plane parameters consist of the normal vector $\mathbf{n}$ to the plane and the orthogonal distance d to the origin. This is discussed further in Section 5.4.

$$\mathbf{n} \cdot \mathbf{p} - d = 0 \tag{4.12}$$

By combining (4.6) with (4.12) it is possible to estimate the scale factor Z and thus the unique 3D point $\mathbf{p_w}$ being imaged. While a singularity appears in 4.13 if $\mathbf{v}$ is orthogonal to $\mathbf{n}$, this situation cannot occur in practice. By definition the vector $\mathbf{v}$ intersects the plane defined by $\mathbf{n}$. As $\mathbf{v}$ must pass through the camera centre, which cannot lie on the plane, the two vectors cannot therefore be orthogonal.

$$Z = \frac{d - \mathbf{n} \cdot \mathbf{p_0}}{\mathbf{n} \cdot \mathbf{v}} \tag{4.13}$$

The uncertainty of $\mathbf{v}$ and $\mathbf{p_0}$ can be propagated into Z using the Jacobian matrix J_z and the input covariance block diagonal matrix Σ containing the covariances Σ_v, Σ_n, σ_d and Σ_{p_0}.

$$\sigma_z^2 = J_z \Sigma J_z^T \qquad \text{where} \qquad J_z = \begin{bmatrix} \frac{\partial Z}{\partial \mathbf{v}} & \frac{\partial Z}{\partial \mathbf{n}} & \frac{\partial Z}{\partial d} & \frac{\partial Z}{\partial \mathbf{p_0}} \end{bmatrix} \tag{4.14}$$

To calculate the final position of the image pixel in world coordinates the value of Z found in (4.13) is used to find $\mathbf{p_w}$ as shown in (4.15). The covariance Σ_{p_w} can be found using (4.16) to account for the uncertainty in both $\mathbf{v}$ and Z.

$$\mathbf{p_w} = \mathbf{p_0} + Z\mathbf{v} \tag{4.15}$$

$$\Sigma_{p_w} = \Sigma_{p_0} + \mathbf{v}\sigma_z^2\mathbf{v}^T + Z^2\Sigma_v \tag{4.16}$$

4.3 Range Feature Extraction

The laser range finder provides a series of range and bearing measurements in a 2D scanning plane. This information can be used to extract structure from the environment. The range values can be converted into 2D cartesian coordinates on the laser plane and line segments can then be extracted. These line segments can be used for plane fitting once they have been converted into 3D space. This is necessary as the line segments will be grouped with those from other 2D scans which were taken at different tilt angles. For further information on the specifics of the LRF used and its calibration, refer to Section 3.5.

This section introduces possible segmentation approaches before detailing the segmentation method used in this work. The transformation of the laser data into the world and camera CFs is then described. This is followed by a description of the line segment representation used.

4.3.1 2D Line Segmentation Literature

A good quality segmentation of the laser range finder measurements into linear segments is very important for the success of the proposed approach as these segments form the basis of the planes that represent the physical surfaces. It is important that the segmentation is accurate and so it should be conservative. This means that false positive points in the line segments are very undesirable, even at the expense of a higher rate of false negative segmentation. In this work, the surface boundaries are not dependent on the laser segments and so it is not necessary for the edges to be detected perfectly. This allows for a conservative approach to be used.

There are several commonly used algorithms for 2D line segmentation and a good summary can be found in the work of Nguyen *et al.* [106]. The approaches most commonly used fall into the categories of incremental line growing, recursive splitting and edge finding. Other methods have also been suggested including the Hough transform and statistical algorithms such as RANSAC and EM.

Incremental Approaches

The incremental approach starts with a small number of consecutive points that form an initial candidate line. Each increment adds the next one or more adjacent points and then refits a 2D line to all the points in the set. A test condition is used to verify the linearity of the point set to determine if the line segment is still linear or not. This condition is usually either the line variance or the point distances from the line. When the condition is violated, the new points are removed and the line set is returned.

Incremental segmentation is a simple algorithm and has been widely used in many applications. Siadat *et al.* [107] called this approach Line Tracking. Their work used a least squares fit and terminated the line when any of the supporting points exceeded a distance threshold. This approach can be inefficient as for long line segments, the constant refitting is wasteful. Nguyen *et al.* [106] suggested a coarse incremental fitting whereby five points were added before each fitting and then a fine search was used to find the exact edge point when needed. Siadat *et al.* [107] used a least squares fit. While this is not an optimal line fit, it allows for an incremental line fit which mitigates the inefficiency. An advantage of incremental approaches is that the segmentation does not have to wait for the entire scan to be acquired to begin the segmentation. However, this is only advantageous if the scanning is a relatively slow process which tends not to be the case with newer range finders.

Recursive Splitting Approaches

Recursive splitting methods are another widely used line segmentation approach. The Split-and-Merge approach involves the fitting of a line to all points in the segment and then finding the supporting point with the greatest distance from the line. If this distance exceeds some threshold then the segment is split in two at that point and the process is repeated for each new segment until a satisfactory line segment is obtained. This is often followed by a merge operation to merge any line

segments that are collinear. Split-and-Merge is fast but Nguyen *et al.* [106] found it to have a higher rate of false positive associations compared to the incremental approach.

An example of the Split-and-Merge approach is the work of Borges *et al.* [108] who combines this splitting approach with a fuzzy clustering method which worked well on a simulated dataset. A similar approach was called Iterative End Point Fit by Siadat *et al.* [107] and instead of fitting a line to the points of each segment as in the splitting methods, the line is formed by joining the two end points. This can be faster as the line fitting becomes trivial but at the cost of higher sensitivity to range noise.

Edge Finding Approaches

An alternative approach to segmentation is to look for edges in the scan data, rather than by line fitting. These approaches can operate on the raw range information which eliminates the need to first transform the data into cartesian coordinates. The simplest method is to split segments when the difference in consecutive range measurements exceeds a threshold. However, this is only reasonable for detecting jump edges but not intersecting edges or changes in surface direction. Siadat *et al.* [107] proposed a version of this approach called Successive Edge Following. This used an adaptive threshold for the distance between successive range points but this does not fully overcome the disadvantages of this approach. It is, nevertheless, useful as a pre-splitting approach to find the easy segment edges.

An alternative approach is suggested by Harati *et al.* [109] which first converted the range and bearing pairs into a Bearing Angle. This was the angle between the laser beam and the line passing through consecutive points. They then showed that edges in this bearing angle provided decent segmentation results using direct thresholding to find changes in the angle. However, this approach can be adversely affected by noisy data and is very sensitive to the threshold value.

Other Approaches

The Hough transform is commonly used in computer vision applications for extracting image lines. Pfister *et al.* [110] used a version adapted to laser line segmentation to find the initial line segments for their line fitting algorithm. The Hough transform does suffer from discretisation and the correct choice of grid size can be difficult.

RANSAC [73] can be applied to line segmentation. It is a random approach to line fitting which selects many lines from random points and selects the best found candidates. EM is another statistical tool that has been applied to segmentation by Liu *et al.* [67] amongst others. Both these approaches are computationally expensive due to their random nature. This makes their performance difficult to guarantee without a large number of iterations as well as disqualifying their usage in this work as computational resources are limited. They do not take advantage of the sequential nature of the data and so occasionally attempt to fit lines to unrelated points. EM can also get stuck in local minima and often requires an initial estimate of the number of line segments.

4.3.2 Proposed 2D Line Segmentation Method

The segmentation method used in this work is based on the incremental line fitting approach. Incremental segmentation and the Split-and-Merge approach are both fast. However, the incremental approach has the advantage due to its lower rate of false positive associations, as noted by Nguyen *et al.* [106]. This is a measure of the number of points incorrectly assigned to a given line segment and does come at the cost of lower true positive point associations. This tradeoff is taken as, in this proposed approach for plane fitting, it is more important to get accurate lines as opposed to accurate line edges. Methods to pre split the data using the image information and post merge the segmented lines are also proposed in an effort to improve the overall quality of the segmentation.

Incremental Segmentation and Proposed Modifications

The incremental line segmentation approach has been modified for use in this work. The 2D lines are fitted using Principal Components Analysis (PCA) and the splitting criteria is the maximum distance of all points to the fitted line. The modifications used are aimed at improving the robustness and accuracy of the segmentation, possibly at the expense of segment endpoint accuracy. As the extracted line segments form the basis of the infinite plane fitting, it is important to ensure that all the segmented points are actually from the true line. This is known as minimising false positive points. The laser line segment edges do not have to be found exactly, as they are not used to determine the planar polygon boundaries.

The task of minimising the false positive rate is achieved with several modifications to the standard incremental method, as shown in Algorithm 4.1. In contrast to most segmentation algorithms, the orthogonal distance of the next candidate point is tested before adding it to the line segment. It thus has no influence on the line parameters. This allows a change in direction to be detected earlier as it reduces the tendency of the incremental approach to follow a curve in the data. This is especially useful at join edges where, unlike jump edges, there is no large change between adjacent range measurements.

Several such advanced points are tested and integrated into the line segment before refitting to improve the speed of the algorithm. Testing 3-5 points in advance of the line, as seen in Fig. 4.3, showed the best results. In this example the next three points are tested and all found to be within the threshold distance so they are added to the line segment and the line is refitted. If the distance of any tested point exceeds the threshold, then the next point is also checked. Two such outlying points indicate a split in the segment whereas a single point is marked as an outlier and henceforth ignored. These two scenarios are shown in Fig. 4.4. Should two adjacent outlier points cause a split in an otherwise linear segment, they will likely be remerged together as discussed later in this section.

The modified incremental segmentation algorithm is shown in Algorithm 4.2.

The initial condition finds the next set of five consecutive points that form a valid line.

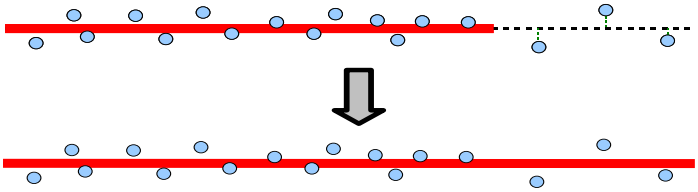


Figure 4.3: 2D line segmentation example. The next three points are tested before integrating and refitting the line.

(a) Splitting at two consecutive outliers.

(b) Single outlier ignored.

Figure 4.4: 2D line segmentation process - Splitting criteria.

Another simple modification is to ignore the first and last points in each extracted segment as these are commonly segmented into the wrong line. In the case where a discarded endpoint was in fact correctly segmented, little is lost during plane fitting unless the total number of points is small. Pre-splitting, or clustering, the range data prior to segmentation using the image lines can also help in this regard and this is discussed below.

Algorithm 4.1 Standard incremental segmentation.

```
start ← i
end ← i + 1
while validLine(fitLine(start:end)) do
  end ← end+1
end while
return fitLine(start:end-1)
```

Algorithm 4.2 Modified incremental segmentation.

```
start ← i
end ← i + 4
while !validLine(fitLine(start:end)) do
  start ← start+1
  end ← end+1
end while
while validLine(fitLine(start:end)) do
  if pointsOnLine(end+1:end+3) then
    end ← end+3
  else
    return fitLine(start+1:end-1)
  end if
end while
return fitLine(start+1:end-1)
```

Pre Segmentation - Image Line Clustering

A common technique to improve the performance of the segmentation is to first split the range scan into point clusters where there are large jumps between consecutive range values. This was shown by Nguyen *et al.* [106] to provide benefits in speed as well as accuracy. The size of the threshold can be kept constant or varied depending on the size of objects in the environment. Overall, this clustering process is similar to the edge based segmentation methods discussed previously.

This work instead uses the image line features for clustering as opposed to range jumps. The images provide valuable information for surface segmentation that can be used to aid the laser line segmentation and this is rarely exploited in this manner. As the image lines are already extracted there is little extra cost to using this information to pre-split the laser line segments.

It is easy to find the corresponding laser bearing in the specific case where the LRF is installed to scan perpendicular to the Y camera axis, the vertical offset between the two sensor apertures is small, and the horizontal offset is small. In this specific configuration the laser scan line projects onto the image as a horizontal line running through the principal point v_0, as seen in Fig. 4.5, or very close to it in the case of small offsets. Any image lines that intersect this line will imply that the laser line should be split at that point. As the two sensors are aligned horizontally, the

horizontal bearing of a feature in one sensor is directly transferable into the other.

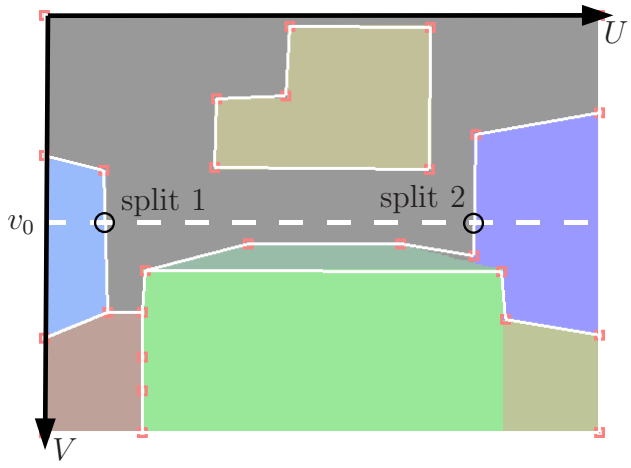


Figure 4.5: Projection of laser line onto image and use of image lines for pre-splitting.

Each image line, represented by its two endpoints $\mathbf{p_1}$ and $\mathbf{p_2}$ in u,v coordinates, is checked for this intersection using (4.17) below. The product will only be negative if one point lies above the line and the other below.

$$(v_1 - v_0)(v_2 - v_0) < 0 \tag{4.17}$$

For each image line found, the u value at the intersection must be interpolated using (4.18) to determine the splitting point. Horizontal image lines do not provide useful intersection information and so are ignored. They would also cause a singularity to occur.

$$u_i = \frac{(v_0 - v_1)(u_2 - u_1)}{v_2 - v_1} + u_1 \tag{4.18}$$

This process provides a set of values $u = u_1 \ldots u_N$ which can be converted into an angular bearing using (4.19) to determine the splitting points in the raw LRF data. The measurement index i can found from (4.20) where R is the angular resolution, f_u is the focal length in the horizontal direction and N is the total number of measurements taken, assuming the scan is centred.

$$\theta_i = \tan^{-1}\left(\frac{u_0 - u_i}{f_u}\right) \tag{4.19}$$

$$i = \frac{\theta_i}{R} + \frac{N}{2} \tag{4.20}$$

Image Line Clustering - General Case

In the general case where the horizontal or vertical sensor offset is significant, a little more work is required to find the corresponding LRF bearing for each image pixel on the horizontal line. If the offset is large or the LRF does not scan parallel to the horizontal axis of the camera, then the horizontal line assumption is invalid. In this work it is assumed that these requirements are not violated.

Due to the presence of a vertical offset between the two sensor frames, the horizontal line will pass through a different point on the image V axis. This can be found by projecting the LRF points from x, y coordinates into u, v image coordinates. The average v coordinate (height) can be found and should be used as the basis for the horizontal scan line. The intersecting image line pixels are then calculated as in (4.17-4.18) above.

In this general case it is not possible to directly correspond the horizontal bearings between sensor frames. The sensor offset results in a different bearing that needs to be found, as seen in Fig. 4.6. The laser points that have been projected onto the image in u, v coordinates track across the image in an increasing fashion due to the consecutive nature of the LRF scanning process. Thus it becomes a 1D search problem to correspond the splitting points u_i with their appropriate laser bearing index. In cases of high noise it is possible that multiple nearby laser readings will correspond to the same image u value and so the laser u values will not be strictly increasing. In this case the first and last readings to intersect with the desired splitting pixel should form the two splitting points, with any measurements in between being ignored.

A similar problem was tackled by Ellekilde *et al.* [24]. However, their work

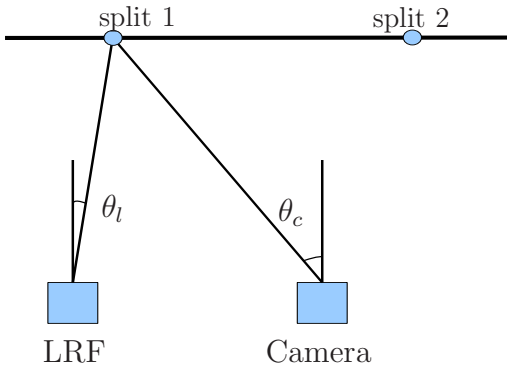


Figure 4.6: Image line pre-splitting - General case with sensor offset.

required a 2D, rather than 1D, search which is a more complicated problem. This splitting process could also be achieved by assigning the pixel colours to the laser range readings and then looking for edges or jumps in a 1D image. This is redundant as it repeats the edge detection already performed in a suboptimal way as the rest of the image data is ignored.

Care should be taken with this approach as spurious lines generated by object textures should not be used to pre split the data and these lines should be filtered out in some way as they do not correspond to physical edges. Care should also be taken to ensure that correct camera calibration has been performed. A post processing merging step can also be used to minimise the impact of this. If the false lines are numerous, then the line segmentation will operate poorly, as each split segment will have few points for segmentation.

This approach also assumes all intersecting image edges are extracted and are linear. This means that some edges will be missing if they are not extracted or if they are curved or otherwise nonlinear. This is not a big issue as the line segmentation will still likely find those segment edges anyway. Nonlinear edges could still be used if the intersecting edge detection was done as an intermediate step during the image edge extraction prior to nonlinear edges being culled.

Post Segmentation - Segment Merging

After the laser lines have been segmented using the above methods, a merging step is beneficial to merge adjacent segments that were over segmented due to the presence of noise, outliers or excessive pre-splitting. This process involves testing each line against adjacent line segments to determine if they are collinear and spatially close. If all three criteria below are satisfied then the two segments should be merged together.

The first criterion requires that the two adjacent lines are collinear. This can be tested by taking the angle between the two segment direction vectors $\mathbf{v_i}$ and $\mathbf{v_{i+1}}$ using the standard dot product in (4.21).

$$\cos^{-1}(\psi) = \cos^{-1}\left(\mathbf{v_i} \cdot \mathbf{v_{i+1}}\right) < \Delta_\psi \tag{4.21}$$

Due to the nature of the line fitting process using PCA the direction vector $\mathbf{v}$ can be rotated 180° for the same line, depending on the implementation. Because of this, if the absolute angle computed is greater than 90°, then the angle should be subtracted from 180° to get the desired angle for testing. This can be seen in Fig. 4.7. An angular threshold of Δ_ψ =5° was used. This value, along with d_{orth} to be introduced shortly, was calculated empirically from a test set of collinear and noncollinear line segments and was found to differentiate robustly between the two cases.

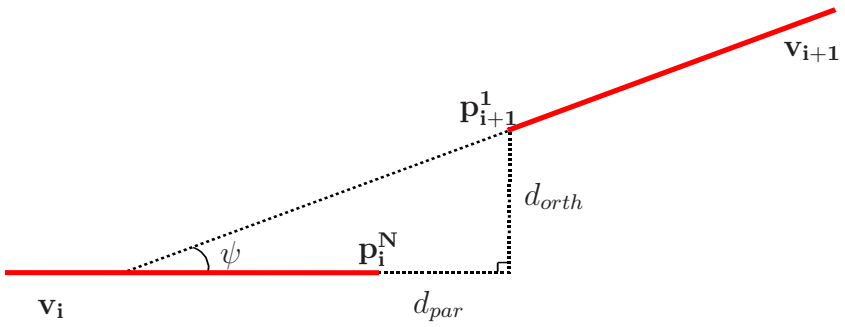


Figure 4.7: Post segmentation merging criteria.

The second criterion requires the endpoints of the two adjacent line segments $\mathbf{v_i}$ and $\mathbf{v_{i+1}}$ to be close spatially. To eliminate the influence of noise, these two endpoints are projected onto their respective lines. This is described later in Section 4.3.5. There are two distances to test. They are the orthogonal distance to the line $\mathbf{v}$, and the distance along the line $\mathbf{v}$ to the orthogonal intersection point. This is shown graphically in Fig. 4.7. These are calculated between the last point on the line segment $\mathbf{v_i}$ ($\mathbf{p_i^N}$) and the first point of the line segment $\mathbf{v_{i+1}}$ ($\mathbf{p_{i+1}^1}$).

Equation (4.22) below describes the calculation of the parallel and perpendicular distances. They should be calculated twice, once for each line direction vector, and the smaller of the two distances should be tested. The threshold values require some tuning and the values used here were d_{par} =30mm and d_{orth} = 5mm. The value for d_{par} was chosen to allow for several outlier points between the endpoints of adjacent segments as they occur more commonly in this modified incremental segmentation approach. This constant threshold does not account for the distance to the line segment as that did not show a significant improvement.

$$\begin{aligned} d_{par} &= (\mathbf{p_i^N} - \mathbf{p_{i+1}^1}) \cdot \mathbf{v} \\ d_{perp} &= \sqrt{(d_{par})^2 - ||\mathbf{p_i^N} - \mathbf{p_{i+1}^1}||^2} \end{aligned} \tag{4.22}$$

A third criterion that can be used is to check that the colour descriptors match between the two adjacent line segments. This can prevent segments being merged that are parallel and connected but are from different surfaces. An example is a sign on a wall having negligible thickness. Such a surface should only have been split in the first instance if the image pre-splitting was also used. This extra criterion should be used in this case as it will otherwise likely lead to those surfaces being falsely merged.

This also checks against the colour of any outlier points between the two segments that were excluded during the line segmentation. If they show a different colour descriptor than the two consecutive segments being tested, then it is likely that the

split was caused by a physical gap or object. In this case the line segments should not be merged.

4.3.3 Results of 2D Line Segmentation

The segmentation methods described above were run on the laser scan data and image data from three datasets, two real and one simulated, to determine the benefits and performance of each combination. The results are shown below for segmentation only, segmentation with pre splitting, segmentation with post merging, and segmentation with both pre splitting and post merging. For each case the segmentation accuracy, true and false positive rates and running time were found.

Experimental Setup

The datasets used to validate the range segmentation methods were acquired as described in Sections 3.3.1 and 3.3.2. Two experimental datasets were acquired. These used identical setups, except that the second scenario also contained three additional nonplanar surfaces. This was used to compare the segmentation performance with a more cluttered scene and the ability to recognise nonlinear line segments. In total there were eighteen 2D scans from the two experimental datasets resulting in 87 line segments. These two scenes are shown below in Figs 4.8(b) and 4.8(c).

The simulated dataset consisted of eleven 2D LRF scans and camera images. This gave a total of 55 line segments that were extracted. The simulated LRF had Gaussian noise added while the simulated camera images were considered noise free. The scene used is shown below in Fig. 4.8(a).

The metrics used for the results below are explained below. Each range point was manually assigned a true segmentation label for the purposes of these results.

True + (%)

A true line segment that was successfully extracted.

False + (%)

An extracted line segment corresponded to a nonlinear surface. This is ap-

(a) Simulated environment.

(b) Real environment.

(c) Real environment containing nonplanar surfaces.

Figure 4.8: Scenes used for laser line segmentation testing.

plicable only to the dataset Real #2, as it was the only test scene containing nonplanar surfaces. This is defined as the number of false segments generated by these surfaces as a percentage of the number of such real surface segments. This rate can be more than 100% as each surface can potentially generate multiple segments.

UnderSeg (%)

An extracted line segment corresponded to more than one true line segment.

OverSeg (%)

A true line segment that was segmented into multiple extracted segments.

EdgeError (%)

A successfully extracted line segment that failed to segment successfully at

the true edge of that line segment. This was defined as at least one segment endpoint being more than one bearing outside the true endpoint range bearing.

Time (ms)

The average time taken to segment each 2D laser scan in milliseconds. This was run in MATLAB on a single core 2.7GHz PC and so these times should be used for comparisons only.

Segmentation Results

The results using simulated data are shown in Table 4.2 below.

Table 4.2: Range segmentation results - Simulated dataset

Type	True +	False +	Under Seg	Over Seg	Edge Error	Time (ms)
Segmentation only	82%	N/A	11%	7%	7%	19.9
Segmentation + Pre Splitting	98%	N/A	2%	0%	4%	32.1
Segmentation + Pre Splitting + Post Merging	98%	N/A	2%	0%	4%	32.4

Table 4.3: Range segmentation results - Real dataset #1

Type	True +	False +	Under Seg	Over Seg	Edge Error	Time (ms)
Segmentation only	75.0%	N/A	25.0%	0%	31.8%	19.7
Segmentation + Pre Splitting	100%	N/A	0%	0%	6.8%	32.8
Segmentation + Pre Splitting + Post Merging	95.5%	N/A	4.5%	0%	6.8%	34.2

Table 4.4: Range segmentation results - Real dataset #2 (nonplanar surfaces)

Type	True +	False +	Under Seg	Over Seg	Edge Error	Time (ms)
Segmentation only	81.4%	200%	11.6%	7.0%	23.2%	19.8
Segmentation + Pre Splitting	90.7%	157%	0%	9.3%	7.0%	32.6
Segmentation + Pre Splitting + Post Merging	97.7%	157%	0%	2.3%	7.0%	34.0

The standard range based incremental segmentation method successfully extracted 82% of the simulated laser line segments. However, it did suffer from some under segmentation and segment edge inaccuracy. By using the image features for pre splitting, the segmentation success rate increased to 98% and the under segmentation and edge error rates reduced. The cost of this extra effort was seen in the

higher runtime, which almost doubled. The addition of the merging step provided a small improvement with negligible additional runtime.

The results for the two experimental datasets are shown in Tables 4.3 and 4.4. The segmentation accuracy was slightly lower than the simulated case with a successful extraction rate of 75-81%. This was expected due to the higher levels of noise in the real data. The edge error rate was also significant at 23-32% of laser line segments. The image line pre-splitting significantly improved the accuracy of the segmentation up to a 91-100% success rate. The edge error rate was also significantly reduced down to 7%.

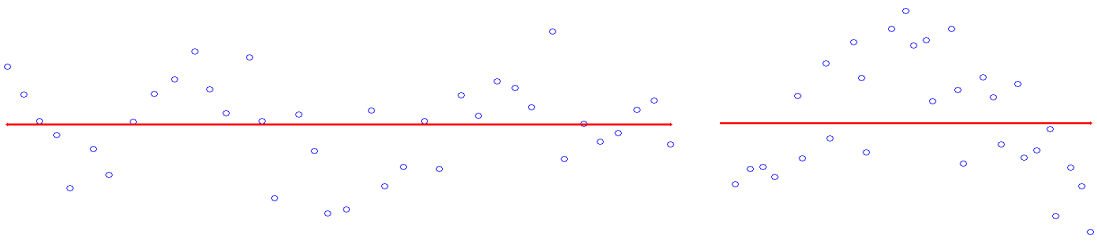
The second experimental scan showed high rates of false positive segments, in the order of 30% of extracted line segments. This was caused by the three curved surfaces shown in Fig. 4.8(c) being approximated by piecewise linear segments. These segments are undesirable as they violate the planar surface assumption used during plane fitting and so a method to differentiate them from true linear segments is needed.

4.3.4 Linearity Verification of Line Segments

The plane fitting method described in Chapter 5 relies on the assumption that the surface features were generated by planar surfaces. As discussed previously, the linear segmentation of the laser range scans can sometimes fit piecewise linear segments to nonplanar surfaces. This can lead to poor planar surfaces being generated that do not correspond to any real surface which could cause major problems for surface merging and footstep planning. This section presents a method to filter out line segments generated by nonplanar surfaces.

An example of such false line segments is shown in Fig. 4.9 below. Both line segments are shown in red with their supporting points in blue and they have identical scale. The line segment in Fig. 4.9(a) is from a linear surface while Fig. 4.9(b) is from a nonlinear surface and a tell-tale curve can be seen.

If a line segment was generated from a planar surface then it is reasonable to



(a) Example linear segment. (b) Example nonlinear segment.

Figure 4.9: Linearity of extracted line segments.

expect that the variance of those points should lie almost exclusively along the direction of that line. A small amount of variance in the orthogonal direction is to be expected but this should be minimal and caused by sensor noise alone. It then stands to reason that a nonplanar surface being approximated by a linear segment should have a relatively higher variance in the orthogonal direction. These segments tend to be shorter in length due to the piecewise nature of approximating a curve by a line segment.

This was seen in Fig. 4.9 where the short length of the nonlinear line segment produces a smaller relative variance along the line in comparison to the orthogonal direction. While both segments appear to have similar levels of orthogonal variance in absolute terms, the key idea is that the relative variance in the orthogonal direction is much larger for a nonlinear segment.

An orthogonal least squares line fit can be performed using PCA which finds a new orthogonal basis for the data points in which the variance along the direction of the line is maximised. It does this by finding the eigenvectors of the covariance matrix of the data, in the order of highest to lowest eigenvalue. Each eigenvalue corresponds directly to the proportion of total variance present along the direction of its corresponding eigenvector. The first eigenvalue λ_1 provides a measure of the variance along the line while λ_2 gives the variance orthogonal to the line. PCA is also used for plane fitting in this work and a full description and derivation can be found in Section 5.4.

A variance based metric is proposed using these two eigenvalues and it will be evaluated on two sets of line segments from the experimental datasets. The first

dataset consisted of linear line segments only while the second set contained only line segments that were generated by piecewise approximation of nonplanar surfaces. The metric used was the percentage of variance explained by the orthogonal direction of the line, as in (4.23), with lower values indicating higher linearity.

$$\left(\frac{\lambda_2}{\lambda_1 + \lambda_2}\right) \times 100 < \Delta \tag{4.23}$$

The value of the threshold Δ can vary depending on the relative importance of minimising false positives or false negatives. This can be seen below in Table 4.5.

Table 4.5: Line segmentation - Linearity verification

Δ	True +	False +
<1.5%	80%	0%
<2.0%	90%	10%
<2.5%	100%	30%

A value of 1.5% gave no false positives as it was able to filter out all the non-linear segments successfully. However, several legitimate linear segments were also removed. Higher thresholds reduced the rate of false negatives but began to introduce more false positives which are generally undesirable. In this work a value of 1.5% was chosen to minimise the number of false line segments passing through to the plane fitting stage. The effect of a false plane being fitted to nonplanar data could potentially be catastrophic whereas missing a single line segment is much less likely to cause the plane fitting to fail due to the redundancy of multiple scans.

This method was implemented on the range segmentation results shown previously in Table 4.4. Updated results can be seen below in Table 4.6. True positive detection was slightly lower but false positive rate was significantly reduced and entirely eliminated when pre-splitting was also used.

Table 4.6: Range segmentation results with nonlinear verification

Type	True + (old)	True + (new)	False + (old)	False + (new)
Segmentation only	81.4%	69.8%	200%	29%
Segmentation + Pre Splitting	90.7%	81.4%	157%	0%
Segmentation + Pre Splitting + Post Merging	97.7%	88.4%	157%	0%

4.3.5 Representation of 2D Line Segments

For the purposes of the feature grouping, as detailed in Section 4.4, it is convenient to represent each laser line segment as a set of two endpoints. It is assumed that the sensors and robot are stationary during data acquisition and that the laser beam width is negligible which is reasonable at close range. A line can be fitted to the points supporting each segment using 2D PCA. This generates a direction vector $\mathbf{v}$ and a centre of gravity point $\mathbf{p_0} = [x_0\ y_0]^T$ that lies on that line. Using these it is possible to map the two end points from each segment onto the line to give the desired endpoints. This can be done orthogonally or radially as seen in Fig. 4.10 below.

The orthogonal approach can be performed as a projection of the vector $[\mathbf{p} - \mathbf{p_0}]$ onto the vector $\mathbf{v}$ and then the location offset $\mathbf{p_0}$ is added, as shown below in (4.24). It is assumed that the vector $\mathbf{v}$ has been normalised.

$$\mathbf{p_{orth}} = \mathbf{p_0} + ([\mathbf{p} - \mathbf{p_0}] \cdot \mathbf{v})\,\mathbf{v} \tag{4.24}$$

The radial approach is performed by calculating the intersection point of the laser beam ($l_1 = b\mathbf{p}$) with the 2D line ($l_2 = \mathbf{p_0} + a\mathbf{v}$). This is shown below in (4.25) where the direction vector $\mathbf{p} = [p_x\ p_y]^T$ is formed by the vector from the origin (0,0) to point $\mathbf{p}$.

$$\mathbf{p_0} + a\mathbf{v} = b\mathbf{p}$$
$$\mathbf{p_{rad}} = \frac{v_y x_0 - v_x y_0}{v_x p_y - v_y p_x}\mathbf{p} \tag{4.25}$$

The radial projection is used as it is more accurate physically, although the orthogonal projection does provide a reasonable approximation if the LRF range noise is small.

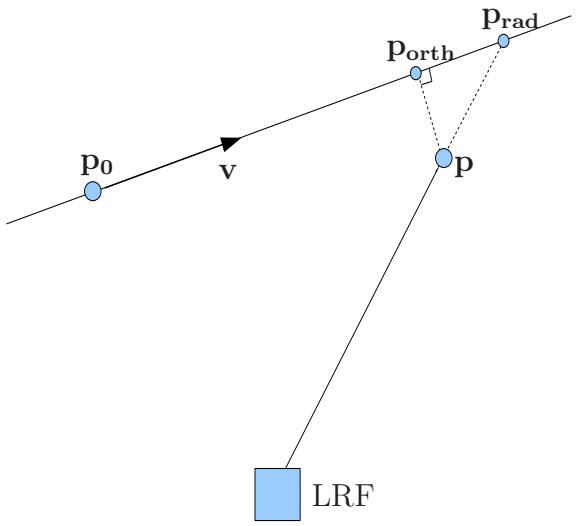


Figure 4.10: Orthogonal and radial projections of a LRF point onto a 2D line.

4.3.6 Transformation of Range Points

Each laser range measurement is a 2D point in the laser CF. To be useful it needs to be transformed into various other CFs as described below.

Laser CF to World CF Transformation

The Hokuyo LRF provides range measurements every 0.36° with the 0° bearing being directly in front of the laser housing and positive bearing angles proceeding anticlockwise. Each range measurement r_i has an associated bearing θ_i and these can be used to convert to cartesian coordinates $\mathbf{p_l}$ within the 2D laser scanning plane using (4.26). The range measurements are in millimetres and the bearing must be in radians.

$$\mathbf{p_l} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}_l = r_i \begin{bmatrix} -\sin(\theta_i) \\ 0 \\ \cos(\theta_i) \end{bmatrix} \tag{4.26}$$

The covariance matrix Σ_{p_l} for $\mathbf{p_l}$ can be found from the range and bearing variances σ_r^2 and σ_θ^2 using (4.27-4.30), as they are independent of each other.

$$\Sigma_{p_l} = \begin{bmatrix} \sigma_x^2 & 0 & \sigma_{xz}^2 \\ 0 & 0 & 0 \\ \sigma_{xz}^2 & 0 & \sigma_z^2 \end{bmatrix} \tag{4.27}$$

where

$$\begin{aligned} x &= f(r_i, \theta_i) = -r_i \sin(\theta_i) \\ \sigma_x^2 &= \left(\frac{\partial f}{\partial r}\right)^2 \sigma_r^2 + \left(\frac{\partial f}{\partial \theta}\right)^2 \sigma_\theta^2 \\ &= \sin^2(\theta_i)\sigma_r^2 + r_i^2 \cos^2(\theta_i)\sigma_\theta^2 \end{aligned} \tag{4.28}$$

$$\begin{aligned} z &= g(r_i, \theta_i) = r_i \cos(\theta_i) \\ \sigma_z^2 &= \left(\frac{\partial g}{\partial r}\right)^2 \sigma_r^2 + \left(\frac{\partial g}{\partial \theta}\right)^2 \sigma_\theta^2 \\ &= \cos^2(\theta_i)\sigma_r^2 + r_i^2 \sin^2(\theta_i)\sigma_\theta^2 \end{aligned} \tag{4.29}$$

$$\begin{aligned} \sigma_{xz}^2 &= \left(\frac{\partial f}{\partial r}\frac{\partial g}{\partial r}\right)\sigma_r^2 + \left(\frac{\partial f}{\partial \theta}\frac{\partial g}{\partial r}\right)\sigma_\theta^2 \\ &= -\sin(\theta_i)\cos(\theta)\sigma_r^2 + r_i^2 \sin(\theta)\cos(\theta_i)\sigma_\theta^2 \end{aligned} \tag{4.30}$$

In many cases the bearing variance can be considered negligible and (4.28-4.30) can be simplified to give Σ_{p_l} as shown in (4.31). This is generally not necessary as the extra calculation is not significant and does not have any further effect on later equations.

$$\Sigma_{p_l} = \sigma_r^2 \begin{bmatrix} \sin^2(\theta_i) & 0 & -\sin(\theta_i)\cos(\theta) \\ 0 & 0 & 0 \\ -\sin(\theta_i)\cos(\theta) & 0 & \cos^2(\theta_i) \end{bmatrix} \tag{4.31}$$

The world CF is setup as a left hand coordinate system, as seen in Fig. (4.11). It is an extension of the 2D image CF used by the camera, where the X axis points to the right and the Y axis points up. The Z axis provides depth to extend the CF into 3D. As the laser range finder is generally mounted in the same direction as the camera, its 2D scanning plane will be the XZ plane.

To determine the position $\mathbf{p_w}$ of the 3D point in the world CF, the kinematics of the sensor motion need to be taken into account. This is the same process as described by the transformations for the camera image points into the world CF in Section 4.2.3. For the simulated and experimental data acquisition setups, described in Section 3.3, the only joint that moves between 2D scans is Joint B which rotates through the angle ϕ. The transformations can, however, be easily derived for any translation and rotation of the sensor suite, such as will be required for the six DOF

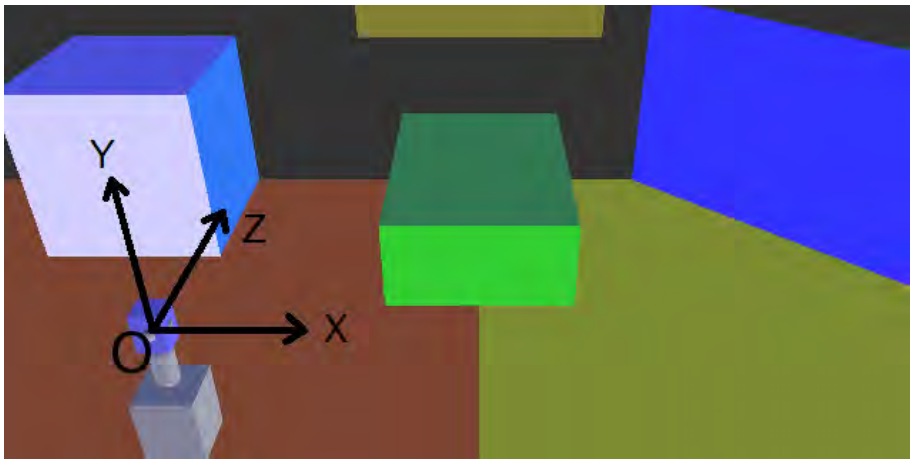


Figure 4.11: 3D left handed coordinate frame with sensors located at point O.

wall climbing robot.

Equation (4.32) is used to transform the cartesian point $\mathbf{p_l}$ from the laser CF into the world CF to get $\mathbf{p_w}$. The translation from the LRF CF origin to joint B, as shown in Figs. 3.3(b) and 3.6, is denoted $T_l = T_L + T_S$. The translation from joint B back to the scan origin at point A is denoted T_B. The rotation about the tilt axis is described by the rotation matrix R_ϕ in (4.7).

$$\begin{aligned} \mathbf{p_w} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}_w &= H(x_l, y_l, z_l, \phi) = R_\phi[T_l + \mathbf{p_l}] + T_B \\ &= R_\phi \mathbf{p_B} + T_B \end{aligned} \tag{4.32}$$

where

$$\begin{aligned} \mathbf{p_B} &= T_l + \mathbf{p_l} \\ \Sigma_{p_B} &= \Sigma_{T_l} + \Sigma_l \end{aligned} \tag{4.33}$$

The covariance of $\mathbf{\Sigma_{p_w}}$ is found from the input covariance $\mathbf{\Sigma_{p_B}}$ and variance σ_ϕ

using (4.34). This is performed using the Jacobian J_H to propagate the error. As the tilt angle ϕ is independent of the 3D point position, the Jacobian can be split and it can be shown that the remaining Jacobian term for $\mathbf{p_l}$ is equivalent to the rotation matrix R_ϕ.

$$\begin{aligned}\Sigma_{p_w} &= J_H \Sigma J_H^T \\ &= \begin{bmatrix}\frac{\partial H}{\partial x} & \frac{\partial H}{\partial y} & \frac{\partial H}{\partial z}\end{bmatrix} \Sigma_{p_B} \begin{bmatrix}\frac{\partial H}{\partial x} & \frac{\partial H}{\partial y} & \frac{\partial H}{\partial z}\end{bmatrix}^T + \begin{bmatrix}\frac{\partial H}{\partial \phi}\end{bmatrix} \sigma_\phi^2 \begin{bmatrix}\frac{\partial H}{\partial \phi}\end{bmatrix}^T + \Sigma_{T_B} \\ &= R_\phi \Sigma_{p_B} R_\phi^T + \begin{bmatrix}\frac{\partial H}{\partial \phi}\end{bmatrix} \sigma_\phi^2 \begin{bmatrix}\frac{\partial H}{\partial \phi}\end{bmatrix}^T + \Sigma_{T_B} \end{aligned} \tag{4.34}$$

Laser CF to Camera CF Transformation

A common task is the projection of a 3D range point onto the corresponding pixel in a camera image. To convert between the laser CF and the camera CF, the transformations in (4.32) and (4.9) are combined. The LRF scan and camera image may have been acquired at different tilt angles ϕ_l and ϕ_c and so the rotation matrices R_{ϕ_l} and R_{ϕ_c} are introduced to allow for this.

$$\begin{aligned}\mathbf{p_c} &= \begin{bmatrix} x \\ y \\ z \end{bmatrix}_c \\ &= R_{\phi_c}^{-1} R_{\phi_l} (T_l + \mathbf{p_l}) - T_c \end{aligned} \tag{4.35}$$

The 3D point $\mathbf{p_c}$ is then normalized by dividing through by z_c and then multiplied with the intrinsic parameter matrix K for the camera from (A.1) to get the image point $\mathbf{p_i}$.

$$\mathbf{p_i} = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K \begin{bmatrix} x_c/z_c \\ y_c/z_c \\ 1 \end{bmatrix} \tag{4.36}$$

4.4 Feature Grouping

The image and range features extracted in Sections 4.2 and 4.3 are of no use individually for plane fitting. It is necessary to associate together those features that were generated by the same physical surface, to which a planar surface can then be fitted. This is a process known as feature grouping. This section proposes a method to achieve this grouping which is detailed below to group together the laser range and image features. This method will then be validated using the extracted features from earlier in the chapter.

Many approaches refer to feature grouping in a different context to that of this work which uses structural grouping. This is defined as grouping together features that were generated by the same physical surface. This is a similar case to perceptual grouping. However, it is more explicit in the requirement that the features originate from the same surface. This work also requires the grouping of multiple feature types including image lines and laser line segments.

It should be clarified that the approach proposed in this work is a pure grouping method and involves no matching. Grouping involves the clustering of multiple features that have something in common within a single image or across multiple images. Matching on the other hand involves finding and tracking the same feature or feature group across consecutive images. This will be clarified further below.

4.4.1 Feature Grouping Literature

Several approaches to grouping image features have been previously suggested based on various criteria. The grouping of image lines is often performed using spatial or perceptual information. Spatial grouping involves grouping image features that are

close to each other. Perceptual grouping relies on the idea that images features, particularly image lines, with similar properties are unlikely to have been generated by chance. Such properties include collinearity and a common termination junction for a pair of line segments.

Mohan *et al.* [111] uses perceptual grouping to extract buildings from aerial images based on parallel lines. Chang *et al.* [112] incorporates both spatial and perceptual grouping of features for matching across images. They group image lines that are parallel or touching and are spatially close in location.

Image line features can be grouped as a precursor to a feature matching algorithm. Rather than match individual features across images, groups of features are matched as they provide a lower dimensionality and more unique feature set than individual features. Shen *et al.* [113] used this approach by using each image corner or junction as a group. Each such group consisted of the image lines terminating at the common point as well as the point location itself. These were then matched across images to provide correspondences for a Structure From Motion (SFM) algorithm. This junction based approach is not suitable for this work, as the grouping criteria are not surface based.

Some grouping methods create a graph structure to store the connectivity between the image lines and image corners. This is a compact and convenient structure to use and it is also followed in this work, as previously mentioned. Grouping methods often attempt to match graph or subgraph structure between images in a matching step. Horaud *et al.* [103] constructed such graphs of image lines from two images and used them for feature matching to find stereo correspondences for stereo vision. This matching was done using a graph search and attempting to match the graph structures or substructures of the two images.

McLauchlan *et al.* [104] combined these two ideas for their SFM algorithm. They extended the junction matching to operate on subgraphs and used a coplanarity test for grouping image lines. They incorporated the plane equation as a constraint for the SFM. However, they did not actually perform any plane fitting. This requires

the image lines to be matched correctly so that a 3D line can be fitted, which allows for coplanarity testing. This difficulty is avoided by using the laser line segments instead.

Image point features are commonly matched in stereo vision [114] and visual SLAM [115] algorithms. These methods use point features that may not necessarily be physical image corners and the matching is usually performed using feature descriptors such as SIFT [29]. An overview and evaluation of some commonly used point feature descriptors is given by Moreels *et al.* [116]. Such feature descriptors allow for robust matching of rich visual features between images but they can be expensive to calculate. This can be a severe restriction for real time operation on smaller robotic systems. These are straight matching approaches, as they involved no grouping of features, and so they are of little use.

Grouping of laser line segments is not usually performed for plane fitting with the exception of scan line grouping. This was introduced by Jiang *et al.* [117] and it is used as a region growing method for fitting planes to dense range data by first segmenting each individual 2D LRF scan into line segments. Adjacent scans are grouped based on a statistical similarity measure. The same approach of grouping line segments is used in this work. However, as it must operate on sparse scans, the scan line grouping approach would likely fail in many scenarios. The gap between each 2D scan is large and there is not always a good overlap between line segments. Therefore, a new feature grouping approach is needed to handle sparse scanning.

4.4.2 Proposed Feature Grouping Method

The feature grouping method proposed in this work differs from most of the methods discussed above. The grouping criterion is structural and requires that grouped features originate from the same physical surface. This is a strict requirement as falsely grouped features will cause the plane fitting to produce poor results. This necessitates that the grouping be conservative.

The feature grouping must be able to handle both laser line segments as well as

image lines. This is a major difference from other approaches which only attempt to group one feature type. Grouping multiple feature types has the benefit of having more available information to work with. Colour information is used which is often neglected in other work and is a valuable addition to improve the robustness of the grouping.

Features from consecutive 2D scans, each consisting of a LRF scan and a camera image, are grouped rather than matched as in other work where the same feature is tracked across multiple images. It is not necessary for the same image features to be detected in each and every image during the grouping stage. This is a major benefit if the feature extraction proves difficult in certain environments.

Feature Grouping Steps

The feature grouping process combines three separate steps of laser-image grouping, image-image grouping and laser-laser grouping. The interaction of these three steps will be outlined briefly below and can also been seen in Fig. 4.12. Each stage will be explained in detail later in the chapter.

1. Extract laser and image features from new 2D scan.
2. **(Laser-Image Grouping)**
   Associate each laser line segment that intersects a surface with the image features that surround it on the new scan.
3. **(Image-Image Grouping)**
   Associate together all image features from the new scan that surround a common surface.
4. **(Laser-Laser Grouping)**
   Associate the grouped features from steps 2 and 3 with those of previous 2D scans by matching coplanar line segments.
5. Get the next scan and go to Step 1.

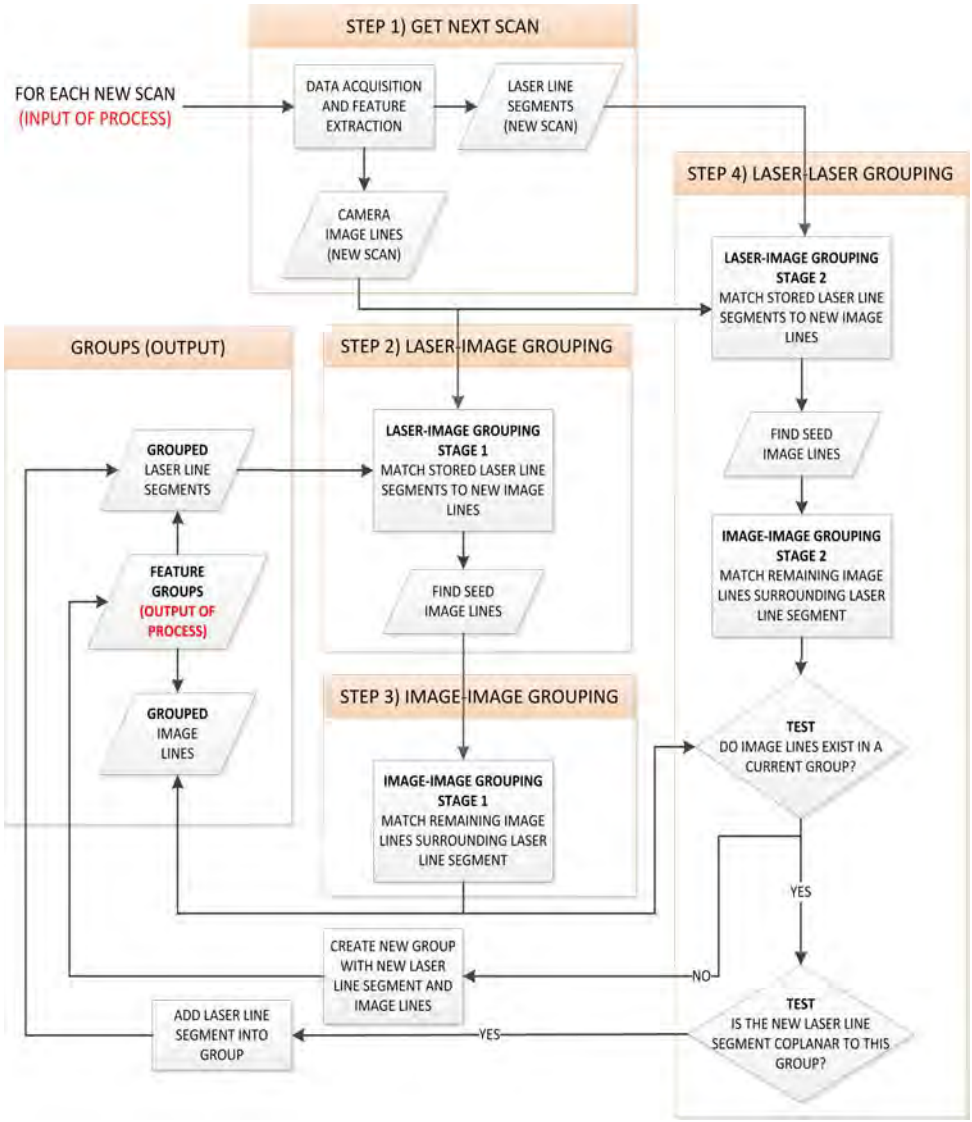


Figure 4.12: Feature grouping process.

An example of grouped features is shown below in Fig. 4.13. The three images show the grouped features for the front of the green box from three consecutive scans. It consists of three laser lines segments (red) and 22 image lines (white). Each image line has a small white tick pointing towards the centre of the surface it is assigned to. This signifies the side of the image line assigned.

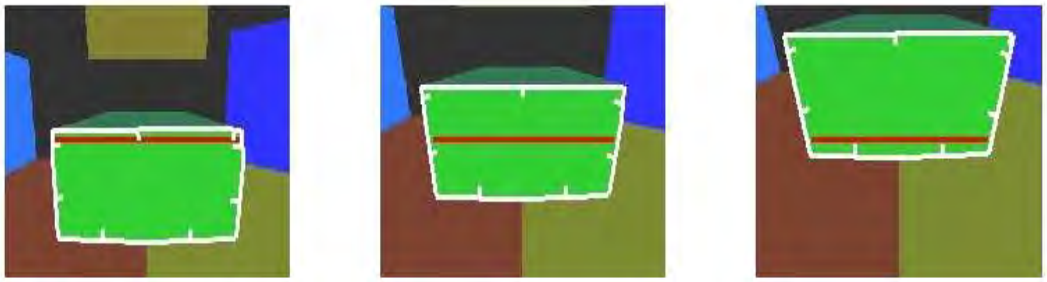


Figure 4.13: Example of grouped features generated by the same physical surface from consecutive 2D scans.

Colour Descriptors

During the grouping processes to be described shortly, one of the major criteria used for testing correct group matches requires that the features share the same colour, texture or pattern. It is reasonable to expect that image features generated from the same surface share a common colour. There are obviously exceptions to this rule. Flat surfaces with non-uniform colour exist, most commonly consisting of images or text, such as a poster, book cover or a sign. The majority of these cases can still be handled with a reasonable degree of success despite this. Extensions to the colour descriptor to include texture could also help, but this has not been implemented.

The descriptor chosen is RGB colour as it is easy to implement and test for potential matches. It will later be shown to provide good matching performance. As RGB camera images are often optimised for human eye perception, they can suffer from low dynamic range. A possible improvement would be to use High Dynamic Range (HDR) images, which could provide for greater contrast and improved performance under varying light conditions. They are more difficult to generate and generally require multiple images, a higher number of bits per colour and some extra computation. This reduces their appeal as these factors will increase the image transfer and computational times and so HDR imaging is not used in this work.

To handle surfaces of varying texture the RGB standard deviation will also be used so that both the colour and the variation are used in the matching process. Therefore, each feature will be assigned a colour descriptor vector C consisting of six parameters, as shown in (4.37).

$$C = [R_\mu \;\; G_\mu \;\; B_\mu \;\; R_\sigma \;\; G_\sigma \;\; B_\sigma] \tag{4.37}$$

To determine matches between any two colour descriptor vectors C_i and C_j, the weighted Euclidean distance is used as in (4.38). The standard deviation terms are weighted using W_σ. This value is dependent on the importance placed on matching the standard deviations. It is also used to normalise the standard deviation values to the RGB values. A weight of $W_\sigma = 0.5$ was used for the simulated data and $W_\sigma = 2$ for the noisier real data, but these can be tuned depending on the environment. A match is determined if the distance $D_{i,j}$ was less than a given threshold. This value should take into account the noise and lighting conditions of the environment and should also be adjusted relative to the size of the weight W_σ.

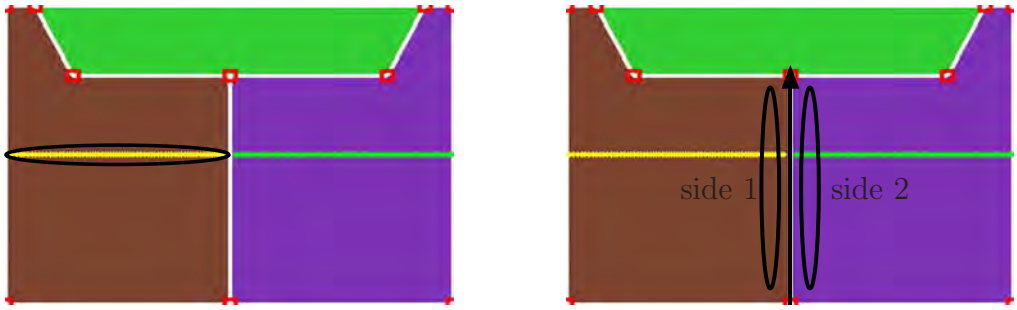
$$\begin{aligned} D_\mu &= \sqrt{(R_{\mu_i} - R_{\mu_j})^2 + (G_{\mu_i} - G_{\mu_j})^2 + (B_{\mu_i} - B_{\mu_j})^2} \\ D_\sigma &= \sqrt{(R_{\sigma_i} - R_{\sigma_j})^2 + (G_{\sigma_i} - G_{\sigma_j})^2 + (B_{\sigma_i} - B_{\sigma_j})^2} \\ D_{i,j} &= \sqrt{D_\mu^2 + W_\sigma^2 D_\sigma^2} \end{aligned} \tag{4.38}$$

Calculation of Colour Descriptors

The colour descriptor for each laser line is calculated by mapping the laser line onto the image from the same scan and extracting the line of N pixels between the endpoints, as seen in Fig. 4.14(a) below. The mean and standard deviation are then calculated from those pixels for the R, G and B values.

Each image line is represented in the graph as consisting of two lines, one for each side of the line. As mentioned earlier in Section 4.2.2, this is because each side will be assigned to a different surface. The colour descriptor for each side is made up of the line of pixels on either side of the image line as shown below in Fig. 4.14(b). This region is chosen to take into account any small error in the line location from the edge fitting to ensure that the pixels are extracted from the correct surface. The several pixels adjacent to each image line endpoint (corner) are also ignored to allow

for acute angles between connecting image lines.



(a) Pixels used for colour descriptor of laser line segment projected onto image.

(b) Pixels used for each image line side colour descriptor. The arrow depicts the direction of the image line.

Figure 4.14: Colour descriptors for features.

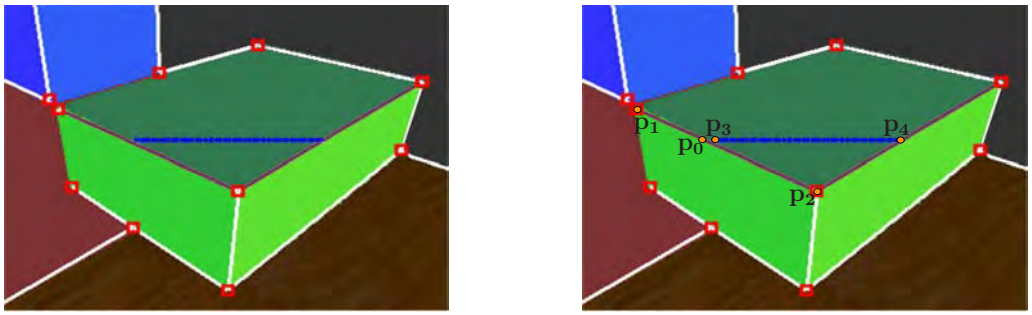
A colour descriptor is not needed for the image corners as they are not directly used in the grouping approach proposed here. While the corners provide location information and anchor the image lines, it is only the laser line segments and image lines that are used during grouping.

4.4.3 Laser-Image Feature Grouping

The second step of Fig. 4.12, after acquiring the new scan, is to group the new laser range and image features together. They are associated with each other by finding the image features that surround a laser line segment. Each laser line segment can be mapped onto any image as a 2D line in u,v coordinates using equations (4.35-4.36) from Section 4.3.6. The image lines that are intersected by the end points of the projected laser line segment are selected as candidates for matching, pending verification tests such as colour matching. An example is shown below in Fig. 4.15(a).

Finding the Line Intersection Point

Each image line is represented by two image end points $(\mathbf{p_1}, \mathbf{p_2})$ and likewise each projected laser line segment is represented by $(\mathbf{p_3}, \mathbf{p_4})$. This grouping task now



(a) Laser line segment (blue) projected onto the image and desired seed image lines (purple).

(b) Image points used to calculate intersection of laser line segment with potential image lines using (4.39-4.40).

Figure 4.15: Laser-Image feature grouping process.

reduces to solving a 2D line intersection problem. Using the parametric equation of a line it is possible to generate two line equations as seen in (4.39) and (4.40). This is demonstrated in Fig. 4.15(b).

$$\mathbf{p_l} = \mathbf{p_3} + u_a(\mathbf{p_4} - \mathbf{p_3}) \tag{4.39}$$

$$\mathbf{p_i} = \mathbf{p_1} + u_b(\mathbf{p_2} - \mathbf{p_1}) \tag{4.40}$$

By finding the intersection point $\mathbf{p_0}$, where $\mathbf{p_l} = \mathbf{p_i}$, it is possible to solve for u_a and u_b as in (4.41) and (4.42).

$$u_a = \frac{(x_4 - x_3)(y_1 - y_3) - (y_4 - y_3)(x_1 - x_3)}{(y_4 - y_3)(x_2 - x_1) - (x_4 - x_3)(y_2 - y_1)} \tag{4.41}$$

$$u_b = \frac{(x_2 - x_1)(y_1 - y_3) - (y_2 - y_1)(x_1 - x_3)}{(y_4 - y_3)(x_2 - x_1) - (x_4 - x_3)(y_2 - y_1)} \tag{4.42}$$

These equations can be simplified if the laser and camera data were acquired at the same pose and the offset between camera and LRF apertures was small. In this

case it can be assumed that the laser line segment mapped onto the image is parallel to the U image axis and thus $y_3 = y_4$. This greatly simplifies the equations to give (4.43) and (4.44).

$$u_a = \frac{-(y_1 - y_3)}{(y_2 - y_1)} \tag{4.43}$$

$$u_b = \frac{(x_2 - x_1)u_a + (x_1 - x_3)}{(x_4 - x_3)} \tag{4.44}$$

Determination of Valid Intersection for Grouping

The parallel distance in pixels from the end of the laser line segment to the intersection point can be calculated using (4.45). For a match to be achieved this distance should be less than some threshold Δ_d. This ensures that only image lines that intersect very near to the end of each line segment are grouped, allowing for some error in the feature extraction locations.

A value of $\Delta_d = 10$ was used. This was based on two factors. Firstly, three standard deviations from the line position were used. This standard deviation was found to be up to 2.5 pixels in Section 4.2.1. Secondly, an allowance was made to account for the conservative nature of the modified laser line segmentation process used. This was discussed in Section 4.3.2.

This distance calculation should be performed for each end of the line as either could be the intersecting end, giving rise to two distances of interest d_3 and d_4. They are the distance from the intersection point $\mathbf{p_0}$ to the endpoints $\mathbf{p_3}$ and $\mathbf{p_4}$ respectively.

$$\begin{aligned} d_3 &= |\mathbf{p_3} - \mathbf{p_0}| \\ &= \sqrt{(x_3 - x_0)^2 + (y_3 - y_0)^2} \end{aligned} \tag{4.45}$$

Again this can be simplified using the horizontal line assumption as discussed previously which gives $y_3 = y_0$. By substituting in (4.39) to give x_0 at u_a, the final equation for d_3 is found in (4.46). Similarly d_4 is found as in (4.47). Both distances are measured in image pixels.

$$\begin{aligned} d_3 &= \sqrt{(x_3 - x_0)^2} \\ &= |x_3 - (x_3 + u_a(x_4 - x_3))| \\ &= |u_a(x_4 - x_3)| \end{aligned} \tag{4.46}$$

$$d_4 = |(1 - u_a)(x_4 - x_3)| \tag{4.47}$$

The u_a parameter can be used to determine where the image line segment is intersected by the laser line segment. Any value within the range $0 > u_a > 1$ means that the intersection lies within the image line segment. The same holds true for u_b and the laser line segment. Therefore the following rules are applied to determine if an image line should be considered a grouping candidate for that laser line segment.

1. $\mathbf{0 > u_a > 1}$ - The intersection point should lie on the image line segment.
2. $\mathbf{d_3 < \Delta_d}$ or $\mathbf{d_4 < \Delta_d}$ - The intersection point should be near the end of the laser line segment.

Choosing the Correct Side of the Image Line

If the above conditions hold, then that image line is deemed to be a match to the laser line segment. Before they can be grouped together, it is necessary to determine which side is the inside of the image line. The inside of an image line is the side that is on the same surface as the laser line segment. Using the notation for image line sides outlined in Section 4.2.2, if a vector $\mathbf{v}$ is created that originates at $\mathbf{p_1}$ (Corner A) and terminates at $\mathbf{p_2}$ (Corner B), then Side 1 is on the left hand side of the vector and Side 2 is on the right hand side.

To determine the side lying on the inside of the image line, a second vector $\mathbf{v_0}$ is created that originates at $\mathbf{p_0}$ and terminates at the midpoint of the laser line segment. The midpoint is used rather than either laser line segment endpoint as it will always lie on the desired surface, whereas the endpoints may occasionally lie on the outside of the image line due to errors in the feature extraction.

The signed angle between these two vectors is found using (4.48). This can be used to find the difference in angle θ_v between two lines if they are represented as vectors originating from the same point.

$$\theta_v = \text{atan2}(\mathbf{v_0}) - \text{atan2}(\mathbf{v}) \tag{4.48}$$

The angle θ_v is used to determine which side of the image line lies on the inside of the surface as shown below. Note that by definition θ_v cannot be 0° as the two lines would be identical in this case.

- **Side 1:** $-180^\circ > \theta_v > 0^\circ$
- **Side 2:** $0^\circ > \theta_v > 180^\circ$

Once the correct side has been determined, all that remains is to check that the colour descriptor for the laser line segment matches the chosen image line. If they do match, then the image line segment side can be added to the same group as the laser line segment. A check should also be made to ensure that this image line side has not already been grouped previously. If so, then the two groups should be considered for merging as each image line side corresponds uniquely to one and only one real surface.

4.4.4 Image-Image Feature Grouping

The third feature grouping step, as shown in Fig. 4.12, is the grouping of the image lines. These form the boundaries of surfaces imaged by the camera. To determine

the polygon boundaries of each surface it is therefore required to group together all the image lines that correspond to the same surface. The method proposed below is based on the idea that the boundary image lines are linked in the image feature graph. Given any image line as a seed, it is then possible to traverse the graph to find connecting lines to group together.

Once an image line side has been grouped to a laser line as described above, that line is used as a seed line to find further matching lines that correspond to the same physical surface. Every image line has two endpoints that are distinct nodes in the graph. Each of these endpoints is chosen in turn as a starting point to traverse the graph. All edges leaving the node corresponding to that endpoint are potential match candidates as they are connected to the seed image line side.

The following criteria are used to determine which of these image line sides, if any, should be grouped.

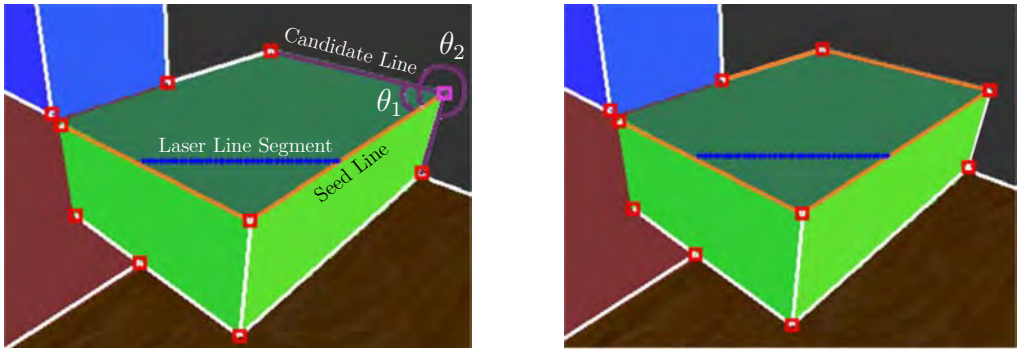
Grouping Criteria: The desired image line side should meet all criteria below.

1. Be connected to an image line currently in the group.
2. Be the first image line radiating from the corner on the same side of the line.
3. Have the same colour descriptor for the corresponding line side.

Criterion 1 selects only those image lines terminating at the seed corner node, excluding the seed line. Criterion 2 then chooses from within these lines, the one image line that continues the image boundary surrounding the surface of interest. This is done by converting each of these image lines into vectors and (4.48) is then used to find their angle relative to the seed image line. The image line with the smallest angle on the same side as the seed line side is chosen and then Criterion 3 is used as a final check by requiring that the colour descriptors match. This is important as if the feature extraction is imperfect then a missing image line could cause the wrong image line exiting the node to be chosen. This image line will likely correspond to a different surface and thus should not be grouped.

If these three grouping criteria are satisfied, then the chosen image line side is added to the group and then becomes the next seed image line. This grouping

process is demonstrated in Fig. 4.16. The extracted seed image lines are shown in orange with the laser line segment in blue as seen in Fig. 4.16(a). The seed corner, in purple, leads to two connected candidate image lines, also shown in purple. The angles θ are evaluated for each of these and θ_1 is chosen. After a successful colour match, this process continues until the grouping is terminated leading to the four grouped image lines shown in Fig. 4.16(b). Note that one image line is missing as it was not extracted.



(a) Candidate image lines connected to seed image line corner, both shown in purple.

(b) Grouped image lines (orange).

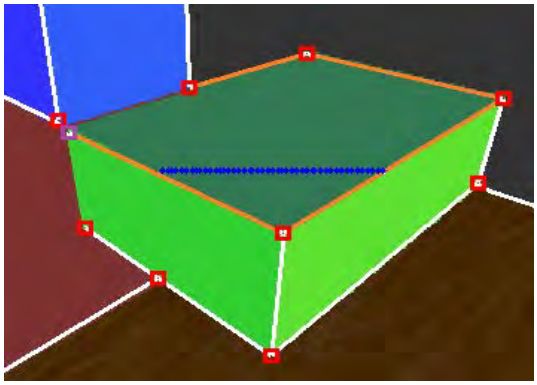
Figure 4.16: Image-Image feature grouping process.

Termination Conditions: Continue until any of the following are true.

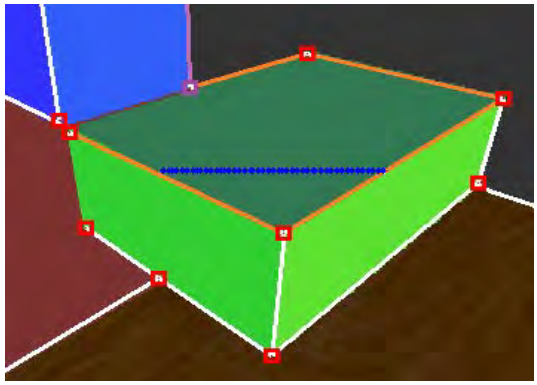
1. No other image lines are connected to the seed line corner.
2. The chosen connected image line does not match the colour descriptor.
3. The chosen connected image line is already in the group.

The first two termination conditions are self explanatory. Termination condition 3 is based on the assumption that the chain has already been followed from the other side of the image line and so further grouping is redundant for this chain. Examples of these conditions are shown in Fig. 4.17. As above, the grouped lines are in orange, the seed image corner is purple and the candidate line to test is also purple.

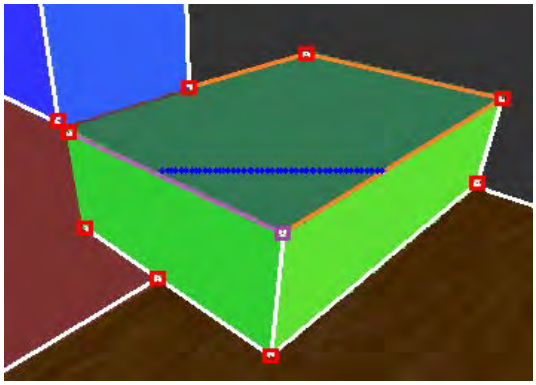
This method to group connected image lines within a single image is reliant on having an accurate image feature graph and good selection of image seed lines.



(a) No ungrouped image line is connected to potential corner (purple). Orange image lines have already been added to the group.



(b) Colour descriptor of potential image line (shown in purple) does not match grouped features.



(c) Potential image line (purple) is already in the group.

Figure 4.17: Image-Image feature grouping termination conditions.

If the feature graph is perfect, then a single seed line should successfully group all surrounding image lines. However, in any realistic scenario this is unlikely. It is therefore important to generate good seed image lines and this is currently done using the laser line segment edge intersections only, as described in Section 4.4.3. To improve the robustness of this grouping it may be necessary to detect additional seed image lines using other means. This should allow the successful grouping of isolated image lines in the situation of poor feature extraction leading to an incomplete image feature graph.

4.4.5 Laser-Laser Feature Grouping

The last grouping step of Fig. 4.12 is to group the laser line segments together. These are the key to grouping the features across images. This can eliminate the need for

feature matching/tracking of the image lines as in other grouping approaches. Each new laser line segment will either be associated with an existing group or will spawn a new feature group.

For each new scan, the new image lines are associated with existing groups as discussed previously in Sections 4.4.3 and 4.4.4. These two steps are then repeated using the new laser line segments. They are projected onto the new image one at a time and the intersecting image lines are extracted as seeds. If these seed lines have already been associated with a group, then the laser line segment is also associated with that same group if several criteria are met.

Grouping Criteria: The laser line segment should meet all criteria below.

1. The colour descriptor of the new laser line should match that of the group.
2. There should exist some continuous path within the image between the new laser line segment and existing line segments.

The first criterion is a standard part of this grouping approach and should already be satisfied if the previous grouping steps have been performed correctly. The second criterion requires that there is no jump or discontinuity on the image between the lines. This would suggest that the line segments originated from different surfaces and that there is some error in the image feature graph. In this case they should not be grouped together.

Verification of Planarity of Grouped Laser Line Segments

Note that so far during the grouping there has been no mention of verifying the coplanarity of the grouped laser line segments. This was deliberately done as non-planar features can still form a valid group if the true surface is nonplanar. However, it is important for planar surface fitting to verify the coplanarity of the grouped laser line segments at some stage prior to plane fitting. This determines if a planar surface should be fitted to each group or if some other surface representation should be used. This verification can be performed at any time prior to plane fitting and

it is convenient to do this during the laser-laser grouping process. Two criteria can be used to satisfy the coplanarity.

Coplanarity Criteria: The grouped laser lines should meet both criteria.

1. A new laser line should be coplanar with all existing laser lines within the group.
2. A quasi-linear RGB image path should exist that links the laser line segments.

The first criterion is a 3D coplanarity test based on the 3D lines fit to each line segment. There are three different situations each requiring different variations of the basic coplanarity test. It should be noted that checking laser line segments from the same scan is redundant. By definition they are coplanar as they both lie on the same 2D laser scanning plane.

Case 1: Two laser line segments

The simplest case involves testing the new laser line segment against only one existing laser line segment from a previous scan. The existing line segment runs from point A to point B in 3D space and the new line segment runs from point C to point D to give vectors AB and CD. A third vector AC can be created which runs between the two line segments' endpoints. These three vectors should all be normalised before using them here. The two line segments are coplanar if (4.49) is true. Due to noise in the sensor data, a small threshold should be used, as in (4.50). In this work a value of $\Delta_{p_1} = 0.05$ radians was used. This allows for a 3° deviation from orthogonal.

$$[AB \times CD] \cdot AC = 0 \tag{4.49}$$

$$|[AB \times CD] \cdot AC| < \Delta_{p_1} \tag{4.50}$$

Case 2: Three or more laser line segments

An extension of the two line case is needed when there are two or more line

segments already within the group to test against. In this case (4.50) should be true for every line segment pair in the group. For large groups this can become computationally costly which is undesirable.

Case 3: Plane and laser line segment

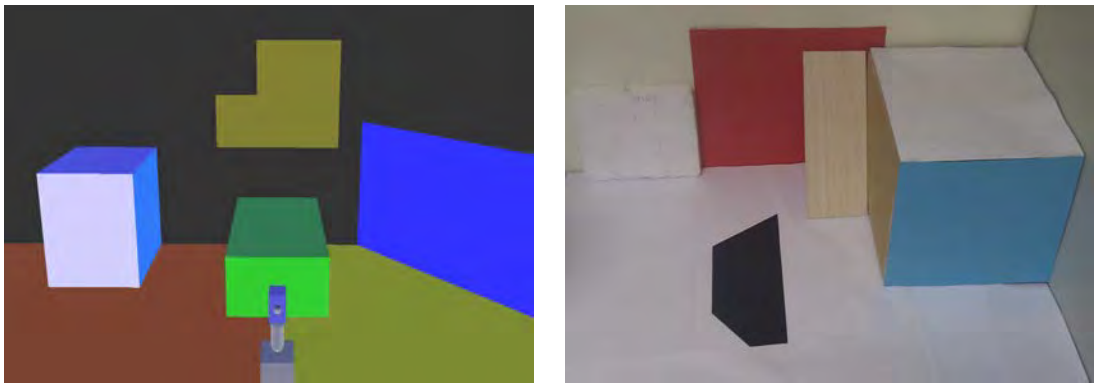
If a plane has already been fitted to the existing laser line segments within the group then it is possible to test if the new line segment lies on the plane. This requires far less computation than individually testing each line segment pair, as in Case 2. If the line segment has endpoints A and B, then all that is required is to test whether both points lie on the plane represented by parameters $\mathbf{n}$ and d. If (4.51) holds true, then the new laser line segment is coplanar and the group can be verified as planar. A threshold of $\Delta_{p_2} = 5$mm was used. This was based on the standard deviation of the laser line segment endpoints.

$$\mathbf{n} \cdot \mathbf{A} - d = 0 \tag{4.51}$$

$$|\mathbf{n} \cdot \mathbf{B} - d| < \Delta_{p_2} \tag{4.52}$$

4.4.6 Results of Feature Grouping

The proposed feature grouping method was evaluated on a simulated scene and a real scene. For each 3D scan a series of metrics was compiled to determine the grouping performance. The simulated scene used was described in detail in Section 3.3.2. It consisted of five 3D scans from differing poses and orientations. Each 3D scan contained eleven 2D scans at an angular resolution of 10°. The experimental scene used was described in Section 3.3.1. It contained three 3D scans from differing poses and orientations, with each scan having nine 2D scans at an angular resolution of 5°. These two scenes are shown below in Fig. 4.18.



(a) Simulated scene. (b) Experimental scene.

Figure 4.18: Test environments.

Tables 4.7 and 4.9 show the accuracy of the extracted groups for the simulated and experimental scenes respectively. The metrics used are described below.

Surfaces

The number of true surfaces visible in the scans.

True Groups

The number of extracted groups that correspond to a true surface.

Missing Groups

The number of true surfaces without a corresponding extracted group.

False Groups

The number of extracted groups not corresponding to any surface. This includes surfaces that have been over grouped where the smaller groups are marked as being false.

Fittable Groups

The number of True Groups that were able to have representative surfaces fitted to them. This required that there were two or more laser line segments in the group and the majority of true surface edges were represented accurately.

Tables 4.8 and 4.10 show the accuracy of the grouping of individual image lines and laser line segments for the simulated and experimental scenes respectively. The metrics used are described below.

Image/Laser Lines

The number of extracted features.

True Group

The number of extracted features that were grouped to their correct surface group.

Wrong Group

The number of extracted features that were grouped to an incorrect surface group.

Table 4.7: Surface grouping results (Simulated dataset)

3D Scan	Surfaces	True Groups	Missing Groups	False Groups	Fittable Groups
A	9	89%	11%	0	100%
B	8	88%	12%	0	100%
C	9	89%	11%	0	88%
D	8	100%	0%	0	75%
E	12	92%	8%	1	60%
ALL	46	91%	9%	1	81%

Table 4.8: Individual feature grouping results (Simulated dataset)

3D Scan	Image Lines	True Group	Wrong Group	Laser Lines	True Group	Wrong Group
A	418	53.5%	0%	27	85.2%	0%
B	323	65.9%	0%	24	100%	0%
C	414	63.3%	0%	30	90.0%	0%
D	341	66.3%	0%	25	100%	0%
E	481	51.4%	0.2%	36	80.6%	0%
ALL	1977	59.3%	0.1%	142	90.2%	0%

Table 4.9: Surface grouping results (Experimental dataset)

3D Scan	Surfaces	True Groups	Missing Groups	False Groups	Fittable Groups
A	9	78%	22%	1	100%
B	7	86%	14%	1	100%
C	9	78%	22%	2	100%
ALL	25	80%	20%	4	100%

Table 4.10: Individual feature grouping results (Experimental dataset)

3D Scan	Image Lines	True Group	Wrong Group	Laser Lines	True Group	Wrong Group
A	391	48.8%	2.8%	32	84.4%	0%
B	324	46.3%	1.5%	26	80.8%	0%
C	429	48.5%	2.1%	33	75.8%	3.0%
ALL	1144	48.0%	2.2%	91	80.2%	1.1%

4.5 Discussion

The quality of the feature extraction and grouping methods introduced in this chapter are critical to the successful fitting of planar surfaces. It is therefore important to examine their performance.

4.5.1 Image Feature Extraction

The image feature extraction was performed on low resolution images using common edge and corner detection schemes as described in Section 4.2. The performance of this edge detection is critical to the success of the plane boundary fitting method described in Section 5.5 to generate the polygon boundary of the planar surface. False positive and false negative features can reduce performance if they are not handled correctly and filtered out during the grouping stage. These are more common in scenarios with difficult surface conditions but such environments cause problems for any vision based methods and so this issue is not restricted to this work.

As the aim of this research is to extract physical structure, physical image corners and edges are obviously desired. Surfaces exhibiting irregular texture, shadow or specular reflectors could cause false edges to be extracted. False edges, particularly those from irregular surface texture, could impact on the feature grouping and surface fitting performance if not handled appropriately. To mitigate any effect, several approaches have been used to filter out these false edges and minimise their impact. This includes the decision to only use linear image edges of a minimum length. This was found to filter out a large proportion of these false edges as short image edges are frequently generated by the linear approximation of textured surfaces and some

specular reflections. The feature grouping process, introduced in Section 4.4, also filtered out a large number of the remaining false edges, as they seldom correspond to other extracted image or range features.

Occasionally false edges from shadows or textured surfaces may make it through to the plane fitting stage. These will generally be those that are large in size and linear in shape. Such false edges can still be handled within the existing surface fitting framework and this leads to extra surfaces being generated. These surface patches could then be merged together with adjacent surfaces if they were found to be coplanar. However, such merging is intentionally not a part of the current approach. It can lead to legitimately distinct surfaces being merged together due to the tolerances involved in the merging process. This could occur when there exists a small physical step change between the two surfaces that falls within the merging tolerances, or when they consist of different materials with systematic range biases that cancel out the range step.

A higher level of surface planarity and accuracy is required for foot placement than can generally be achieved through the merging of adjacent surfaces. If a wall climbing robot attempted to walk on a merged surface with a physical step change it could lead to poor adhesion, which is undesirable. However, this lack of merging is generally only a minor inconvenience as it can lead to fragmentation of some surfaces and it does not adversely affect performance if the surfaces are large enough to fit the robot's foot. Mapping for scenarios other than foot placement may be more tolerant of any merged step change and so this merging process could be easily implemented if desired.

As the proposed mapping method fuses information from both vision and range sources it is less sensitive than vision only methods. It is, however, also more sensitive than pure range methods to such visually difficult surfaces. Due to the redundancy in this approach, as multiple images are used for each 3D scan, it is not crucial to see every image feature in every image. Successful surfaces can still be mapped if most of the relevant surface features are seen in at least one image.

Currently only linear image edges are being extracted. This simplifies their storage and transformation through CFs as each line can be easily represented by its two endpoints. This limits the applicability of this method to structured or indoor environments where the surfaces contain a high proportion of straight surface edges. This is, however, not a major disadvantage for two reasons. Firstly, the wall climbing robot requires structured environments with a high proportion of planar surfaces to be able to walk on and these planar surfaces commonly exhibit linear edges. Secondly, this approach could be extended to use some other compact point based representation for the nonlinear image edges. Possibilities include a piece-wise linear approximation or the use of splines.

4.5.2 Range Feature Extraction

The line segments extracted from the laser range scans are to be used for plane fitting in Section 5.4. It is more important that the 2D segmentation minimises the number of false positive points rather than attempting to find the segment edges exactly. These edges are not used to find the planar polygon boundaries, unlike other plane fitting methods. For this reason the LRF 2D line segmentation was performed conservatively and modifications to the traditional incremental line segmentation were proposed to further this aim.

Many line segmentation algorithms, including the incremental approach, have a tendency to want to follow edges past the true join edge point. Often the line criteria will still be satisfied for the next several points past the true edge. Modifications were proposed to counter this behaviour. They included using the image lines to pre split the laser range scans at known surface boundaries, ignoring the first and last segmented points and reformulating the incremental segmentation algorithm as seen in Algorithm 4.2.

The benefit of these modifications and improvements can be seen in the results shown in Tables 4.2-4.4. The percentage of line segments containing at least one incorrectly segmented edge reduced significantly when the image line pre-splitting

was introduced, especially in the noisier experimental datasets. This will flow onto the accuracy of the planar surfaces fitted to these line segments and provide more accurate surface mapping.

During the range segmentation process, some nonlinear surfaces were piece-wise approximated by linear line segments. This is undesirable as the plane fitting procedure described in the next chapter assumes that the points it is fitting to were all generated by the same planar surface. Methods to mitigate this were outlined in Section 4.3.4 by attempting to determine if an extracted line segment was from a linear or nonlinear surface.

A metric based on the relative variances parallel to and orthogonal to the line was able to filter out over 80% of the false linear segments. It could filter out even more if a low rate of false negatives is acceptable. This choice of threshold depends on the aim of the method and in this work it was deemed more important to be robust against nonlinear extracted line segments. This led to several small linear segments being also removed. These were very short segments and in most cases there already existed at least two other lines from that surface that could be successfully grouped and used for plane fitting.

4.5.3 Feature Grouping

The feature grouping method outlined in Section 4.4 showed promising results. As seen in Table 4.7, it was able to group 91% of the simulated surfaces successfully, with 81% of those being able to have a good planar surface fitted to them. The remaining 19% of these surfaces were unable to have a planar surface fitted to them as they either contained only one laser line segment or they were missing several key image edges. Neither of these problems stemmed from the grouping method itself as the first was caused by the sparse scanning resolution and the second caused by the image feature extraction process.

It should also be noted that within these groups the grouping of individual features was performed accurately but conservatively, as in Table 4.8. Just under

60% of 1977 image lines were assigned to a surface group and of these only 1 image line was incorrectly grouped. The grouping of the laser line segments performed even better with 90% of 142 line segments grouped and no false positives.

The grouping method was then applied to a real dataset, as seen in Tables 4.9 and 4.10. From the three 3D scans, 80% of the surfaces were successfully represented by a feature group and all of these were able to have a good surface fitted to them. Within these groups the rate of image line side and laser line segment grouping was slightly lower than in the simulated case as the additional noise caused more difficult features to exist. However, this rate was still just under 50% and still provided a good grouping of features for use in surface fitting. The false grouping rate was also higher for similar reasons.

This grouping method is not dependent on the matching and tracking of individual features unlike other feature grouping approaches previously mentioned. This is a major benefit as it allows for some level of failure in the feature extraction process, particularly with difficult image extraction scenarios. The most important aspect is the image feature graph which this grouping method uses to associate grouped image features. The grouping performance degrades as the quality and completeness of the image feature graph reduces. It can be seen from the results that the proposed grouping method is still reasonably robust to this.

Grouping failures can either be false negative groupings or false positive groupings. False negative groupings involve a feature not being grouped at all. As multiple images are used, each feature can appear in multiple scans and has more than one chance to be grouped if it is missed from an individual image. The worst case scenario for a false negative is when the feature is not seen in any image and this leads to that edge being unable to be represented on the planar surface fit to that group. This is undesirable but not fatal.

On the other hand, a false positive grouping can lead to a catastrophic failure. This occurs when a feature is associated to a surface group representing a different surface. This leads to that group containing features from multiple surfaces and the

resulting attempted planar surface fit will most likely bear no resemblance to any real surface. This could lead to the wall climbing robot hitting an obstacle or falling off the wall. It is for this reason that the grouping is performed conservatively, as discussed above, and the low level of false positive feature grouping is a testament to the success of this. Several post processing steps will also be introduced in Section 5.5.4. These aim to further tidy and remove any falsely grouped image features once they have been merged together as polygon boundaries.

4.6 Summary

This chapter has presented the extraction and grouping of the image and range features used in this thesis. These features are the basis for the planar surface fitting performed in Chapter 5.

The extraction of the image lines and corners was discussed and the transformation of each image point into the world CF was detailed. The 2D range scans were linearly segmented using several modifications to the existing incremental segmentation approach. The addition of a pre-segmentation splitting step using image edges was shown to be beneficial. This led to an improvement in the accuracy of the segmentation edges. A method was also proposed which verified that each line segment was generated from a linear surface. It successfully filtered out those line segments that were caused by nonplanar surfaces being segmented in a piece-wise fashion. This is important because of the planar assumption inherent to the surface fitting process.

The proposed surface fitting approach requires that the extracted features are grouped together and a method to perform this was presented. This grouping was based upon structural similarity so that features generated by the same physical surface were grouped together. It avoided feature matching and tracking across images, as in other grouping approaches, and was able to handle multiple different feature types. This grouping method successfully grouped features from 80% of surfaces in the experimental test environment and the rate of false positive groupings

was very low, at just over 2%. This will allow for good quality surfaces to be fitted to the majority of the planar surfaces in an environment using the methods outlined in Chapter 5.

Chapter 5

Fitting Planar Surfaces to Grouped Features

5.1 Introduction

The previous chapter described the process of extracting and grouping together range and image features that were generated from the same physical surface. This chapter proposes a method for fitting a planar surface patch to each of these feature groups. The extracted planar surface patches will later be merged together with those extracted from other 3D scans acquired at nearby poses. This will extend the field of view, improve the surface accuracy and build a more complete 3D surface map of the environment. This surface merging is the focus of Chapter 6.

The surface fitting method described in this chapter is a two step process. It first involves fitting an infinite plane to the grouped laser line segments. This is followed by the generation of polygon boundaries by projecting the image features onto the recently fitted plane. The uncertainty information of each surface will also be estimated. This surface fitting method will be described in detail and results will be shown to demonstrate its accuracy. Several real and simulated datasets will be used to demonstrate the basic approach while a series of further datasets will be used to investigate the surface fitting performance using a variety of challenging

realistic surfaces.

5.2 Surface Fitting Literature

The approach commonly used for fitting planar surface patches involves the use of dense range images from either a 3D laser range finder [37][118] or a stereo vision system [59][114]. These range images are segmented through either region growing or edge based methods and then planes are fitted to each segmented point cloud. This requires dense scanning and the segmentation can often be expensive and slow. Because of this, an alternative approach was proposed in Chapter 4 to extract surface features for plane fitting.

The proposed surface fitting method differs in that sparse laser and image data are fused together and used to generate planar surfaces from a minimal number of scans. This will speed up both the data acquisition and processing time to allow real time operation using the limited computational resources of the CRACbot. This is discussed further in Chapter 7.

While the idea of using both range and vision information to produce better 3D maps is not a new one, the approach presented here is novel. A similar idea was introduced by Zureiki *et al.* [37], who discussed the use of infinite line segments from a camera image to add additional features for their EKF 3D SLAM approach. They did not, however, implement this idea and their planar segmentation was based on dense range scanning. Their approach also differs from the one presented in this work, as their image lines were only intended to augment the planar surfaces as additional EKF features. In this work the image line features are an integral part of the planar surface segmentation and fitting processes and allow faster segmentation by using sparse data scanning.

As discussed in Chapter 2, the simplest fusion approaches only applied colour information to their 3D point clouds. This was useful for visualization purposes only. Both Andreasson *et al.* [58] and Diebel *et al.* [57] used vision to interpolate between laser range measurements. They used dense scans and high resolution im-

ages to provide accurate and detailed surface models. These methods were, however, computationally expensive and produced high density point clouds. Vision has been also used to aid in segmentation by Andreasson *et al.* [69] and to add additional range points to occluded areas in laser range images by Dias *et al.* [59] and Jiang *et al.* [60]. All these approaches were aimed at increasing the data density rather than reducing it and none of them used any planar surface fitting.

5.3 Overview of Proposed Surface Fitting Method

The method proposed in this thesis for fitting planar surface patches to sparse range and image data is summarised below in Fig. 5.1.

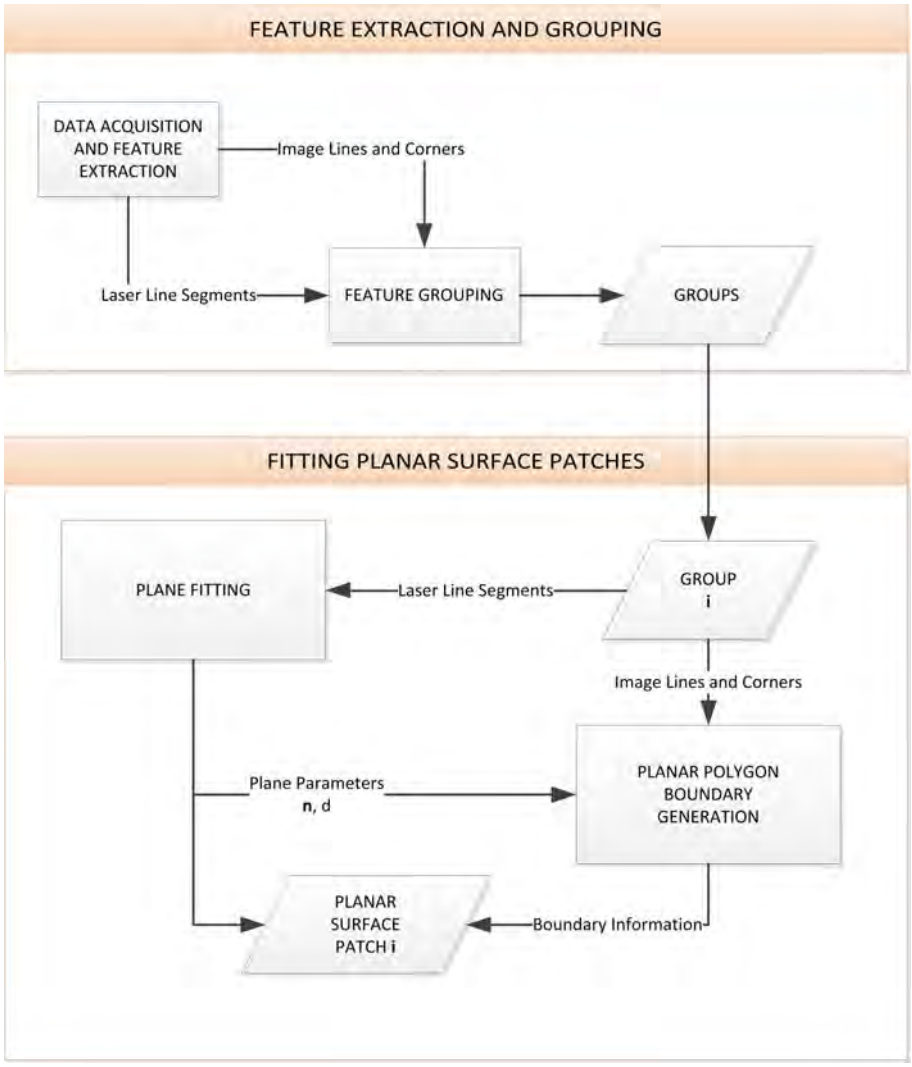


Figure 5.1: Overview of surface fitting methodology.

At each scanning position, a 3D surface scan is acquired. This is achieved through taking a series of 2D scans by tilting the sensor suite through one angular DOF. This is a simplification of the general case where consecutive 2D scans can be acquired at any nearby pose. It simplifies the kinematics and thus the computational burden to help achieve real time performance. This is also the easiest way to ensure that there exists sufficient overlap between the 2D scans.

Features are extracted from the raw sensor data to be used for surface fitting. They are then associated together into groups corresponding to the physical surfaces that generated them. This was discussed in detail in Chapter 4.

Infinite Plane Fitting

A planar surface is fitted to the features from each group using the laser line segments in the world CF. A minimum of two such line segments from different scans are needed to fit a plane and additional line segments allow for an improved fit.

This plane fitting process assumes that the physical surface that generated the range data was planar. This assumption needs to be confirmed by some means and there are several aspects to consider. Firstly, the feature grouping process must be robust and accurate to avoid incorrect associations. This was achieved in Section 4.4 by conservatively matching features so that marginal data was ignored. Secondly, planes should only fit to points from linear laser segments. This will filter out the majority of potential nonplanar surfaces. A method to verify the linearity of these line segments was detailed in Section 4.3.4.

Polygon Boundary Generation

A series of polygons are projected onto the recently found planar surface for each feature group. This uses the grouped image corners and lines. Each image provides a separate boundary estimate and these are then merged together to provide the final polygon boundary for the planar surface patch. This merging allows the corners to be estimated more accurately as they are seen multiple times and it allows for

redundancy in the feature extraction process. It also enables the planar surface boundaries to be extended for large surfaces or occluded areas.

This step is necessary as, unlike other dense plane fitting approaches, the sparse range data does not provide enough information to generate the surface polygon boundaries. However, this approach does allow for very sparse and fast scanning of the environment, as very few scans are needed to map the majority of the planar surfaces in the scene. This is one of the major advantages of this approach, as the number of scans can be reduced by an order of magnitude. The 2D Scans can be acquired at an angular step of $5° - 10°$ as opposed to other plane fitting methods that acquire dense range images by scanning at a step of $0.5° - 1°$.

The sparse scanning of this approach opens up the possibility for using an adaptive tilting step. The scanning can be sparser and faster when the Field of View (FoV) consists of a few large objects such as walls. Areas of the scene with many smaller objects could also be scanned more densely. This idea is discussed further in Section 7.3.3.

5.4 Infinite Plane Fitting

Fitting infinite planes is the first half of the surface mapping method, as introduced in Section 5.3. The fitting of each infinite plane involves two steps. Firstly, the parameters of the plane must be found. Secondly, the uncertainty of these parameters must be estimated.

There are several viable plane models that can be used to represent a plane in 3D space. The general equation of a plane has the four parameters A,B,C,D as seen below in (5.1). The Hessian plane model is a simple reformulation of the general plane equation to physically meaningful parameters. Here A,B,C are normalised to give the normal direction vector $\mathbf{n}$ and D is normalised by the same value to get the distance d. This model is seen in (5.2).

$$A\mathbf{x} + B\mathbf{y} + C\mathbf{z} + D = 0 \tag{5.1}$$

$$\mathbf{n} \cdot \mathbf{p} - d = 0 \tag{5.2}$$

Other plane models have been derived from (5.1) and one useful model is seen in (5.3). This is obtained by rearranging in terms of $\mathbf{z}$ and dividing through by the C parameter. It has eliminated one parameter and this allows for a more compact representation but at the cost of introducing singularities. This means that some planes are impossible to represent using this model. It should be noted that while each model discussed thus far may have used similar symbols they are, however, not always representing the same value. This will be clarified later where appropriate for those models that are used.

$$\mathbf{z} = A\mathbf{x} + B\mathbf{y} + D \tag{5.3}$$

Other models exist with different paramaterisations but the three models mentioned are the most useful, especially the Hessian plane model of (5.2) which is used in many plane fitting approaches. This work uses the Hessian plane model for fitting the infinite plane parameters but the uncertainty of these parameters will be found using the model shown in (5.3). This is the approach taken by Weingarten [35]. However, their covariance calculations were incomplete and unproven. This work will extend their approach by showing and proving the full derivation of the plane parameter covariance equations in Section 5.4.2.

5.4.1 Calculation of Plane Parameters

For each group, an infinite plane is fitted to the set of supporting 3D points from each grouped laser line segment, providing that two or more such segments from different 2D scans are available. The points are transformed into the world CF and

the plane is fitted using Principal Components Analysis (PCA).

The Hessian plane model shown in (5.2) has been used in this work. The parameters of this model are a unit normal vector $\mathbf{n}$ which is orthogonal to the plane and the orthogonal distance d to the origin. These can be seen in Fig. 5.2 below. The infinite plane is shown as dashed lines with an example polygon boundary shown by the thicker red lines.

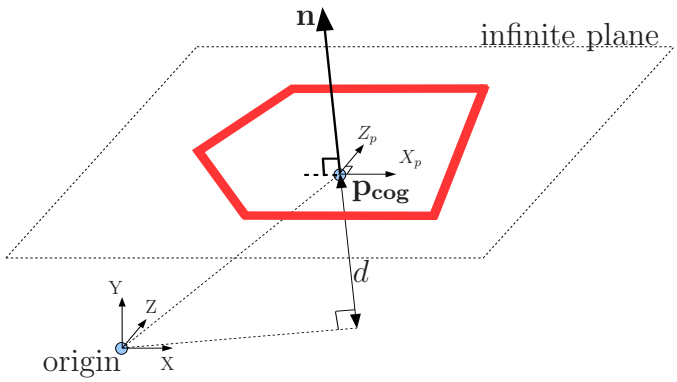


Figure 5.2: Hessian plane model for infinite plane with planar polygon boundary shown in red.

PCA is a commonly used method for fitting planes to 3D data points. It was originally proposed for use in plane fitting by Pearson [119] and has also found extensive uses in statistics and data analysis. More recent work using PCA for plane fitting, particularly in the context of robotic mapping, has been done by Weingarten *et al.* [118] and Poppinga *et al.* [70] amongst others.

PCA acts to generate a new orthonormal basis for a set of data points such that the greatest variance lies in the direction of the first Principal Component (PC) vector. The second PC direction then accounts for as much of the remaining variance as possible and so on. In this manner, PCA can be used to fit an orthogonal plane to a set of 3D points. The normal vector $\mathbf{n}$ of (5.2) is the third PC vector found.

Assuming the data was generated by scanning a plane and that the line segments used were coplanar and from different 2D scans, then the third PC direction should have the least variance. This also makes sense intuitively as the greatest variance should lie along the plane in the first two PC directions, with the third direction accounting only for noise and uncertainty. A useful tutorial on PCA is provided by Shlens [120].

Using PCA for Plane Fitting

To use PCA to fit a plane, it is first required to find the centre of gravity $\mathbf{p_{cog}}$ of the 3D point dataset P. This is equivalent to the mean of the data points as shown in (5.4).

$$\mathbf{p_{cog}} = \frac{1}{N}\sum_{i}^{N}\mathbf{p_i} = \begin{bmatrix}\bar{x}\\ \bar{y}\\ \bar{z}\end{bmatrix} \tag{5.4}$$

where

$$P = \begin{bmatrix}\mathbf{p_1} & \dots & \mathbf{p_i} & \dots & \mathbf{p_N}\end{bmatrix} = \begin{bmatrix}\mathbf{x}\\ \mathbf{y}\\ \mathbf{z}\end{bmatrix} = \begin{bmatrix}x_1 & \dots & x_i & \dots & x_N\\ y_1 & \dots & y_i & \dots & y_N\\ z_1 & \dots & z_i & \dots & z_N\end{bmatrix} \tag{5.5}$$

PCA finds the eigenvectors of the covariance Σ_P of the dataset P, in the order of highest to lowest eigenvalues. This corresponds to the order of highest to lowest variance directions. Each eigenvalue is proportional in size to the variance explained by its corresponding eigenvector direction. The required covariance Σ_P is found in (5.6).

$$\Sigma_P = \frac{1}{N-1}\sum_{i}^{N}\left[\mathbf{p_i} - \mathbf{p_{cog}}\right]\left[\mathbf{p_i} - \mathbf{p_{cog}}\right]^T \tag{5.6}$$

The normal vector $\mathbf{n}$ is then found using (5.7) where eig(X) generates the eigenvectors of a matrix X. The vectors $\mathbf{v_1}$ and $\mathbf{v_2}$ are the first two eigenvectors generated

by this technique. They are chosen to be normalised so that V is an orthonormal basis which can be used later as a rotation matrix.

$$V = \text{eig}(\Sigma_P) = \begin{bmatrix} \mathbf{v_1} & \mathbf{v_2} & \mathbf{n} \end{bmatrix} \qquad \text{where} \qquad \lambda_{v_1} > \lambda_{v_2} > \lambda_n \tag{5.7}$$

After $\mathbf{n}$ has been found it is used to find the d parameter. This is done by substituting $\mathbf{n}$ and any point lying on the plane into (5.2). The centre of gravity point $\mathbf{p_{cog}}$ is one such point as it can be shown to lie on the plane [118]. It can be used to calculate d as shown in (5.8).

$$d = \mathbf{n} \cdot \mathbf{p_{cog}} \tag{5.8}$$

The principal component vectors of V form an orthonormal basis of a new CF aligned to the fitted plane. The inverse of V, which is also its transpose, can be used as a rotation matrix to transform the input points $\mathbf{p_w}$ from the world CF into the plane CF to give points $\mathbf{p_p}$ as in (5.9). The covariance Σ_w of the point in the world CF is also used to generate the corresponding point covariance Σ_p in the plane CF.

$$\begin{aligned} \mathbf{p_p} &= V^T \begin{bmatrix} \mathbf{p_w} - \mathbf{p_{cog}} \end{bmatrix} \\ \Sigma_p &= V^T \Sigma_w V \end{aligned} \tag{5.9}$$

Likewise, V can be used to rotate a point in the plane CF back into the world CF as in (5.10).

$$\begin{aligned} \mathbf{p_w} &= V \mathbf{p_p} + \mathbf{p_{cog}} \\ \Sigma_w &= V \Sigma_p V^T \end{aligned} \tag{5.10}$$

Robustness of Plane Fitting with PCA

The method of using PCA to fit planar surfaces is commonly employed in robotic mapping applications, as mentioned previously. It is however important to recognise that PCA is highly influenced by outliers and this can lead to poor plane fitting performance if not properly addressed. In this thesis, several methods have been proposed to minimise the appearance of outliers, particularly during the 2D line segmentation process detailed in Section 4.3.

Single outlier points were detected during 2D line fitting, as seen in Fig. 4.4. These outliers were removed from the line data set and so would not feed through to the plane fitting process. The end points of each extracted line segment were also ignored as seen in the proposed modified incremental line segmentation approach shown in Algorithm 4.2. These boundary points were occasionally incorrectly segmented due to the nature of the standard incremental line segmentation approach.

As PCA assumes it is fitting to planar data, it was important to also filter out those extracted line segments that were generated by the piecewise linear fitting to a curved surface. A method to achieve this was proposed in Section 4.3.4 to verify the linearity of each line segment.

Incorrect grouping of line segments that appear to belong to the same plane can also lead to problems with plane fitting using PCA. This is why the grouping method proposed in Section 4.4 is performed conservatively. Before attempting to fit a planar surface, the grouped line segments are checked to ensure that they are all coplanar. This was detailed in Section 4.4.5. This grouping can occur when two adjacent and non-coplanar surfaces are difficult to distinguish between and are subsequently grouped together, as if they were from the same surface. In this situation a plane fit would not be attempted, as is the case with other non-planar surfaces. Some other method for mapping these surfaces could be used instead, but this is beyond the scope of this thesis.

Alternative plane fitting methods, such as RANSAC [72], EM [67] and the Hough Transform [110] are generally more robust than PCA to the presence of outliers in

the segmented data. They do however suffer from other weaknesses which make them impractical for use by a small wall climbing robot. RANSAC and EM are both statistical plane fitting approaches and are very computationally expensive. It is often only practical to run them offline which rules out their usage for real time applications. The Hough Transform is predominantly used for line fitting, particularly in computer vision applications. While it can be used for plane fitting too, it suffers from discretisation and the correct choice of grid size can be difficult.

5.4.2 Derivation of Plane Parameter Covariance

The PCA method described above to find the plane parameters of a given point set does not provide any uncertainty information. This must be found separately. A least squares approach can be used to find this covariance for the plane parameters in (5.2).

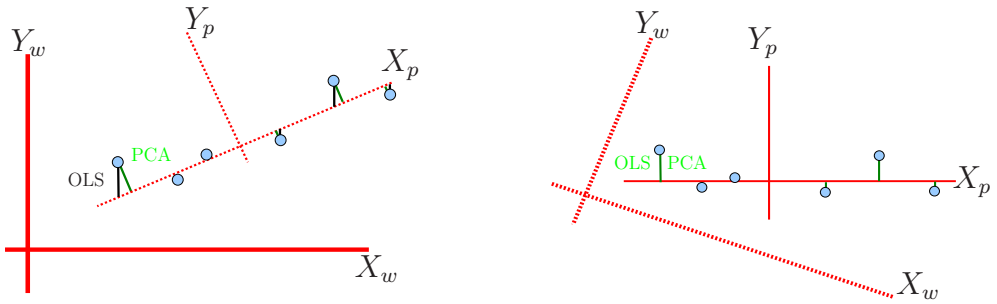
An Ordinary Least Squares (OLS) plane fit can be applied to any data set to provide both the plane parameters and associated covariance. The plane model used is (5.3) which was found by rearranging the general plane equation of (5.1), as previously mentioned. This is restated below in (5.12) and reparameterised in terms of β. It is then possible to derive the covariance of the OLS parameters β_0,β_1,β_2 and then use these to calculate the covariances of $\mathbf{n}$ and d.

The OLS plane is, in general, not orthogonal to the data and so is less optimal than an orthogonal fitting approach, such as PCA. This OLS fit only accounts for fitting error in the Z direction of the 3D point set and so it is not identical to the PCA plane fit found previously. Because of this, the use of OLS will not calculate the desired orthogonal covariance of the PCA fitted plane, in the general case. The desired covariance will only be found when the OLS plane fit aligns exactly with the PCA plane fit.

It is, however, possible to first transform the data points into the PCA fitted

plane CF defined by the PCA orthonormal basis whose origin is at the centre of gravity point. In this CF the OLS plane fit aligns exactly with the PCA fit as required. As the OLS covariance equations can be fit to any data set in any CF, the choice is made to find the OLS covariance with the original 3D points having first been rotated to align with the PCA plane in the plane CF. The desired covariances of $\mathbf{n}$ and d can then be found and transformed back into the original CF.

This idea is demonstrated for the 2D case of a line in Fig. 5.3(a) below. In the original world CF the OLS residuals (black), which operate on the error in the Y direction only (Z direction for the 3D case), are not identical to the Euclidean distance used for the residual by PCA (green) and other orthogonal fitting approaches. This leads to the difference in the fitted planes. After rotating the points into the line CF frame found using the PCA line fit, as shown in Fig. 5.3(b), the two residuals now become identical and so the fitted lines will also be identical.



(a) Data in world CF.

(b) Data transformed into PCA line CF.

Figure 5.3: 2D Example of aligning OLS with the orthogonal PCA fit.

An alternative approach to using a this combination of PCA and OLS could be to use Total Least Squares (TLS) [121]. The TLS approach accounts for error in all residual directions during the fitting process rather than just the Z direction used by OLS. The TLS fit is an orthogonal plane fit and so the coordinate transformation required by the OLS approach above is unnecessary. The TLS plane fit would align with the orthogonal PCA plane fit by definition. The TLS equations are however more complicated to use, especially for the covariance calculations and so this approach is not used here. The approach of using PCA and OLS provides plane

fitting equations which are easier to calculate, particularly with the simplifications introduced later in this chapter.

This PCA and OLS approach to calculating the plane covariance is an extension of that used by Weingarten *et al.* [118], as described later in this section. They outlined this method as a way to find the plane covariance but they did not provide the derivations or proofs that will be presented here to fully demonstrate this approach.

An alternative method for calculating the plane covariance was put forward by Pathak *et al.* [122]. They derived an analytical method, using a newly proposed LRF sensor model *et al.* [123], that calculated the covariance as the generalised inverse of a Hessian matrix containing second partial derivatives. This model had no analytical solution and they acknowledged that incremental or numerical solutions are unsuitable for real time applications, which is especially relevant for a computationally limited system such as the robot used in this work. They then went on to derive an approximate model to allow a solution to be calculated and this did alleviate some, but not all, of the computational burden.

Ordinary Least Squares Plane Fitting

For a point set P consisting of N points $\mathbf{p_w}$ in the world CF, the general plane equation for OLS fitting can be restated as in (5.11). Here $\mathbf{x}, \mathbf{y}, \mathbf{z}$ are column vectors containing the individual x,y,z values of each point in P^T, as in (5.5).

$$A\mathbf{x} + B\mathbf{y} + C\mathbf{z} + D = 0 \tag{5.11}$$

By rearranging as a function for $\mathbf{z}$ and changing the parameters to $\beta_0, \beta_1, \beta_2$, an amenable form is achieved, as in (5.12).

$$\begin{gathered} \mathbf{z} = \beta_0\mathbf{x} + \beta_1\mathbf{y} + \beta_2 \\ \beta_0 = \frac{-A}{C} \quad \beta_1 = \frac{-B}{C} \quad \beta_2 = \frac{-D}{C} \end{gathered} \tag{5.12}$$

With the plane equation in this form, the individual x,y,z coordinates of each point $\mathbf{p_w}$ are substituted. The equation is converted into matrix form consisting of N rows, one for each of the N 3D points. This can be simplified and rearranged for β as in (5.13), where M is a matrix containing N rows of the x,y values.

$$\mathbf{z} = M\beta \tag{5.13}$$

This equation consist of three matrices. Firstly, $\mathbf{z}$ is the column vector consisting of the z values of each point as above. Secondly, M is the Nx3 matrix containing the x and y values. Finally, $\beta = [\beta_0\ \beta_1\ \beta_2]^T$ is the 3x1 vector containing the desired plane parameters. Using these matrices, (5.13) can be expanded as (5.14).

$$\begin{bmatrix} z_1 \\ \vdots \\ z_i \\ \vdots \\ z_N \end{bmatrix} = \begin{bmatrix} x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots \\ x_i & y_i & 1 \\ \vdots & \vdots & \vdots \\ x_N & y_N & 1 \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix} \tag{5.14}$$

To solve for β, (5.13) is rearranged using the pseudo-inverse of M as in (5.15). For the inverse of M^TM to exist, M must be full rank. This requires M to contain at least three points that are linearly independent of each other. This should be the case for any set of three or more points that were generated from at least two different tilt angles.

$$\beta = [M^TM]^{-1}M^T\mathbf{z} \tag{5.15}$$

Equation (5.15) can be used to find the covariance of the parameters β if it is expanded symbolically to allow partial derivatives to be found.

This can be easily calculated numerically to find β to provide an OLS plane fit. However, this is undesirable and unnecessary. This OLS fit is not an orthogonal fit as it only considers errors in the Z and ignores any in the X and Y directions. It

does not therefore align with the PCA plane parameters found previously.

This problem is solved by first transforming the points from the world CF onto the plane CF. In this CF the Z axis aligns with the normal vector of the plane. The errors in the X and Y directions now lie along the surface of the plane and so are not relevant. All the remaining error is now normal to the plane and lies exclusively in the Z direction. The OLS plane now aligns with the PCA plane fit. This is the approach taken by Weingarten *et al.* [118]. The remainder of this section extends their approach by fully deriving, proving and simplifying the necessary equations.

It is also important to note that this approach to calculating the covariance does not fit the plane twice even though two different plane models are used. This was claimed by Pathak *et al.* [122] as their main criticism of this covariance calculation method and this will be shown not to be the case. The Hessian plane model of (5.2) is fitted using PCA and this is the only time that plane fitting is performed. The OLS plane model of (5.12) is used only to derive the covariance equations that follow.

Using the OLS Fit to Calculate the Plane Covariance

The covariance Σ_β of the OLS plane parameters β can be calculated from the covariances of each input point. This is done using (5.16) where Σ_{in} is a matrix with the covariance matrices of each input point along its diagonal, as in (5.17), J_β is the Jacobian of β with respect to each input point's x, y and z coordinates, as in (5.19).

$$\Sigma_\beta = J_\beta \Sigma_{in} J_\beta^T \tag{5.16}$$

where

$$\Sigma_{in} = \begin{bmatrix} \Sigma_1 & & & 0 \\ & \ddots & & \\ & & \Sigma_i & \\ & & & \ddots \\ 0 & & & \Sigma_N \end{bmatrix} \tag{5.17}$$

$$J_{\beta i} = \begin{bmatrix} \frac{\partial \beta_0}{\partial x_i} & \frac{\partial \beta_0}{\partial y_i} & \frac{\partial \beta_0}{\partial z_i} \\ \frac{\partial \beta_1}{\partial x_i} & \frac{\partial \beta_1}{\partial y_i} & \frac{\partial \beta_1}{\partial z_i} \\ \frac{\partial \beta_2}{\partial x_i} & \frac{\partial \beta_2}{\partial y_i} & \frac{\partial \beta_2}{\partial z_i} \end{bmatrix} \tag{5.18}$$

and where

$$J_\beta = \begin{bmatrix} J_{\beta 1} \dots J_{\beta i} \dots J_{\beta N} \end{bmatrix} \tag{5.19}$$

The Jacobian $J_{\beta i}$ of (5.18) can be found by taking the partial derivatives of (5.15). This requires the inverse $[M^T M]^{-1}$ to be found symbolically, rather than numerically, to allow for the partial derivatives to be found. Firstly, $M^T M$ needs to be expanded as shown below in (5.20).

$$
\begin{aligned}
M^T M &= \begin{bmatrix} \mathbf{x^T} \\ \mathbf{y^T} \\ \mathbf{1^T} \end{bmatrix} \begin{bmatrix} \mathbf{x} & \mathbf{y} & \mathbf{1} \end{bmatrix} \\
&= \begin{bmatrix} x_1 & \dots & x_i & \dots & x_N \\ y_1 & \dots & y_i & \dots & y_N \\ 1 & \dots & 1 & \dots & 1 \end{bmatrix} \begin{bmatrix} x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots \\ x_i & y_i & 1 \\ \vdots & \vdots & \vdots \\ x_N & y_N & 1 \end{bmatrix} \\
&= \begin{bmatrix} \sum_i x_i^2 & \sum_i x_i y_i & \sum_i x_i \\ \sum_i x_i y_i & \sum_i y_i^2 & \sum_i y_i \\ \sum_i x_i & \sum_i y_i & N \end{bmatrix}
\end{aligned} \tag{5.20}
$$

The symbolic inverse of (5.20) is found using the standard formula for a 3x3 matrix inverse.

$$
[M^T M]^{-1} = \frac{F}{|M^T M|} \tag{5.21}
$$

where

$$
F = \begin{bmatrix} N\sum_i y_i^2 - (\sum_i y_i)^2 & \sum_i x_i \sum_i y_i - N\sum_i x_i y_i & \sum_i x_i y_i \sum_i y_i - \sum_i x_i \sum_i y_i^2 \\ \sum_i x_i \sum_i y_i - N\sum_i x_i y_i & N\sum_i x_i^2 - (\sum_i x_i)^2 & \sum_i x_i y_i \sum_i x_i - \sum_i x_i^2 \sum_i y_i \\ \sum_i x_i y_i \sum_i y_i - \sum_i x_i \sum_i y_i^2 & \sum_i x_i y_i \sum_i x_i - \sum_i x_i^2 \sum_i y_i & \sum_i x_i^2 \sum_i y_i^2 - (\sum_i x_i y_i)^2 \end{bmatrix} \tag{5.22}
$$

and where

$$
\begin{aligned}
|M^T M| =& N \sum_{i=1}^{N} x_i^2 \sum_{i=1}^{N} y_i^2 + 2 \sum_{i=1}^{N} x_i \sum_{i=1}^{N} y_i \sum_{i=1}^{N} x_i y_i \\
& - \sum_{i=1}^{N} y_i^2 \left(\sum_{i=1}^{N} x_i \right)^2 - \sum_{i=1}^{N} x_i^2 \left(\sum_{i=1}^{N} y_i \right)^2 - N \left(\sum_{i=1}^{N} x_i y_i \right)^2
\end{aligned} \tag{5.23}
$$

The remaining term of (5.15) is expanded symbolically in (5.24).

$$
\begin{aligned}
M^T\mathbf{z} &= \begin{bmatrix} \mathbf{x^T} \\ \mathbf{y^T} \\ \mathbf{1^T} \end{bmatrix} \mathbf{z} \\
&= \begin{bmatrix} x_1 & \dots & x_i & \dots & x_N \\ y_1 & \dots & y_i & \dots & y_N \\ 1 & \dots & 1 & \dots & 1 \end{bmatrix} \begin{bmatrix} z_1 \\ \vdots \\ z_i \\ \vdots \\ z_N \end{bmatrix} \\
&= \begin{bmatrix} \sum_i x_i z_i \\ \sum_i y_i z_i \\ \sum_i z_i \end{bmatrix}
\end{aligned} \tag{5.24}
$$

It is now possible to write equations explicitly for $\beta_0, \beta_1, \beta_2$ in terms of the input points x_i, y_i, z_i. Each row vector of (5.21) is multiplied with (5.24) to give (5.25 - 5.27). The notation F_{ab} refers to the entry at row a, column b of (5.22).

$$\beta_0 = \frac{1}{|M^T M|} \begin{bmatrix} F_{11} \sum_i x_i z_i & F_{12} \sum_i y_i z_i & F_{13} \sum_i z_i \end{bmatrix} \tag{5.25}$$

$$\beta_1 = \frac{1}{|M^T M|} \begin{bmatrix} F_{21} \sum_i x_i z_i & F_{22} \sum_i y_i z_i & F_{23} \sum_i z_i \end{bmatrix} \tag{5.26}$$

$$\beta_2 = \frac{1}{|M^T M|} \begin{bmatrix} F_{31} \sum_i x_i z_i & F_{32} \sum_i y_i z_i & F_{33} \sum_i z_i \end{bmatrix} \tag{5.27}$$

The required partial derivatives for the Jacobian J_{β_i} in (5.18) can be found using the quotient rule for differentiation shown in (5.28). The numerator term, denoted by f_i, consists of the 1×3 matrix from the RHS of (5.25-5.27).

$$\frac{\delta\beta_i}{\delta x} = \frac{\frac{\delta f_i}{\delta x}|M^T M| - \frac{\delta |M^T M|}{\delta x} f_i}{(|M^T M|)^2} \qquad \text{where} \qquad \beta_i = \frac{f_i}{|M^T M|} \qquad x = x_i,\ y_i,\ z_i \quad (5.28)$$

These partial derivatives are long and complicated if taken in the original world CF. As mentioned above, the OLS fit used above also does not align with the orthogonal plane fit found using PCA in (5.7). These two issues can be solved by transforming the 3D points into the CF defined by the orthonormal basis supplied by the PCA plane fitting. This idea was proposed by Weingarten *et al.* [118] but their work did not provide the full derivations here. The simplified covariance equations and proofs presented below are new contributions over and above that of Weingarten.

It will be shown that the partial derivatives can be significantly simplified using lemmas (5.4.1) and (5.4.2) below. The covariance can be found in the plane CF and the inverse transformation can then be applied to get the plane covariance in the world CF, as required.

Covariance Calculation in Plane CF

The input data points are transformed into the plane CF defined by the axes and origin found using PCA in (5.9). In this new CF, the fitted plane lies along the X, Y axes as expected, because it is aligned with the CF axes and passes through the origin by definition. Therefore the plane parameters are exactly $\mathbf{n} = [0\ 0\ 1]^T$ and $d = 0$. This will be shown to be true later. These parameters are not of interest as PCA has already provided them in the original world CF. It is possible, however, to calculate the covariance matrix Σ_β as outlined above and use this to determine the covariance matrix Σ_n and variance σ_d.

The following two lemmas allow the partial derivatives for equations (5.25-5.27) to be significantly simplified, as in the plane CF many of the sum terms equate exactly to zero. They can also be used to show that the parameters β from (5.12) also all equal 0.

Lemma 5.4.1. *In the plane CF defined by the orthonormal basis vectors and origin supplied from a PCA planar fit, the vector sum of the data points is exactly equal to 0. For clarity the coordinate frame subscripts will be noted as superscripts with* $\mathbf{p^p}$ *being in the plane CF and* $\mathbf{p^w}$ *being in the world CF.*

$$\sum_{i=1}^{N} \mathbf{p_i^p} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \qquad \textit{which leads to} \qquad \sum_{i=1}^{N} x_i^p = \sum_{i=1}^{N} y_i^p = \sum_{i=1}^{N} z_i^p = 0$$

Proof. Equation (5.9) outlines the transformation to obtain a point $\mathbf{p^p}$ in plane coordinates as below.

$$\mathbf{p^p} = V^T[\mathbf{p^w} - \mathbf{p_{cog}}]$$

By taking the sum of all N points and substituting in (5.4) for $\mathbf{p_{cog}}$, the desired result is achieved.

$$\begin{aligned}
\sum_{i=1}^{N} \mathbf{p_i^p} &= \sum_{i=1}^{N} \left(V^T[\mathbf{p_i^w} - \mathbf{p_{cog}}]\right) \\
&= V^T \left(\sum_{i=1}^{N} \mathbf{p_i^w} - \sum_{i=1}^{N} \mathbf{p_{cog}} \right) \\
&= V^T \left(\sum_{i=1}^{N} \mathbf{p_i^w} - N\mathbf{p_{cog}} \right) \\
&= V^T \left(\sum_{i=1}^{N} \mathbf{p_i^w} - N\frac{1}{N} \sum_{i=1}^{N} \mathbf{p_i^w} \right) \\
&= \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}
\end{aligned}$$

□

Lemma 5.4.2. *In the plane CF defined by the orthonormal basis vectors and origin supplied from a PCA planar fit, the vector sum of product data points are exactly equal to 0. For clarity the coordinate frame subscripts will be noted as superscripts as above.*

$$\sum_{i=1}^{N} x_i^p y_i^p = \sum_{i=1}^{N} x_i^p z_i^p = \sum_{i=1}^{N} y_i^p z_i^p = 0$$

Proof. Equation (5.9) outlines the transformation to obtain a point $\mathbf{p^p}$ in the plane CF as below.

$$\mathbf{p^p} = V^T[\mathbf{p^w} - \mathbf{p^w_{cog}}]$$

The point set P^p in the plane CF, as in (5.5), can then be written as below using P_0^w which is the mean centred point set in the world CF.

$$P^p = V^T P_0^w \qquad \text{where} \qquad P_0^w = P^w - P_{cog}$$

P^p can be multiplied by its transpose to get the matrix $P^p(P^p)^T$. This contains the desired plane CF sum of product terms which are located outside the main diagonal. All that remains is to then show that $P^p(P^p)^T$ is a diagonal matrix and therefore the off diagonal terms must by definition be exactly 0.

$$P^p(P^p)^T = \begin{bmatrix} \sum_i (x_i^p)^2 & \sum_i x_i^p y_i^p & \sum_i x_i^p z_i^p \\ \sum_i x_i^p y_i^p & \sum_i (y_i^p)^2 & \sum_i y_i^p z_i^p \\ \sum_i x_i^p z_i^p & \sum_i y_i^p z_i^p & \sum_i (z_i^p)^2 \end{bmatrix}$$

By substituting back in the original equation for P^p it can be shown that the matrix $P^p(P^p)^T$ is the product of a square matrix $P_0^w(P_0^w)^T$ and V matrices. A square matrix is diagonalised by its eigenvector matrix in exactly this form so it now remains to show that V is the eigenvector matrix of P_0^w.

$$\begin{aligned} P^p(P^p)^T &= V^T P_0^w (V^T P_0^w)^T \\ &= V^T P_0^w (P_0^w)^T V \end{aligned}$$

From (5.7) it is known that V consists of the eigenvectors of the covariance Σ_P^w of P^w from (5.6). This can be rewritten using P_0^w.

$$\begin{aligned} \Sigma_P^w &= \frac{1}{N-1} \sum_{i=1}^{N} \left[\mathbf{p_i^w} - \mathbf{p_{cog}^w} \right] \left[\mathbf{p_i^w} - \mathbf{p_{cog}^w} \right]^T \\ &= \frac{1}{N-1} P_0^w (P_0^w)^T \end{aligned}$$

The $\frac{1}{N-1}$ term has no influence on the eigenvectors of Σ_P^w, as they are normalised, and so it can ignored. The only influence this constant term has is on the size of the eigenvalues. This is not an issue as only their relative size is important in ordering the eigenvectors.

$$\begin{aligned} V &= \text{eig}(\Sigma_P^w) \\ &= \text{eig}(\frac{1}{N-1} P_0^w (P_0^w)^T) \\ &= \text{eig}(P_0^w (P_0^w)^T) \end{aligned}$$

It can therefore be seen that the square matrix $P_0^w (P_0^w)^T$ is diagonalised by its eigenvector matrix V and so $P^p(P^p)^T$ is therefore a diagonal matrix. This necessitates that the off diagonal sum of product terms are exactly equal to zero as required. □

The two above lemmas can now be used to simplify the partial derivative equations significantly and they allow the Jacobian terms of (5.18) to be easily calculated. It can also now be seen from (5.24) that $M^T\mathbf{z}$ is equal to 0 and therefore using (5.25-5.27) it can be seen that $\beta_0 = \beta_1 = \beta_2 = 0$. The partial derivative equation from (5.28) can now be simplified, as the numerator f_i for β_i is equal to zero.

$$\frac{\partial \beta}{\partial x} = \frac{\frac{\partial f}{\partial x}}{|M^T M|} \tag{5.29}$$

The derivation for $\frac{\partial \beta_0}{\partial x_i}$ is shown below. The remaining partial derivatives are found similarly.

$$\frac{\partial \beta_0}{\partial x_i} = \frac{1}{|M^T M|}\frac{\partial f_0}{\partial x_i} \tag{5.30}$$

First the partial derivative $\frac{\partial f_0}{\partial x_i}$ is found.

$$\begin{aligned} f_0 = &\left(N\sum_{i=0}^{N} y_i^2 - \left(\sum_{i=0}^{N} y_i\right)^2\right)\sum_{i=0}^{N} x_i z_i + \left(\sum_{i=0}^{N} x_i \sum_{i=0}^{N} y_i - N\sum_{i=0}^{N} x_i y_i\right)\sum_{i=0}^{N} y_i z_i \\ &+ \left(\sum_{i=0}^{N} x_i y_i \sum_{i=0}^{N} y_i - \sum_{i=0}^{N} x_i \sum_{i=0}^{N} y_i^2\right)\sum_{i=0}^{N} z_i \end{aligned} \tag{5.31}$$

$$\begin{aligned} \frac{\partial f_0}{\partial x_i} = &\left(N\sum_{i=0}^{N} y_i^2 - \left(\sum_{i=0}^{N} y_i\right)^2\right) z_i + \left(\sum_{i=0}^{N} y_i - N y_i\right)\sum_{i=0}^{N} y_i z_i \\ &+ \left(y_i \sum_{i=0}^{N} y_i - \sum_{i=0}^{N} y_i^2\right)\sum_{i} z_i \end{aligned} \tag{5.32}$$

This partial derivative can be greatly simplified by using lemmas 5.4.1 and 5.4.2 which allow most of the sum terms to be set to zero. This leaves the only remaining non-zero terms as in 5.33.

$$\frac{\partial f_0}{\partial x_i} = \left(N \sum_{i=1}^{N} y_i^2 \right) z_i \tag{5.33}$$

Next the denominator $|M^T M|$ from (5.23) is found and also simplified in the same manner to leave only non-zero terms.

$$|M^T M| = N \sum_{i=1}^{N} x_i^2 \sum_{i=1}^{N} y_i^2 \tag{5.34}$$

Substituting (5.33) and (5.34) into (5.30) gives the desired partial derivative in the plane CF, as seen in (5.35) below. This is used in the Jacobian J_{β_i} of (5.18).

$$\begin{aligned} \frac{\partial \beta_0}{\partial x_i} &= \frac{(N \sum_i y_i^2)\, z_i}{N \sum_i x_i^2 \sum_i y_i^2} \\ &= \frac{z_i}{\sum_i x^2} \end{aligned} \tag{5.35}$$

The other eight partial derivatives can be derived in a similar manner and substituted into (5.18) to give J_{β_i} as shown below in (5.36). Four of these partial derivatives equate exactly to zero as they do not contain any non-zero terms.

$$J_{\beta i} = \begin{bmatrix} \frac{\partial \beta_0}{\partial x_i} & \frac{\partial \beta_0}{\partial y_i} & \frac{\partial \beta_0}{\partial z_i} \\ \frac{\partial \beta_1}{\partial x_i} & \frac{\partial \beta_1}{\partial y_i} & \frac{\partial \beta_1}{\partial z_i} \\ \frac{\partial \beta_2}{\partial x_i} & \frac{\partial \beta_2}{\partial y_i} & \frac{\partial \beta_2}{\partial z_i} \end{bmatrix} = \begin{bmatrix} \frac{z_i}{\sum_i x_i^2} & 0 & \frac{x_i}{\sum_i x_i^2} \\ 0 & \frac{z_i}{\sum_i y_i^2} & \frac{y_i}{\sum_i y_i^2} \\ 0 & 0 & \frac{1}{N} \end{bmatrix} \tag{5.36}$$

This finally allows the covariance matrix Σ_β to be found in the plane CF using (5.16). This equation can be rewritten as an iterative sum as in (5.37) below. Note that Σ_{in} is the block diagonal input covariance matrix which contains all the input point covariances.

$$\Sigma_\beta = J_\beta \Sigma_{in} J_\beta^T = \sum_{i=1}^{N} J_{\beta_i} \Sigma_i J_{\beta_i}^T \tag{5.37}$$

Covariance Transformation to the World CF

It now remains to use Σ_β to find the covariance Σ_n and variance σ_d in the plane CF. These can then be transformed back into the world CF to find the desired plane parameter uncertainty. This is done by using the partial derivatives of the transformation from β to $\mathbf{n}$ and d to transform the uncertainty.

Using (5.12) with $C = 1$ allows the general plane parameters of (5.11) to be found in terms of β.

$$A = -\beta_0 \qquad B = -\beta_1 \qquad C = 1 \qquad D = -\beta_2 \tag{5.38}$$

Equation (5.11) can be rearranged to mirror (5.2) and it can be easily seen that A, B, C is related to $\mathbf{n}$ and D is related to d.

$$\begin{aligned} A\mathbf{x} + B\mathbf{y} + C\mathbf{z} + D &= 0 \\ \begin{bmatrix} A \\ B \\ C \end{bmatrix} \cdot \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \\ \mathbf{z} \end{bmatrix} + D &= 0 \end{aligned} \tag{5.39}$$

Equation (5.39) can be normalised so that the normal vector $[A \quad B \quad C]^T$ has a magnitude of 1. This leads to equations for $\mathbf{n}$ and d in terms of β.

$$\begin{aligned} \mathbf{n} = \begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix} &= \frac{1}{\sqrt{A^2 + B^2 + C^2}} \begin{bmatrix} A \\ B \\ C \end{bmatrix} \\ &= \frac{1}{\sqrt{\beta_0^2 + \beta_1^2 + 1}} \begin{bmatrix} -\beta_0 \\ -\beta_1 \\ 1 \end{bmatrix} \end{aligned} \tag{5.40}$$

$$\begin{aligned} d &= \frac{-D}{\sqrt{A^2 + B^2 + C^2}} \\ &= \frac{\beta_2}{\sqrt{\beta_0^2 + \beta_1^2 + 1}} \end{aligned} \tag{5.41}$$

It can now be seen by substituting $\beta_0 = \beta_1 = \beta_2 = 0$ into (5.40) and (5.41) that in the plane CF, $\mathbf{n} = [0\ 0\ 1]^T$ and $d = 0$, as previously stated.

To calculate the covariance matrix Σ_n of the normal vector $\mathbf{n}$, the Jacobian J_n is used, as shown in (5.45). The component partial derivatives are easily calculated. The derivation of $\frac{\partial n_x}{\partial \beta_0}$ is shown below.

$$n_x = \frac{\beta_0}{\sqrt{\beta_0^2 + \beta_1^2 + 1}} = \frac{f}{g} \tag{5.42}$$

The quotient rule is again used to find the partial derivatives as shown below in (5.43). This is simplified by using $\beta_0 = \beta_1 = 0$ to get $f = 0$ and $g = 1$.

$$\begin{aligned} \frac{\partial n_x}{\partial \beta_0} &= \frac{\frac{\partial f}{\partial \beta_0}}{g} \\ &= \frac{1}{\sqrt{1}} \\ &= 1 \end{aligned} \tag{5.43}$$

The remaining partial derivatives are found similarly to get the jacobian J_n. This allows Σ_n to be found in the plane CF, as in (5.45) below.

$$J_n = \begin{bmatrix} \frac{\partial n_x}{\partial \beta_0} & \frac{\partial n_x}{\partial \beta_1} & \frac{\partial n_x}{\partial \beta_2} \\ \frac{\partial n_y}{\partial \beta_0} & \frac{\partial n_y}{\partial \beta_1} & \frac{\partial n_y}{\partial \beta_2} \\ \frac{\partial n_z}{\partial \beta_0} & \frac{\partial n_z}{\partial \beta_1} & \frac{\partial n_z}{\partial \beta_2} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{5.44}$$

$$\Sigma_n = J_n \Sigma_\beta J_n^T = \begin{bmatrix} \sigma_{\beta_{00}}^2 & \sigma_{\beta_{01}}^2 & 0 \\ \sigma_{\beta_{01}}^2 & \sigma_{\beta_{11}}^2 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{5.45}$$

Likewise the variance σ_d^2 of d is found in the place CF, as in (5.47).

$$J_d = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \tag{5.46}$$

$$\sigma_d^2 = J_d \Sigma_\beta J_d^T = \sigma_{\beta_{22}}^2 \tag{5.47}$$

These give the desired uncertainty information for the plane parameters. All that remains is to transform them back into the original world CF, as shown below in (5.48) and (5.49). The distance parameter d is the same in both CFs because the coordinate transformation is orthonormal and therefore the variance σ_d^2 remains unchanged.

$$(\Sigma_n)_w = V(\Sigma_n)_p V^T \tag{5.48}$$

$$(\sigma_d^2)_w = (\sigma_d^2)_p \tag{5.49}$$

5.5 Generation of Planar Polygon Boundaries

The previous section detailed the method for fitting an infinite plane to the acquired range data. Unlike other surface mapping approaches that operate on dense point clouds, sparse range scanning does not supply enough information to determine the polygon boundaries of each plane. Instead, the method proposed here uses the extracted image features to estimate the boundary information. The image corners act as the primary basis for the boundary of the plane and the image lines provide information as to the connectedness of those corners.

The simplest plane fit involves two laser lines and one camera image. In this case, the polygon is bounded by the extracted corners and, by adding a second or third image, it is possible to improve the accuracy of the boundary. It can also be expanded if the camera FoV is limited or occluded. These new corners need to be matched and fused incrementally with the boundary corner estimates.

An example of this process is shown below in Fig. 5.4. The grouped features from three consecutive images for the black paper surface are shown in Fig. 5.4(a). The projected boundaries in the plane CF of each of the three images can be seen in Fig. 5.4(b). The corner location uncertainty is demonstrated using ellipses set at one standard deviation in the X and Y directions. The merged polygon boundary is shown in Fig. 5.4(c).

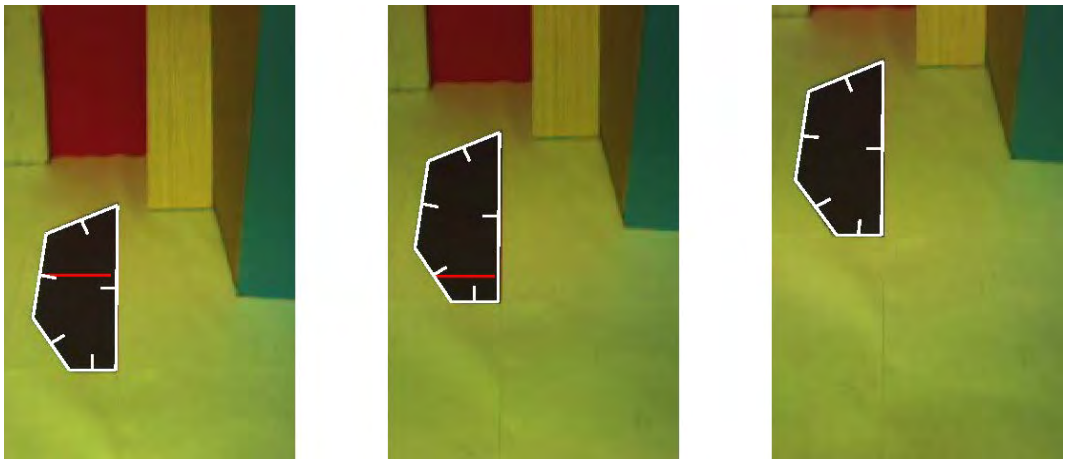
Prior to the matching and fusion of the image corners, it is first necessary to transform the pixel location $\mathbf{p_i}$=[u v 1] from the camera into the world CF. They can then be projected onto the plane to get their 2D corner location in the plane CF. The equations to perform this transformation were derived in Section 4.2.3.

5.5.1 Corner and Line Registration

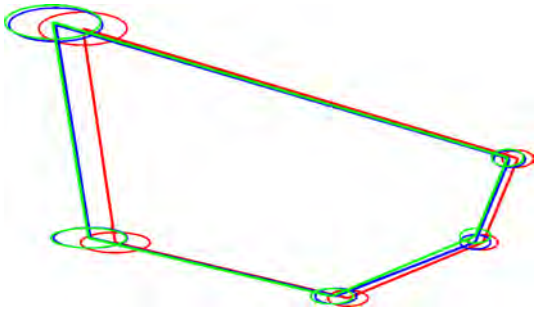
The image feature boundary information from the first scan is set as the initial polygon boundary. Each new scan can be overlaid on top of the current boundary with full uncertainty information. This new scan can then be incrementally registered and merged into the current polygon boundary estimate. A similar problem was tackled by Neira *et al.* [124] for data association of features for 2D SLAM.

In this work, each new line and corner is checked for a match to an existing boundary feature. If a correspondence is found, they are marked as being a match to await merging. This is done by finding the closest corner pair and assigning them as a match. Then each line from that corner is followed in turn and the subsequent corners are also checked for a match. This allows the connectivity of the image lines to be preserved whilst using the corners to anchor those lines to the plane.

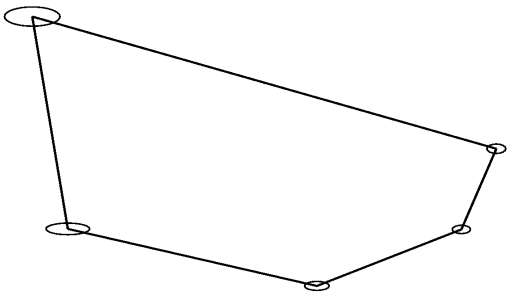
Once these chains are exhausted, the remaining unmatched corners are checked,



(a) Grouped features for black paper surface.



(b) Individual planar polygon boundaries projected onto infinite plane.



(c) Merged polygon boundary.

Figure 5.4: Example of polygon boundary merging with uncertainty ellipses.

the next closest corner match is used and then the process repeats until no matched corners remain. Any corners that are still unmatched at this point are considered for addition to the boundary, if they extend it sufficiently.

Image Line and Corner Types

The registration process requires different approaches to handle the different types of image corners or lines being matched. As mentioned in Section 4.2.1, the outer edges of the camera image introduce virtual image corners and lines that do not correspond to physical surface corners or edges. These are useful as they allow the image boundaries to be closed when the FoV is limited. They are preserved until subsequent images expand the surface boundary and eventually the true image edge can be seen. The matching of these virtual features is more complicated than the standard features and each matching case will be discussed below.

The following definitions are used in the remainder of this work. An example of each can also be seen in Fig. 5.5.

Corner: An image corner corresponding to a physical corner.

Line: An image line corresponding to a physical surface edge.

Virtual Corner: An image corner introduced by an image line intersecting the outer image edge. The four outer corners of the image are also virtual corners, but these correspond to virtual lines instead.

Virtual Line: An image line introduced by the outer image edge that connects two virtual corners.

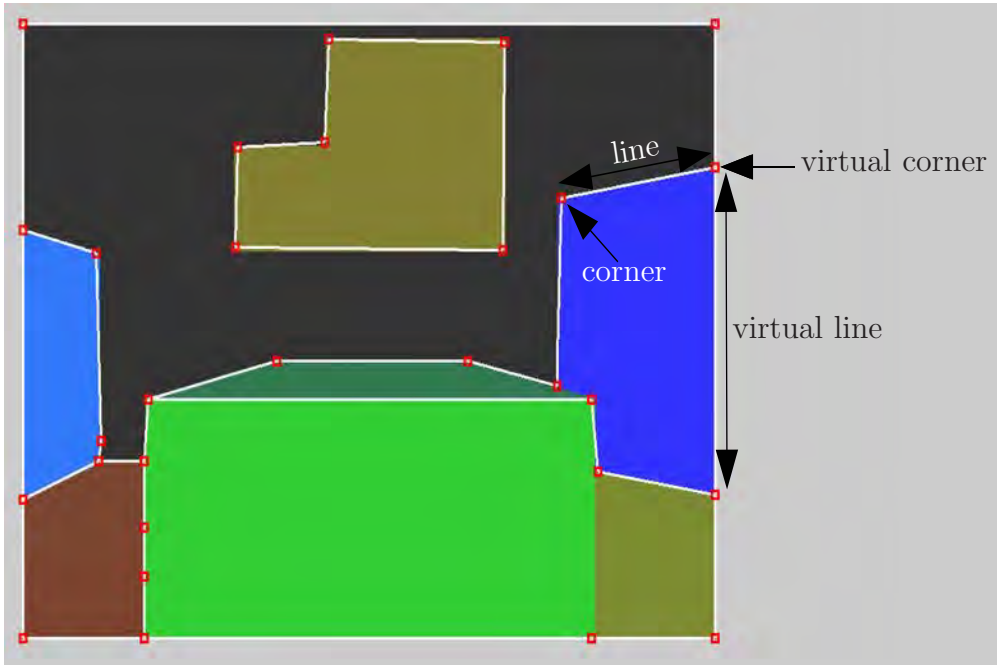


Figure 5.5: Examples of real and virtual image features.

5.5.2 Registration Combinations

During the registration process each combination of feature types, virtual or real, must be handled differently. The virtual features represent different feature information with respect to the underlying physical surface.

Firstly, corners should be matched with other corners until all possible matches are corresponded. The next two stages involve matching the lines of remaining unmatched corners and virtual corners to lines present in the other scan. This is a two way process as the new scan corners should be matched to boundary lines and vice versa. The concept is the same for both cases of line matching. However, they each generate different points to be merged with. This is explained in more detail below. The final matching step is to use the remaining unmatched features to extend the boundary, if they lie outside it.

Matching a Corner to a Corner

The matching of corners from the new scan to those already existing in the boundary is performed by finding the Mahalanobis distance between each potential pair using (5.50). This is a standard measure of the distance between two vectors. It is essentially a 2D cartesian distance with each direction normalised by the combined point covariance.

$$d_M = \sqrt{[\mathbf{p_1} - \mathbf{p_2}]^T [\Sigma_1 + \Sigma_2]^{-1} [\mathbf{p_1} - \mathbf{p_2}]} \tag{5.50}$$

If this distance d_M is less than a threshold value of 2.5 standard deviations, then the two points are matched with each other. This value was chosen as it corresponds to a 99% probability that the two points coincide and thus are in fact the same point. At this stage an additional check is also made to verify that these two matched corners have similar line connectivity to other matched corners between the boundary and the new scan. This ensures that the image line connectivity of the polygon boundary is preserved.

Matching a Corner to a Line

If a corner is not close to any existing corner, then it may be possible to match it to an existing line. This occurs most often in the situation where an image edge is over segmented in one scan and so a corner is generated in the middle of an image

line. This can be seen in Fig 5.6(a) below.

This matching process requires lines to be matched together rather than the corners themselves. The candidate lines are found by first taking the lines terminating at the corner in question and checking whether any of the corners at the opposite end of these lines have already been matched. If such a corner is found, the angle and distance between the two lines are checked to see if they are similar. When a match is found, the corner is projected onto the line using (5.51). This splits the image line at $\mathbf{p_0}$ and generates a new corner to merge with. This is shown in Fig. 5.6(b).

(a) Over segmented image line (blue) matched to an image line (red).

(b) Projected corner position $\mathbf{p_0}$ for merging with $\mathbf{p_3}$.

Figure 5.6: Matching of a corner to a line.

$$\mathbf{p_0} = \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} = \mathbf{p_1} + u(\mathbf{p_2} - \mathbf{p_1}) \qquad \text{where} \qquad u = \frac{(\mathbf{p_3} - \mathbf{p_1}) \cdot (\mathbf{p_2} - \mathbf{p_1})}{|\mathbf{p_2} - \mathbf{p_1}|^2} \tag{5.51}$$

This can be expanded out for the components of $\mathbf{p_0}$ as in (5.52) and (5.53). These will be used to find the partial derivatives, as required for (5.54) below. This will allow the uncertainty of this new point to be found.

$$x_0 = x_1 + \frac{(x_3 - x_1)(x_2 - x_1)^2 + (y_3 - y_1)(y_2 - y_1)(x_2 - x_1)}{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{5.52}$$

$$y_0 = y_1 + \frac{(x_3 - x_1)(x_2 - x_1)(y_2 - y_1) + (y_3 - y_1)(y_2 - y_1)^2}{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{5.53}$$

The covariance Σ_{p_0} of this new point $\mathbf{p_0}$ can be calculated from the individual covariances of the three input points using the Jacobian of $\mathbf{p_0}$, as shown in (5.54).

$$\Sigma_{p_0} = \sum_{i=1}^{3} J_{p_i} \Sigma_{p_i} J_{p_i}^T \qquad \text{where} \qquad J_{p_i} = \begin{bmatrix} \frac{\partial x_0}{\partial x_i} & \frac{\partial x_0}{\partial y_i} \\ \frac{\partial y_0}{\partial x_i} & \frac{\partial y_0}{\partial y_i} \end{bmatrix} \tag{5.54}$$

A final check should be done to ensure that the point lies within the line segment it is being projected onto. This is the case if u from (5.51) lies between 0 and 1. If this is true, then the corner is added into the boundary and it is assigned as a match to the original corner $\mathbf{p_3}$.

Matching a Virtual Corner to a Line

Virtual corners are matched to lines in a very similar manner to that of corners to lines, as above. The major difference is that virtual corners do not correspond to actual physical corners, as they were instead generated by the outer boundary of the image. The line they terminate likely continues some distance outside the camera FoV and so it makes sense to merge with the corner there, rather than looking for one near the location of the virtual corner. It stands to reason that if the virtual corner was generated by the same physical line that it was matched with, then the length of its line should also be similar.

After matching a virtual corner to a line, instead of projecting the corner onto the line as above, the virtual corner's line should be extended to match the length of the line it was matched to. This is seen in Fig. 5.7(a). This new corner, generated at the endpoint of the extended line, can then be merged with the matched corner.

Equations (5.51) and (5.54) can also be used to find this point as above. However, the points involved are slightly different, as shown in Fig. 5.7(b) below. The other difference requires that the value of u should lie outside the range 0 to 1 to indicate that the projected point does not lie on the line segment.

(a) Virtual corner on image line (blue) matched to image line (red).

(b) Projected corner position $\mathbf{p_0}$ for merging with $\mathbf{p_0}$.

Figure 5.7: Matching of a virtual corner to a line.

Unmatched Corners and Lines

Any remaining corners and lines in the new scan that have not yet been matched can be used to extend the current polygon boundary. This is a similar process to taking the union of the current boundary and the new scan boundary after first merging the matched feature pairs. These missing features fall into two categories.

Firstly, some difficult surface edges may not have been extracted or grouped in previous scans and so they should be added into the boundary. This is done by finding, if possible, a matched corner in the boundary and then adding the missing lines and corners that extend from that point.

Secondly, virtual lines should be added to the boundary if they occur outside the image boundary and if they extend other virtual lines. Virtual lines should not be merged together as they do not correspond to physical surface edges. The outer most virtual line should be kept and any interior virtual lines should be removed from the boundary as they become redundant. This removal can be done here or as a post processing step, as detailed in Section 5.5.4.

5.5.3 Merging of Registered Corners

After the matching process has been completed for the new scan, all matched corner pairs are merged together to produce an updated boundary estimate. This merging process is performed incrementally with the addition of each new scan to simplify the corner registration process. The merging itself, however, could be undertaken

as a batch operation at any stage if a different registration process was used.

The merging is based on the standard Kalman Filter (KF) equations to provide an optimal linear estimate. A motion model is not needed as the planar surface can be considered stationary. The state variable is the 2D point $\mathbf{p^b}$, with covariance matrix Σ^b, which is the corner location in the plane CF of the current boundary. The Z coordinate is not used because in the plane CF the corner lies on the plane and so it is equal to zero. The matched corner location from the new scan is denoted $\mathbf{p^n}$, with a 2×2 covariance matrix Σ^n.

The subscript denotes the index of the new scan being integrated into the polygon boundary with $\mathbf{p^b_{k-1}}$ being the boundary prior to merging. The polygon boundary is denoted as $\mathbf{p^b_k}$ after it has been merged with the new scan. The superscripts n and b denote a point from the new scan and boundary respectively.

The state and covariance update steps both require the calculation of the Kalman Gain K_k as in (5.55). This is a measure of the relative uncertainties of the new corner measurement $\mathbf{p^n_k}$ and the current corner boundary location $\mathbf{p^b_{k-1}}$.

$$K_k = \Sigma^b_{k-1}[\Sigma^b_{k-1} + \Sigma^n_k]^{-1} \tag{5.55}$$

The state update step is shown in (5.56). This generates the new corner location $\mathbf{p^b_k}$ as a weighted sum of the old corner location $\mathbf{p^b_{k-1}}$ and the new corner measurement $\mathbf{p^n_k}$. The weight used is the Kalman Gain K_k from (5.55).

$$\mathbf{p^b_k} = \mathbf{p^b_{k-1}} + K_k[\mathbf{p^n_k} - \mathbf{p^b_{k-1}}] \tag{5.56}$$

Similarly, the covariance update step of (5.57) is used to generate the new corner location covariance Σ^b_k where I is a 2×2 Identity matrix.

$$\Sigma^b_k = [I - K_k]\Sigma^b_{k-1} \tag{5.57}$$

5.5.4 Post Processing of Polygon Boundaries

Due to the unique nature of this surface mapping approach, it was necessary to develop the polygon boundary merging methods detailed above. The boundaries that are produced are, however, still short of the quality required for robotic use. After the registration and merging steps are complete, a post processing step is required to improve the quality of the polygon boundaries. Several methods are proposed below to solve common problems in the boundaries that are generated.

Erroneous boundary lines are removed if they made it through the registration process or were caused by poor feature grouping. It is also possible to close small gaps in the polygon boundary using surrounding information. These both help to ensure that a quality closed polygon boundary is generated. Finally, any collinear adjacent boundary segments are merged to simplify the boundary representation.

Removal of Inconsistent Edges

The registration process sometimes produces two types of erroneous boundary lines that should be removed to improve the quality of the polygon boundary.

The first type of edge that should be removed are those that lie inside the polygon. It is important to differentiate these bad edges from legitimate edges that lie internal to the polygon because non-convex hulls are valid. The erroneous internal lines generally connect to a corner lying on the polygon boundary and so are easily identified by looking for corners that connect more than two boundary lines. The image lines that were extracted and grouped contain information about the inside of the line and this allows the inside of the boundary to be distinguished from the outside. Such internal lines are most often caused by virtual lines that, by definition, are internal to a surface and should be removed once they have been extended and become redundant. This removal can be performed during boundary merging or as a post processing step.

The second type of erroneous edge are those that lie outside the polygon boundary. They are often caused by a false grouping during the feature grouping stage of

Section 4.4. These are identified in a similar manner to the false internal boundary lines. As the inside of each image line is known, the outside of the polygon boundary can be determined. Any image line that lies on the outside of the boundary, especially if it connects to a corner that already has two other lines on the boundary, is likely a false line and should be removed.

An example of a surface containing such edges that require removal is shown below in Fig. 5.8. The resulting polygon boundary, shown in Fig. 5.8(d), was generated from the image lines of three images. It has two lines that should be removed. This first is an internal line caused by a virtual line from the bottom edge of Fig. 5.8(a). The second line to be removed is a falsely grouped line from Fig. 5.8(b). The third image has all four image lines correctly extracted and grouped as seen in Fig. 5.8(c).

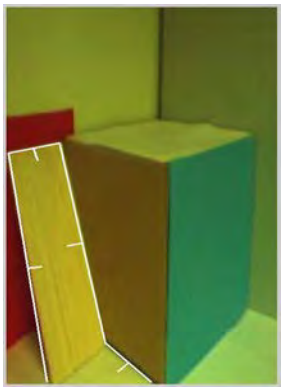
Closing of Small Gaps in the Boundary

Another useful post processing step is to join small gaps in the boundary to provide a closed polygon. Often these gaps are small and easily joined, as is the case where a small segment of an image edge was missed during extraction or grouping. An example of such small gaps being filled is shown in Fig. 5.9(a) by the red lines, which are also circled for clarity.

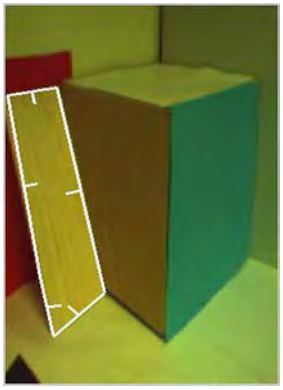
The criterion used to find such gaps requires a pair of nearby corners that each have less than two lines connected to them. If the Mahalanobis distance between these two corners, found using (5.50), is less than 2.5, then they are connected by a new boundary line.

If the gap is too large, then more effort is required to estimate a boundary, if at all possible. This generally occurs in situations when whole image lines are not extracted or grouped. This is usually due to difficult feature extraction conditions such as similar surface material and colour. In this case the two corners are tentatively joined but the line is marked as being a temporary line in a similar fashion to virtual lines. This is shown by the dotted red line in Fig. 5.9(b).

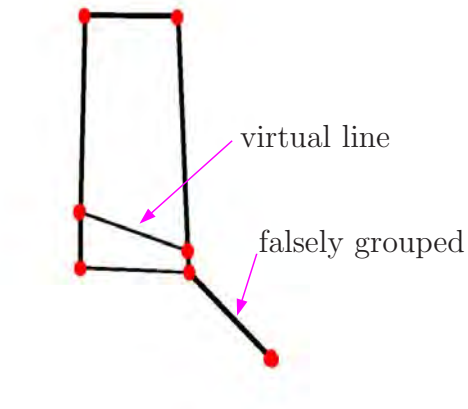
(a) Extracted image lines (white) with virtual line from bottom edge.



(b) Extracted image lines with falsely grouped line from wrong surface (ground).



(c) All four image lines correctly extracted and grouped.



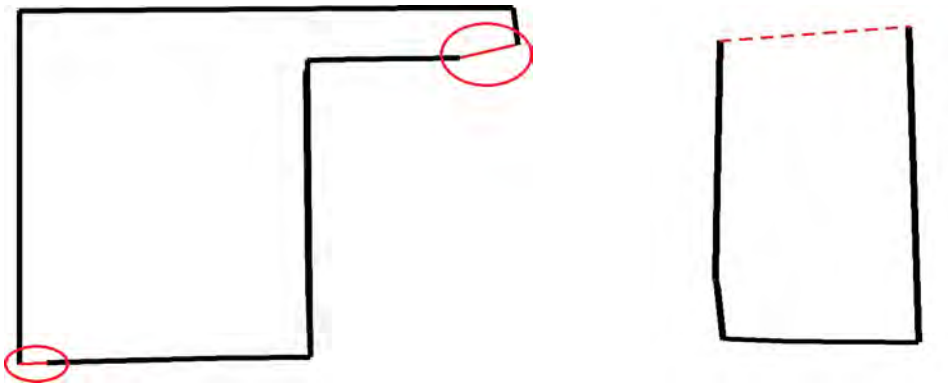
(d) Polygon boundary with erroneous lines to be removed.

Figure 5.8: Polygon boundary simplification by removing erroneous lines.

This allows the polygon boundary to be closed for visualisation purposes and allows navigation to proceed cautiously in those areas near this new line until more information is acquired. When this surface is merged with that of following 3D scans it may be possible to fill in the gap with real lines and thus the temporary line can then be removed.

Merging of Linear Boundary Lines

This registration and merging method has a tendency to produce corners within linear boundary edges. These should be removed as the edge is otherwise linear and this allows the boundary representation to remain compact. The main reason for



(a) Two small gaps filled (circled red lines).

(b) One large gap temporarily filled (dotted red line).

Figure 5.9: Examples of filling gaps in the polygon boundary.

these extra corners is the occasional over segmentation of image lines during feature extraction as well as over segmentation at image T junctions. This leads to new corners being introduced within linear edges, as described in Section 5.5.2.

These extra corners provide useful information, as the algorithm itself cannot distinguish between an over segmented line segment and a true corner. The merging process takes care of this because a falsely over segmented line will get merged back into the line with subsequent observations if it is indeed truly linear. In this case the corner is redundant and can be removed without any loss of information.

The angle θ_c of each corner in the boundary can be calculated, as shown in Fig. 5.10, using (5.58). This allows redundant corners to be identified if this angle is close to 180°. A threshold of 8° was used, as this was the threshold used during the extraction of image corners in Section 4.2.1.

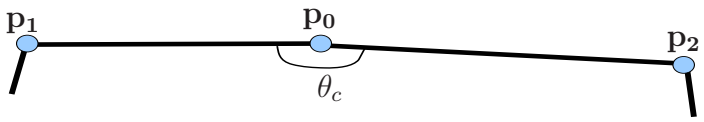
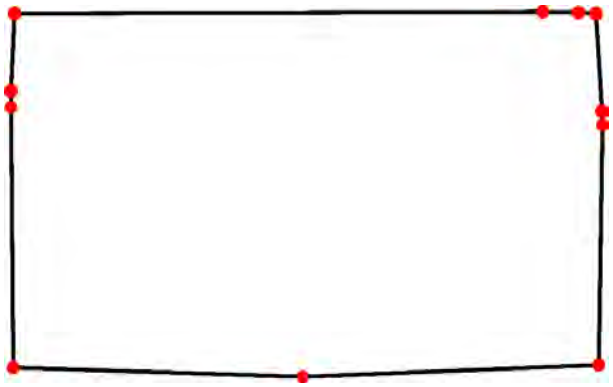


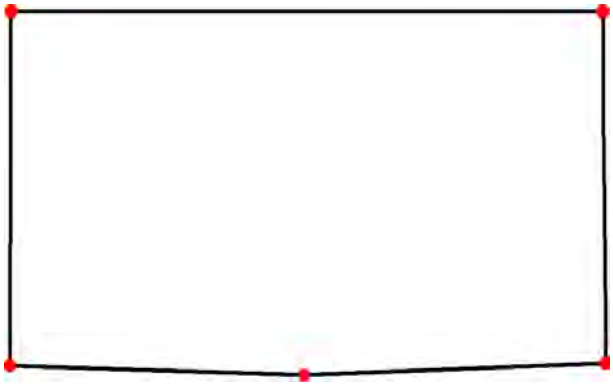
Figure 5.10: Image corner angle calculation.

$$\theta_c = \cos^{-1}\left([\mathbf{p_2} - \mathbf{p_0}] \cdot [\mathbf{p_1} - \mathbf{p_0}]\right) \tag{5.58}$$

Each corner that is identified with $\theta_c < 8^\circ$ is removed from the boundary and the two lines it connects are merged into one. This simplifies the storage and subsequent usage of the polygon boundary. An example from the real dataset is shown below in Fig 5.11. A polygon boundary with several redundant corners is shown in Fig. 5.11(a). After identifying and removing several corners, the final boundary is seen in Fig. 5.11(b) with only the significant boundary corners remaining.



(a) Polygon boundary with several redundant corners from experimental dataset.



(b) Linear corners merged leaving only significant corners.

Figure 5.11: Polygon boundary simplification by removing redundant corners.

5.6 Results of Planar Surface Fitting

The methods proposed in this chapter for fitting planar surfaces to sparse range and vision data have been evaluated. Both real and simulated datasets were used to validate the performance. The plane parameter fitting accuracy and polygon boundary corner accuracy are presented below and compared with their ground truth values. Each 3D scan was then reconstructed and compared with the original scene to give a visual representation of the quality of this surface mapping approach. These results will then be discussed in detail in Section 5.8.

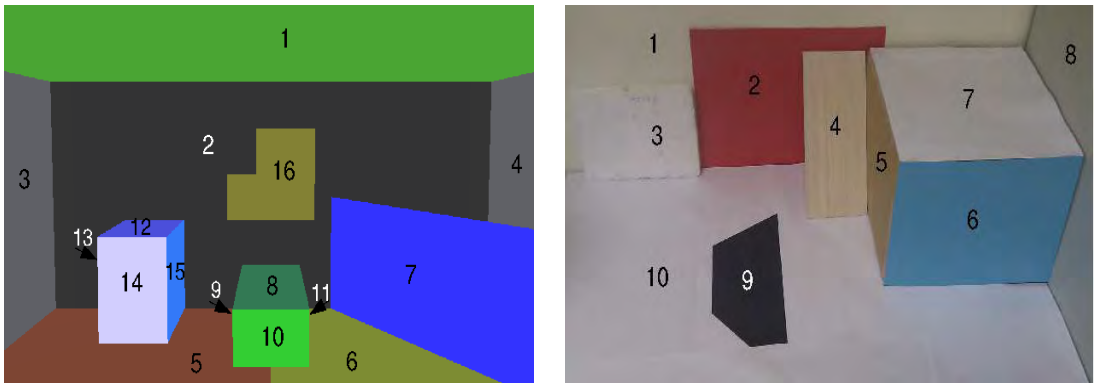
Experimental Setup

The experimental and simulated environments used in this chapter were the same as those used previously to validate the feature grouping methods of Section 4.4.6. These environments were fully described in Sections 3.3.1 and 3.3.2.

Five simulated 3D scans were taken, at different poses denoted A-E, with each containing eleven 2D scans at an angular resolution of 10°. The experimental dataset consisted of three 3D scans, at different poses denoted A-C, with each scan having nine 2D scans at an angular resolution of 5°. Refer to Sections 3.3.1 and 3.3.2 for more information regarding the pose of each 3D scan. The simulated range measurements contained Gaussian noise only. The experimental range measurements, however, contained both Gaussian noise and systematic biases as they were generated by a real sensor. This was discussed in Section A.3.

These two scenes can be seen below in Figs. 5.12(a) and 5.12(b). Each surface is labeled numerically, as detailed in Tables 5.3 and 5.4. Several surfaces are not visible in Fig. 5.12(a) below due to the viewpoint of the image and these are indicated with arrows. Note that these two scenario environments, while similar, are intentionally different and the surface numbering scheme is unique to each environment.

The input feature groups used for these plane fitting results were extracted and grouped as described in the previous chapter. The groupings were thus imperfect but realistic.



(a) Simulated scene. (b) Experimental scene.

Figure 5.12: Test environments with surface numbering.

5.6.1 Infinite Plane Fitting Results

The plane parameters that were fitted using the equations from Section 5.4 are presented below. For each 3D scan the mean errors were calculated both in absolute terms and normalised by the covariances calculated in Section 5.4.2. The normal vector $\mathbf{n}$ error was calculated as the angle between the true normal vector and the fitted vector while the error in the d parameter was the absolute scalar error. The calculation and explanation for each term is shown below. The results for the five simulated 3D scans are shown in Table 5.1 and the results for the three real data 3D scans are shown in Table 5.2.

$\mu_{\theta_\mathbf{n}}$

The mean of the absolute angular error in degrees between each fitted normal vector $\mathbf{n}$ and the corresponding ground truth normal vector $\mathbf{n^0}$.

$$\mu_{\theta_n} = \frac{1}{N}\sum_{i=0}^{N}|\theta_{n_i}| \qquad \text{where} \qquad \theta_{n_i} = \cos^{-1}(\mathbf{n}_i \cdot \mathbf{n}_i^0) \tag{5.59}$$

$\mu_{\theta_\mathbf{n}}$ (σ)

The same as above, except each angle is normalised by its covariance in a manner analogous to calculating the Mahalanobis distance. The normalising angle using the covariance is the angle between the normal vector and a vector

that is one standard deviation away.

$$\mu_{\theta_{n(\sigma)}} = \frac{1}{N}\sum_{i=0}^{N}\frac{|\theta_{n_i}|}{|\theta_{\sigma_i}|} \tag{5.60}$$

where

$$\theta_{\sigma_i} = \cos^{-1}\left(\mathbf{n_i}\cdot\frac{\mathbf{n}_i+\mathbf{n}_{\sigma_i}}{||\mathbf{n}_i+\mathbf{n}_{\sigma_i}||}\right) \quad \text{and} \quad \mathbf{n}_{\sigma_i} = \begin{bmatrix}\sigma_x\\ \sigma_y\\ \sigma_z\end{bmatrix}_i \tag{5.61}$$

$\mu_{\mathbf{\Delta d}}$

The mean scalar error in millimetres between the calculated d parameter and the ground truth value d^0.

$$\mu_{\Delta d} = \frac{1}{N}\sum_{i=0}^{N}\left(d_i - d_i^0\right) \tag{5.62}$$

$\mu_{\mathbf{\Delta d}}\ (\sigma)$

The mean normalised error, which is the error in d normalised by its standard deviation σ_d.

$$\mu_{\Delta d(\sigma)} = \frac{1}{N}\sum_{i=0}^{N}\left(\frac{d_i - d_i^0}{\sigma_{d_i}}\right) \tag{5.63}$$

Table 5.1: Simulated dataset - Plane parameter fitting error

3D Scan	Planes	μ_{θ_n}	$\mu_{\theta_n}\ (\sigma)$	$\mu_{\Delta d}$	$\mu_{\Delta d}\ (\sigma)$
A	7	0.15°	0.66	1.20mm	5.63
B	7	0.24°	1.97	1.05mm	5.48
C	8	0.13°	0.49	1.10mm	4.69
D	6	0.23°	0.78	1.16mm	2.69
E	7	0.19°	0.79	1.82mm	4.47
ALL	35	0.18°	0.93	1.26mm	4.65

Table 5.2: Real dataset - Plane parameter fitting error

3D Scan	Planes	μ_{θ_n}	$\mu_{\theta_n}\ (\sigma)$	$\mu_{\Delta d}$	$\mu_{\Delta d}\ (\sigma)$
A	7	2.40°	3.35	12.9mm	32.6
B	5	1.20°	2.77	7.9mm	28.1
C	8	1.46°	2.11	7.6mm	27.8
ALL	20	1.73°	2.71	9.5mm	27.8

5.6.2 Polygon Boundary Generation Results

The accuracy of the polygon boundaries generated by this method has been evaluated. The boundary corner locations have been used for this task as ground truth information is available for these. The results using the simulated dataset are shown below in Table 5.3 while the results from the real dataset are shown in Table 5.4.

Only those corners that corresponded to known physical corners were used. Virtual corners do not have any matching physical corner and so ground truth is unavailable for these. This is the reason for the relatively low corner count for some of the planar surfaces. Some large surfaces, such as walls, extended beyond the camera FoV in some of the images and so relatively few non-virtual corners were seen.

To fairly assess the corner accuracy, the boundaries were projected onto the ground truth plane rather than using the plane parameters fitted to the laser range data. This allows the accuracy of the polygon boundary generation process to be examined independently of any errors in the plane parameters. A comparison is shown in Table 5.5 to express the difference in accuracy between these two cases.

The distance metrics used are described below.

$\mu_{\mathbf{\Delta c}}$

: The 3D Euclidean distance in millimetres between the measured corner and the ground truth.

$\mu_{\mathbf{\Delta c}}$ (σ)

: The Mahalanobis distance between the measured corner and the ground truth. This normalises the Euclidean distance by the corner location covariance and the units used are standard deviations.

To show the difference in corner location error introduced by using the fit plane parameters instead of the ground truth plane parameters, the corner error was recalculated using the plane parameters fitted to the acquired data for scan A from the real dataset as seen below in Table 5.5.

Table 5.3: Simulated dataset - Polygon boundary corner error

Planar Surface	# of Corners	$\mu_{\Delta c}$	$\mu_{\Delta c}$ (σ)
1 - Ceiling	3	59.3mm	3.12
2 - Back Wall	22	11.0mm	3.24
3 - Left Wall	2	16.0mm	2.99
4 - Right Wall	1	22.3mm	2.98
5 - Red Floor	16	8.2mm	1.51
6 - Yellow Floor	8	8.2mm	2.54
7 - Blue Diagonal RHS	1	13.8mm	2.85
8 - Green Box Top	4	15.0mm	4.62
9 - Green Box LHS	3	14.2mm	2.27
10 - Green Box Front	8	4.5mm	2.66
11 - Green Box RHS	0	-	-
12 - Blue Box Top	3	18.2mm	3.08
13 - Blue Box LHS	0	-	-
14 - Blue Box Front	7	9.0mm	2.18
15 - Blue Box RHS	10	8.4mm	2.03
16 - Back Wall Yellow Inset	16	8.0mm	2.23
ALL	104	11.1mm	2.54

5.6.3 3D Scan Reconstructions

For each 3D scan, using both the real and simulated datasets, the planar surfaces were reconstructed to give a qualitative representation of the plane fitting approach. Each reconstruction is shown below alongside the corresponding view of the environment. Note that the images do not completely align as they have different perspectives. The 3D reconstructions are built using multiple images and so the corresponding true scene image should be used as a comparative aid only. The five simulated data 3D reconstructions are shown in Figs. 5.13 - 5.17 while the three real data 3D reconstructions are shown in Figs. 5.18 - 5.20.

5.7 Further Results and Validation

A further series of 3D datasets were acquired and processed with the aim of investigating the performance of the proposed surface fitting approach for more difficult scenarios and also to determine its limitations. Three scenarios consisting of eight additional 3D scans were used.

Each scenario contained several planar surfaces with the addition of complicat-

Table 5.4: Real dataset - Polygon boundary corner error

Planar Surface	# of Corners	$\mu_{\Delta c}$	$\mu_{\Delta c}$ (σ)
1 - Back Wall	10	6.2mm	1.02
2 - Red Paper	10	6.8mm	1.42
3 - White Foam	0	-	-
4 - Balsa Wood	8	9.6mm	2.48
5 - Box Side	7	18.1mm	2.95
6 - Box Front	11	8.6mm	2.26
7 - Box Top	0	-	-
8 - PC Side	4	9.6mm	1.91
9 - Black Floor	8	12.7mm	3.01
10 - White Floor	15	13.6mm	2.95
ALL	73	10.6mm	2.27

Table 5.5: Polygon boundary corner error - Fit vs true plane parameters

3D Scan	# of Corners	$\mu_{\Delta c}$	$\mu_{\Delta c}$ (σ)
Real Scan A (true $\mathbf{n}$,d)	22	13.8mm	2.55
Real Scan A (fit $\mathbf{n}$,d)	22	14.7mm	1.88

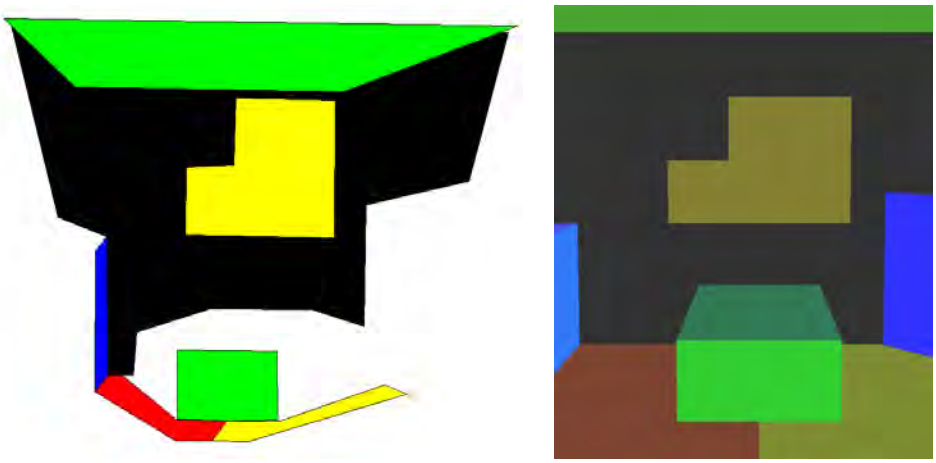
ing factors such as clutter, nonplanar or irregular surfaces, textured surfaces and shadows from different lighting conditions. It should be noted that the intention of using these difficult surfaces was not to attempt to fit surfaces to the nonplanar objects, as that is outside the scope of this work. Instead the aim was to investigate if these complicating factors degraded the surface fitting performance of the dominant planar surfaces and if so, to what degree.

The resulting 3D reconstructions of each scene are presented below and the performance will be discussed in detail later in Section 5.8.5.

5.7.1 Validation Scenario 1 - Clutter

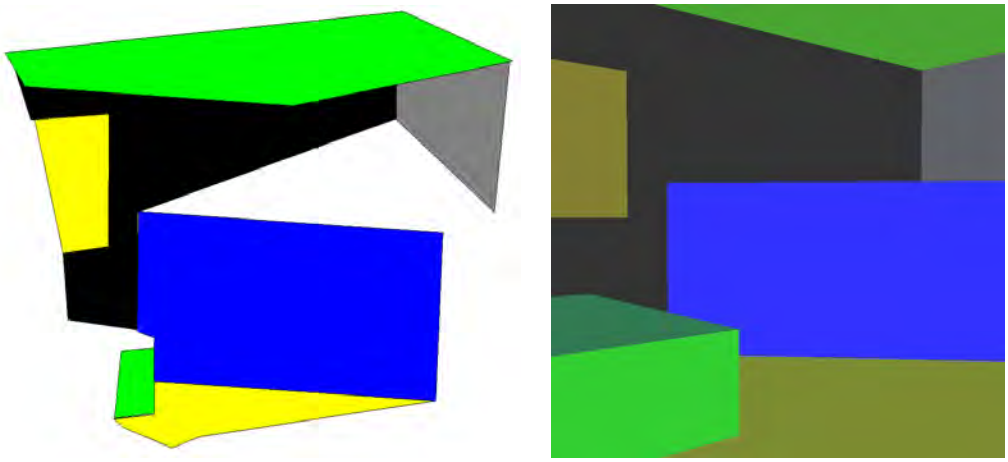
The first validation scenario was designed to investigate the effect of the level of clutter in the scene. Three 3D scans were taken of a scene with increasing levels of clutter and non-planar or irregular objects. The first scene, shown in Fig. 5.21, contained three basic planar surfaces with no clutter. These planar surfaces were the wall, floor and a book cover and they were present in all three scans.

It is important to note for each 3D reconstruction that the image of the true scene, such as that shown in Fig. 5.21(b), is presented for comparison purposes only



(a) 3D reconstruction - Sim 1. (b) Actual environment.

Figure 5.13: 3D reconstruction of simulated scan #1 for qualitative comparison.

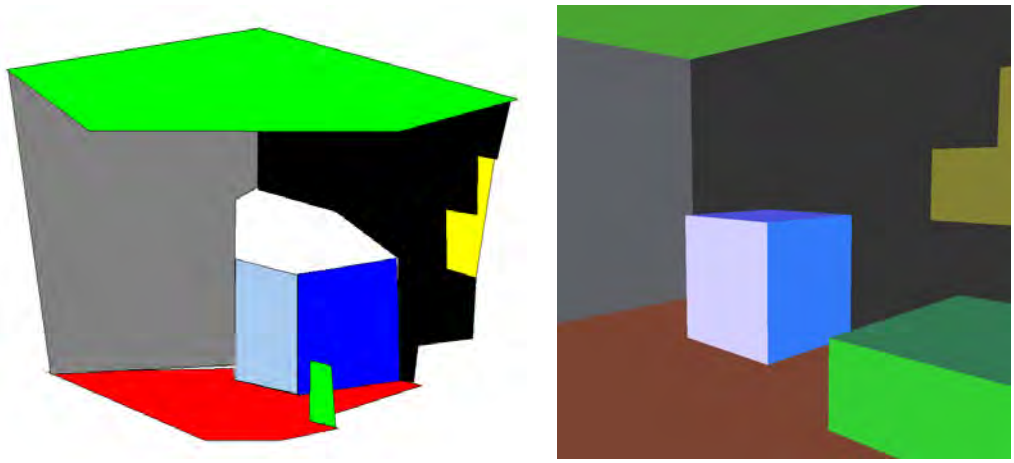


(a) 3D reconstruction - Sim 2. (b) Actual environment.

Figure 5.14: 3D reconstruction of simulated scan #2 for qualitative comparison.

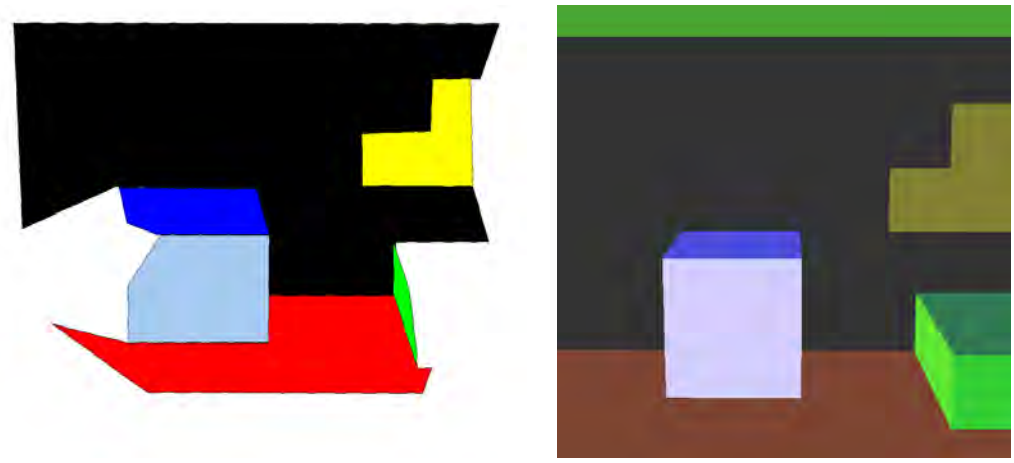
due to the different perspective used by the 3D scene reconstruction from multiple images. This perspective also explains the apparent missing corners in each 3D reconstruction.

The second scene, shown in Fig. 5.22, and the third scene, shown in Fig. 5.23, contained the same three basic planes as Fig. 5.21, but with the addition of increasing levels of clutter. This clutter included a can of drink, a set of keys, a small aluminium plate, a plastic bag, a coaster, an apple, a pen and a piece of balsa wood. These objects were generally small in size with irregular edges or nonplanar surfaces and contained various textures. They were chosen to test the surface fitting performance of the major planes in the scene, rather than attempting to fit surfaces to



(a) 3D reconstruction - Sim 3. (b) Actual environment.

Figure 5.15: 3D reconstruction of simulated scan #3 for qualitative comparison.



(a) 3D reconstruction - Sim 4. (b) Actual environment.

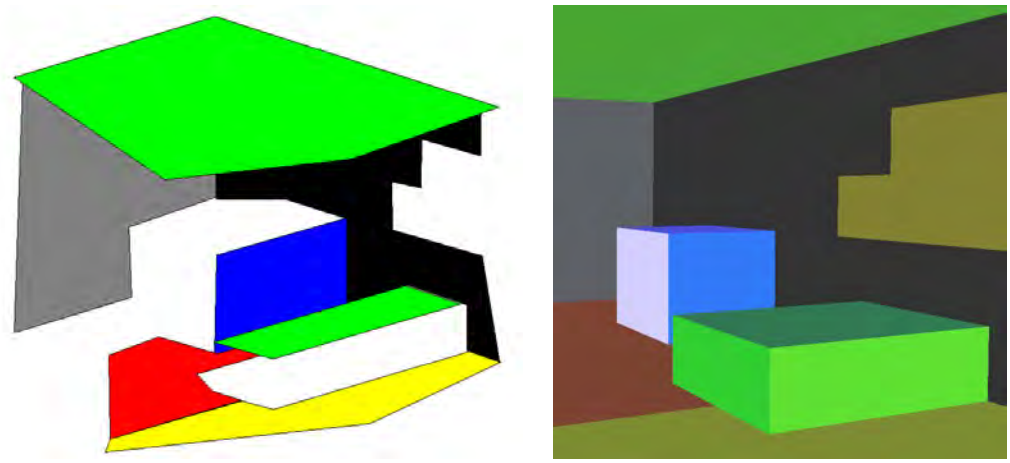
Figure 5.16: 3D reconstruction of simulated scan #4 for qualitative comparison.

them explicitly.

5.7.2 Validation Scenario 2 - Walls with Shadow

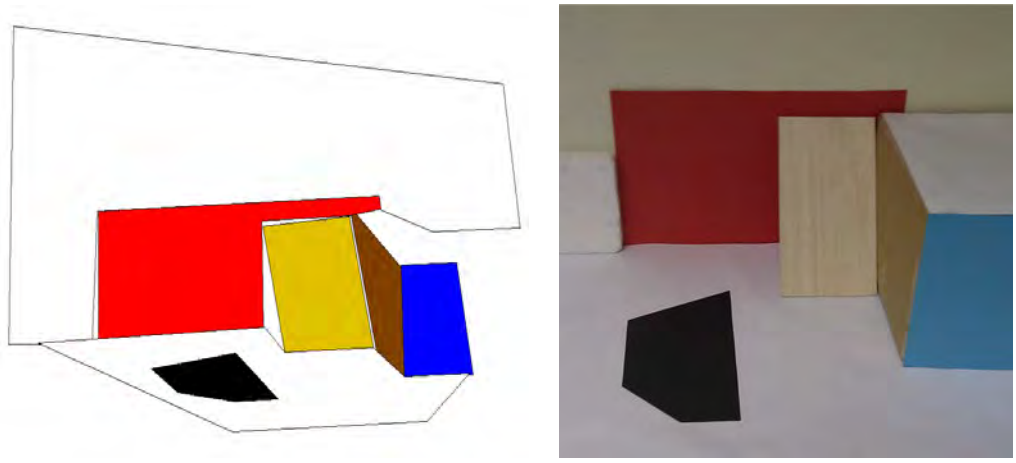
The second validation scenario was designed to investigate the surface fitting performance using a typical set of office walls. The three main surfaces, seen in Fig. 5.24, consisted of very similar colours and so were also used to test the ability to extract image edges and group surface features in the absence of sharp image gradients. A third complicating factor was the presence of some small cable conduits.

The second scene, shown in Fig. 5.25, introduced a fourth planar surface made of textured wood, along with a light source to generate some shadow. This was



(a) 3D reconstruction - Sim 5. (b) Actual environment.

Figure 5.17: 3D reconstruction of simulated scan #5 for qualitative comparison.



(a) 3D reconstruction - Real 1. (b) Actual environment.

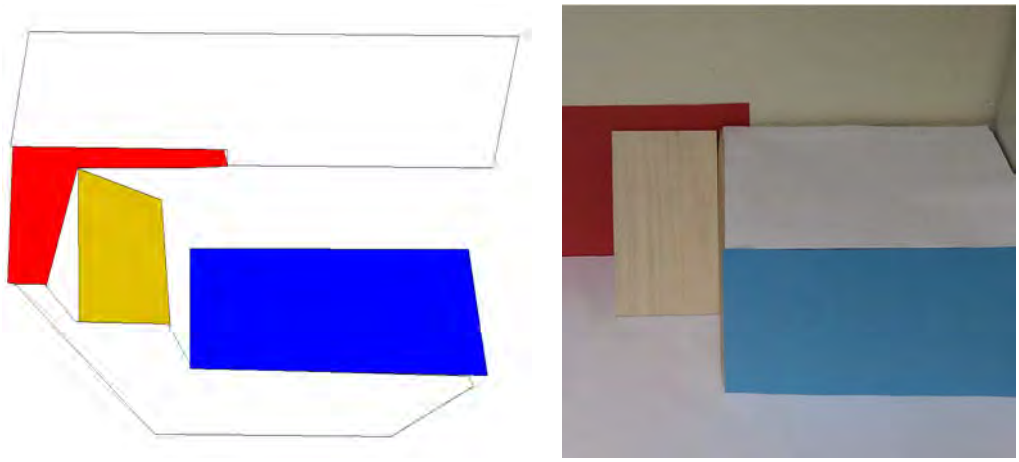
Figure 5.18: 3D reconstruction of real scan #1 for qualitative comparison.

designed to investigate the effect of edge extraction for textured surfaces as well as the effect of shadows and the ability to distinguish a shadow from a distinct surface.

5.7.3 Validation Scenario 3 - Outdoors

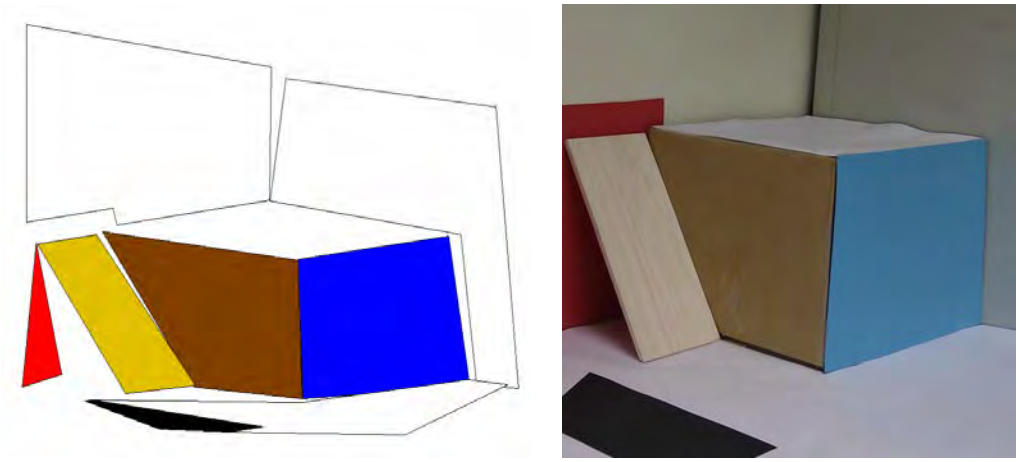
The third validation scenario was designed to investigate the surface fitting performance in outdoor conditions with realistic walls and surfaces consisting of various materials and textures.

The first outdoor scene, seen in Fig. 5.26, contained several walls along with other planar surfaces including a ledge, concrete floor and a window. These surfaces contained various textures, along with surface marks, and clutter in the form of



(a) 3D reconstruction - Real 2. (b) Actual environment.

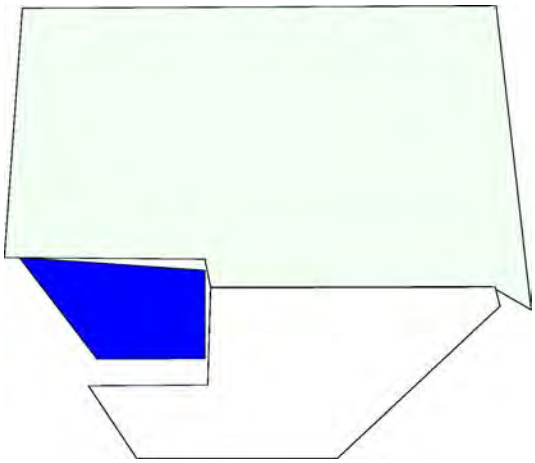
Figure 5.19: 3D reconstruction of real scan #2 for qualitative comparison.



(a) 3D reconstruction - Real 3. (b) Actual environment.

Figure 5.20: 3D reconstruction of real scan #3 for qualitative comparison.

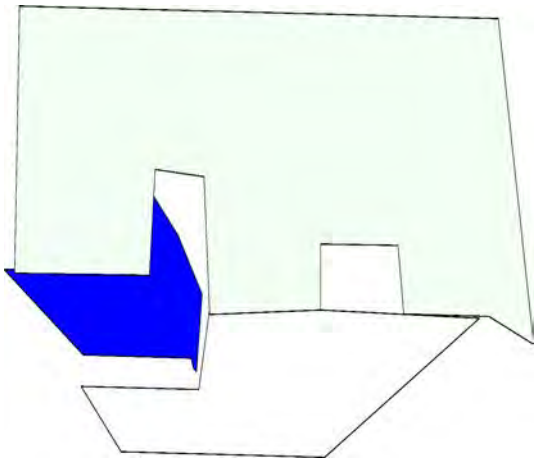
leaves. The second outdoor scene, seen in Fig. 5.27, contained a brick wall and several pipes. The third outdoor scene, seen in Fig. 5.28, consisted of the side of a shipping container along with a concrete floor that was irregular both in slope and colouring with a small plant in the bottom left corner. Each of the four surfaces is numbered to aid in comparing the corresponding surfaces in this scene as they are less obvious than the previous 3D reconstructions.



(a) 3D reconstruction.

(b) Actual environment.

Figure 5.21: 3D reconstruction of Validation scene 1a - No Clutter.



(a) 3D reconstruction.

(b) Actual environment.

Figure 5.22: 3D reconstruction of Validation scene 1b - Moderate Clutter.

5.8 Discussion

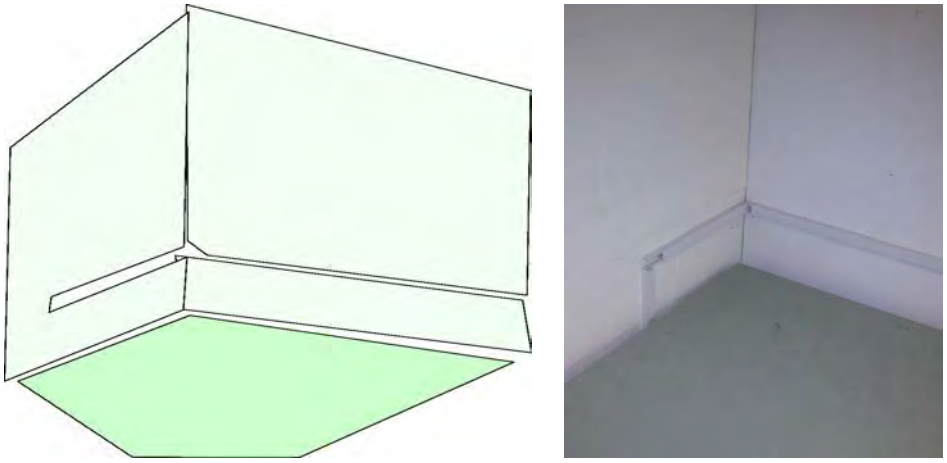
The methodology and results of the planar surface fitting approach presented in this chapter will now be evaluated and discussed.

5.8.1 Plane Parameter Fitting

The results seen in Tables 5.1 and 5.2 show that the sparseness of the range data has little impact on the accuracy of the plane parameters fitted using PCA. This allows planar surfaces to be mapped using 5-10x less raw data points, which will significantly improve the speed of the plane fitting. Such sparseness also makes

(a) 3D reconstruction. (b) Actual environment.

Figure 5.23: 3D reconstruction of Validation scene 1c - Substantial Clutter.

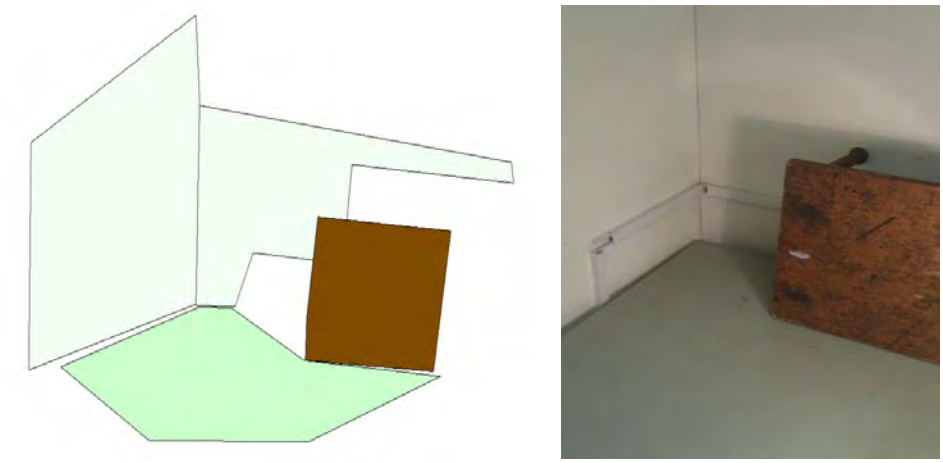


(a) 3D reconstruction. (b) Actual environment.

Figure 5.24: 3D reconstruction of Validation scene 2a - Basic Walls.

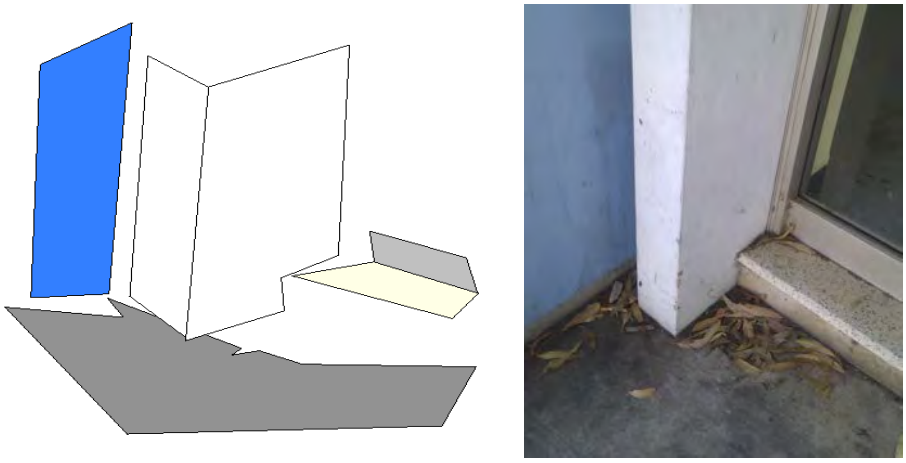
real time operation using limited processing resources viable as the computational and memory requirements are an order of magnitude less than when using dense scanning. The computational cost is further eased by the use of planar features as opposed to trying to operate on a map consisting of a large raw point cloud, as in many other robotic mapping approaches. The reductions in computation and memory usage gained by this approach will be further explored in Chapter 7.

There are two main limitations of this plane fitting approach and these can limit the variety of viable environments for robotic operation, if not handled appropriately. Firstly, any nonplanar surfaces cannot be fit properly using this plane fitting approach. It is assumed that the feature groups correspond to planar surfaces only



(a) 3D reconstruction. (b) Actual environment.

Figure 5.25: 3D reconstruction of Validation scene 2b - Shadows and Texture.

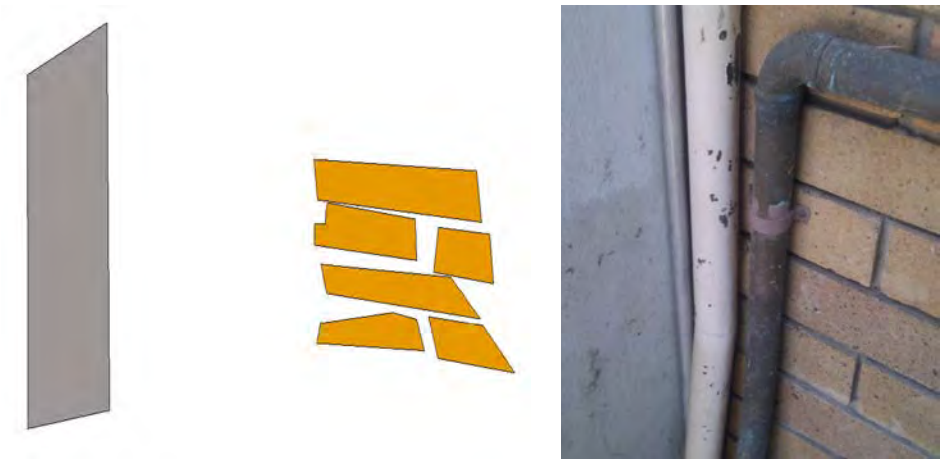


(a) 3D reconstruction. (b) Actual environment.

Figure 5.26: 3D reconstruction of Validation scene 3a - Outdoor Walls.

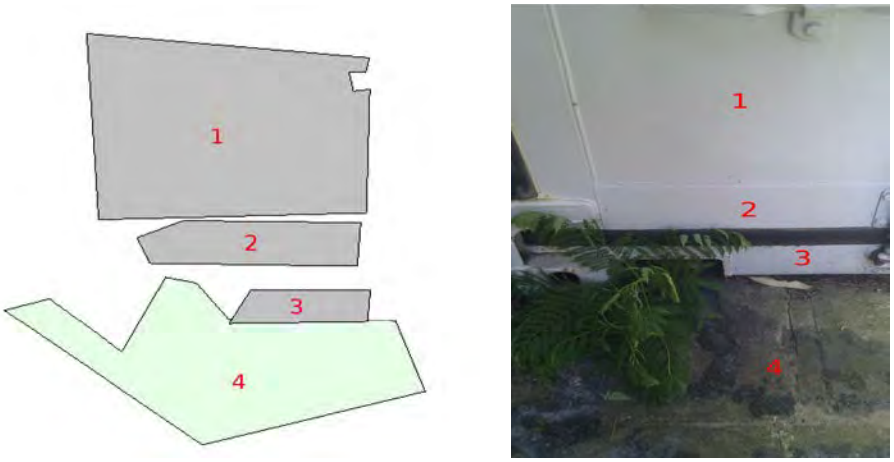
and any attempt to fit a plane to a nonplanar surface will obviously lead to an inaccurate surface fit. A method was described in the LRF line segmentation process of Section 4.3.4 to ensure that only linear line segments were extracted. The grouping stage of Section 4.4.5 also ensured that only coplanar laser line segments were grouped together.

These techniques allow any nonplanar surfaces or features to be filtered out prior to the plane fitting and an alternative surface fitting method is not proposed here. Such surfaces are unsuitable for a wall climbing robot to use as footholds because a planar surface is required to achieve an adequate vacuum seal with a suction cup. Those surfaces should nevertheless be represented in some fashion in the robot's map



(a) 3D reconstruction. (b) Actual environment.

Figure 5.27: 3D reconstruction of Validation scene 3b - Outdoor Bricks.



(a) 3D reconstruction. (b) Actual environment.

Figure 5.28: 3D reconstruction of Validation scene 3c - Outdoor Container.

as obstacles to be avoided. Such representations could include cylinders, spheres, bounding boxes or some type of polynomial function.

While the introduction of extra surface representations is outside the scope of this work, it is worth noting that the mapped planar surfaces required direct line of sight to the sensors. This means that there should exist free space between the sensor CF origin and those planar surface. Unmapped surfaces, nonplanar or otherwise, will appear as holes in the planar surface map and extra care should be taken about these areas during motion planning. The issue of motion planning for the CRACbot, and its predecessor, was addressed by Ward *et al.* [8].

A second limitation of this surface mapping approach is introduced by the sparse

nature of the range scanning. Some smaller surfaces can be missed by the LRF scanning plane and so a plane is unable to be fitted to that surface. Two linear segments from different LRF scans are required to fit a plane and this compounds the issue by requiring at least two LRF scans to intersect the surface and possibly more if one is unable to be successfully segmented or grouped. This can be seen in the 3D reconstructions as a few surfaces are missing because of this.

The obvious solution to this problem is to scan more densely but this quickly erodes the computational benefits of the sparse scanning. A better solution would be to introduce an adaptive scanning angular resolution to allow dense scanning in areas with smaller objects and then sparser scanning in areas with large surfaces such as walls. This will be discussed further in Section 7.3.3.

Despite these missing surfaces, the nature of this scanning process allows for multiple 3D scans to be merged from different poses and this process is the subject of the next chapter. Therefore, there is a level of redundancy built into the mapping process. Any surfaces that do not have sufficient range information in one 3D scan can be identified and a suitable pose chosen to acquire the required range scans for that surface. An alternative method would be to modify the plane fitting process to operate on range data from multiple 3D scans or incorporate pseudo range features using the image information. This is not considered at this stage.

5.8.2 Plane Parameter Accuracy

The plane parameters $\mathbf{n}$ and d were accurately fitted to the acquired and grouped range data, using the equations of Section 5.4. The results of fitting these parameters to the simulated dataset were shown in Table 5.1 above. A total of 35 planes were fitted from the five simulated 3D scans. The mean angular error of the normal vector $\mathbf{n}$ was 0.18° and the mean error in the orthogonal distance d was 1.26mm.

The simulated range data contained only Gaussian noise and this was almost completely averaged out. The small amount of remaining error was caused by errors in the LRF line segmentation of Section 4.3. This is to be expected as no

segmentation process is perfect and this issue affects all linear and planar segmentation approaches. Despite this, the planes were still fitted to an acceptable level of accuracy using this approach.

The results of plane fitting for the real dataset were shown in Table 5.2. From the three 3D scans, a total of 20 planes were fitted. The mean angular error of the normal vector $\mathbf{n}$ was 1.73° and the mean error in the orthogonal distance d was 9.5mm. It should be mentioned that these errors are taken against ground truth values that were difficult to measure exactly as they correspond to real objects.

As expected, the accuracy is lower for the real dataset due to the much noisier range data from a real LRF. Like the simulated dataset, the Gaussian noise gets averaged away. The linear range segmentation again causes some of the fitting error and this is slightly more prominent than in the simulated case as the noisy real range points are harder to segment accurately.

The majority of the real dataset plane fitting error was caused by unmodeled systematic range errors. This was discussed in Section A.3 and a basic range error model was used. More complicated sensor models have been proposed that try to account for errors due to the surface range, angle of incidence and material. They do, however, greatly complicate the plane fitting and covariance calculations as well as increasing the computational load. For many cases this extra accuracy is not required and the accuracy found here is very reasonable.

In general, it is difficult to compare the accuracy of this plane fitting method with those from the literature as quantitative results are rarely provided. However, in a qualitative comparison, the surfaces fit here show similar accuracy to those found by Weingarten *et al.* [118], Poppinga *et al.* [70] and Pathak *et al.* [46] amongst others.

5.8.3 Plane Covariance Calculations

The uncertainty of the plane fitting parameters is an important part of the mapping process and it is therefore important to calculate the covariance Σ_n and variance σ_d^2.

This uncertainty information allows planar surfaces from different 3D scans to be merged in a manner that ensures the most reliable surface information is weighted highest. Without it, such merging can only be undertaken by equally weighting each plane regardless of the relative uncertainty.

The plane uncertainty is also important as it allows the robot to make informed decisions about the best surfaces to use as footholds and avoid those surfaces that have high uncertainty. This is important because attempting to step onto an unreliable surface may fail, as it may be unsuitable as a foothold. In extreme cases this could also result in a catastrophic fall or collision.

The method used in this work to calculate the covariance of the plane parameters was described in Section 5.4.2. It was based on a method proposed by Weingarten *et al.* [118] which used an OLS plane fit, aligned with the PCA plane fit, to derive the covariance equations. They did not fully derive these covariance equations and so this work has extended their approach by deriving and proving a significantly simplified covariance calculation in the plane CF, as shown in (5.16)-(5.37). This allows the covariance to be calculated in a fast and efficient manner as an iterative sum.

The results of the covariances of the plane parameters can be seen in Tables 5.1 and 5.2. The plane parameter errors were calculated in absolute terms as well as being normalised by the parameter uncertainty, as represented by $\mu_{\theta_n}(\sigma)$ and $\mu_{\Delta d}(\sigma)$. The Mahalanobis error for d was straightforward to calculate using (5.63). The simulated dataset had a mean distance error of 4.65 standard deviations while the real dataset was 27.8 standard deviations.

The angular error of the normal vector θ_n did not allow a standard Mahalanobis distance calculation so the covariance Σ_n was also converted into an angular representation using (5.61). This allowed a normalised angular error to be found. The simulated dataset had a mean angular error of 0.93 standard deviations while the real dataset was 2.71 standard deviations.

These values indicate that the covariance is being significantly underestimated,

particularly for σ_d^2. This is most likely due to the assumption that the range noise is Gaussian with no systematic bias, which influences the d parameter significantly more than $\mathbf{n}$. This Gaussian noise assumption has been clearly shown to be false and yet the covariance does not account for it. The reason for this can be seen in (5.36), which forms the Jacobians for (5.37). The input Jacobian terms are inversely proportional to the number of points N and the sum of the x_i^2 and y_i^2 points, which are in the plane CF and have been mean centred. These denominator terms increase proportionally to the number of points supporting the plane as well as the size of the plane as the sum terms are a measure of the spread in the X and Y directions within the plane.

The result of this is that the covariance matrix converges towards zero as the number of points supporting each plane increases. This level of uncertainty is clearly overly optimistic and unrealistic when systematic biases are present. A secondary issue is the small size of the covariance values can cause numerical issues in practice. Despite the sparse scanning approach used in this thesis, this issue is still apparent and this would be compounded when dense range scanning is used. This problem was also noted by Weingarten [35] who proposed to add a small amount of constant noise to the diagonal elements of the covariance matrix.

An alternative approach would be to use a more powerful sensor model, such as those detailed in Section A.3, to estimate these systematic errors and eliminate them prior to calculating the covariance or even integrate them within the covariance calculations, as done by Pathak *et al.* [122].

There is a cost to using these more complicated sensor models, as in general no analytical solutions exist for them. As numerical or incremental methods are too computationally costly for real time operation on limited resources, approximate solutions become the only viable option. This does reduce the benefits gained by the extra complexity and so while some improvement in the accuracy of the covariance is possible, it is not necessarily significant.

If computational constraints allow, other sensor models or covariance calculations

could be substituted in place of the methods proposed here and there would be no effect on the remainder of the plane fitting approach. Having said that, the covariances calculated here still provide valid results, especially for Σ_n, and their usefulness will be shown in the following chapter.

5.8.4 Polygon Boundary Generation

As previously mentioned, the sparse nature of the range scanning does not provide enough raw points to generate an accurate polygon boundary on the planar surface. Mapping approaches using dense range scanning are able to use convex hulls of the segmented points, or similar methods, but sparse scanning does not allow for this. The image information is therefore used to provide the polygon boundary information and this approach has produced planar patches of comparable quality to other plane fitting methods, as seen in the 3D reconstructions of Figs. 5.13 - 5.20. The reconstructions show good agreement when comparing them side-by-side with images of the true environment and quantitative results have also shown the accuracy of this method.

The accuracy of the polygon boundaries was measured using the corner location error μ_{Δ_c}, both in absolute and Mahalanobis distances. Results were shown for the boundaries of 35 simulated surface observations and 20 real surface observations, as in Figs. 5.3 and 5.4 respectively. Only corners in the boundary that corresponded to a physical corner were used, as ground truth information was available for these. Virtual corners do not correspond to a physical surface corner and so their accuracy cannot be determined. This is the reason for the seemingly low number of corners tested, as many of the larger surfaces exceeded the camera FoV in most of the images.

The mean corner error was 11.1mm (2.54 standard deviations) for 104 simulated corners while the real dataset had a mean corner error of 10.6mm (2.27 standard deviations) for 73 corners. These values were calculated using the ground truth plane parameters for each plane to characterise the corner error independent of any

errors in the fitted plane parameters $\mathbf{n}$ and d.

A comparison was performed in Table 5.5 that compared the extent of the additional error caused by using the fitted, rather than ground truth, plane parameters for 3D scan A of the real dataset. For the 30 corners in this scan, the mean corner location error increased slightly from 13.8mm to 14.7mm when using the fit parameters. It is interesting to note that despite this increase in absolute terms, there was actually a decrease in the Mahalanobis distance error. The covariance increased proportionally more due to the extra uncertainty of the plane parameters. Overall the difference was not significant and this gives some confidence in the use of the covariance and uncertainty calculations.

In general, this corner accuracy is very good. It is not often characterised in other plane fitting approaches, which tend to rely on visual inspection to determine qualitatively the accuracy of their planar patches. It is therefore difficult to compare this boundary accuracy with that of other work. Nevertheless in absolute terms, the error of this approach is very low. This is especially good when the low resolution of the images is taken into account, as this places a lower bound on the achievable accuracy. The effect of the image resolution on the boundary accuracy will be further examined later in Section 7.3.1, but these results show that it is possible to fit good surface patches using such low resolution cameras.

It should be noted that several corners showed much higher absolute error. For example Surface 1 (Ceiling) of the simulated dataset in Table 5.3 had a mean error of 59.3mm. This value seems high at first glance but the Mahalanobis distance was only 3.12 standard deviations which is in line with the rest of the surface corners. These corners were relatively far from the sensor, at a range of almost two metres, and so naturally they were more uncertain than closer corners and this was accounted for by the larger covariance of these points.

The generation of the polygon boundaries is limited by the feature extraction and grouping process preceding it. Missing features or falsely grouped features will translate to errors in the polygon boundary. This is partly dealt with by the post

processing steps described in Section 5.5.4. However, those methods can only do so much. It is therefore important to ensure that the extraction and grouping of the image features are performed accurately and robustly.

Another limitation inherent to this approach of using image features, rather than range information to generate the polygon boundaries, is that the boundary is limited by the FoV of the camera. This especially concerns the horizontal FoV in the scanning configuration used here, as the sensors tilt vertically. In this work the simulated camera had a horizontal FoV of 60° while the CMUcam3 camera only had a horizontal FoV of 42°. This weakness is partially mitigated by the use of the virtual lines and corners to ensure that the maximum amount of image information is used to generate the polygon boundaries. The fact still remains that a camera with a wider FoV, such as one with a fish eye lens, would provide larger surface boundaries. However, the accuracy would be lower if the image resolution is held constant to limit the computational cost.

Despite these limitations, the results indicate that the accuracy for most of the fitted surfaces was within the required tolerances, as detailed in Section 3.2.1. Therefore, the planar surface patches produced are good enough to allow a wall climbing robot to use them as footholds. This is especially true after they are merged together to build a larger 3D map, as detailed in the following chapter.

5.8.5 Further Results and Validation

The basic experimental and simulated datasets, shown in Fig. 5.12, were used to demonstrate the proposed surface mapping approach and the results of which have been discussed above. A series of more difficult 3D scenarios were generated to examine validate the surface mapping performance against a range of complicating factors. The 3D reconstructions of these scenes were presented in Section 5.7.

Validation Scenario 1 - Clutter

The first additional set of three 3D scans was designed to test the surface fitting performance against increasing levels of clutter. The three scans, seen in Figs. 5.21 - 5.23, contained the same three main planar surfaces to allow for qualitative comparisons to be performed.

The first scene, shown in Fig. 5.21, contained no clutter. The three dominant planar surfaces consisting of the wall, the floor and the front cover of the book were extracted very well. A fourth surface was visible, which is the bottom of the book consisting of the white pages. This surface was extracted and grouped successfully, however, only one 2D laser scan intersected it and so a plane was unable to be fitted. Like all previous 3D reconstructions, the bottom corners are missing as they were not seen from the camera's perspective.

The second scene, shown in Fig. 5.22, introduced several small objects into the scene including a can of drink, a set of keys, a coaster and a small piece of aluminium. The three main planar surfaces were again extracted reasonably well. The back wall was not affected but the other two surfaces showed some effects from the clutter.

The book cover was successfully grouped and fitted but the image edges from the can of drink were grouped together with the book edges due to a similar colour and texture existing where the two objects intersected. A non-linear laser line segment was extracted for this non-planar can but it was filtered out to prevent it being falsely grouped and used for plane fitting. However, this left the image edges corresponding to the can free to be falsely grouped elsewhere as they were not already grouped. A potential modification could be introduced to allow line segments that are recognised as non-linear through to the grouping process but to prevent them from being used for plane fitting.

The floor was extracted reasonably well, except the coaster and the set of keys were not extracted as holes in the 3D reconstruction. Several linear piecewise edges were extracted for these objects, but this was not sufficient for them to be grouped in this case. The small piece of aluminium was not extracted, for the same reason

as the front side of the book. The ability to extract small surfaces is hindered by the lack of two intersecting laser scan lines. This is a weakness of this method in that it can struggle to extract some small objects in a cluttered scene. However, these small objects would not be suitable for use by a wall climbing robot as they are not large enough to place a foot on.

The third scene, shown in Fig. 5.23, had substantial levels of clutter and the plane fitting performance degraded further. Despite this, two of the major planes were still mostly extracted and the balsa wood was also fitted well. The area of the floor that was unoccupied by clutter was successfully extracted despite the irregular edges of the clutter. This was bordered by the plastic bag (top right), the pen (bottom right), the keys (bottom left), the book (top left) and the stapler, (top middle).

The book cover was not extracted but it was almost completely covered with other objects. It could not be reasonably expected to be extracted by this method, or any other surface fitting method for that matter. It should also be noted that none of the nonplanar or irregularly shaped clutter had a false surface fitted to them because their non-planar surface features were filtered out using the methods described earlier in Sections 4.3.4 and 4.4.5.

Validation Scenario 2 - Walls with Shadow

The second validation scenario was designed to investigate the surface fitting performance using a typical example of office walls.

The three main planar surfaces, seen in Fig. 5.24, were all of a very similar colour and this was used to further test the surface extraction and grouping robustness. The presence of some small cable conduits, which were the same colour as the wall behind it, was another complicating factor.

The three dominant surfaces were extracted well, despite the similar colours of each surface. This had little effect on the success of the surface fitting. The cable conduits were also seen and acted to split the walls into smaller surfaces. This is

important because these conduits protrude from the wall and so would be unsuitable to act as a foothold. They would possibly be difficult for other methods to detect due to their small size and could easily be mistaken for noise in a dense 3D range scan.

The second scene, shown in Fig. 5.25, introduced a fourth planar surface made of textured wood along with a light source to generate some shadows. This was designed to investigate the effect of edge extraction for textured surfaces as well as the ability to distinguish a shadow from a distinct surface.

The introduction of a textured surface did not impede the surface fitting. This was helped by the regular and repeating nature of the textured wood grain. This resulted in many small and non-linear edges being extracted during the image feature extraction process and the majority of these were easily filtered out.

This is in contrast to textured surfaces that are irregular or contain a small number of dominant textural areas, such as the book cover from Fig. 5.21. Such texture can in some cases generate large enough and straight enough image edges to impact on the grouping and surface fitting performance. These textured surfaces are less common in the target environment for a wall climbing robot, as common industrial and other structured environments usually consist of surfaces with regular texture.

The shadow introduced by the light source did affect the surface extraction performance of this scene. The area of shadow on the back wall was seen as a distinct surface on the grouping from several images, due to the sharp image gradients, but in other images it was grouped together with the wall behind it. This is the reason why the back wall was not extracted as high as the left wall. This redundancy did allow the shadow to be ignored as it presented as internal edges. These were ignored according to the post processing rules of Section 5.5.4. Part of the desk that was covered in shadow was not grouped together with the rest of the desk. This presented as a hole in the 3D reconstruction because only one laser scan line intersected this small area and so a separate surface could not be fitted.

Validation Scenario 3 - Outdoor Scenes

Three outdoor scenes were analysed and the resulting 3D reconstructions were presented above in Figs. 5.26-5.28. These scenes were designed to test the surface fitting performance with a range of outdoor walls and surfaces that might be encountered by a wall climbing robot in the real world.

The first scene, shown in Fig. 5.26, was of the side of a building. It contained several walls and small planar surfaces as well as the ground and a glass window. As expected, the window was not extracted, but of the other planar surfaces in the scene, 6 out of 7 were successfully extracted. These surfaces included different materials and textures and most of the plain coloured surfaces contained surface marks. The concrete pavement was partially covered in leaves and the surface fitting algorithm was able to successfully extract the visible surface despite this additional visual clutter.

The second outdoor scene, shown in Fig. 5.27, was of a brick wall with some piping. The bricks were all extracted successfully, however, the edge accuracy was somewhat diminished. This is still a good result as the bricks exhibit surface texture and are individually small in size. The left wall was also extracted but the two pipes were not. This is important as they are non-planar surfaces and were filtered out.

The third outdoor scene, shown in Fig. 5.28, was of the side of a shipping container with a plant growing out the bottom. Several planar surfaces were fitted to the side of the container as well as the concrete ground. The irregularly shaped plant presented as a hole in the concrete ground but the edges were not followed exactly. The ground did exhibit a variety of texture and colour and so its extraction is a good result.

Validation Scenario Performance

The extra eight 3D reconstructions have shown that the proposed 3D surface fitting algorithm can perform reasonably well against a variety of difficult environmental and surface conditions. The overall results were promising and while the test sce-

narios were not exhaustive, they fulfilled their main intention of demonstrating the feasibility of the proposed surface fitting approach and identified several limitations for robotic usage in its current form.

Clutter and irregular edges: The presence of clutter in a scene did degrade the surface fitting performance but in many cases it did not prevent the predominant planar surfaces being extracted. Those small objects that were successfully grouped were in many cases unable to have surfaces fit to them and so presented as holes in the 3D reconstruction. Other clutter that had irregular edges were sometimes filtered out through the image edge extraction and grouping processes and so were completely missed in some cases.

It should also be noted some of this small clutter can be difficult for any surface fitting approach, especially those based solely on range scans. The small thin objects, such as the aluminium plate and the coaster, would be almost indistinguishable from the planar surfaces behind them in such dense 3D range scans, whereas the method presented here method does recognise some of them, albeit in a limited fashion.

In all the 3D reconstructions, every non-planar surface was successfully filtered out and none had a plane fit attempted. This is important in minimising erroneous planar surface fits which could lead to catastrophic failures for a climbing robot.

There is further potential to improve the surface fitting success and robustness by using less restrictive, but more computationally expensive, edge representations to allow small clutter to be extracted more successfully. An adaptive scan tilting angle would also help in such scenes to scan the small clutter more densely. This idea was introduced earlier in Section 5.3 and is discussed in more detail later in Section 7.3.3.

Shadows and illumination:

Shadows were found to slightly degrade the surface fitting performance in some cases. Fig. 5.25 contained some shadows which affected the surface fitting performance for several surfaces. The back wall surface was able to ignore and merge part of the shadow. The shadow falling on the desk and the rest of the back wall pre-

sented as holes in the 3D reconstruction as neither had the required two intersecting laser lines to allow a surface to be fitted.

Shadows are a problem for any visual mapping approach but the availability of range information could allow them to be merged with the background surface the shadow falls upon, rather than fitting them as two distinct surfaces as in the current approach. Such merging of adjacent coplanar surfaces, as would be done in a pure range based geometric 3D mapping, is intentionally not performed as part of the proposed mapping approach.

Adjacent coplanar surfaces could actually be distinct and slightly different surfaces containing a small step change. This can be caused by merging tolerances or systematic biases inherent to the range data canceling out any true range differences. For a wall climbing robot, attempting to place a foot on such a step could cause a lack of adhesion and a potential fall. An example of this is the small aluminium plate in Fig. 5.22. For other applications, such as wheeled robots, it could be acceptable to perform such merging and this would handle most examples of shadow.

Texture:

Those surfaces with regular texture were handled well in most cases. These generally presented with a large number of small false image edges which were easily filtered out during the extraction and grouping stages. This texture was also modeled reasonably well using the RGB colour descriptors for feature grouping. Successful examples were the textured wooden surface of Fig. 5.25, the speckled step in Fig. 5.26, the bricks in Fig. 5.27 and the concrete ground in Fig. 5.28.

Several surfaces which contained irregular texture, such as the book cover in Fig. 5.21, did generate large straight edges which degraded the performance in some cases. In the likely wall climbing environment the majority of texture surfaces should exhibit regular texture and so this is not considered a major issue.

5.9 Summary

This chapter has described a novel approach to fitting planar surface patches to 3D scans consisting of a limited number of laser range scans and camera images. The first half of this chapter described a plane fitting method using principal components analysis. This included estimating the uncertainty of the plane parameters. A full derivation and proof of the plane covariance was shown which had not been performed in similar work. A method for fitting the polygon boundaries using the image feature information was also presented. This involved the projection of image corners onto the infinite plane and a procedure for merging the boundaries from multiple images was detailed.

The results showed, both qualitatively and quantitatively, that this method is able to produce accurate planar surface patches using sparse range and image data. Their quality was comparable to the those produced by other surface mapping methods from the literature. More difficult scenarios were also tested for further validation. The proposed surface mapping approach was still able to successfully extract most major planar surfaces in each scenario despite the presence of complicating factors such as clutter, texture, non-planar or irregular surfaces, shadows and outdoor environments.

The surface patches extracted in this chapter will enable a wall climbing robot to build a local 3D map of the surrounding environment by merging together multiple 3D surface scans. This surface merging is the subject of the next chapter.

Chapter 6

Merging of Surfaces to Generate 3D Maps

6.1 Introduction

Chapter 5 detailed the process of fitting planar surface patches to an individual 3D scan. Multiple such scans can be acquired at different poses from the same foothold location. This chapter describes the process of merging them together to create a local 3D map about the robot's current position that is more useful than the individual scans on their own.

Before planar surface merging can take place, the change in pose between each new 3D scan is estimated and corresponding surfaces are registered together. A method for merging these planar surfaces using a Kalman Filter is then detailed along with two proposed modifications to alleviate several practical and numerical issues. The equations to transform the individual polygon boundaries onto the new merged plane are derived. These projected boundaries are merged together to improve the surface accuracy and field of view.

The proposed merging method will be validated using the individual real and simulated 3D scans found in Section 5.6. This will be followed by a discussion of the merits of this merging method and the suitability of the resulting 3D surface maps

for use within planar SLAM algorithms from the literature.

6.2 Scan Alignment and Surface Registration

As individual 3D scans are acquired at different locations in space, there needs to be some way of estimating their relative pose so that they can be aligned within the same CF. A separate data association problem, known as feature registration, involves determining which planar surfaces within these scans correspond to the same physical surface and should thus be merged together.

A common method for pose estimation is to use odometry. This uses proprioceptive sensor measurements of the robot's internal state to estimate its motion and thus infer the change in position over time. These measurements, such as joint rotation from a shaft encoder, are combined with a motion model to estimate the pose change. This approach is an open loop process but if the movement between each scanning pose is small then the estimated pose from odometry alone is likely to be reasonably accurate. In general the pose estimate will drift over time, due to odometry errors such as noise and slippage at the ground contacts.

To correct for drift in the pose estimate, a SLAM approach is useful as it closes the loop with exteroceptive measurements of the robots position relative to its environment. This may include real time kinematic GPS position information or environmental features or landmarks such as planar surface patches. An EKF is a popular framework for implementing SLAM and neatly combines these two processes. The prediction step uses the odometry and motion model to predict the robot's pose until an environmental measurement is acquired and the update step corrects the pose to provide a more accurate estimate.

Kohlhepp *et al.* [32] proposed an EKF based 3D SLAM approach to build a 3D map using planar surfaces. However, they separated out the SLAM based pose estimation and the 3D planar map generation and so the two components were only loosely integrated. Weingarten *et al.* [33] used the SPmap [34] as a way to integrate planar surfaces as features within the EKF SLAM framework.

Their results showed good pose estimation. However, their wheeled robot was ground constrained and so limited to 2D motion. Changes in the ground height degraded their performance and this was a weakness of their implementation. They also modified their approach to use planar segment information during data association [35] which showed some improvement in the pose accuracy as well as improving the visualisation. This SLAM approach was also used by de la Puente *et al.* [36] in a simulated rescue robotics scenario, with similar results.

An alternative planar SLAM approach was proposed by Pathak *et al.* [74] which operates without odometry or other motion information. They estimated the pose change by solving the registration problem first using an algorithm called Minimally Uncertain Maximum Consensus (MUMC). In this manner it is similar to scan matching approaches, such as ICP [39], but uses planar surfaces rather than the raw point clouds [125]. The other key difference is that it estimates the rigid transformation between scanning poses in a closed form after maximising planar surface correspondences in a search space. This operated on infinite planes only, with planar polygons used for visualisation. Despite this it was able to build 3D maps successfully in a variety of real world environments and was significantly faster than ICP based scan matching methods.

6.2.1 Pose Estimation

For the CRACbot, introduced in Section 3.2, the change in pose between the acquisition of each 3D scan will be estimated using odometry as the joint displacements are relatively small. The estimated pose can be used to fuse together multiple 3D scans taken from a single foothold to build a local 3D map about that position. It is assumed that the suction cup does not slip during this process. This is justified because the centre of gravity of the CRACbot does not change significantly during the acquisition of the 3D scans, as the joint displacements are small.

Each time the robot takes a step, this local 3D map will be merged into a global 3D SLAM framework as the pose will have possibly changed considerably.

While the implementation of this is outside the scope of this thesis, the process and applicability of using these extracted planar surface patches within 3D SLAM will be discussed later in Section 6.6.2.

Transformation of Planar Surfaces to Common CF

Using odometry information, an estimate of the change in pose is available in the form of a rigid transformation described by a rotation matrix R_α and a translation vector T_α . These can then be used to transform the new 3D scan back into the local foothold CF so that it can be merged into the local 3D map.

The transformations are described below in (6.1-6.2) and the polygon boundary information can also be transformed point-wise using (6.3). The subscript denotes the CF of each term where necessary, with i denoting the new 3D scan and 0 denoting the local foothold CF.

$$\begin{aligned} \mathbf{n_0} &= R_\alpha \mathbf{n_i} \\ \Sigma_{n_0} &= R_\alpha \Sigma_{n_i} R_\alpha^{\mathrm{T}} + \left(\frac{\partial \mathbf{n_0}}{\partial \alpha}\right) \sigma_\alpha^2 \left(\frac{\partial \mathbf{n_0}}{\partial \alpha}\right)^{\mathrm{T}} \end{aligned} \tag{6.1}$$

$$\begin{aligned} d_0 &= d_i + \mathbf{n_0^T} T_\alpha \\ \sigma_{d_0}^2 &= \sigma_{d_i}^2 + \mathbf{n_0^T} \Sigma_{T_\alpha} \mathbf{n_0} + T_\alpha^{\mathrm{T}} \Sigma_{n_0} T_\alpha \end{aligned} \tag{6.2}$$

$$\begin{aligned} \mathbf{p_0} &= R_\alpha \mathbf{p_i} + T_\alpha \\ \Sigma_{p_0} &= R_\alpha \Sigma_{p_i} R_\alpha^{\mathrm{T}} + \left(\frac{\partial \mathbf{p_0}}{\partial \alpha}\right) \sigma_\alpha^2 \left(\frac{\partial \mathbf{p_0}}{\partial \alpha}\right)^{\mathrm{T}} + \Sigma_{T_\alpha} \end{aligned} \tag{6.3}$$

The uncertainty of each term is dependent on the form on the rotation matrix used. The chosen form is the rotation matrix R_α, which describes rotation by an angle α about a single joint axis. When this is coupled with the translational offset T_α, they describe the transformation about a single rotating joint, such as those

used on the CRACbot in this thesis. Thus for multiple joints the above equations can be chained together to achieve the desired pose transformation with respect to the world CF.

6.2.2 Planar Surface Registration

Once the new 3D scan has been transformed into the local foothold CF, the planar surface pairs that correspond to the same physical surface need to be associated together in a process known as registration. Because the relative motion between the scans is small, the uncertainty of the estimated change in pose is also small. Therefore the desired surface pairs should align and overlap to a large degree, taking into account the measurement and motion uncertainty.

There are four criteria that must be satisfied for a pair of surfaces to be registered together for merging.

Surface Orientation

The two planar surfaces should have similar orientations in space. The covariance normalised angular error between the two normal vectors $\mathbf{n_i}$ and $\mathbf{n_j}$ should be less than a threshold of $t_n = 3$ standard deviations. The normalising angle θ_σ is found using (5.61).

$$\mu_\theta = \frac{|\cos^{-1}(\mathbf{n}_i \cdot \mathbf{n}_j)|}{\theta_{\sigma_i} + \theta_{\sigma_j}} < t_n \tag{6.4}$$

Surface Location

The two planar surfaces should be physically close in space. The Mahalanobis distance error between the two distances d_i and d_j should be less than a threshold of $t_d = 3$ standard deviations.

$$\Delta d(\sigma) = \left(\frac{d_i - d_j}{\sqrt{\sigma_{d_i}^2 + \sigma_{d_j}^2}} \right) < t_d \tag{6.5}$$

Surface Colour

The colour descriptors of the two surfaces should match, as detailed previously in Section 4.4.2.

Surface Polygons

The planar polygon boundaries of the two surfaces should overlap to some degree, taking into account the uncertainty of each polygon boundary.

6.3 Infinite Plane Merging

Once a planar surface patch from the new 3D scan has been registered and transformed into the foothold CF, it needs to be merged into the local 3D map. The first step of this process is to merge the infinite plane parameters ($\mathbf{n}$ and d) of the two planar surfaces. This is performed using a KF as detailed below. The filter is initialised with the parameters of the first 3D scan surface and these make up the state variable $\mathbf{x_k}$ with the associated covariance matrix Σ_k. This is shown in (6.6), where k denotes the index of the 3D scan that the planar surface was observed in.

$$\mathbf{x_k} = \begin{bmatrix} \mathbf{n} \\ d \end{bmatrix} \qquad \Sigma_k = \begin{bmatrix} \Sigma_n & 0 \\ 0 & \sigma_d^2 \end{bmatrix} \tag{6.6}$$

The KF equations described here are essentially the same as those described in Section 5.5.3. The Kalman gain K_k is found using (6.7) which then allows the state and covariance to be updated using (6.8-6.9). The new surface observation being merged into the filter is denoted $\mathbf{y_k}$ with associated covariance R_k. This observation has already been transformed into the original CF for merging using (6.1-6.2).

$$K_k = \Sigma_{k-1}[\Sigma_{k-1} + R_k]^{-1} \tag{6.7}$$

$$\mathbf{x_k} = \mathbf{x_{k-1}} + K_k[\mathbf{y_k} - \mathbf{x_{k-1}}] \tag{6.8}$$

$$\Sigma_k = [I - K_k]\Sigma_{k-1} \tag{6.9}$$

There are two issues with this approach that must be considered. Firstly, the two normal vectors may point in opposite directions for different observations of the same plane. Secondly, the merged normal vector may not remain a unit vector.

Pathak *et al.* [122] also noted these issues and proposed an alternative plane fusion approach which avoided these two problems. Their method found the merged state covariance using a modified version of (6.9) and then used this to find the state variable $\mathbf{x_k}$. This relied on their plane fitting equations which placed the plane parameters within the null space of their plane covariance and so the reverse transform could be used to find the merged plane parameters $\mathbf{x_k}$.

Their approach is not applicable here as they used a different plane fitting method to the one used in this thesis and it was also based on a more complicated sensor model. Nevertheless, there are simple solutions to the two problems mentioned above and they are detailed below. It will be shown later in the results of Section 6.5 that the basic KF approach of (6.7-6.9) provides good quality plane merging when these two modifications are applied.

6.3.1 Directional Consistency of the Normal Vector

The normal vector sign problem is a quirk of the plane fitting process. While $\mathbf{n}$ and $-\mathbf{n}$ can both represent the same physical plane in space, any attempt to merge them using this KF approach, or indeed any other merging approach, will fail. They will cancel each other out and the resulting estimate of the surface normal will be wildly inaccurate.

A solution to this issue involves modifying the plane fitting equations of Section 5.4.1 by adding an additional step to ensure consistency in the sign of the normal vector for each surface. This is shown below in (6.10) and ensures that each normal vector always points outwards from each surface.

$$\text{if } (\mathbf{n} \cdot \mathbf{p_0} - d < 0) \qquad \text{then } (\mathbf{n} := -\mathbf{n} \quad d := -d) \tag{6.10}$$

This works by checking the distance of a point $\mathbf{p_0}$ to the planar surface. If this distance is negative then that point is below the surface when looking in the direction of $\mathbf{n}$. Any point in space can be used for $\mathbf{p_0}$ but for meaningful results some choices are better than others. Any point that might intersect a surface, such as the world origin which likely intersects the ground plane, will introduce a singularity and so is not a good choice. A better point to use is the origin of the LRF taking each particular 3D scan. This ensures that all normal vectors point towards the LRF and thus outwards from each surface. This also eliminates the possibility of that plane intersecting with this point because such a plane would not have been seen by the LRF in the first place.

6.3.2 Ensuring Normal Vector Remains Normalised

When using the basic KF to merge a pair of planar surfaces, the merged normal vector $\mathbf{n_k} = [\mathbf{x_k}(1)\ \mathbf{x_k}(2)\ \mathbf{x_k}(3)]^T$ may no longer be a unit vector as required. This vector is formed from the first three components of the state vector $\mathbf{x_k}$ and must be renormalised in some manner. This can be handled in several ways.

An approximate solution is to explicitly normalise the normal vector component of $\mathbf{x_k}$, as in (6.11), to give the normalised state vector as $\mathbf{x_k^+}$. This is done by dividing the normal vector component by its Euclidean norm.

$$\mathbf{x_k^+} = \mathbf{x_k} N \qquad\qquad \Sigma_k^+ = N \Sigma_k N^{\mathrm{T}}$$

where

$$N = \begin{bmatrix} \frac{1}{||\mathbf{n_k}||} & 0 & 0 & 0 \\ 0 & \frac{1}{||\mathbf{n_k}||} & 0 & 0 \\ 0 & 0 & \frac{1}{||\mathbf{n_k}||} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.11}$$

A second approach to this problem is to integrate the requirement that $\mathbf{n}$ remain a unit vector into the KF equations as a state constraint. There are a number of methods to achieve this and they are surveyed and explained in detail by Simon [126].

The method used is called Perfect Measurements [127] and introduces the state constraint as an extra measurement with an uncertainty of zero. This requires the full KF equations to be used (excluding the prediction step), rather than the simplified ones shown in (6.7-6.9). The H matrix, which was previously an identity matrix, is reintroduced into the equations below.

$$\mathbf{y_k} = H\mathbf{x_k} + \mathbf{v} \tag{6.12}$$

$$\begin{bmatrix} \mathbf{n} \\ d \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \mathbf{x_k}(1) & \mathbf{x_k}(2) & \mathbf{x_k}(3) & 0 \end{bmatrix} \mathbf{x_k} + \begin{bmatrix} \mathbf{v_k} \\ 0 \end{bmatrix} \tag{6.13}$$

Using this augmented form of $\mathbf{y_k}$ and H, the KF equations become (6.14-6.16).

$$K_k = \Sigma_{k-1} H [H \Sigma_{k-1} H^T + R_k]^{-1} \tag{6.14}$$

$$\mathbf{x_k} = \mathbf{x_{k-1}} + K_k[\mathbf{y_k} - H\mathbf{x_{k-1}}] \tag{6.15}$$

$$\Sigma_k = [I - K_k H]\Sigma_{k-1} \tag{6.16}$$

This state constraint approach does not seem to work well in practice, showing similar results to the basic KF case, and in some cases showed slightly worse performance. This was likely due to numerical instability inherent to this method due to the singularity introduced to the covariance matrix by the perfect measurement. This was compounded by the small absolute size of the covariance values due to the

normalised nature of the state normal vector. Some extra method of normalisation, such as the explicit approach described above, is still required which negates the value of using these state constraints.

In cases where the error was very large and the observed planes differed significantly, the state constraint did hold the norm of the normal vector closer to 1 than the basic Kalman Filtering approach and so may have value in such cases. Despite this, these state constraints were not used due to the numerical issues and the explicit normalisation approach was favoured instead.

6.4 Polygon Boundary Merging

Once the new plane observation has been merged with the current plane estimate, as described above, the old boundary corner locations become obsolete and must be projected onto the merged plane estimate. This must be performed for both the current boundary estimate and the observed plane boundary to allow them to be merged together.

6.4.1 Projection of Polygon Boundaries onto Merged Plane

Each 2D corner location $\mathbf{p_1}$, on its original plane, must be first transformed into the world CF using equation (5.10). This 3D point can then be projected onto the newly found merged plane as follows.

The orthogonal distance D of this 3D point to the merged planar surface is found using the Hessian plane equation as shown in (6.17). In general $\mathbf{p_1}$ will not lie on the plane and so D is not equal to zero. This D parameter is the distance from $\mathbf{p_1}$ to a corresponding point $\mathbf{p_2}$ that lies on the plane, conforming to (6.18), and is the orthogonal projection of $\mathbf{p_1}$. It stands to reason that $\mathbf{p_2}$ can be found from $\mathbf{p_1}$ by subtracting the distance D in the direction of the normal vector $\mathbf{n}$.

$$\mathbf{n} \cdot \mathbf{p_1} - d = D \tag{6.17}$$

$$\mathbf{n} \cdot \mathbf{p_2} - d = 0 \tag{6.18}$$

Using these two equations it is possible to derive the equation for $\mathbf{p_2}$ as follows. Firstly, D is added to each side of (6.18) and then $\mathbf{n} \cdot \mathbf{n}$, which is equal to 1 by definition, is also introduced. This is then rearranged to a form matching that of (6.17) to give (6.20).

$$\mathbf{n} \cdot \mathbf{p_2} + \mathbf{n} \cdot \mathbf{n}D - d = D \tag{6.19}$$

$$\mathbf{n} \cdot (\mathbf{p_2} + D\mathbf{n}) - d = D \tag{6.20}$$

By comparing (6.20) with (6.17) it can be seen that $\mathbf{p_1}$ is found using (6.21).

$$\mathbf{p_1} = \mathbf{p_2} + D\mathbf{n} \tag{6.21}$$

This is then rearranged to give the desired equation for $\mathbf{p_2}$ using (6.22).

$$\mathbf{p_2} = \mathbf{p_1} - D\mathbf{n} \tag{6.22}$$

The uncertainty of $\mathbf{p_2}$ is calculated by first finding the variance of D as in (6.23) below. The Jacobians containing the partial derivatives of (6.17) are trivial to calculate in this case. The desired covariance Σ_{p_2} is then found using (6.24).

$$\sigma_D^2 = \mathbf{p_1^T}\Sigma_n\mathbf{p_1} + \mathbf{n^T}\Sigma_{p_1}\mathbf{n} + \sigma_d^2 \tag{6.23}$$

$$\Sigma_{p_2} = \Sigma_{p_1} + D^2\Sigma_n + \mathbf{n}\sigma_D^2\mathbf{n^T} \tag{6.24}$$

All that remains is to transform the 3D point $\mathbf{p_2}$ into the merged plane CF as a 2D corner point to allow the polygon boundaries to be merged. This is performed using (5.9). There are, however, some minor complications that need to be addressed first.

As the normal vector $\mathbf{n}$ of the merged plane has been modified, the rotation matrix V used in (5.9) will no longer be orthogonal due to small changes in the direction of $\mathbf{n}$. The solution is to generate a new orthonormal rotation matrix by selecting any two vectors that are mutually orthogonal to $\mathbf{n}$. This can be done by using Gram-Schmidt orthogonalisation [128], with the vectors $\mathbf{v_1}$ and $\mathbf{v_2}$ that lie on the original plane, to give the new rotation matrix $V^{'}$ as in (6.25).

$$V^{'} = \left[\frac{\mathbf{u_2}}{||\mathbf{u_2}||} \quad \frac{\mathbf{u_3}}{||\mathbf{u_3}||} \quad \mathbf{n}\right] \tag{6.25}$$

where

$$\begin{aligned}
\mathbf{u_1} &= \mathbf{n} \\
\mathbf{u_2} &= \mathbf{v_1} - \frac{\mathbf{v_1}\cdot\mathbf{n}}{\mathbf{n}\cdot\mathbf{n}}\mathbf{n} \\
\mathbf{u_3} &= \mathbf{v_2} - \frac{\mathbf{v_2}\cdot\mathbf{n}}{\mathbf{n}\cdot\mathbf{n}}\mathbf{n} - \frac{\mathbf{v_2}\cdot\mathbf{u_2}}{\mathbf{u_2}\cdot\mathbf{u_2}}\mathbf{u_2}
\end{aligned} \tag{6.26}$$

In the same manner, the origin of the original plane CF no longer lies on the new plane and so a new CF origin $\mathbf{p^{'}_{cog}}$ must be generated. This can be performed using (6.22) to give (6.27).

$$\mathbf{p^{'}_{cog}} = \mathbf{p_{cog}} - D\mathbf{n} \tag{6.27}$$

6.4.2 Merging of Projected Polygon Boundaries

The final step in the merging process is to merge the two polygon boundaries that have just been projected from their respective infinite planes onto the new merged planar surface. This process is performed in the same manner as the merging of the individual polygon boundaries from each 2D image to originally generate the planar surfaces, as described in Section 5.5.

To summarise this process briefly, each projected boundary is represented by a series of linked corner points lying on the 2D planar surface. Pairs of corners that are physically close are registered together and merged. The remainder of the boundary is merged by taking the union of the unmatched corners and edges and removing any internal or otherwise erroneous edges.

6.5 Surface Merging Results

The methods described in this chapter to merge the planar surfaces from multiple 3D scans are validated in this section. The experimental and simulated datasets described in Section 3.3, and the planar surfaces extracted according to Chapter 5, were used for this.

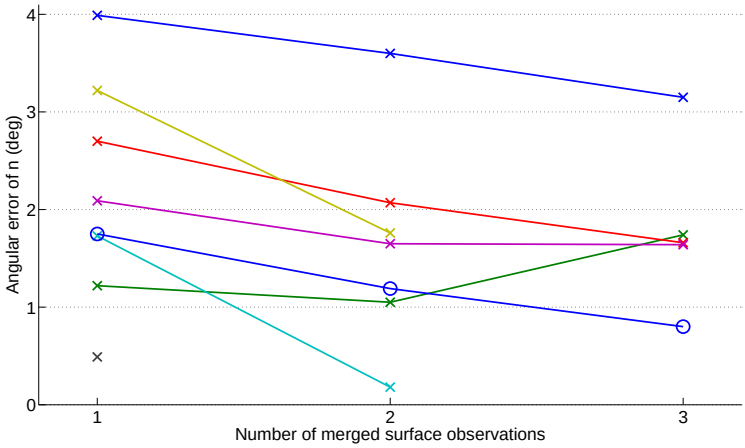
6.5.1 Results for Experimental Dataset

The results of merging the three 3D scans of the experimental dataset are shown below. These were found using the same methods as those used for the results of the individual 3D scans, as shown in Section 5.6.

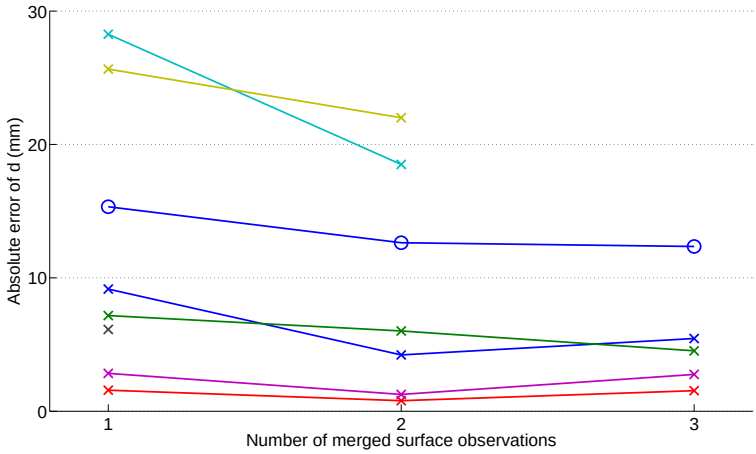
Experimental Plane Fitting Results

The plane parameter fitting errors are shown in Fig. 6.1 for the eight surfaces that were extracted in at least one of the real 3D scans. The change in error is shown as each new surface observation is merged into the current surface estimate. The change in the angular error of $\mathbf{n}$ in degrees is shown in Fig. 6.1(a) while the change

in the error of d in millimetres is shown in Fig. 6.1(b). The X axis depicts the number of observations of each surface.



(a) Angular error (°) of normal vector $\mathbf{n}$.



(b) Absolute error (mm) of distance d.

Figure 6.1: Change in plane parameters during merging (Experimental dataset).

Experimental Polygon Boundary Results

The mean corner location error, $\mu_{\Delta c}$ in millimetres, is shown in Table 6.1 after the three experimental 3D scans were merged together. This error value was found using the ground truth plane parameters to show the corner error independent from any plane fitting error. The mean Mahalanobis error $\mu_{\Delta c}$ (σ) is also shown in terms of the standard deviations. The average of the individual 3D scan mean corner errors is also shown for comparison.

Table 6.1: Real dataset - Polygon boundary corner error (true $\mathbf{n}$,d)

3D scan	Corners	$\mu_{\Delta c}$	$\mu_{\Delta c}$ (σ)
Average of Individual Scans	73	10.6mm	2.27
Merged Scan	39	8.3mm	2.06

The total corner error, including any errors from the plane parameters, was also found as in Table 6.2. This was found by projecting the corners onto the plane parameters fit to the raw data, rather than using the ground truth parameters as in Table 6.1 above. This error is compared to the total error from Scan A, as taken from Table 5.5.

Table 6.2: Real dataset - Polygon boundary corner error (fit $\mathbf{n}$,d)

3D Scan	Corners	$\mu_{\Delta c}$	$\mu_{\Delta c}$ (σ)
Scan A (found in Table 5.5)	22	14.7mm	1.88
Merged Scan	39	13.0mm	2.38

Experimental 3D Reconstruction Results

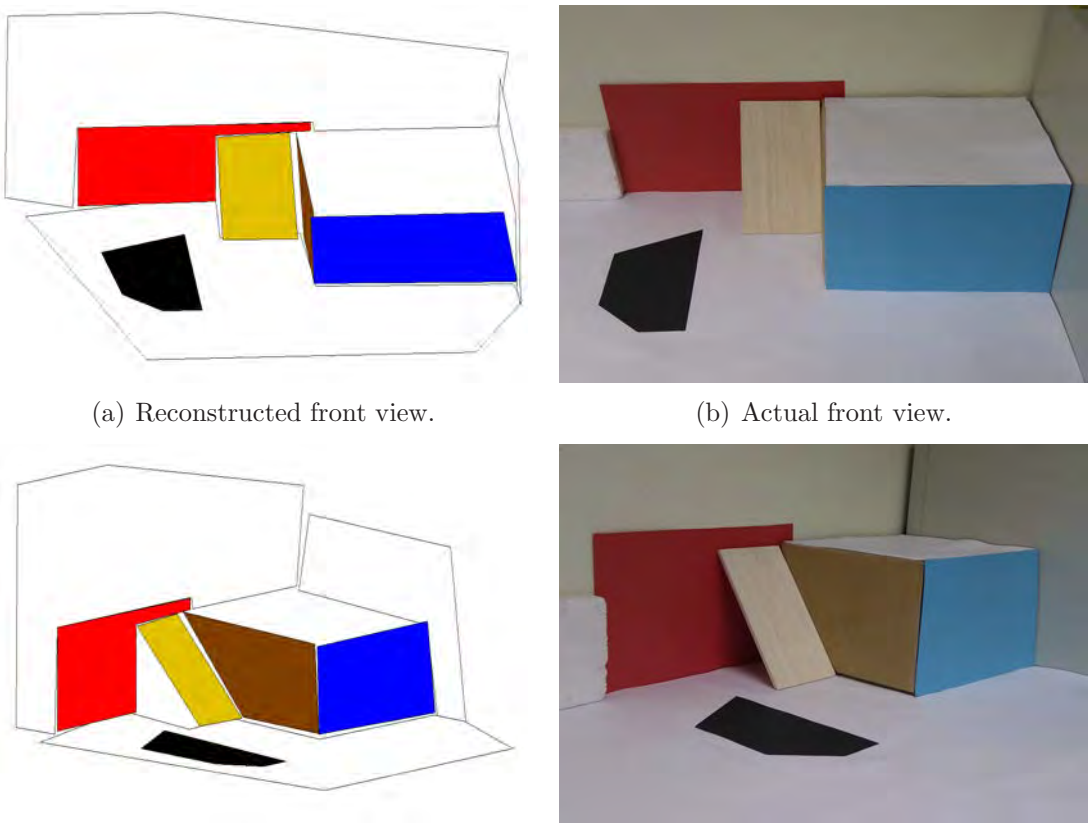
The 3D reconstruction of the local 3D map produced by merging the three 3D scans from the experimental dataset are shown in Fig. 6.2.

6.5.2 Results for Simulated Dataset

The results of merging the five 3D scans of the simulated dataset are shown below. These results were calculated using the same methods as those used for the individual 3D scans shown in Section 5.6. The same layout is followed as that of the experimental results shown in Section 6.5.1 above.

Simulated Plane Fitting Results

The plane parameter fitting errors are shown in Fig. 6.3 for the fourteen surfaces that were extracted in at least one simulated 3D scan. The change in error is shown as each new planar surface observation is merged into the current surface estimate. Note that the scale of the Y axis is different to that used in Fig. 6.1 and it is important to take this into account when making comparisons.



(a) Reconstructed front view.

(b) Actual front view.

(c) Reconstructed side view.

(d) Actual side view.

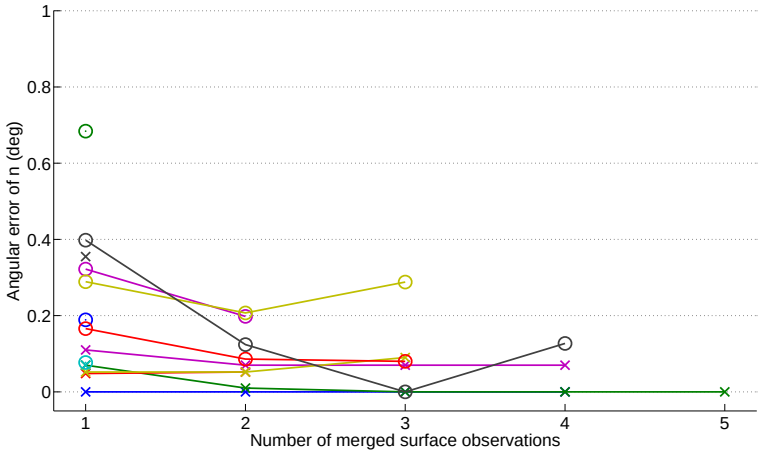
Figure 6.2: 3D reconstruction of merged experimental scene.

Simulated Polygon Boundary Results

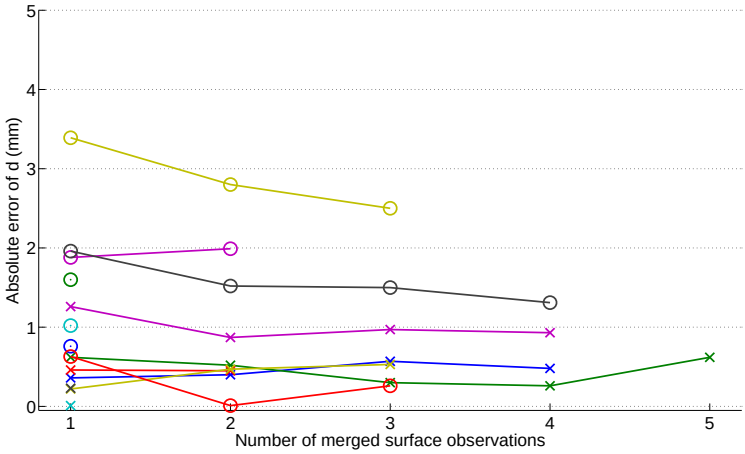
The mean corner location error is shown in Table 6.3 after the five simulated 3D scans have been merged together. The mean error of the five individual 3D scans is also shown for comparison. Unlike the experimental results shown in Section 6.5.1 above, there is no need to show the corner error using the fit parameters, as in Table 6.2. This would be redundant as the plane fitting error was negligible for this simulated dataset, as was seen in Fig. 6.3, and the corner error is virtually identical to that seen below in Table 6.3.

Table 6.3: Simulated dataset - Polygon boundary corner error (true $\mathbf{n}$,d)

3D Scan	Corners	$\mu_{\Delta c}$	$\mu_{\Delta c}$ (σ)
Average of Individual Scans	104	11.1mm	2.54
Merged Scan	56	11.1mm	2.69



(a) Angular error (°) of normal vector **n**.



(b) Absolute error (mm) of distance d.

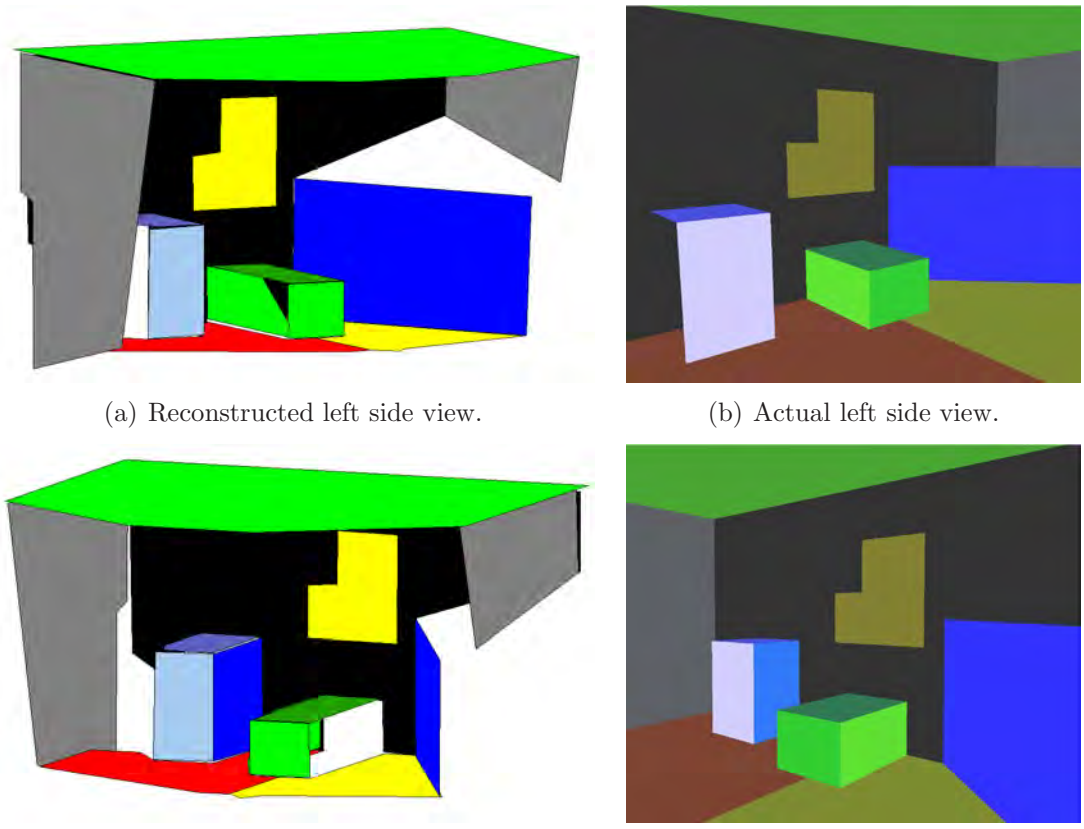
Figure 6.3: Change in plane parameters during merging (Simulated dataset).

Simulated 3D Reconstruction Results

The 3D reconstruction of the local 3D map produced by merging the five 3D scans from the simulated dataset are shown in Fig. 6.4.

6.6 Discussion

The merging of individual 3D scans was required to generate a 3D map about the WCR's current position. This map must contain the majority of planar surfaces from the environment and they need to be mapped accurately enough to allow the WCR to use them as footholds. It is necessary to determine if the generated merged



(a) Reconstructed left side view.

(b) Actual left side view.

(c) Reconstructed right side view.

(d) Actual right side view.

Figure 6.4: 3D reconstruction of merged simulated scene.

3D maps meet these requirements.

6.6.1 Merging of Planar Surfaces

For both experimental and simulated datasets, the merged 3D maps were better than the individual 3D scans, both qualitatively and quantitatively. This was seen as improvements to the field of view, the number of mapped surfaces and a reduction in occlusions, as well as an improvement in the accuracy of the planar surfaces. Almost all relevant planar surfaces were observed and had a planar surface fit to them, the accuracy of which was within the tolerances laid out in Section 3.2.1. The 3D maps are therefore good enough to allow the WCR to plan its next foothold location and navigate towards it. Such navigation and motion planning is, however, outside the scope of this work. Refer to the work of Ward [75] for a motion planning approach that was developed for the CRACbot.

Infinite Plane Fusion

The experimental 3D map reconstruction successfully mapped eight of the ten visible surfaces, as seen in Fig. 6.2. It can also be seen that the field of view is wider than the individual scans, shown in Figs. 5.18-5.20. The three individual scans each contained seven, five and eight of the ten surfaces respectively, however, several of these were only partially observed.

The accuracy of each planar surface plane in the merged 3D map showed general improvement, as seen in Fig. 6.1. This shows the change in the error of the $\mathbf{n}$ and d parameters of each plane after integrating each successive surface observation during the merging process. The generally decreasing trend is apparent and all but one surface had an angular error of less than 2°. All but two surfaces had a distance error of less than 15mm. This level of error is decent considering that they were calculated against ground truth values that were hand measured and so not known exactly. It is also within the tolerances required for footstep placement of $\pm 3^\circ$ and $\pm$20mm. The higher level of error in the d parameter was mostly caused by unmodeled systematic errors in the LRF range measurements, as mentioned in Chapter 5.

Similar results were seen for the simulated dataset and its 3D reconstruction of the merged 3D map, as in Fig. 6.4. The merged 3D map contained 14 out of the 15 visible surfaces and the level of occlusion was significantly less than that seen in the individual 3D scans, shown in Figs. 5.13-5.17. The plane parameter accuracy, as in Fig. 6.3, did not change significantly. The absolute accuracy is already excellent, due to the lack of unmodeled systematic biases, and so there is little room for the error to improve.

The plane fitting results for both datasets show that the basic KF merging approach, as described in Section 6.3, was sufficient to merge the planar surface observations. The proposed modifications effectively solved the two problems pertaining to the unit normalisation and the directional consistency of the normal vector.

Polygon Boundary Fusion

The polygon boundary fusion process followed the same method used to merge the boundaries of the individual scans, as detailed in Section 5.5. This method was found to work well for the individual scan boundary generation and similar results were also seen in the merging of the boundaries to generate the planar surfaces of the merged 3D map. This can be seen qualitatively in the 3D reconstructions shown in Figs. 6.2 and 6.4, which bear a good resemblance to the true scenes being mapped. The field of view of the polygon boundaries are also more complete than those seen in the individual 3D scans which were limited by of the camera field of view.

For the experimental dataset the merged corner error, from Table 6.1, was 8.3mm which was a 22% decrease from the mean error of the individual scans when using the ground truth plane parameters. When using the fit plane parameters, the corner error was 13.0mm (12% decrease). These show that the polygon boundary merging process is taking the corner uncertainty into account and providing a better estimate than using the average corner position. This is also a validation of the corner projection method, detailed in Section 6.4.1, to project each boundary onto the merged plane and calculate the new corner locations and uncertainties.

The simulated dataset merged corner error, shown in Table 6.3, was 11.1mm which was the same as the mean corner error of the individual 3D simulated scans. Due to the simplified nature of the simulated noise models, the level of uncertainty of each surface boundary was not significantly different. The merging process therefore weighted each corner location similarly and essentially provided the average corner position.

6.6.2 Usage of Planar Surfaces for SLAM

After the generation of a local 3D map, the WCR can use this environmental knowledge for navigation by some means. The map is locally accurate with low relative distortion between surfaces and this will allow the WCR to plan its motion towards the next foothold location. These surfaces are globally uncertain but this is not an

issue during motion planning. It is more important to be able to successfully move to the next foothold rather than knowing the exact global location of that foothold. For a WCR, being slightly lost is less of a concern than attempting to step onto a poor quality surface which could lead to a collision or fall.

After the footstep motion has finished, the global robot position can be estimated by a 3D SLAM implementation as odometry alone will no longer be sufficient to track the robot's pose. While the implementation of such a SLAM algorithm is outside the scope of this work, it remains to demonstrate that the planar surface maps generated in this thesis are able to be integrated within the planar SLAM approaches from the literature.

EKF SLAM using Planar Features

EKF planar SLAM uses planar surfaces as features as opposed to the point features that are more commonly used in SLAM. The work of Kohlhepp *et al.* [32], Weingarten *et al.* [33] and de la Puente *et al.* [36] in this area was introduced in Section 6.2. They all proposed similar approaches, with Weingarten [35] proposing to use the SPmap framework to maintain an estimate of the planar surfaces. This required the planar surface parameters $\mathbf{n}$ and d, as well as the local plane covariance. This information is readily available from the surface mapping approach of this thesis. Therefore, the planar surfaces from the local 3D map could be used within such an EKF SLAM framework with little modification. The ability of the proposed method to generate the surface uncertainty is especially useful because many plane fitting methods do not provide this and so they could not be used for SLAM.

Weingarten [35] also suggested using the polygon boundary information to aid in the data association and again this information is available. The surface colour and texture information from the colour camera could also be used to improve the robustness of the data association.

Planar Scan Matching SLAM

The planar scan matching approach of Pathak *et al.* [74], as described previously, registers planar surfaces together and then determines their change in pose between observations. Their method also used the Hessian plane parameters $\mathbf{n}$ and d, with corresponding uncertainties, and again this is the same plane representation used in this work.

The main requirement for successful registration using their MUMC approach [125] was that each 3D map needed to contain at least three nonparallel planar surfaces that could be matched. The local 3D maps of Figs. 6.2 and 6.4 each contained enough such surfaces for use within their MUMC registration and SLAM framework. This approach operates on the infinite plane parameters only and so could benefit from the use of polygon boundary information and surface colour. These are available for each planar surface extracted in this work and this could improve the speed and robustness of MUMC by reducing the search space during the data association phase.

6.7 Summary

This Chapter has detailed the creation of local 3D maps by merging together a number of individual 3D scans taken from the same WCR foothold position. These were extracted, as described in Chapter 5, and a methodology was proposed to merge together both the infinite plane parameters and the polygon boundaries. The resulting local 3D map showed an improvement in the field of view, a reduction in occlusions and increased accuracy when compared with the individual 3D scans.

The accuracy of the merged surfaces was within the allowable tolerances set out in Section 3.2.1 and over 80% of the visible planar surfaces were successfully mapped. It follows that these local 3D surface maps are sufficient to allow the CRACbot to locate and maneuver to a suitable nearby foothold. A 3D SLAM implementation could then be used at this point for longer term mapping and localisation. This

integration was discussed and found to be viable.

It now remains to show that the presented surface mapping methodology can build these 3D surface maps in real time. This is the focus of the following chapter.

Chapter 7

Performance Analysis

7.1 Introduction

The previous three chapters have detailed the proposed 3D mapping method. The accuracy and quality of the resulting planar surfaces have been experimentally validated and found to be sufficient to enable navigation and motion planning to occur by some means. This chapter analyses the computational performance of the proposed mapping method. This will determine the viability of real time operation using the limited computational resources available to a small wall climbing robot, such as the CRACbot.

The computational cost of each mapping component will be analysed theoretically and the running time will be found experimentally. This will show that real time operation is possible and also highlight any potential bottlenecks. Areas for improvement will be discussed with a particular focus on optimising the level of sparseness used during the data acquisition process.

7.2 Computational Performance

This section analyses the computational performance of the 3D surface mapping method presented in this thesis. The computational complexity of each component of the proposed mapping method will be theoretically analysed. This is followed by

runtime results for the experimental datasets which will then be compared to other planar surface fitting methods.

7.2.1 Computational Complexity Analysis

Feature Extraction

The extraction of image features uses the approach and implementation of He *et al.* [102]. A Canny edge detection stage is used which has a complexity of $O(uv)$ for an image consisting of $u \times v$ pixels. This is followed by a scale space corner detection stage to find the corners within each detected edge. This has a complexity of $O(n_{il})$ for an image containing n_{il} extracted image lines. For further information refer to Section 4.2.1. The edge detection complexity tends to dominate due to the quadratic nature of the image pixel count and this is a major reason why low resolution images were used.

The incremental 2D line segmentation of each LRF scan is $O(Sn_p^2)$ [106] for a scan consisting of n_p points and S extracted line segments. This quadratic complexity suggests that subsampling the LRF scans would provide large improvements in the runtime of this segmentation process. The level of sparseness used for feature extraction and its effect on runtime will be discussed further in Section 7.3.1.

Feature Grouping

The feature grouping process has a computational complexity of $O(n_{2D}^2 n_g n_{ll} n_{il})$ for a 3D scan consisting of n_{2D} 2D scans. The total number of feature groups is n_g and the average number of image line and laser line features per image are denoted n_{il} and n_{ll} respectively. This complexity is dominated quadratically by the number of 2D scans taken and so this feature grouping method performs better when a larger tilting angular step is used. This will be discussed further in Section 7.3.2.

Plane Fitting

PCA plane fitting using (5.4-5.8) is $O(n_p)$ when fitting to a set of n_p raw 3D data points. The calculation of the covariance of the plane parameters using (5.36-5.37) is also $O(n_p)$. So the plane fitting process is overall linear in complexity when fitting each individual plane. The overall complexity of the plane fitting process is $O(n_s n_p)$ when fitting n_s planar surfaces. Note that n_s is less than or equal to n_g as some feature groups may not be able to have a planar surface fitted to them.

Polygon Boundary Generation

The generation of polygon boundaries for each planar surface in each 3D scan has a complexity of $O(n_{2D} n_c^2)$ where n_c is the number of image corners on each polygon boundary. Due to this quadratic complexity in terms of n_c, the corner postprocessing techniques of Section 5.5.4 becomes especially important to remove redundant corners.

Merging of 3D Scans

The process of incrementally merging two 3D scans has two components. The fusion of the plane parameters is $O(n_s)$ in complexity while the fusion of the two polygon boundaries is $O(n_s n_c^2)$. If this process is performed as a batch operation, once n_{3D} individual 3D scans have been acquired, then the overall complexity is $O(n_{3D} n_s n_c^2)$.

7.2.2 Runtime Results

The total running time of the mapping method depends on the data acquisition and motion times along with the time taken to process the data to build each 3D surface map. For real time operation to be viable, the processing of each 3D scan should be equal to or less than the time taken to acquire it and move to the next position. If more time is required then the robot will have to pause to allow the computation to catch up and any delay of more than a few seconds is generally undesirable.

Runtime Results for Mapping Components

The running time of each proposed mapping component was found experimentally using the three 3D scans of the real dataset, as shown below in Table 7.1. These values were found using implementations of each mapping component written in C++ and MATLAB. A conservative factor of 3:1 was used to estimate the runtime of the MATLAB implementations when running in equivalent C++ code. All mapping components were run on a single core 2.7GHz Intel processor with 1GB of RAM.

The feature extraction required 79ms per 2D scan with a total of 711ms needed to process the nine 2D scans of each individual 3D scan. The feature grouping process required 6ms per group to give a total of 54ms for an average of nine groups. The plane fitting and polygon boundary generation runtime was 18ms per plane. This gave a runtime of 126ms for an average of seven planar surfaces for each 3D scan. The merging of the individual 3D scans to generate the local 3D map took 8ms per plane pair. These values are summarised in Table 7.1.

The CRACbot embedded computer is a GumStix Verdex Pro [11] with a processor clock speed of 600MHz. A factor of 1:8.1 was used to estimate the runtime of each mapping component when running in C++ code on the Verdex Pro. This value was found by taking into account benchmark information for the Verdex Pro, the relative clock speeds and differences in architecture and available memory. For further information about the Verdex Pro specifications and the assumptions used to arrive at the factor of 8.1, refer to Appendix A.1. This process was necessary as the code has not yet been fully converted and implemented for operation on the CRACbot at this time. The estimated runtimes are shown below in the final column of Table 7.1.

Data Acquisition Time

The data acquisition time is the product of the number of 2D scans to be acquired and the time taken to tilt the sensors to the next position and acquire the LRF and camera data. The CMUcam3 can acquire images at 26Hz (38.5ms per image). The

Table 7.1: Runtime for components of proposed mapping method

Mapping Component	2.7GHz PC Time (Average)	Verdex Time (Average)	Average Number	Verdex Time (Total)
Image Feature Extraction	59ms	478ms	9 2D scans	4.30s
LRF Feature Extraction	20ms	162ms	9 2D scans	1.46s
Feature Grouping	6ms	49ms	9 groups	0.44s
Plane Fitting	9ms	73ms	7 planes	0.51s
Polygon Boundary	9ms	73ms	7 planes	0.51s
3D Scan Merging	8ms	65ms	12 plane pairs	0.78s

Hokuyo URG-04LX LRF can acquire 2D range scans at 10Hz (100ms per scan). As the motor runs continuously it may require at worst case 199ms to acquire a scan, depending on the initial position of the LRF scanning motor when a scan is requested.

Both sensors were interfaced using a RS232 serial connection operating at a baud rate of 115,200bps. The protocol used was eight data bits and one stop bit which leads to an effective data transfer rate of 12.5KBps. The camera supplies a 16 bit raw RGB image with 50,512 pixels which equates to 99KB in size. This will require eight seconds to transfer each image in this format. The onboard hardware can compress the image to jpeg format to reduce the file size to 5-7KB (400-560ms transfer time) and approximately 200ms is required for this conversion to jpeg.

The 2D LRF scan consists of 682 range measurements, requiring 2 bytes each, to give a total size of 1.33KB. This will take just over 100ms to transfer . If intensity and the receiver gain value for each range measurement are also desired, then 320ms will be required to transfer this information. It should be noted that higher transfer rates are possible for the Hokuyo URG-04LX but are not implemented.

As both sensors can acquire their data simultaneously, a minimum time of 200ms is required at each 2D scan location after the joint has settled. Taking into account the transfer time of each sensor, a worst case minimum time between acquisition of consecutive scans is 800ms, which is predominantly limited by the camera transfer rate. This transfer can occur during the tilting motion to the next 2D scan position.

From Table 7.1 it was seen that the feature extraction process for each 2D scan

required an estimated 640ms. This is within the minimum data acquisition time required for each 2D scan of 800ms. It is therefore possible to extract the features in real time.

These data acquisition timings are summarised in Fig. 7.1 below. This diagram shows the relative timing of the acquisition and feature extraction for each 2D scan. The image feature extraction for 2D scan i is performed during the acquisition and transfer of the image from the next scan $(i + 1)$ to prevent the need for a pause. Note also that these times were estimates only and the conversion factors used were conservative. It is likely that the feature extraction process speed could be further improved when converted from MATLAB to optimised C++ code.

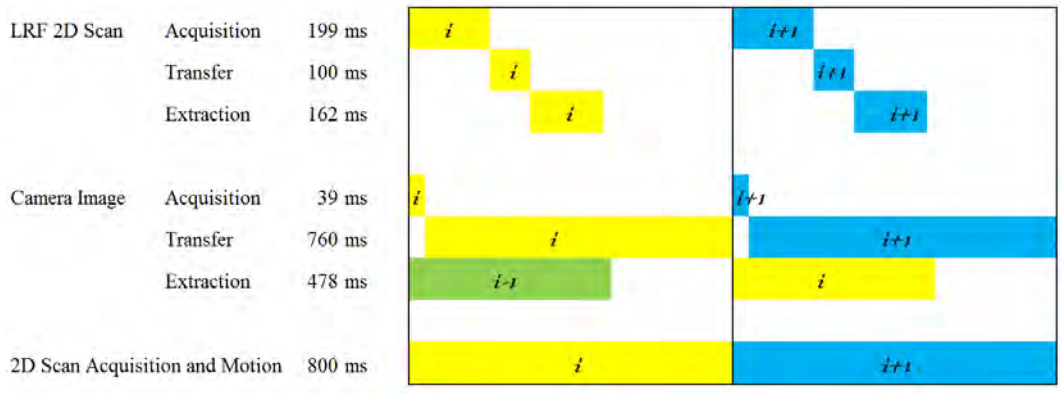


Figure 7.1: Data acquisition timing diagram.

3D Scan Acquisition Time

The tilting joint on the CRACbot has a speed of 3RPM which equates to an angular velocity of 0.31 radians (18°) per second. The CRACbot joint controller was designed and analysed by Szwec [77]. Based on this information, and the motor time constant of 9ms, an estimated 100ms should be allowed for speed up and settling time per tilting movement. The time required to acquire each 2D scan is thus 300ms, including 200ms for data acquisition, plus the time taken to tilt through the desired angular step to arrive at the next 2D scan tilt angle.

Possible scanning times are shown below in Table 7.2. A number of tilting

angular steps that are commonly used in the literature (0.5° and 1.0°), as well as the angular steps used in this thesis (5.0° and 10.0°), are shown. The time taken to acquire a 3D scan which tilts through a range of 40°, as used in the experimental dataset, is shown along with the number of 2D scans required. These scanning times are, however, limited by the minimum time required for the transfer of the raw sensor data, which is 800ms.

Table 7.2: 3D scan acquisition time

Tilting Angular Step	2D DAQ Time	Tilting Time	Total Time per 2D Scan	40° Scan Time	2D Scans Required
0.5°	300ms	28ms	800ms (328)	65s	81
1.0°	300ms	56ms	800ms (356)	33s	41
5.0°	300ms	278ms	800ms (578)	7.2s	9
10.0°	300ms	556ms	856ms	4.3s	5

From Table 7.1 it was seen that the feature grouping, plane fitting and polygon boundary generation processes required an estimated 1.5s for each 3D scan. This is currently performed after the acquisition of the entire 3D scan, as most of the time available during data acquisition is required for feature extraction. This time is comparable to the time required to move to the next 3D scan pose and so no pause is required to complete the mapping. If these surfaces are needed to plan the next 3D scanning pose, then a small pause of less than two seconds will be required before continuing.

3D Scan Memory Usage

Each 2D scan consists of a set of LRF range measurements and a camera image. As mentioned above, the camera image requires 5-7KB when converted to jpeg format, while each LRF scan requires 1.33KB when subsampling is not used. Assuming these worst case values, each 40° 3D scan, consisting of nine 2D scans, will require a maximum of 75KB of storage space for the raw data. Once each image has been used for feature extraction it can then be deleted. This reduces the memory requirement to 19KB for the storage of nine LRF scans and the latest camera image to operate on.

Once feature extraction has been performed then the raw image data is no longer needed. The image corner features themselves consist of a list of (u,v) coordinate pairs and a second list of edges that join corners together. A typical set of features consisting of 40 corners and edges will require approximately 160 bytes. The LRF segment information is negligible in size, as it is simply a list of the endpoint indexes with each 2D LRF scan. In total, the image and LRF features can typically be represented by 200 bytes per scan to give a total of 1.8KB per 3D scan.

Each planar surface requires 32 floating points values to store the plane parameters, covariances, origin and rotation matrix. This gives a total of 128 bytes per plane. The average polygon boundary consists of 10 corners, each requiring 6 floating point values for their 2D location and covariance. This requires an additional 260 bytes, once the edge information is also included, to give a total of 388 bytes per planar surface. Assuming an average of seven planes per 3D scan, the total memory required to store each 3D surface scan is 2.7KB. At this point the raw LRF data is no longer needed.

Overall the process of fitting planar surfaces to each 3D scan requires the temporary storage of less than 20KB throughout the whole mapping process. Only 3KB are needed to store the resulting planar surface patches for an average 3D scan.

Local 3D Map Acquisition Time

The time taken to acquire a local 3D map about a particular foothold location depends on a number of factors but several assumptions can be used to estimate this time. It is assumed that the 3D scan poses do not differ significantly. This allows odometry to provide the change in pose, as assumed in Section 6.2.1, as well as ensuring some level of overlap between scans. Another benefit of this assumption is that the robot can operate with less environmental knowledge if it keeps its movements short between observations. Practically, this assumption limits the number of joints moving to two and those joints are limited to moving through a maximum of 90°. At a speed of 3RPM this will take 4.6 seconds, including the 100ms allowance

for settling time.

It is also assumed that each local 3D map consists of a maximum of five individual 3D scans and so the total time for the acquisition of each local 3D map is 40 seconds (10.0° tilt step) to 54 seconds (5.0° tilt step). From Table 7.1 it was seen that the 3D scan merging process required 65ms to merge each planar surface pair within a set of individual 3D scans. Assuming an average of twelve matched planar surface pairs, this merging will take an estimated 780ms to merge a new 3D scan into the local 3D map. This is well within the data acquisition and motion time for each new 3D scan.

7.2.3 Discussion of the Computational Performance

The results and estimates of the mapping runtimes showed that the proposed mapping approach can operate in real time when using the limited onboard computational power of the Verdex Pro. The estimated runtimes, for onboard operation of each mapping component, were all within their respective data acquisition and motion time frames. However, it should be noted that these times were estimates only, as the code has not yet been fully implemented on the onboard Verdex Pro processor. Nevertheless, even if these estimates are out by a factor of 2, real time operation is still achievable if short pauses in the order of a few seconds are acceptable.

The data acquisition of each 2D scan was limited by the transfer speed of the camera images. The maximum available baud rate is 115,200 bps for the CMUcam3 and despite the low resolutions used, these images can take approximately 800ms to acquire, convert to jpeg format and then transmit to the Verdex Pro board. It was mentioned in Section 3.4.1 that the CMUcam3 could be interchanged with any low resolution camera. Using a different camera with a faster serial or USB transfer rates would significantly improve the speed of the data acquisition process.

In general, the acquisition of each 3D scan is still significantly faster than dense scanning approaches. The number of 2D scans required is an order of magnitude less when scanning sparsely. This was seen in Table 7.2 where a dense scan at an

angular step of 0.5° required over a minute for acquisition while the sparse scans required only 5-7 seconds. This allows for both faster scanning and faster mapping, as less data is acquired and operated on.

Performance Compared with Literature

It has been shown that the proposed mapping approach is capable of generating planar 3D surface maps in real time. It is beneficial to compare this performance against other 3D planar mapping approaches to ascertain any relative advantages or disadvantages. The mapping approaches used for comparison are those of Zureiki *et al.* [37], Pathak *et al.* [46] along with Weingarten *et al.* [72] who proposed both the Grid Based Segmentation (GBS) and Region Growing Segmentation (RGS) approaches. Where appropriate, their runtimes will be scaled in a similar manner to that used to generate the estimated Verdex Pro runtimes. This will allow for fairer comparisons. These times are summarised in Table 7.3 below.

Table 7.3: Running time for mapping methods in the literature

Planar Mapping Approach	Processor Speed	Tilt Range	Point Cloud	Run Time	Verdex Time
Weingarten [72] - GBS	1.4GHz	270°	58k	4.5s	19s
Pathak [46]	1.6GHz	180°	195k	6.4s	31s
Weingarten [72] - RGS	1.4GHz	270°	58k	14s	59s
Zureiki [37]	3.0GHz	97°	61k	10s	90s
Proposed Method - 40° Tilt Range	2.7GHz	40°	2.5k	0.9s	7s
Proposed Method - 90° Tilt Range	2.7GHz	90°	5k	1.9s	15s

It is difficult to make quantitative comparisons between the different methods due to variations in factors such as the angular tilting range and the amount of plane fitting and polygonalisation performed. Despite this, it can be seen that the proposed mapping method compares favourably and performs at least as fast, if not faster than the other mapping methods in the literature. It also operates on significantly smaller input datasets. This is important for embedded operation where memory and storage space are limited, but this is not factored into the running time estimates. All the other methods were also fitting 3D maps from ground based

robots using large or heavy sensors. Therefore they would not be directly suitable for use on a small wall climbing robot undertaking 3D motion.

7.3 Discussion of the Level of Sparseness

The use of sparse input data from the LRF and camera is a key component of the mapping method proposed in this thesis. It is important to investigate the effect on the performance of the proposed mapping method when the level of sparseness is varied. This sparseness originates from both the sensors themselves and the tilting angular step between each 2D scan. By analysing each source of sparseness carefully it will be possible to provide clues as to the best levels of sparseness to use.

7.3.1 Sensor Sparseness

Camera Pixel Resolution

The resolution of the camera images influences the running time in several ways. The image acquisition and transfer time, which was shown to be the main bottleneck in the data acquisition process, along with the image feature extraction process are both significantly faster operations when using lower resolution images. These processes are both quadratic in complexity for an image consisting of $u \times v$ pixels.

This justifies the use of low resolution images in this thesis which were only 287 $\times$ 176 pixels (0.05MP) in size. The use of such images provides improvements in runtime of approximately 6$\times$ over VGA images (640$\times$480=0.3MP) and 26$\times$ over XVGA images (1280$\times$1024=1.3MP). The storage space and memory required to store and operate on each image are also reduced by a similar factor. This is another advantage when using limited onboard computational resources.

There is a tradeoff when using low resolution images in terms of reduced accuracy. The image resolution limits the achievable accuracy of the reconstructed polygon boundary corners to the size of half a pixel. However, it is only one of several factors that contribute to the boundary corner error. Other factors include the quality of

the camera calibration and the accuracy of the feature extraction process.

Increasing the image resolution would not provide a significant improvement in accuracy but would come at the cost of a large increase in runtime. For example, if the image resolution was doubled in both the vertical and horizontal directions, the runtime would increase fourfold due to the quadratic complexity of the image feature extraction operations. The corner error would, however, not halve as the influence of the other error sources has remained unchanged.

The image resolution should therefore be as low as possible while still maintaining a reasonable level of error. If the resolution is too low then the corner error becomes dominated by the image resolution. The ideal pixel resolution should provide a level of corner error that is similar in magnitude to the error provided by the other sources which are independent of the image resolution. Lower resolutions would begin to dominate the polygon boundary corner error while higher resolutions would greatly increase the runtime with little improvement to the corner error.

For the low resolution (0.05MP) CMUcam3, a surface corner at a range of 500mm would have a minimum error of ± 2mm. This corresponds to half a pixel in each of the horizontal and vertical directions. From Table 5.4 it can be seen that the average corner reconstruction error was 10.6mm. The majority of this error was not caused by the image resolution and so higher resolution images would provide a negligible improvement in the corner accuracy.

LRF Angular Resolution

The angular resolution of the 2D LRF scans influence the accuracy and running time of the plane fitting and line segmentation operations. Lower scanning resolutions will supply less raw data points. This will decrease the plane fitting runtime as the calculation of the plane parameters and covariance are both $O(n_p)$ in complexity for a 2D scan consisting of n_p points. The incremental 2D line segmentation is $\mathrm{O}(Sn_p^2)$ and would benefit even more from a lower scanning angular resolution, or subsampling of the acquired data.

The Hokuyo URG-04LX used an angular resolution of 0.36° and this was the default setting. At this resolution a surface at a range 500mm and 0° angle of incidence will be intersected by adjacent laser beams that are approximately 3mm apart. The proposed surface mapping method does not use the LRF data to detect the surface boundaries and so decreasing the angular resolution will provide large improvements in performance at negligible cost to the accuracy of the planar surfaces.

The limiting factor to reducing the angular resolution of each scan is the size of the objects in the scene. If the surfaces are small in size, then fewer beams will intersect them. This may cause some surfaces to be missed or inaccurately segmented if the resolution is too low, as the influence of the range noise becomes more significant. Nevertheless, such small surfaces are likely to be unsuitable as footholds and so the accuracy of their extraction is of less concern. An angular resolution of 1.08° (subsampling every third measurement) would improve the plane fitting speed threefold and the 2D line segmentation speed ninefold while having little effect on the surface accuracy.

7.3.2 Tilting Angle Sparseness

The second source of sparseness in this mapping approach is introduced by the tilting angular step between the acquisition of each 2D scan. In this thesis a step of 5-10° was used. This is an order of magnitude larger than the 0.5-1.0° step that is commonly used for dense 3D range scanning in other robotic mapping approaches.

This sparseness influences the plane fitting process in two ways. Halving the angular step will double the number of raw data points n_p used for plane fitting. This will result in the time taken to calculate the plane parameters and covariances to double as both processes are linear in complexity. However, there will be negligible improvement in their accuracy. A plane can be fitted to a minimum of two line segments from different angular steps and this will provide the fastest plane fit. Further line segments provide little extra improvement to the plane accuracy in general.

Having said that, if the tilting angular step is too large, then some surfaces will not be intersected twice and thus cannot have a planar surface fit to them. Assuming optimal feature extraction and grouping, the ideal tilting angular step should intersect each surface in the environment twice but no more. In practice this should be relaxed somewhat to provide extra intersections to allow for grouping and extraction failures.

This ideal tilting angular step should also take into consideration the vertical FoV of the camera. The CMUcam3 has a vertical FoV of $\pm 15°$. This means that each surface point is seen in seven consecutive images for a tilting step of 5°. Each surface feature needs to be successfully extracted from at least one image. However, if it is observed several times, then the accuracy can be improved through the boundary merging process. In general, two to three observations of each corner are sufficient to generate a good quality polygon boundary. In this case, the tilting step could be increased to 10° increments (four images per feature) if the size of the surface objects allow the LRF to intersect each one at least twice.

7.3.3 Adaptive Tilting Angle

As discussed in the previous section, the ideal angular tilting step is largely dependent on the size of the surface objects being scanned. A large angular step has significant performance benefits but risks missing smaller objects in the scene. The solution to this problem may be to introduce an adaptive tilting step that can attempt to determine the size of the objects in the scene and adjust the size of the angular step accordingly. This would involve using the current camera image to find the largest angular step for the next 2D scan that will intersect any currently grouped surface that still requires a second LRF line segment for plane fitting.

For example, when scanning large surfaces, such as a wall, the angular step can be maximised to the vertical FoV of the camera (15° for the CMUcam3). As soon as smaller objects are encountered, the angular step can be reduced to acquire the necessary two LRF line segments. This would significantly speed up the data acqui-

sition and surface fitting operations but would introduce some extra computation to implement the adaptive step planner.

Another approach is to use a small constant angular step to ensure that smaller objects are observed sufficiently but during surface fitting the number of observations actually used could be limited. It was previously mentioned that plane fitting requires a minimum of two LRF line segments for successful fitting. Further line segment observations generally provide little improvement to the accuracy of the planar surface but significantly increase the fitting time. Thus large surfaces may produce many line segments of which most are largely redundant.

By using the most accurate two or three line segments, a faster plane fit could be found with little sacrifice in accuracy. A similar idea could be used after a sufficient number of corner observations have been extracted and merged to generate the polygon boundary of a particular surface. The position of that surface region could be estimated and ignored in subsequent images to speed up the extraction of the image features.

Both these approaches have the potential to reduce significantly the runtime of the proposed mapping approach. They have not been implemented at this time and are presented here as future work.

7.4 Summary

This chapter has discussed and analysed the computational performance of the proposed planar surface fitting method. Experimental results have shown that real time operation is achievable when operating on the limited computational resources available to the CRACbot. This performance was also compared to the speed of other similar planar surface mapping approaches. It was found to perform at least as fast, if not faster, than these other methods.

The degree of sparseness was analysed for both the raw sensor data and the angular step between each 2D scan. It was determined that a significant improvement in computational performance could be achieved by optimising the level of sparse-

ness used. The image resolution could be decreased further and subsampling the LRF scans could speed up the mapping performance with negligible effect on the accuracy of the planar surfaces. The tilting angular step size could also be modified adaptively to scan small objects sufficiently whilst also scanning larger objects the minimum number of times.

Chapter 8

Conclusions and Future Work

A wall climbing robot is able to move in full three dimensional (3D) space by attaching itself to any suitable surface within reach. These robots have the potential for use in areas such as the inspection, surveillance and maintenance of environments that are hazardous or inaccessible to ground based robots or people.

This thesis has presented research into a methodology for the creation of 3D surface maps to enable a small wall climbing robot with limited computational resources to map and navigate in real time. Using experimental results from realistic surfaces, the presented 3D mapping method was shown to produce accurate planar surfaces using sparse scanning and data acquisition from both a laser range finder and a colour camera. These planar surfaces were found to be sufficient for use as footholds by a wall climbing robot and this will facilitate motion planning and navigation. Real time operation was also shown to be achievable.

The main contributions of the research presented in this thesis are summarised in this chapter. Potential areas and prospects for future work are also discussed.

8.1 Thesis Contributions

This thesis has provided a unique contribution to the field of robotics by allowing small wall climbing robots to build planar surface maps for navigation in real time. The main contributions are summarised below.

8.1.1 Planar Surface Fitting

A wall climbing robot requires knowledge of nearby surfaces to navigate through its environment. The CRACbot uses suction cups to attach itself to surfaces and therefore only planar surfaces are suitable to act as footholds. The basis of the surface mapping approach in this thesis was a method for planar surface fitting to construct a 3D map using the fusion of range and image information. This was detailed in Chapter 5 and consisted of two steps. Infinite planes were fitted to 3D points extracted from laser range finder scans while edge features from camera images were used to provide polygon boundary information for each surface.

Due to the unique nature of this surface mapping approach, a method to merge the polygon boundary information from several images was needed. Section 5.5 presented an approach to handle this merging and construct the planar polygon boundaries. The concept of virtual corners and lines was introduced. These virtual image features were generated at the physical image boundary. They allowed closed polygons to be generated for surfaces that were not fully observed due to limitations in the camera field of view. The registration and merging of each combination of real and virtual features were presented along with post processing stages to clean up the generated polygon boundaries. This process allowed for multiple images to improve the boundary accuracy and extend the field of view, despite the simplicity of the registration and merging used. This provides additional benefits in terms of computational performance.

The novel approach presented for the fusion of range and image information allowed for the use of sparse scanning and low resolution data acquisition. This contrasts with the normal paradigm in robotic mapping of densely scanning the environment and attempting to fit highly detailed surface models. The proposed planar surface fitting methods can therefore be performed quickly and in real time using the limited computational resources available to a small wall climbing robot. The amount of data being operated on was orders of magnitude less when compared to the dense point clouds used by most other robotic 3D mapping approaches. This

will allow the presented mapping approach to be generalised for use by other small or miniature mobile robots that have similar computational limitations.

8.1.2 Surface Feature Extraction

It was found that for many plane fitting methods, the planar segmentation of the raw 3D data was slow and computationally expensive. An alternative approach was proposed in Chapter 4. This was based on the extraction of features from the range and image data that could then be grouped together to provide the basis for the plane fitting process.

While the extraction of image features used standard edge and corner extraction algorithms, the extraction of range features from the 2D LRF scans relied on significant modifications to the standard incremental segmentation algorithm. Firstly, image line features were used to pre-split the raw range scan. Secondly, the adjacent points were not added to the line segment set until several consecutive points were all found to belong to the estimated line segment. Thirdly, the two endpoints of each segment were ignored during plane fitting. These modifications showed an improvement in the accuracy of the planar surfaces fitted to the line segments by significantly reducing the rate of false positive segmentations.

8.1.3 Feature Grouping

The extracted features were of little use individually and therefore some means for grouping them together was required before planar surfaces could be fitted. A method was proposed in Section 4.4 to group together image and range features generated by the same physical surface. This was required as no previous method existed that could handle the sparseness of the scanning and data acquisition used in this approach. The grouping method was designed to be conservative and the results showed that only 2% of features were incorrectly grouped. This low rate is important as false associations can lead to the generation of inaccurate planar surfaces during the plane fitting stage.

8.1.4 Accuracy and Uncertainty of Planar Surfaces

The calculation of uncertainty information for each planar surface was required for both the merging of planar surfaces and the determination of suitable and reliable foothold locations. The plane parameter covariance equations were fully derived in Section 5.4.2 by extending the approach of Weingarten *et al.* [118]. It was shown that by performing these calculations in the plane coordinate frame, the covariance equations simplified dramatically. Most of the sum and product terms equated to zero and proofs were shown to confirm this. Using these simplified equations allowed for faster computation of the plane uncertainty. This is beneficial when computational resources are limited, as is the case with the CRACbot or other similar small robots.

For the experimental dataset, the error of each planar surface fit in Chapter 5 was less than 3° in orientation and less than 30mm in distance. All but one surface had an error of less than 2° in orientation and less than 15mm in distance. The surface orientation accuracy was therefore good enough for use by the CRACbot but the distance accuracy was borderline for several surfaces. This was caused by the range errors from the laser range finder and a more complicated sensor model could be used to improve this, albeit at a higher computational cost.

Overall, this level of surface accuracy indicates that the planar surfaces fitted using the proposed approach are sufficient to allow a wall climbing robot to successfully use them as footholds. The accuracy was also comparable to that of other surface fitting approaches, despite the sparseness of the scanning used. The mean corner error of the merged planar polygon boundaries was found to be 8mm. This polygon boundary accuracy is rarely shown in other methods and so comparisons are difficult to make. This level of error does, however, appear to be very reasonable considering the low resolution of the images used to generate them.

8.1.5 Real Time Operation

The generation of a 3D map must be performed in real time for a wall climbing robot as it cannot move without a surface map of some kind. The computational hardware that a wall climbing robot can carry onboard is often severely limited due to payload restrictions. A lightweight algorithm was needed as most dense 3D mapping approaches generate a large amount of raw data which is slow to acquire, store and operate on.

The proposed surface mapping approach was shown in Chapter 7 to be lightweight and fast because of the usage of sparse scanning, low resolution data acquisition and feature extraction. Real time operation using the 600MHz processor onboard the CRACbot was shown to be viable. Each mapping component was able to be processed within the time required for the relevant data acquisition and motion. This minimises the need for the robot to pause and wait for the 3D map to be built which is preferable for speed and battery life considerations.

8.1.6 3D Map Construction for Navigation

Individual planar surfaces are of little use by themselves for navigation. It was shown in Chapter 6 that it was possible to merge surface observations from nearby 3D scans to build a local 3D surface map about a wall climbing robot's current foothold position. This would allow the robot to use this information to locate a suitable nearby foothold and move towards it successfully. Any implementation of such navigation and motion planning is, however, outside the scope of this work.

A standard Kalman Filter was used to merge the planar surface parameters. Several simple modifications were proposed and implemented to ensure that the surface normal vector direction and sign remained consistent throughout the merging process. Previous planar merging approaches noted these problems but used alternative surface merging approaches instead. It is beneficial if a Kalman Filter can be used directly for surface merging. The proposed modifications allowed its usage while also maintaining the correctness of the normal vector. This is important

because the Kalman Filter is easy to use and is a well understood estimator.

The use of the generated planar surfaces within planar 3D SLAM frameworks from the literature was discussed and analysed. It was found that they could be used, with little modification, within both feature based and scan matching planar SLAM approaches. This will allow for longer term navigation and map building which is an important part of autonomous operation for a wall climbing robot.

8.2 Future Work

The planar surface mapping method presented in this thesis opens up new avenues for research and development. Areas of potential future work are introduced below along with possible refinements to the method to validate it further and improve its robustness.

8.2.1 Full Implementation on CRACbot Prototype

The CRACbot was the target robotic system and motivation for the surface mapping approach presented in this thesis. It was not however, fully operational at the time of writing. The proposed methods were instead validated using an experimental test rig which simulated the robot scanning motion. The surface mapping approach was shown to be fast enough to allow for real time usage. This will need to be fully validated using the CRACbot once it is operational. This will confirm that the robot can map successfully its environment and move between footholds in real time. The use of the generated planar surfaces within a 3D SLAM framework was discussed in Section 6.6.2 and found to be viable. This also needs to be implemented on the CRACbot to provide a long term navigation capability.

8.2.2 Estimation of Surface Properties

The presented mapping approach was able to provide information regarding each planar surface in the map, but this was limited to colour and texture only. The

ability to provide richer surface information, such as surface type or material, would be very useful for the end user robot. This could allow a more detailed sensor model to correct for material specific range biases from the LRF scans. Another benefit could be to allow a wall climbing robot to avoid surface materials that are known to provide poor suction. The availability of both range and vision data provides the opportunity for further exploration of this concept.

8.2.3 Integration of Nonplanar Surfaces

Currently, the surface mapping is limited to planar surfaces. Nonplanar surfaces are unsuitable as footholds for the CRACbot and so have not been the focus of this work. While several methods were detailed in Sections 4.3.4 and 4.4.5 to detect and filter out any features generated by nonplanar surfaces, no representation was detailed to handle them within the current framework. This is important for obstacle avoidance and visualisation purposes. Possible extensions to accommodate nonplanar surfaces include the addition of spheres, cylinders or piecewise planar surfaces to represent them.

8.2.4 Optimisation of Sparseness

Section 7.3 analysed the sparseness of both the sensor data and the tilting angular resolution. It was found that by optimising the level of sparseness used, the computational performance of the mapping process could be improved further with little cost to the surface accuracy. Several suggestions were proposed including the introduction of an adaptive tilting angular step to achieve this. It also appears that even lower resolution images and LRF scans could be used in some situations. Further investigation is warranted to determine the level of possible improvement to the speed of the approach and any resulting loss in accuracy.

References

[1] W. Yan, L. Shuliang, X. Dianguo, Z. Yanzheng, S. Hao, and G. Xueshan, "Development and application of wall-climbing robots," *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, vol. 2, pp. 1207–1212 vol.2, 1999.

[2] J. Xiao, A. Sadegh, M. Elliott, A. Calle, A. Persad, and H. M. Chiu, "Design of mobile robots with wall climbing capability," *Advanced Intelligent Mechatronics. Proceedings, 2005 IEEE/ASME International Conference on*, pp. 438 – 443, 2005.

[3] H. Tang, Z. Zhu, and J. Xiao, "Stereovision-based 3D planar surface estimation for wall-climbing robots," *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pp. 97 –102, 10-15 2009.

[4] M. Taylor, X. Chen, M. Lang, T. McKee, J. Robertson, and S. Aston, "Tigbot - a wall climbing robot for tig welding of stainless steel tanks," *Mechatronics and Machine Vision in Practice, 2008. M2VIP 2008. 15th International Conference on*, pp. 550 –554, 2-4 2008.

[5] C. Menon and M. Sitti, "A biomimetic climbing robot based on the gecko," *Journal of bionic engineering*, vol. 3, no. 3, pp. 115–125, 2006.

[6] H. Prahlad, R. Pelrine, S. Stanford, J. Marlow, and R. Kornbluh, "Electroadhesive robots: wall climbing robots enabled by a novel, robust, and electrically controllable adhesion technology," *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pp. 3028–3033, 2008.

[7] M. Minor, H. Dulimarta, G. Danghi, R. Mukherjee, R. Tummala, and D. Aslam, "Design, implementation, and evaluation of an under-actuated miniature biped climbing robot," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 31, pp. 1999–2005, 2000.

[8] J. Ward and J. Katupitiya, "Free space mapping and motion planning in configuration space for mobile manipulators," *Robotics and Automation, 2007 IEEE International Conference on*, pp. 4981–4986, 2007.

[9] J. Casper and R. Murphy, "Human-robot interactions during the robot-assisted urban search and rescue response at the world trade center," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 33, no. 3, pp. 367 – 385, 2003.

[10] W. Meiting, T. Shili, D. Junjian, and Y. Liwen, "Complete coverage path planning of wall-cleaning robot using visual sensor," *Electronic Measurement and Instruments, 2007. ICEMI '07. 8th International Conference on*, pp. 4–159 –4–164, aug. 2007.

[11] Gumstix, "Gumstix, 'http://www.gumstix.com/'," 2010.

[12] J. Xiao, "Cooperative wall-climbing robots in 3D environments for surveillance and target tracking," tech. rep., City College of the City University of New York, 2009.

[13] T. Balch, "Avoiding the past: A simple but effective strategy for reactive navigation," pp. 678–685, IEEE, 1993.

[14] R. Arkin, "Integrating behavioral, perceptual, and world knowledge in reactive navigation," *Robotics and Autonomous Systems*, vol. 6, no. 1-2, pp. 105–122, 1990.

[15] R. Smith and P. Cheeseman, "On the representation and estimation of spatial uncertainty," *The international journal of Robotics Research*, vol. 5, no. 4, p. 56, 1986.

[16] R. Smith, M. Self, and P. Cheeseman, "Estimating uncertain spatial relationships in robotics," *Autonomous Robot Vehicles*, vol. 1, pp. 167–193, 1990.

[17] H. Durrant-Whyte, "Uncertain geometry in robotics," *Robotics and Automation, IEEE Journal of*, vol. 4, pp. 23 –31, feb 1988.

[18] M. Dissanayake, P. Newman, S. Clark, H. 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, 2001.

[19] H. Durrant-Whyte and T. Bailey, "Simultaneous localisation and mapping (slam): Part i the essential algorithms," *Robotics and Automation Magazine*, vol. 13, pp. 99–110, 2006.

[20] S. Thrun, "Simultaneous localization and mapping," *Robotics and cognitive approaches to spatial mapping*, pp. 13–41, 2008.

[21] J. Guivant and E. Nebot, "Optimization of the simultaneous localization and map-building algorithm for real-time implementation," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 3, pp. 242–257, 2001.

[22] J. Leonard and H. Feder, "A computationally efficient method for large-scale concurrent mapping and localization," *Robotics Research - International Symposium*, vol. 9, pp. 169–178, 2000.

[23] R. Martinez-Cantin and J. Castellanos, "Unscented slam for large-scale outdoor environments," pp. 3427–3432, IEEE, 2005.

[24] L. Ellekilde, S. Huang, J. Valls Miro, and G. Dissanayake, "Dense 3D map construction for indoor search and rescue," *Journal of Field Robotics*, vol. 24, no. 1-2, p. 71, 2007.

[25] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "Fastslam: A factored solution to the simultaneous localization and mapping problem," *Proceedings of the AAAI National Conference on Artificial Intelligence*, pp. 593–598, 2002.

[26] P. Besl and H. McKay, "A method for registration of 3D shapes," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 14, no. 2, pp. 239–256, 1992.

[27] J. Nieto, J. Guivant, and E. Nebot, "Denseslam: Simultaneous localization and dense mapping," *International Journal of Robotics Research*, vol. 25, no. 8, pp. 711–744, 2006.

[28] S. Se, D. Lowe, and J. Little, "Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks," *The International Journal of Robotics Research*, vol. 21, no. 8, p. 735, 2002.

[29] D. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[30] A. Davison, I. Reid, N. Molton, and O. Stasse, "Monoslam: Real-time single camera slam," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2007.

[31] O. Stasse, A. Davison, R. Sellaouti, and K. Yokoi, "Real-time 3D slam for humanoid robot considering pattern generator information," *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pp. 348–355, 2006.

[32] P. Kohlhepp, P. Pozzo, M. Walther, and R. Dillmann, "Sequential 3D-slam for mobile action planning," *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, vol. 1, 2004.

[33] J. Weingarten and R. Siegwart, "Ekf-based 3D slam for structured environment reconstruction," *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*, pp. 3834–3839, 2005.

[34] J. Castellanos, J. Montiel, J. Neira, and J. Tardos, "The spmap: a probabilistic framework for simultaneous localizationand map building," *Robotics and Automation, IEEE Transactions on*, vol. 15, no. 5, pp. 948–952, 1999.

[35] J. Weingarten, *Feature-Based 3D SLAM.* PhD thesis, École Polytechnique Fédérale de Lausanne, 2006.

[36] P. de la Puente, D. Rodriguez-Losada, A. Valero, and F. Matia, "3D feature based mapping towards mobile robots' enhanced performance in rescue missions," *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pp. 1138 –1143, 10-15 2009.

[37] A. Zureiki, M. Devy, and R. Chatila, "Slam and data fusion from visual landmarks and 3D planes," *Proc. 17th IFAC World Congress*, 2008.

[38] H. Surmann, A. Nüchter, and J. Hertzberg, "An autonomous mobile robot with a 3D laser range finder for 3D exploration and digitalization of indoor environments," *Robotics and Autonomous Systems*, vol. 45, no. 3-4, pp. 181–198, 2003.

[39] A. Nuchter, H. Surmann, K. Lingemann, J. Hertzberg, and S. Thrun, "6d slam with an application in autonomous mine mapping," *Robotics and Automation, IEEE International Conference on*, vol. 2, 2004.

[40] A. Nuchter, "Parallel and cached scan matching for robotic 3D mapping," *Journal of Computing and Information Technology*, vol. 17, no. 1, pp. 51–65, 2009.

[41] D. Borrmann, J. Elseberg, K. Lingemann, A. Nüchter, and J. Hertzberg, "Globally consistent 3D mapping with scan matching," *Robotics and Autonomous Systems*, vol. 56, no. 2, pp. 130–142, 2008.

[42] P. Biber and W. Straßer, "The normal distributions transform: A new approach to laser scan matching," *2003 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings*, vol. 3, 2003.

[43] M. Magnusson, A. Lilienthals, and T. Duckett, "Scan registration for autonomous mining vehicles using 3D-ndt," *Journal of Field Robotics*, vol. 24, no. 10, pp. 803–827, 2007.

[44] D. Viejo and M. Cazorla, "3D plane-based egomotion for slam on semi-structured environment," *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pp. 2761–2766, 2007.

[45] A. Harati and R. Siegwart, "Orthogonal 3D-slam for indoor environments using right angle corners," *Proc. of 3rd European Conf. on Mobile Robots (ECMR)*, 2007.

[46] K. Pathak, N. Vaskevicius, J. Poppinga, M. Pfingsthorn, S. Schwertfeger, and A. Birk, "Fast 3D mapping by matching planes extracted from range sensor point-clouds," *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pp. 1150 –1155, 10-15 2009.

[47] D. Hahnel, W. Burgard, and S. Thrun, "Learning compact 3D models of indoor and outdoor environments with a mobile robot," *Robotics and Autonomous Systems*, vol. 44, no. 1, pp. 15–27, 2003.

[48] I. Mahon and S. Williams, "Three-dimensional robotic mapping," *Australasian Conference on Robotics and Automation*, 2003.

[49] I. S. Kweon, R. Hoffman, and E. Krotkov, "Experimental characterization of the perceptron laser rangefinder," Tech. Rep. CMU-RI-TR-91-01, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, January 1991.

[50] J. Saez and F. Escolano, "A global 3D map-building approach using stereo vision," *Proceedings of the 2004 IEEE International Conference on Robotics and Automation*, pp. 1197–1202, 2004.

[51] S. May, B. Werner, H. Surmann, and K. Pervolz, "3D time-of-flight cameras for mobile robotics," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2006.

[52] J. Underwood, S. Scheding, and F. Ramos, "Real-time map building with uncertainty using colour camera and scanning laser," *Proceedings of the 2007 Australasian Conference on Robotics and Automation*, 2007.

[53] V. Sequeira, J. Goncalves, and M. Ribeiro, "3 d environment modelling using laser range sensing," *Rob Autom Syst*, vol. 16, no. 1, pp. 81–91, 1995.

[54] K. Pervolz, A. Nuchter, H. Surmann, and J. Hertzberg, "Automatic reconstruction of colored 3d models," *Proc. Robotik 2004*, 2004.

[55] C. Frueh, S. Jain, and A. Zakhor, "Data processing algorithms for generating textured 3D building facade meshes from laser scans and camera images," *International Journal of Computer Vision*, vol. 61, no. 2, pp. 159–184, 2005.

[56] Y. Bok, Y. Hwang, and I. Kweon, "Accurate motion estimation and high-precision 3D reconstruction by sensor fusion," *Robotics and Automation, IEEE International Conference on*, pp. 4721–4726, 2007.

[57] J. Diebel and S. Thrun, "An application of markov random fields to range sensing," *Proceedings of Conference on Neural Information Processing Systems (NIPS), Cambridge, MA, USA*, 2005.

[58] H. Andreasson, R. Triebel, and A. Lilienthal, "Vision-based interpolation of 3D laser scans," *Proc. of ICARA*, pp. 469–474, 2006.

[59] P. Dias, V. Sequeira, F. Vaz, and J. Goncalves, "Registration and fusion of intensity and range data for 3D modelling of real world scenes," *3D Digital Imaging and Modeling, Fourth International Conference on*, pp. 418–425, 2003.

[60] W. Jiang and J. Lu, "Panoramic 3D reconstruction by fusing color intensity and laser range data," *Robotics and Biomimetics, IEEE International Conference on*, pp. 947–953, 2006.

[61] H. Moravec, "Sensor fusion in certainty grids for mobile robots," *AI Magazine*, vol. 9, no. 2, pp. 61–74, 1988.

[62] H. Moravec, "Robot spatial perception by stereoscopic vision and 3D evidence grids," *CMU Robotics Institute Technical Report CMU-RI-TR-96-34*, 1996.

[63] J. Ryde and H. Hu, "3D mapping with multi-resolution occupied voxel lists," *Autonomous Robots*, vol. 28, no. 2, pp. 169–185, 2010.

[64] P. Pfaff, R. Triebel, and W. Burgard, "An efficient extension to elevation maps for outdoor terrain mapping and loop closing," *The International Journal of Robotics Research*, vol. 26, no. 2, p. 217, 2007.

[65] R. Triebel, P. Pfaff, and W. Burgard, "Multi-level surface maps for outdoor terrain mapping and loop closing," *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2276–2282, 2006.

[66] A. Birk, K. Pathak, N. Vaskevicius, M. Pfingsthorn, J. Poppinga, and S. Schwertfeger, "Surface representations for 3D mapping," *KI-K ”unstliche Intelligenz*, pp. 1–6, 2010.

[67] Y. Liu, R. Emery, D. Chakrabarti, W. Burgard, and S. Thrun, "Using em to learn 3D models of indoor environments with mobile robots," *IEEE International Conference on Machine Learning (ICML)*, 2001.

[68] R. Lakaemper and L. Latecki, "Extended em for planar approximation of 3D data," *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pp. 1173–1179, 2006.

[69] H. Andreasson, R. Triebel, and W. Burgard, "Improving plane extraction from 3D data by fusing laser data and vision," *Intelligent Robots and Systems, International Conference on*, pp. 2656–2661, 2005.

[70] J. Poppinga, N. Vaskevicius, A. Birk, and K. Pathak, "Fast plane detection and polygonalization in noisy 3D range images," *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pp. 3378–3383, Sept. 2008.

[71] P. de la Puente, D. Rodriguez-Losada, R. Lopez, and F. Matia, "Extraction of geometrical features in 3D environments for service robotic applications," *Hybrid Artificial Intelligence Systems*, pp. 441–450, 2008.

[72] J. Weingarten, G. Gruener, and R. Siegwart, "A fast and robust 3D feature extraction algorithm for structured environment reconstruction," *International Conference on Advanced Robotics, Coimbra, Portugal, July*, 2003.

[73] M. Fischler and R. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[74] K. Pathak, A. Birk, N. Vaskevicius, M. Pfingsthorn, S. Schwertfeger, and J. Poppinga, "Online three-dimensional slam by registration of large planar surface segments and closed-form pose-graph relaxation," *Journal of Field Robotics*, vol. 27, no. 1, pp. 52–84, 2010.

[75] J. Ward, *Motion Planning of Bipedal Wall Climbing Robots*. PhD thesis, Mechanical and Manufacturing Engineering, University of New South Wales, Sydney, Australia, 2009.

[76] B. King, "Design of a wall climbing robot," tech. rep., Undergraduate thesis, School of Mechanical and Manufacturing Engineering, University of NSW, 2008.

[77] A. Szwec, "Development of a micro wall climbing robot finalising mechanical design and electronic integration," tech. rep., Undergraduate thesis, School of Mechanical and Manufacturing Engineering, University of NSW,, 2009.

[78] CMUcam, "Cmucam3 wiki - 'http://www.cmucam.org/'," Sep 2009.

[79] S. Thrun, "Robotic mapping: A survey," *Exploring artificial intelligence in the new millennium*, pp. 1–35, 2002.

[80] M. Hebert, "Active and passive range sensing for robotics," *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, vol. 1, 2000.

[81] M. Adams, "Lidar design, use, and calibration concepts for correct environmental detection," *Robotics and Automation, IEEE Transactions on*, vol. 16, no. 6, pp. 753–761, 2000.

[82] C. Ye and J. Borenstein, "A novel filter for terrain mapping with laser rangefinders," *Robotics, IEEE Transactions on [see also Robotics and Automation, IEEE Transactions on]*, vol. 20, no. 5, pp. 913–923, 2004.

[83] W. Boehler, M. Bordas Vicent, and A. Marbs, "Investigating laser scanner accuracy," *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 34, pp. 696–701, 2003.

[84] C. Ye and J. Borenstein, "Characterization of a 2d laser scanner for mobile robot obstacle negotiation," *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, vol. 3, 2002.

[85] H. Kawata, A. Ohya, S. Yuta, W. Santosh, and T. Mori, "Development of ultra-small lightweight optical range sensor system," *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*, pp. 1078–1083, 2005.

[86] L. Kneip, F. Tache, G. Caprari, and R. Siegwart, "Characterization of the compact hokuyo urg-04lx 2d laser range scanner," *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pp. 1447–1454, May 2009.

[87] Y. Okubo, C. Ye, and J. Borenstein, "Characterization of the hokuyo urg-04lx laser rangefinder for mobile robot obstacle negotiation," *Proceedings of SPIE*, vol. 7332, p. 733212, 2009.

[88] K. Lee and R. Ehsani, "Comparison of two 2d laser scanners for sensing object distances, shapes, and surface patterns," *Computers and Electronics in Agriculture*, vol. 60, no. 2, pp. 250–262, 2008.

[89] J. Pascoal, L. Marques, and A. de Almeida, "Assessment of laser range finders in risky environments," *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pp. 3533–3538, Sept. 2008.

[90] H. Surmann and R. Worst, "New applications with lightweight 3D-sensors," *VDI Berichte*, vol. 1956, p. 171, 2006.

[91] J. Weingarten, G. Gruener, and R. Siegwart, "A state-of-the-art 3D sensor for robot navigation," *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, vol. 3, 2004.

[92] X. Brun and F. Goulette, "Modeling and calibration of coupled fish-eye ccd camera and laser range scanner for outdoor environment reconstruction," *Proceedings of the Sixth International Conference on 3D Digital Imaging and Modeling (3DIM 2007)-Volume 00*, pp. 320–327, 2007.

[93] D. Ortin, J. Neira, and J. Montiel, "Relocation using laser and vision," *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, vol. 2, 2004.

[94] H. Yoshitaka, K. Hirohiko, O. Akihisa, and Y. Shin'ichi, "Mobile robot localization and mapping by scan matching using laser reflection intensity of the sokuiki sensor," *IEEE Industrial Electronics, IECON 2006-32nd Annual Conference on*, pp. 3018–3023, 2006.

[95] I. Stamos and P. Allen, "Integration of range and image sensing for photo-realistic 3D modeling," *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, vol. 2, 2000.

[96] R. Kurazume, K. Nishino, Z. Zhang, and K. Ikeuchi, "Simultaneous 2d images and 3D geometric model registration for texture mapping utilizing reflectance attribute," *Fifth Asian Conference on Computer Vision*, vol. 1, pp. 99–106, 2002.

[97] A. Carballo, Y. Hara, H. Kawata, T. Yoshida, A. Ohya, and S. Yuta *Multisensor Fusion and Integration for Intelligent Systems, 2008. MFI 2008. IEEE International Conference on*, pp. 261–266, 2008.

[98] Q. Zhang and R. Pless, "Extrinsic calibration of a camera and laser range finder (improves camera calibration)," *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, vol. 3, 2004.

[99] G. Li, Y. Liu, L. Dong, X. Cai, and D. Zhou, "An algorithm for extrinsic parameters calibration of a camera and a laser range finder using line features," *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pp. 3854–3859, 2007.

[100] D. Scaramuzza, A. Harati, and R. Siegwart, "Extrinsic self calibration of a camera and a 3D laser range finder from natural scenes," *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pp. 4164–4169, 2007.

[101] J. Canny, "A computational approach to edge detection," *Readings in computer vision: issues, problems, principles, and paradigms*, vol. 184, 1987.

[102] X. He and N. Yung, "Corner detector based on global and local curvature properties," *Optical Engineering*, vol. 47, p. 057008, 2008.

[103] R. Horaud and T. Skordas, "Stereo correspondence through feature grouping and maximal cliques," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, no. 11, pp. 1168–1180, 1989.

[104] P. McLauchlan, X. Shen, A. Manessis, P. Palmer, and A. Hilton, "Surface-based structure-from-motion using feature groupings," *Proceedings of the Fourth Asian Conference on Computer Vision*, pp. 699–705, 2000.

[105] K. Arras, "An introduction to error propagation," *Swiss Federal Institute of Technology Lausanne, Tech. Rep*, 1998.

[106] V. Nguyen, S. Gächter, A. Martinelli, N. Tomatis, and R. Siegwart, "A comparison of line extraction algorithms using 2d range data for indoor mobile robotics," *Autonomous Robots*, vol. 23, no. 2, pp. 97–111, 2007.

[107] A. Siadat, A. Kaske, S. Klausmann, M. Dufaut, and R. Husson, "An optimized segmentation method for a 2d laser-scanner applied to mobile robot navigation," *Proceedings of the 3rd IFAC symposium on intelligent components and instruments for control applications*, 1997.

[108] G. Borges and M. Aldon, "A split-and-merge segmentation algorithm for line extraction in 2d range images," *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, vol. 1, 2000.

[109] A. Harati and R. Siegwart, "A new approach to segmentation of 2d range scans into linear regions," *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2007. IROS 2007*, pp. 2083–2088, 2007.

[110] S. Pfister, S. Roumeliotis, and J. Burdick, "Weighted line fitting algorithms for mobile robot map building and efficient data representation," *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, vol. 1, pp. 1304–1311 vol.1, Sept. 2003.

[111] R. Mohan and R. Nevatia, "Using perceptual organization to extract 3D structures," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, pp. 1121–1139, 1989.

[112] Y.-L. Chang and J. K. Aggarwal, "Line correspondences from cooperating spatial and temporal grouping processes for a sequence of images," *Computer Vision and Image Understanding*, vol. 67, no. 2, pp. 186 – 201, 1997.

[113] X. Shen and P. Palmer, "Uncertainty propagation and the matching of junctions as feature groupings," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, pp. 1381 –1395, dec. 2000.

[114] L. Iocchi, K. Konolige, and M. Bajracharya, "Visually realistic mapping of a planar environment with stereo," *Proceesings of the 2000 International Symposium on Experimental Robotics*, 2000.

[115] P. Newman and K. Ho, "Slam-loop closing with visually salient features," *Robotics and Automation, 2005. Proceedings of the 2005 IEEE International Conference on*, pp. 635–642, 2005.

[116] P. Moreels and P. Perona, "Evaluation of features detectors and descriptors based on 3D objects," *International Journal of Computer Vision*, vol. 73, no. 3, pp. 263–284, 2007.

[117] X. Jiang and H. Bunke, "Fast segmentation of range images into planar regions by scan line grouping," *Machine Vision and Applications*, vol. 7, no. 2, pp. 115–122, 1994.

[118] J. Weingarten, G. Gruener, and R. Siegwart, "Probabilistic plane fitting in 3D and an application to robotic mapping," *Robotics and Automation, IEEE International Conference on*, vol. 1, 2004.

[119] K. Pearson, "On lines and planes of closest fit to systems of points in space," *Philosophical Magazine Series 6*, vol. 2, no. 11, pp. 559–572, 1901.

[120] J. Shlens, "A tutorial on principal component analysis," *Systems Neurobiology Laboratory, University of California at San Diego*, 2005.

[121] P. de Groen, "An introduction to total least squares," *Arxiv preprint math/9805076*, 1998.

[122] K. Pathak, N. Vaskevicius, and A. Birk, "Revisiting uncertainty analysis for optimum planes extracted from 3D range sensor point-clouds," *IEEE International Conference on Robotics and Automation, Kobe, Japan*, 2009.

[123] K. Pathak, A. Birk, J. Poppinga, and S. Schwertfeger, "3d forward sensor modeling and application to occupancy grid based sensor fusion," pp. 2059–2064, IEEE, 2007.

[124] J. Neira and J. Tardós, "Data association in stochastic mapping using the joint compatibility test," *Robotics and Automation, IEEE Transactions on*, vol. 17, no. 6, pp. 890–897, 2001.

[125] K. Pathak, A. Birk, N. Vaskevicius, and J. Poppinga, "Fast registration based on noisy planes with unknown correspondences for 3D mapping," *Robotics, IEEE Transactions on*, vol. 26, pp. 424 –441, june 2010.

[126] D. Simon, "Kalman filtering with state constraints: a survey of linear and nonlinear algorithms," *Control Theory & Applications, IET*, vol. 4, no. 8, pp. 1303–1318, 2010.

[127] L. Wang, Y. Chiang, and F. Chang, "Filtering method for nonlinear systems with constraints," vol. 149, pp. 525–531, IET, 2003.

[128] H. Weber and G. Arfken, *Essential mathematical methods for physicists*. Academic Pr, 2004.

[129] Z. Zhang, "Flexible camera calibration by viewing a plane from unknown orientations," *International Conference on Computer Vision 99*, vol. 1, pp. 666–673, 1999.

[130] Bouguet, "Camera calibration toolbox for matlab - 'http://www.vision.caltech.edu/bouguetj/calib_doc/'," Sep 2010.

[131] SICK, "Sick inc, 'http://www.sick.com/'," 2009.

[132] Hokuyo, "Hokuyo automatic co ltd, 'http://www.hokuyo-aut.jp/'," 2009.

[133] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics. 2005*. MIT Press, 2005.

[134] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer, *et al.*, "Autonomous driving in urban environments: Boss and the urban challenge," *The DARPA Urban Challenge*, pp. 1–59, 2009.

[135] M. Hebert and E. Krotkov, "3D measurements from imaging laser radars: how good are they?," *Intelligent Robots and Systems' 91.'Intelligence for Mechanical Systems, Proceedings IROS'91. IEEE/RSJ International Workshop on*, pp. 359–364, 1991.

[136] A. Reina and J. Gonzales, "Characterization of a radial laser scanner for mobile robotnavigation," *Intelligent Robots and Systems, 1997. IROS'97., Proceedings of the 1997 IEEE/RSJ International Conference on*, vol. 2, 1997.

[137] J. Lang and D. Pai, "Bayesian estimation of distance and surface normal with a time-of-flight laser rangefinder," *3D Digital Imaging and Modeling, 1999. Proceedings. Second International Conference on*, pp. 109–117, 1999.

List of Figures

3.1 Wall climbing robot - Current prototype. 30
3.2 Scanning motion for data acquisition. Sensors tilt through angle θ about the X axis. 33
3.3 Experimental setup for data acquisition. 35
3.4 Experimental (Real) scenario used for data acquisition. 36
3.5 Simulated scenario used for data acquisition. 37
3.6 Extrinsic calibration parameters for sensor suite. 43

4.1 Image feature extraction process. 49
4.2 Graphical representation of image features. The image corners are nodes and the image lines are directed edges. 52
4.3 2D line segmentation example. The next three points are tested before integrating and refitting the line. 62
4.4 2D line segmentation process - Splitting criteria. 62
4.5 Projection of laser line onto image and use of image lines for pre-splitting. 64
4.6 Image line pre-splitting - General case with sensor offset. 66
4.7 Post segmentation merging criteria. 67
4.8 Scenes used for laser line segmentation testing. 70
4.9 Linearity of extracted line segments. 73
4.10 Orthogonal and radial projections of a LRF point onto a 2D line. . . 76
4.11 3D left handed coordinate frame with sensors located at point O. . . 79
4.12 Feature grouping process. 85
4.13 Example of grouped features generated by the same physical surface from consecutive 2D scans. 86
4.14 Colour descriptors for features. 88
4.15 Laser-Image feature grouping process. 89
4.16 Image-Image feature grouping process. 94
4.17 Image-Image feature grouping termination conditions. 95
4.18 Test environments. 99

5.1 Overview of surface fitting methodology. 110
5.2 Hessian plane model for infinite plane with planar polygon boundary shown in red. 114
5.3 2D Example of aligning OLS with the orthogonal PCA fit. 119
5.4 Example of polygon boundary merging with uncertainty ellipses. . . . 136
5.5 Examples of real and virtual image features. 137
5.6 Matching of a corner to a line. 139
5.7 Matching of a virtual corner to a line. 141
5.8 Polygon boundary simplification by removing erroneous lines. 145
5.9 Examples of filling gaps in the polygon boundary. 146
5.10 Image corner angle calculation. 146
5.11 Polygon boundary simplification by removing redundant corners. . . . 147
5.12 Test environments with surface numbering. 149
5.13 3D reconstruction of simulated scan #1 for qualitative comparison. . 154
5.14 3D reconstruction of simulated scan #2 for qualitative comparison. . 154
5.15 3D reconstruction of simulated scan #3 for qualitative comparison. . 155
5.16 3D reconstruction of simulated scan #4 for qualitative comparison. . 155
5.17 3D reconstruction of simulated scan #5 for qualitative comparison. . 156
5.18 3D reconstruction of real scan #1 for qualitative comparison. 156
5.19 3D reconstruction of real scan #2 for qualitative comparison. 157
5.20 3D reconstruction of real scan #3 for qualitative comparison. 157
5.21 3D reconstruction of Validation scene 1a - No Clutter. 158
5.22 3D reconstruction of Validation scene 1b - Moderate Clutter. 158
5.23 3D reconstruction of Validation scene 1c - Substantial Clutter. 159
5.24 3D reconstruction of Validation scene 2a - Basic Walls. 159
5.25 3D reconstruction of Validation scene 2b - Shadows and Texture. . . . 160
5.26 3D reconstruction of Validation scene 3a - Outdoor Walls. 160
5.27 3D reconstruction of Validation scene 3b - Outdoor Bricks. 161
5.28 3D reconstruction of Validation scene 3c - Outdoor Container. 161

6.1 Change in plane parameters during merging (Experimental dataset). . 189
6.2 3D reconstruction of merged experimental scene. 191
6.3 Change in plane parameters during merging (Simulated dataset). . . 192
6.4 3D reconstruction of merged simulated scene. 193

7.1 Data acquisition timing diagram. 204

A.1 Wall climbing robot prototypes. 241
A.2 Kinematic configuration of the CRACbot. 241
A.3 CAD model of CRACbot. 242

List of Tables

3.1 3D scan poses for experimental dataset. 36
3.2 3D scan poses for simulated dataset. 38

4.1 Image feature extraction results 50
4.2 Range segmentation results - Simulated dataset 71
4.3 Range segmentation results - Real dataset #1 71
4.4 Range segmentation results - Real dataset #2 (nonplanar surfaces) . 71
4.5 Line segmentation - Linearity verification 74
4.6 Range segmentation results with nonlinear verification 75
4.7 Surface grouping results (Simulated dataset) 100
4.8 Individual feature grouping results (Simulated dataset) 100
4.9 Surface grouping results (Experimental dataset) 100
4.10 Individual feature grouping results (Experimental dataset) 101

5.1 Simulated dataset - Plane parameter fitting error 150
5.2 Real dataset - Plane parameter fitting error 150
5.3 Simulated dataset - Polygon boundary corner error 152
5.4 Real dataset - Polygon boundary corner error 153
5.5 Polygon boundary corner error - Fit vs true plane parameters 153

6.1 Real dataset - Polygon boundary corner error (true $\mathbf{n}$,d) 190
6.2 Real dataset - Polygon boundary corner error (fit $\mathbf{n}$,d) 190
6.3 Simulated dataset - Polygon boundary corner error (true $\mathbf{n}$,d) 191

7.1 Runtime for components of proposed mapping method 203
7.2 3D scan acquisition time 205
7.3 Running time for mapping methods in the literature 208

A.1 CRACbot power consumption 243
A.2 Camera specifications. 245
A.3 Camera intrinsic parameters 246
A.4 LRF specifications 247

List of Algorithms

4.1 Standard incremental segmentation. 62
4.2 Modified incremental segmentation. 63

Appendices

Appendix A

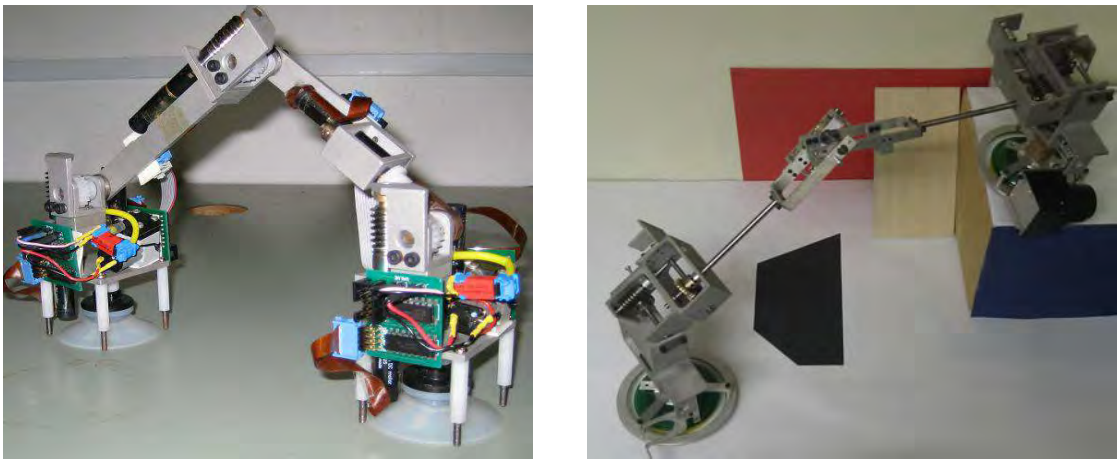
Hardware Specifications

A.1 CRACbot Specifications

This appendix details the target robot system to provide more detailed specifications than those outlined in Section 3.2. The mechanical design and choice of motors will be discussed. An analysis of power consumption and battery life is also provided along with details of the onboard computational resources.

The Compact Remote Articulating Climbing Robot (CRACbot) described in this thesis is a custom built wall climbing robot. It is the second generation prototype and has evolved from the first generation robot designed by Ward [75]. This was designated Hirudo and is seen in Fig. A.1(a). As such the major design decisions have already been made. The bipedal configuration was originally chosen as it provides a more agile wall climbing robot compared with wheeled configurations. Suction cups were chosen as they are a well known adhesion technology and can attach to a wide variety of surface materials.

The main improvements developed for the CRACbot, shown in Fig. A.1(b), include the addition of onboard electronics and sensors as well as improvements to the joint mechanisms, gearboxes and feet designs. The majority of this redesign was performed by King [76] and Szwec [77] in collaboration with the author.



(a) Hirudo (missing umbilical cord). (b) CRACbot (some hardware missing).

Figure A.1: Wall climbing robot prototypes.

Mechanical Design

The CRACbot has a bipedal design with a total of seven revolute joints providing motion in six DOF, as seen in Fig. A.2 below. This provides a large amount of flexibility and agility that allows it to scan the environment from any desired pose within its workspace. A 3D CAD model of the CRACbot is shown in Fig A.3.

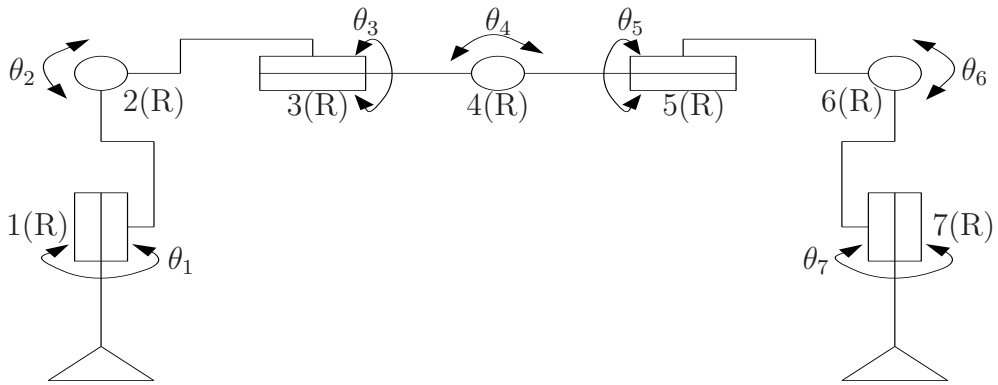


Figure A.2: Kinematic configuration of the CRACbot.

The robot consists of two legs joined at a hip joint and attaches to surfaces via a suction cup on each foot. The robot is 760mm long when fully extended. Each foot is circular with an 120mm diameter annulus surrounding the suction cup. This annulus improves the surface attachment of the foot and reduces the chance of the suction cup being peeled off the wall when subjected to high moments.

Figure A.3: CAD model of CRACbot.

The CRACbot has a total mass of 3.0kg when fully equipped. This includes a payload of 500g consisting of the sensors and computational hardware. The maximum allowable payload varies depending on mass distribution as well as the intended movements of the robot. If certain worst case movements are avoided then a higher payload can be carried.

Motors

Each joint is powered by a 12V motor. Two sizes are used because the central joints require less torque than those on each foot, which only have to support half the weight of the robot in the worst case. The chosen motor for these joints is the EC-16 from Maxon Motors (`www.maxonmotor.com.au`). It is a 12V brushless DC motor with a 16mm output shaft which will run at 60RPM. The larger motors are the EC-22 which will run at 50RPM. Each motor also contains an absolute encoder to allow for position control feedback and is driven by a brushless electronic speed controller.

Each motor has a worm gear on its output shaft with a 20:1 reduction ratio to generate the necessary output torque. The worm gear also allows the joints to be self-locking as the torque cannot feed back from the joint to the motor. This allows for great reductions in power usage as the motors do not have to be running continuously to maintain their position. This results in output speeds of 2.5-3.0RPM

for each joint, which is equivalent to an angular velocity of 0.044-0.052 radians per second (15-18° per second).

Power

The onboard hardware is powered by an 11.1 Lithium Polymer (LiPo) battery cell with a capacity of 3000 mAh. The operating time can be found by adding up the power consumption of the individual components. The main components are the seven motors, the sensors and the remaining electronics and vacuum system. The EC-22 motor requires 12W, while the EC-16 motor requires 5W for operation. These motors are not in continual usage and so their average power consumption is much lower.

It is assumed that the robot spends half its time acquiring 3D scans and the other half moving. During 3D scan acquisition only one of the EC-22 motors is in operation and only one of each sensor is used. During robot motion it is assumed that each motor is running 50% of the time. This leads to a operating time of 50% for the sensors (one LRF and one camera) and 25% for six motors with the remaining EC-22 motor operating at 75%. This is summarised in table A.1 below.

Based on these assumptions about operating duty cycles, the average power consumption is 30.3 W. The LiPo battery has a capacity of 33.3 Whours and so this leads to an average operating time of 66 minutes.

Table A.1: CRACbot power consumption

Component	Power Consumption	Duty Cycle	Average Power Consumption
Motors - Motion	63 W	25%	15.8 W
Motors - Scanning	12 W	50%	6 W
Sensors - Motion	3 W	0%	0 W
Sensors - Scanning	3 W	50%	1.5 W
Other Components	7 W	100%	7 W
TOTAL			30.3 W

Computation

The CRACbot is currently controlled by an onboard embedded computer consisting of a Gumstix Verdex Pro board [11] and two expansion boards. The Verdex Pro runs at a clock speed of 600MHz with 128MB of RAM and runs Linux as its operating system. It has 32MB of flash memory and an SD card slot for additional storage. A Robostix expansion board provides the interface to the motor controllers, vacuum system and sensors. A WiFi expansion board allows for wireless communication with a base station. The Verdex board is an entirely self contained module with a size of 80mm $\times$ 20mm $\times$ 6.3mm. With the two expansion boards in place the height increases to approximately 15mm.

The performance of the Gumstix Verdex Pro processor has been benchmarked (`http://www.gumstix.org/hardware-design/verdex-pro-coms/`) and found to operate at just over twice as fast as an AMD K6 233MHz processor. As the code has not yet been fully implemented on the Verdex board, this information will be used to estimate the running time of the surface mapping method. These runtimes were found using a Core i7 2.7GHz PC with 6GB of RAM and running Windows 7 OS. This was artificially limited to single core operation at 2.7GHz with 1GB of RAM for the purposes of the runtime experiments in Section 7.2.2. These times will be multiplied by an appropriate factor to estimate the runtime when using the Verdex Pro onboard computer.

This factor is found by first assuming the Verdex Pro runs at a speed equivalent to an AMD K6 processor at 500MHz using the benchmark data. The raw clock speed is 5.4 times slower than the 2.7GHz PC. An extra factor of 50% is added to account for the difference in RAM and processor limitations as well as being conservative with clock speed comparisons between different architectures. Thus the run time of each component of the mapping method is multiplied by a factor of 8.1 to estimate the runtime on the Verdex Pro. This same process was also used to estimate the runtimes of comparable mapping methods in the literature, as seen in Table 7.3. The feature extraction process was coded in MATLAB and so a conservative speedup

factor of 3 was used to convert to an equivalent C++ implementation runtime.

A.2 Image Sensor

The specifications and calibration of the chosen colour camera, the CMUcam3, are detailed below.

Camera Specifications

The relevant specifications for the CMUcam3 and the simulated camera are summarised in Table A.2. The CMUcam3 is a colour CMOS camera with a resolution of 176 × 287 pixels and a horizontal field of view of ±21°. This resolution equates to 0.05 megapixels which is significantly less than a standard webcam or other cameras used in robotics which are often at least 1 megapixel and sometimes even higher resolutions. Image data is acquired via a RS232 serial interface and the average power consumption is 275 mW while the camera is powered on.

During simulation using MRDS, the VSE provides a camera module that is used to simulate the real camera. This is a noiseless pinhole camera and the intrinsic parameters are precisely known. The resolution is 320 × 240 pixels which is approximately 50% higher than the resolution used for the CMUcam3 images. Nevertheless, this resolution is still very low and the simulated camera does provide a wider horizontal field of view of ±30°.

Table A.2: Camera specifications.

Parameter	CMUcam3	Simulated Camera
Image Sensor	Colour CMOS	Simulated
Camera Model	Pinhole	Pinhole
Image Resolution	176 × 287 pixels	320 × 240 pixels
Field of View (Horizontal)	±21°	±30°
Interface	RS232 Serial at 115,200 bps	Simulated
Operating Current (Average)	55 mA	-
Operating Current (Maximum)	130 mA	-
Operating Voltage	5 V	-

Intrinsic Camera Calibration

It is necessary to calibrate each camera to determine the intrinsic parameters such as the focal length (f_u and f_v) and principal point (u_0, v_0). The subscripts u and v refer to the width and height images axes respectively. These parameters form the camera intrinsic parameter matrix K, shown in (A.1), and they allow image points to be transformed into the world coordinate frame as seen later in Section 4.2.3. The cameras were calibrated with a standard checkerboard pattern [129] by using the MATLAB camera calibration toolbox [130]. The parameters are detailed in Table A.3. For the CMUcam3 images there was no skew evident. Some lens distortion was shown to exist and this was removed from the images.

$$K = \begin{bmatrix} f_u & 0 & u_0 \\ 0 & f_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{A.1}$$

A.3 Range Sensor

The specifications and calibration of the chosen laser range finder, the Hokuyo URG-04LX, are detailed below.

LRF Specifications

The specifications of the Hokuyo URG-04LX LRF [85] are summarised in Table A.4. It is lightweight at only 160g and it is small in size at 50mm × 50mm × 70mm, both of which are very desirable traits for usage on a WCR or other small autonomous vehicle. It is also relatively low power, requiring only 2.5W at 5V on average which

Table A.3: Camera intrinsic parameters

Parameter		CMUcam3	Simulated Camera
Focal Length	f_u	205.7	208
	f_v	450.3	208
Principal Point	u_0	74.6	160
	v_0	169.6	120

means it can run off the same power supply as any onboard electronics. It has a wide field of view providing an angular range of 240° with an angular resolution of 0.36°.

This Hokuyo LRF has a maximum range of just over four metres which is low compared to some other commercial LRFs. This is, however, more than sufficient for the target WCR as in general the objects of interest will be at ranges of less than 1000mm. The quoted range accuracy is ±10mm for ranges less than 1000mm which is reasonable but the true error is higher than this in some cases. This is discussed later in Section A.3. This value was generated under ideal conditions and the calibrations and error characterisations discussed in the following section will show that this error value should be used with care, as there are numerous sources of systematic and random error that need to be carefully controlled.

The simulated environment provided a simulated LRF whose specifications are also shown in Table A.4. While Gaussian errors were introduced into the simulated range measurements, the simulated LRF was otherwise highly accurate. There were no systematic biases introduced, unlike a real LRF, as they are difficult to model and reproduce. This was not significant, however, as the Hokuyo LRF allowed the effect of real systematic errors to be tested on the proposed mapping methods.

Table A.4: LRF specifications

	Hokuyo URG-04LX	Simulated LRF
Range	20-4095mm	25-8000mm
Accuracy (bias)	±10mm (range < 1000mm), then ±2%	0mm
Accuracy (std. dev.)	σ = 5mm	σ = 5mm
Range Resolution	1mm	0.001mm
Angular Range	240°	360°
Angular Resolution	0.36°	0.5°
External Dimensions	50mm × 50mm × 70mm	-
Weight	160g	-
Power Usage	2.5W at 5V(4W max)	-
Interface	RS-232C and USB2.0 FS	-

Laser Range Finder Calibration and Characterisation

In many robotic applications that use modern laser range finders, such as the SICK LMS 200 [131] and the Hokuyo URG-04LX [132], the accuracy of the range measurements provided is more than adequate and is easily handled using probabilistic techniques for SLAM [133] and other perception requirements. This is especially true when the environment or the robot itself are large in scale, such as with the autonomous vehicles participating in the DARPA Urban Challenge [134]. In these and other similar situations, a range error of 10mm is negligible and in some case errors of up to 50mm or higher may even be acceptable, especially for tasks such as obstacle avoidance and some 2D SLAM.

In other situations, however, the range measurement accuracy becomes significant. For small robotic systems a range error of 10mm cannot be considered negligible. The small WCR described in Section 3.2 needs to map surfaces accurately in 3D to determine possible foothold locations. This goes beyond obstacle avoidance to actively trying to walk on the object surfaces and so the accuracy requirements are more stringent, particularly when considering the relatively small scale involved.

The Hokuyo URG-04LX range accuracy must therefore be well known and characterised. This will allow good quality range data to be acquired. This is also important as underestimating the range accuracy could have catastrophic consequences.

Central Bearing Calibration

The first calibration that should be performed is to find the true 0° measurement step. This is the range bearing directly in front of the LRF housing and is not necessarily the same as the central measurement step returned from the LRF. A target was setup on the base plate shown in Fig. 3.3(a) at a range of 600mm from the LRF and at an incidence angle of 0° to the laser beam. After taking 1000 readings, the step with the minimum average distance was taken as the true 0° step.

The minimum mean range occurred at step 389 not 384 as quoted by the manu-

facturers. This corresponds to a rotational bias of 1.8° which is not insignificant but is easily compensated for. The necessity for this calibration is demonstrated in other work where the central (0°) step was found at step 387 [86] and 386 [87] in other work. This variability is likely due to the tolerances of the mechanical mounting and appears to differ for each laser individually.

Startup Drift Calibration

LRFs exhibit a property known as warmup or measurement drift. This is where for a period of time immediately after providing power to the LRF, the mean range will drift before settling to a steady state value. All LRFs drift to differing extents for anywhere up to 2 hours [84][135]. This drift has been demonstrated for the Hokuyo URG-04LX by Lee *et al.* [88], Pascoal *et al.* [89] and others. They found that the mean range drifted by 6-20mm for 30-90 minutes after startup. The usual approach to handling this problem involves waiting for a set period of time before acquiring the more useful steady state values.

Tests were performed to confirm these values and it was found that the drift varied substantially depending on the target range and surface material, amongst other factors. The drift lasted up to 60 minutes in some cases before settling to steady state. During the data acquisition process, described in Section 3.3, the LRF was powered on for a minimum of 2 hours before acquiring the datasets.

For practical robotics tasks, this waiting period is not ideal, especially for time critical applications such as search and rescue. It would therefore be useful if this drift could be compensated for. Two possible methods involve using the time elapsed or the temperature of the receiver electronics as suggested by Hebert *et al.* [135] and Kneip *et al.* [86]. Excluding the first 15 minutes of operation, the LRF drift appears to exhibit a common shape that shows a weak correlation to the LRF temperature, which heats up by about 12°C during startup. Initial tests showed some promise in using this temperature to estimate the drift error. However, more work is needed to fully verify it on a range of operating conditions outside of the laboratory and so

any such compensation is not used further at this stage.

Range Error Characterisation

The URG-04LX LRF manufacturer (Hokuyo) quoted a range accuracy of ±10mm [85]. It is unclear exactly what this represents in terms of the range error. It appears to be an upper bound on the mean range bias for a small number of range readings taken under ideal operating conditions. This figure takes no account of the numerous systematic error sources and gives no indication of the level of Gaussian noise present in the range measurements.

The true range error of a single measurement can be much higher than ±10mm due to a variety of factors. In addition to the calibration errors discussed above, other common error sources are surface dependent. These include the target range, surface material and angle of incidence to the laser beam. Range errors have been seen in the order of 30mm in some extreme but not uncommon cases and this error consists of a mix of systematic biases and random error. Considering the need for accurate range data it is important to ensure that these error sources are carefully considered and calibrated for if possible.

Several studies have characterized various different LRF models to show the existence and extent of systematic range error sources [84][135][136]. The Hokuyo URG-04LX range error has also been characterised in this way. Lee *et al.* [88] compared it to the SICK LMS200, errors due to surface material and the surface patterns of the object and found that both LRFs showed similar levels of error. Pascoal *et al.* [89] looked at the Hokuyo LRF to determine their usefulness in difficult environmental conditions. They found that the range accuracy deteriorates significantly and becomes unusable in environments containing adverse conditions, including smoke or excessive heat, such as those found at a disaster site or building fire.

Kneip *et al.* [86] and Okubo *et al.* [87] both followed the approach that Ye *et al.* [84] used to characterize the SICK LMS200. Both focussed particularly on the

influence of surface material and angle of incidence on the Hokuyo range error as well as applying a linear model to the dependency of range error to true range. Overall, these characterisation studies found that while the surface dependent systematic error sources could be modeled to some degree of success individually, they were difficult to properly model and compensate for in combination.

Others have proposed LRF sensor models, but they only provided limited success at the expense of complicating any algorithms wishing to utilise them. One such model was proposed by Lang *et al.* [137] which modeled the incidence angle error in a Bayesian fashion. While some improvement in range accuracy was seen, the model had to be individually calibrated for each new material which limited its use.

It is possible to provide an approximate calibration if the target environment is known in advance. The target environment consists of surfaces at ranges of less than one metre which allows most of the range dependent range error to be compensated for. The angle of incidence and surface material are harder to compensate for. The range distributions for a variety of expected surface materials and angles of incidence were acquired at ranges of 200mm to 600mm.

A systematic bias of 17mm was estimated and this was removed from the raw range measurements during acquisition. The level of Gaussian noise was also measured and differed from 3-5mm depending on the target conditions. A value of 5mm was used in this work as a conservative estimate.