A FRAMEWORK FOR NON-ICP LIDAR-BASED POSE ESTIMATION USING AN OPTIMALLY CONSTRAINED 3D TARGET

by

Pierre Saint-Cyr
B.Eng, Ryerson University, 2008

A thesis
presented to Ryerson University
in partial fulfillment of the
requirements for the degree of
Master of Applied Science
in the Program of
Aerospace Engineering

Toronto, Ontario, Canada, 2010

I hereby declare that I am the sole author of this thesis.

I authorize Ryerson University to lend this thesis or dissertation to other institutions or individuals for the purpose of scholarly research.

I further authorize Ryerson University to reproduce this thesis or dissertation by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Abstract

A FRAMEWORK FOR NON-ICP LIDAR-BASED POSE ESTIMATION USING AN OPTIMALLY CONSTRAINED 3D TARGET

by

Pierre Saint-Cyr
B.Eng, Ryerson University, 2008

A thesis presented to Ryerson University in partial fulfillment of the requirements for the degree of Master of Applied Science in the Program of Aerospace Engineering

Ryerson University
Toronto, Ontario, Canada, 2010

©Pierre Saint-Cyr 2010

This thesis describes a non-ICP-based framework for the computation of a pose estimate of a special target shape from raw LIDAR scan data. In previous work, an ideal unambiguously-shaped 3D target (the Reduced Ambiguity Cuboctahedron, or RAC) was designed for use in LIDAR-based pose estimation. The RAC was designed to be used in an ICP algorithm, without an initial guess at the pose. This property is, however, not robust to LIDAR measurement noise and data artefacts. The pose estimation technique described in the present work is based upon the geometric non-ambiguity criteria used originally to design the target, and is robust to the aforementioned LIDAR data characteristics. This technique has been tested using simulated point clouds representing a full range of views of the RAC. The technique has been validated using real LIDAR scans of the RAC, generated at Neptec's Ottawa facility with their Laser Camera System (LCS). Experimental results using LCS data show that pose estimates can be generated with mean errors (relative to ICP) of 1.03 [deg] and 1.08 [mm], having standard deviations of 0.56 [deg] and 0.67 [mm] respectively.

Acknowledgements

Firstly, I would like to acknowledge my supervisors, Dr. John Enright and Dr. Galina Okouneva, for all the advice and support they have given me. Their encouragement and patience, as well as their criticism, have been instrumental to me in the writing of this thesis. I would also like to thank Dr. Don McTavish, both for inspiring me to pursue my graduate degree, and for introducing me to his work on constraint analysis.

Secondly, I would like to thank everyone in the department of Aerospace Engineering at Ryerson University, as well as our research partners, Neptec Design Group, and the Canadian Space Agency.

Finally, I want to thank everyone else who provided me with support and encouragement when I needed it. My parents, Lise & Jean Saint-Cyr, for being absolutely steadfast in their support of my academic interests. They have lovingly given whatever was in their power to provide, and deserve the highest praise. My brother Yves, for always being there with a sympathetic ear, a relaxing night out, or whatever else I needed. My friends Aradhana Choudhuri, Rhett Lunn and Adrian Marek, for their help and encouragement, and for all the times I started a sentence with "tell me if this makes sense…". And Chris Vogel, both for his unwavering support, and for being a part of my life.

Cette thèse est dédiée à Rita Bourdon

je me souviens

Table of Contents

List of Tables

List of Figures

Terms and Abbreviations

CSVS	Canadian Space Vision System
DOF	Degree Of Freedom
FNPE	Face Normal Pose Estimation
ICP	Iterative Closest Point
LCS	(Neptec's) Laser Camera System
LIDAR	LIght Detection And Ranging
RAC	Reduced Ambiguity Cuboctahedron
RANSAC	RANdom SAmple Consensus
RMS	Root-Mean-Squared
RMSE	Root-Mean-Squared Error
SVS	Space Vision System

Chapter 1 Introduction and Prior Work

In recent decades, computer vision has become an integral part of robotic space systems. The Canadian Space Vision System (CSVS) installed on the ISS in 1992, was developed by Neptec Design Group. It works by tracking black and white targets attached to the surfaces of spacecraft and ISS modules. Laser-based space vision systems (also built by Neptec) are installed on the MDA-built Remote Manipulator System (Canadarm I), and Space Station Remote Manipulator System (Canadarm II).

Pose estimation is a core engine of any computer vision system. The pose of an object is a six DOF vector defining its position and orientation in 3D space, relative to a given reference frame. A pose estimate is typically determined by establishing a correspondence between sensor data and a computer model, in the form of a rigid transformation. Pose estimation is used to perform various tasks, including 3D shape reconstruction, object identification and target tracking. These typically involve using sensor data to extract features from which some combination of range, position or orientation can be inferred, either in 2D or 3D. These features include things such as corners, edges, surfaces and intensity gradients.

In orbital rendezvous and docking operations, pose estimation is particularly important. In this context, the use of fiduciary markers is already well-established [1]. These markers are typically retro-reflectors or visual targets (such as with the CSVS) that can be easily identified in the sensor data. The arrangement of these markers (as seen in the data) is then compared to the known arrangement. From this, the pose can be calculated. This approach has some associated difficulties. Extreme glare or shadows, for example, can prevent some markers from being registered.

In contrast, LIDAR (LIght Detection And Ranging) technology can be used to acquire highly accurate pose estimates under any lighting conditions. This technology is based on the use of lasers to establish range measurements at multiple locations (by various methods such as time-of flight, triangulation, etc...), resulting in arrays of discrete range data. The object being scanned

will register as a collection of points in 3D space, called a "point cloud".

The Iterative Closest Point (ICP) algorithm [2] is a method commonly used to determine pose estimates from point clouds [2] [3] [4]. Iterative rigid transformations are used to minimise the mean-squared distance between input data points, and the closest corresponding points on a computer model of the target shape. Although this method can be highly accurate, its iterative nature can make it very computationally expensive when used with large numbers of data points [4]. Also, a poorly constrained shape[1] (which will often exhibit a high degree of symmetry) such as a cube, can cause the algorithm to converge on a false solution, since there can be many ways to match data representing a cube to a model of a cube. In this case, there are multiple global minima, only one of which represents the true pose of the cube. The algorithm will simply converge on the global minimum nearest to the initial guess. Depending on the initial guess at the pose, objects that are geometrically well-constrained (i.e. non-symmetric and therefore unambiguous) can still cause the algorithm to fall into *local* minima, possibly leading to false solutions. If the algorithm has fallen into a local minimum, it means that any incremental modification of the orientation and translation will cause the registration error to increase. In this case, even if the shape has only have one global minimum (i.e. the true pose), the algorithm may converge on a local minimum instead. Typically, these problems necessitate an initial estimate of the pose that is as close as possible to the true pose.

Many spacecraft have simple geometries that lead to ambiguous pose estimates. Even when spacecraft geometries are adequately constrained, many features are not visible during close-range docking operations. This leads to inaccurate pose estimation. In previous work done at the Ryerson Space Vision Lab [5][8], a specially shaped target (designed for use with ICP) was designed to deal with this problem. This shape is called a Reduced Ambiguity Cuboctahedron (RAC), and is pictured in Figure 1.1. By design, sets of neighbouring faces will exhibit unique arrangements of normal vectors. This approach eliminates unwanted local (and global) minima and ensures that the shape is always geometrically constrained. Good geometric constraint

[1] Constraint analysis is an application of Principal Component Analysis. The constraint of a given shape can be used to describe the degree to which that shape is unambiguous. Further information can be found in [5-7]

enables accurate ICP pose estimates from any view geometry. Under ideal circumstances, LIDAR scans of such a shape can be used with an ICP algorithm to obtain the true pose without an initial guess. A physical model of the RAC was scanned with Neptec's LCS, a LIDAR camera system developed for spacecraft pose estimation. Previously, Choudhuri [6] used a portion of the data from these scans to validate the shape design.

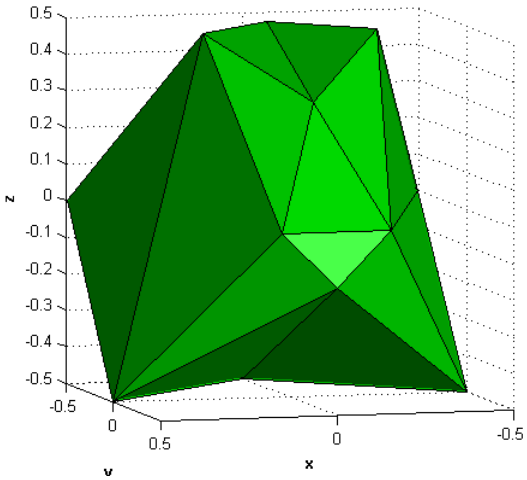

Figure 1.1: The reduced ambiguity cuboctahedron (RAC).

1.1 Motivation

The RAC was designed such that an initial guess at the pose was not necessary for ICP to converge on the correct solution. ICP, however, can be very computationally intensive for dense point clouds. Accurate ICP pose estimates require high resolution scans, which will necessarily result in large numbers of points. This makes real-time estimates more difficult. In addition, although the shape was shown to behave as anticipated under near ideal conditions, its optimality depends on ideal viewing, and is not robust to noise in the data, or artefacts such as edge effects or outliers. If these artefacts are present in the data, ICP is not guaranteed to converge on the correct solution without an initial guess at the pose. As it is impractical to prune these artefacts from the scan data in real-time, an alternative method was sought to estimate the pose of the RAC from raw scan data.

Extracting the normal vectors from the observed surfaces provides a convenient means of directly estimating the relative orientation between the LIDAR and the target shape, since

measurements based upon these normals will be invariant in translation and rotation. In many situations these estimates may be sufficient, but if additional accuracy is required, the direct pose estimates can be used as an initial guess for a classical approach such as ICP.

The present work is a logical extension of the ideal target design work described in [6]. Although the RAC's geometry was intended to facilitate ICP convergence, there is no reason to refrain from exploring other potential ways in which this geometry could be used for pose estimation. A great deal of work currently exists concerning geometric feature extraction from LIDAR point clouds, as seen in the following references [9], [11], [10], [12-14] (which is by no means an exhaustive list). The present work in not the only one to describe a non-ICP 3D registration technique that is based upon local relative geometry [15]. To date, however, no other source has been found which is designed to take advantage of the geometric properties of a shape such as the RAC.

1.2 Scope of thesis

The major contribution of this thesis is the practical implementation of a pose estimation algorithm based on the use of the unique properties of the RAC as a target. The implementation described here takes into account real LIDAR data, which is not edited other than basic cropping to isolate the shape from its surroundings. The LCS data used consists of point clouds of anywhere from six to ten thousand points. These data contain edge effects, some outliers, and some geometric distortions (see section 4.1 for an overview of the data characteristics). In addition to real LIDAR data, the work described here is tested with simulated data from a range of perspectives. Specifically, this thesis deals with the implementation of a catalogue lookup algorithm to generate orientation estimates, the extraction of the required input data from the raw point cloud, the determination of a displacement estimate, and testing of the accuracy of the resulting pose estimates. Although real-time implementation of these tasks is considered in the algorithmic development, it is not the primary focus of this thesis. It should also be noted that this work considers only static pose estimation. If the “motion blur” resulting from a scan of a moving target is not excessive, then the present work should provide a reasonable estimate of performance during dynamic scanning situations. Other issues such as the identification and isolation of the target from a wide scan, target tracking, the processing of sequential pose

estimates, and the precise simulation of LIDAR error characteristics are beyond the scope of this thesis. An ICP algorithm is used in some of the experimental validation and the implementation is briefly described, but the design and optimisation of this algorithm is not a part of this work.

1.3 Face Normal Pose Estimation (FNPE) Overview

Face Normal Pose Estimation (FNPE) is the estimation of the RAC's pose from LIDAR scan data without using ICP. The goal is to provide a faster pose estimate than ICP (for rough estimation) which can be used to initialise ICP (for more precise estimation) if desired. Orientation estimation is based on the unique geometry of the target as represented in the point cloud, and relies on the identification of planar surfaces and the calculation of their associated normal vectors. Measurements are generated from combinations (called "tuples") of these normal vectors, and then sent to a catalogue lookup algorithm where they are matched to reference data. The catalogue lookup algorithm then uses measurement-to-reference correspondence (if possible) to output a point cloud orientation estimate[2]. The orientation portion of the pose estimate is more difficult to generate than the displacement portion, but can be solved separately. Once the orientation has been determined, displacement calculations are relatively simple. The basic process is shown in Figure 1.2.

1.4 Thesis Outline

Chapter 2 presents the theoretical basis for the use of the RAC to generate unique pose estimates for a given view, as well as the methodology for the construction of the reference catalogue, and the catalogue lookup algorithm. Chapter 3 describes the methods investigated to extract the geometry of the RAC from a raw point cloud, and the integration of these methods with the algorithm described in Chapter 2, to generate orientation estimates. Two different point cloud geometry extraction methods were investigated. The initial method (FNPE1) is described in Section 3.1, but was later replaced by a better-performing algorithm (FNPE2), described in Section 3.2. The initial method was judged to be inadequate when considering only orientation, so no displacement estimation algorithm was written. It is provided mainly as context for the

[2] The orientation of the point cloud is defined as the rotation which will rotate the computer model from a default attitude to the observed attitude of the point cloud.

development of the FNPE2 algorithm. A displacement estimation technique using least-squared fitting was developed for use with the second method (FNPE2). Chapter 4 provides a brief outline of the characteristics of the point cloud data used in this thesis, as well as a very brief overview of the simulations and experiments in which they are used. Chapter 5 presents the setup and results of all of the simulations and experiments performed. In Chapter 6, the results from the FNPE2 simulation are compared and contrasted with those from the FNPE2 LCS experiment, and general observations are made. Final conclusions and future work are described in Chapter 7.

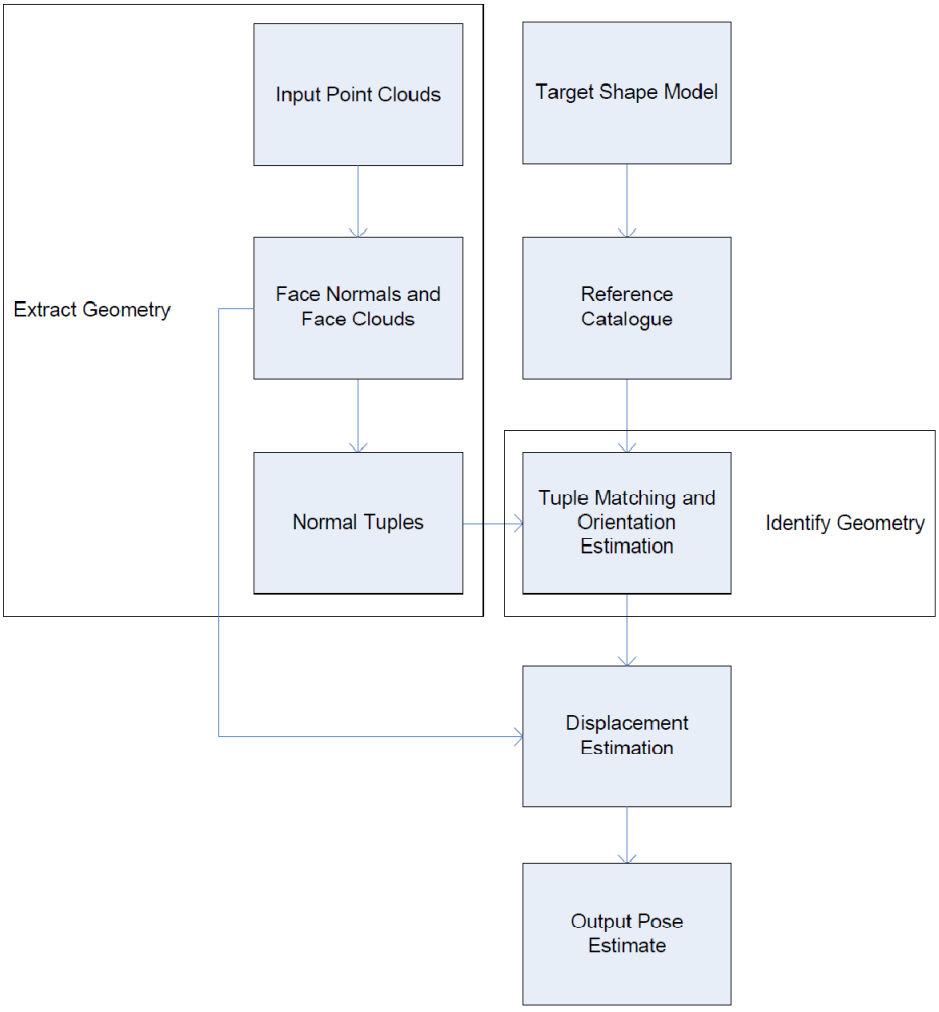

Figure 1.2: FNPE overview

Chapter 2 Catalogue Lookup

The usefulness of the RAC is derived from the unique orientation of neighbouring faces. Extracting the normal vectors from the observed surfaces provides a means of directly estimating the relative orientation between the LIDAR and the target shape. The relative orientations of neighbouring sets of normal vectors are unique for any part of the shape, meaning that experimental measurements of the angles between these vectors can be matched to unique entries in a reference catalogue. This permits the identification of the specific faces being observed. Calculation of the rotation necessary to bring the reference vectors to the measured vectors provides an estimate of the orientation of the RAC. The displacement portion of the pose (used in FNPE2) is estimated separately (see section 3.2.6).

2.1 Required Point Cloud Measurement Geometry

The unique target geometry that facilitates orientation estimation also imposes a requirement on the manner in which the angular measurements are made. A specific geometric arrangement of faces must be identified in the point cloud, and the normal vector of each face must be used differently depending on its relative position within the pattern. A set of faces **F** = {F1, F2, ... , F6} defined by vertices **V** = {V1, V2, ... , V9} is shown in Figure 2.3. This figure highlights the geometric arrangement required for orientation estimation.

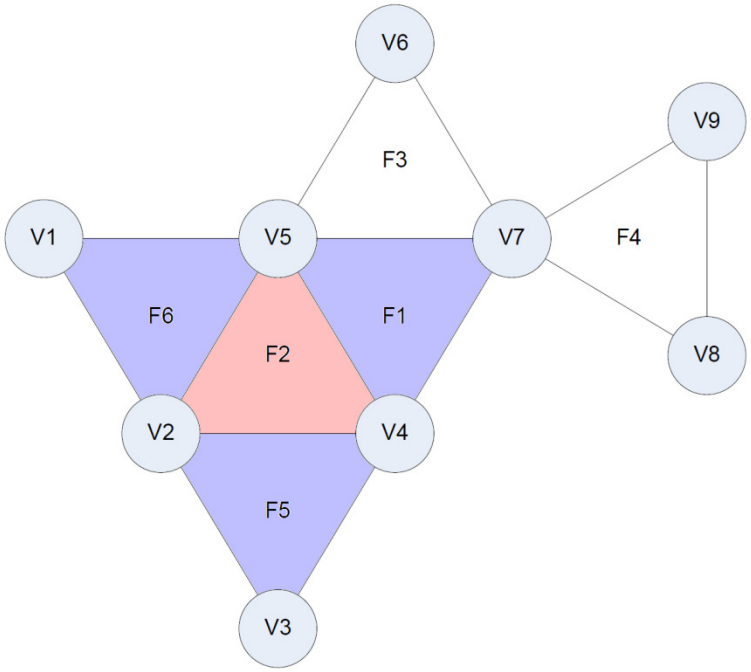

Figure 2.3: Arrangement of faces necessary for orientation estimation.

The only requirement for the formation of this pattern is that it contains a central face (shown in red), which shares each of its edges with a different adjacent face (shown in blue)[3]. If no such central face can be identified from the point cloud data, the FNPE technique cannot be used to generate an orientation estimate.

2.2 Catalogue Construction and the Lookup Process

The uniqueness of the target shape is based upon the angles between a central face normal and the normals of its three adjacent faces. Thus for each face there is a tuple of angles:

$$\tau \equiv (\theta_a \quad \theta_b \quad \theta_c) \tag{1}$$

This tuple forms a unique set when the angles are correctly ordered, as shown in Table 2.1. This order depends upon the method that will be used to compare estimated data and catalogued data. Considerations relating to measurement error must therefore be included in the design

[3] The terms "central face" and "adjacent face" are used later in this thesis, and refer to the relative arrangement of faces shown in Figure 2.3.

methodology of the angle ordering criteria. These criteria, applied to the error-free data, are used to compile the reference catalogue.

Table 2.1: Tuples of ordered angles

Central Face	θ A	θ B	θ C
1	49.0	15.9	43.1
2	69.4	43.1	36.1
3	69.4	36.1	43.1
4	55.5	43.1	15.9
5	80.7	9.8	59.3
6	66.1	55.5	9.8
7	40.8	60.3	59.3
8	35.6	40.8	40.8
9	35.6	15.9	15.9
10	58.8	69.4	66.1
11	74.1	80.7	81.3
12	9.6	52.3	49.0
13	74.1	9.6	58.8
14	40.8	52.3	60.3
15	69.4	58.8	58.8
16	74.1	58.8	27.0
17	74.1	27.0	58.8
18	81.3	36.3	60.3
19	81.3	60.3	36.3
20	81.3	74.1	74.1

Both the angles, as well as the corresponding unit normals, are necessary for the determination of the target's orientation. Some pairs of tuples contain identical angles, and can only be distinguished from each other using the normal vectors and relative positions of the observed surfaces. In order to match measured angles to their counterparts in the catalogued tuples, both the measured and reference data must be ordered in the same way (i.e. $\theta_{a,m}$ must be compared to $\theta_{a,r}$, etc...). A method is therefore needed to provide a specifically defined order to the angles of each tuple, while maintaining the required uniqueness. Our approach is to designate the largest angle θ_a. θ_b and θ_c can then be determined by rotating counterclockwise about the central normal (n_0), from the unit vector associated with θ_a, through to the unit vectors associated with θ_b and θ_c (see Figure 2.4).

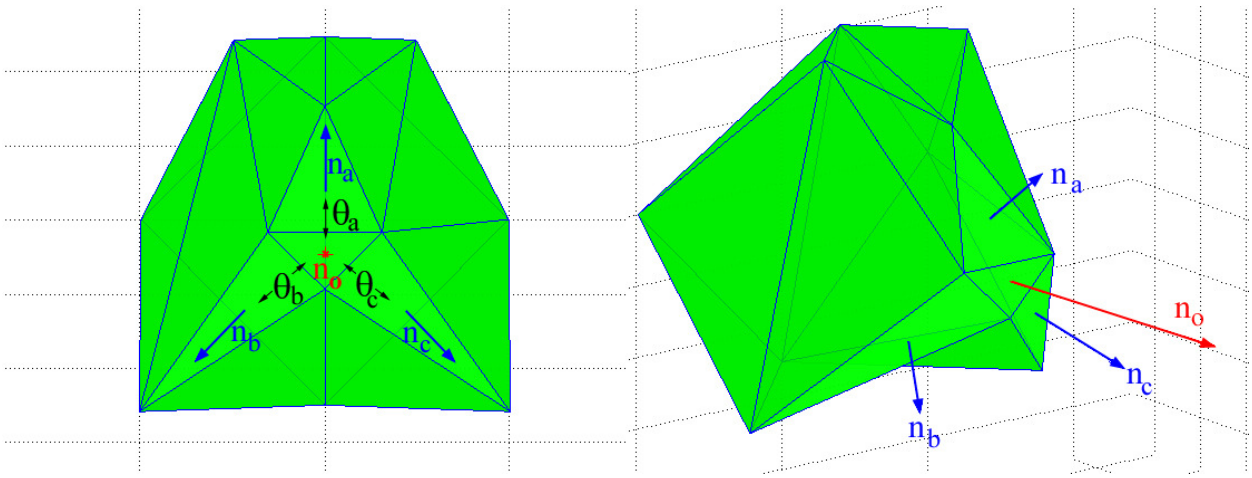

Figure 2.4: Angles and normals for tuple number 8.

Tuple 8 (pictured in Figure 2.4) has two angles which are identical. There are tuples on the shape which have duplicate large angles as well as duplicate small angles. Therefore neither the largest nor smallest angle is sufficient to uniquely determine the ordering. When measurement error is considered, this type of ambiguity is introduced into some (but not all) of the otherwise unambiguous sets of data as well.

Angle ambiguity can be avoided by grouping the possible measurements by magnitude, and associating each angle with the specific angular range within which it falls. These angular ranges are given identification numbers, referred to as "group numbers". Each angle is therefore associated with a specific group number. The following is an example of this process (Note that the angular values and angular ranges used in this example are purely demonstrative, and do not relate to any other part of this thesis).

Example:

A set of angles is given as shown in Figure 2.5 (upper left). The angles are then rearranged Figure 2.5 (lower left), sorted by magnitude, and divided into angular ranges Figure 2.5 (lower right). These angular ranges are given "group numbers". Each set of angles is then associated with a set of group numbers Figure 2.5 (upper right).

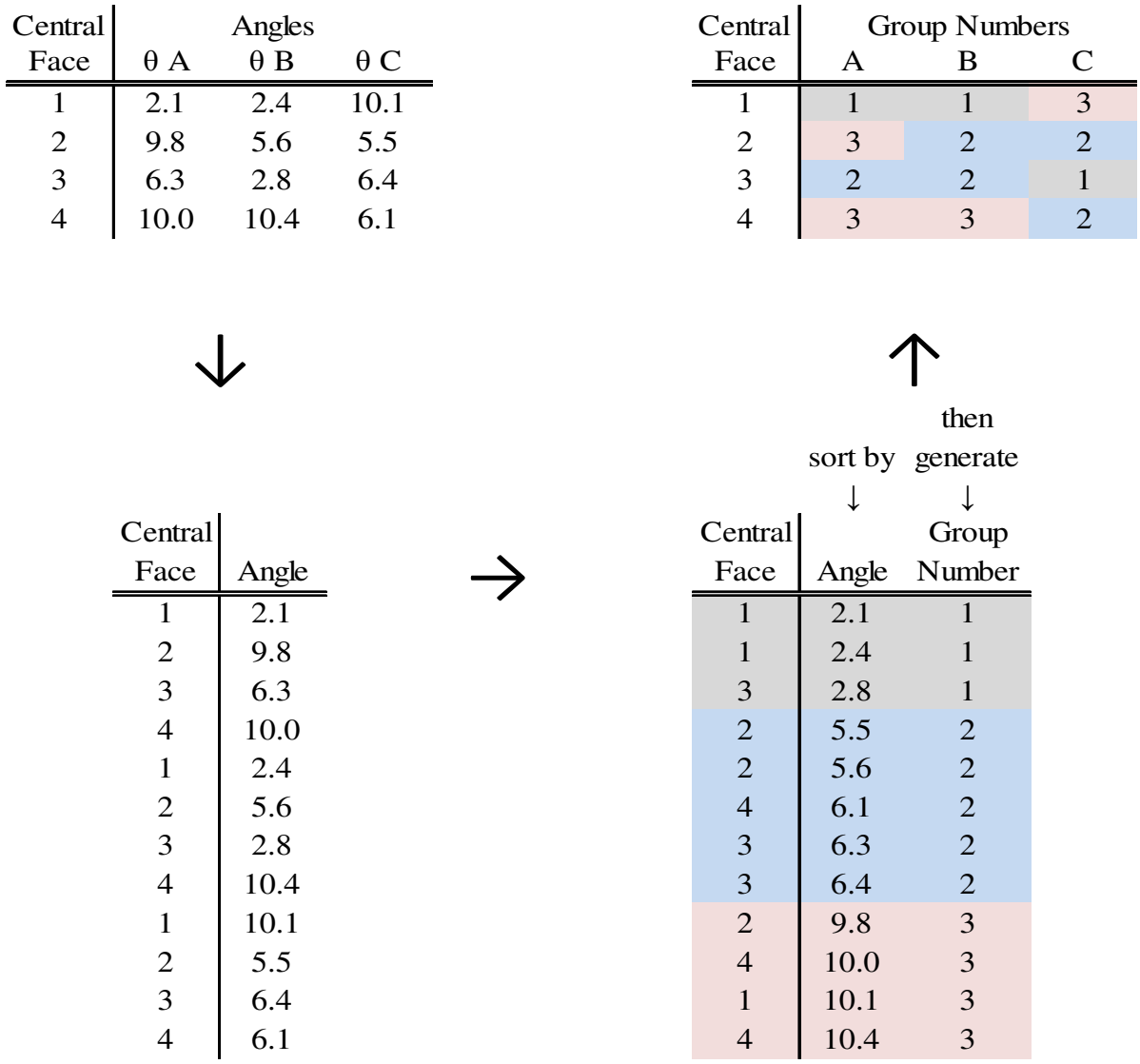

Figure 2.5: Generation of group numbers

2.2.1 Generation of the Reference Catalogue

The individual angles from the reference catalogue have been separated into nine groups. Each group has a minimum or maximum value at least 4.48° away from the closest angle in a neighbouring group. By associating each angle in each tuple with a particular group, a great deal can be learned about the likely identity of a measured tuple. Range errors from the LIDAR scan introduce errors in the measurements of the face normals, and consequently the angles between

faces. Thus, it is far easier to determine group membership than it is to determine the precise angular value, as some of the reference angles differ by only 0.2˚, as with θ_c from tuple 6, and θ_b from tuple 13. The group numbers are unique (irrespective of sequence within the tuple) for 45% of the angular sets. For the rest, the group numbers are common (again, irrespective of sequence within the tuple) to no more than two tuples, of which no more than one will ever require a specific angle sorting criterion. The group numbers can therefore be used to determine when to apply a tuple angle sorting criterion based specifically on a maximum or minimum angular value. When the sets of reference tuples are sorted using this information, tuples of ordered angles are generated as seen in Table 2.1. The reference catalogue consists of these ordered tuples (and their sorting criteria), as well the corresponding ordered normal vectors and group numbers. The basic functionality is illustrated in the block diagram in Figure 2.6.

2.2.2 Parsing of Measurement Data

Measured tuples (and their associated normal vectors) can be identified, ordered and matched against the catalogue, using a similar procedure as was used to generate the reference catalogue. The following is an example of the basic catalogue lookup procedure. The input is a measured tuple of angles, and the vectors used to calculate them. If all of the measured angles are associated with measurement errors of less than 2.24 [deg], the following will always work as described. Otherwise, the following will function only for certain tuples, depending on the tuple and the amount of error involved.

The basic catalogue lookup process is as follows: First, determine the group numbers of the angles in the measured tuple. The measured group numbers are then compared to the catalogued group numbers. A "simple match" is made between the measured group numbers and those from a set in the reference catalogue, if the reference set contains the same three group numbers (in any order). A simple match will identify which angle sorting criterion (i.e. max or min angle) to use. The measured normal vectors can then be used to put the measured angles (and group numbers) in order. An "ordered match" is made between the measured group numbers and those from a set in the reference catalogue, if the reference set contains the same three group numbers *in exactly the same order*. For 90% of the possible tuple measurements, only one ordered match will be found. The other 10% will find no more than two. If two ordered matches are found, an

error metric is generated for each one. Orientation estimates are determined as part of the error metric generation process. The match with the minimum error is used to identify the correct tuple match, and the corresponding orientation estimate. This basic structure is outlined in Figure 2.6.

Using the catalogue lookup process described so far, measured tuples can only be identified if the error in the measured angles does not cause them to be classified in the wrong group. Unfortunately, this requires that all angular measurements be accurate to within 2.24º. To improve error tolerance, the method of grouping is taken further. Each measurement is now given a primary group number, and an alternative group number (for example, a measurement that is in the upper half of group three, would be given an alternative group number of four). Every combination of primary and alternative group numbers is then parsed in the same manner as with the basic catalogue lookup algorithm. The amount of error tolerance added by this approach varies from tuple to tuple (minimum error tolerance added was 0.5º, with errors as high as 4.25º returning correct solutions). Whereas the basic algorithm produces no more than two ordered matches for a given tuple measurement, the robust algorithm has not been seen to produce more than four. This robust configuration of the catalogue lookup algorithm is illustrated in the block diagram shown in Figure 2.7, and is the configuration used with all the experiments presented in this thesis.

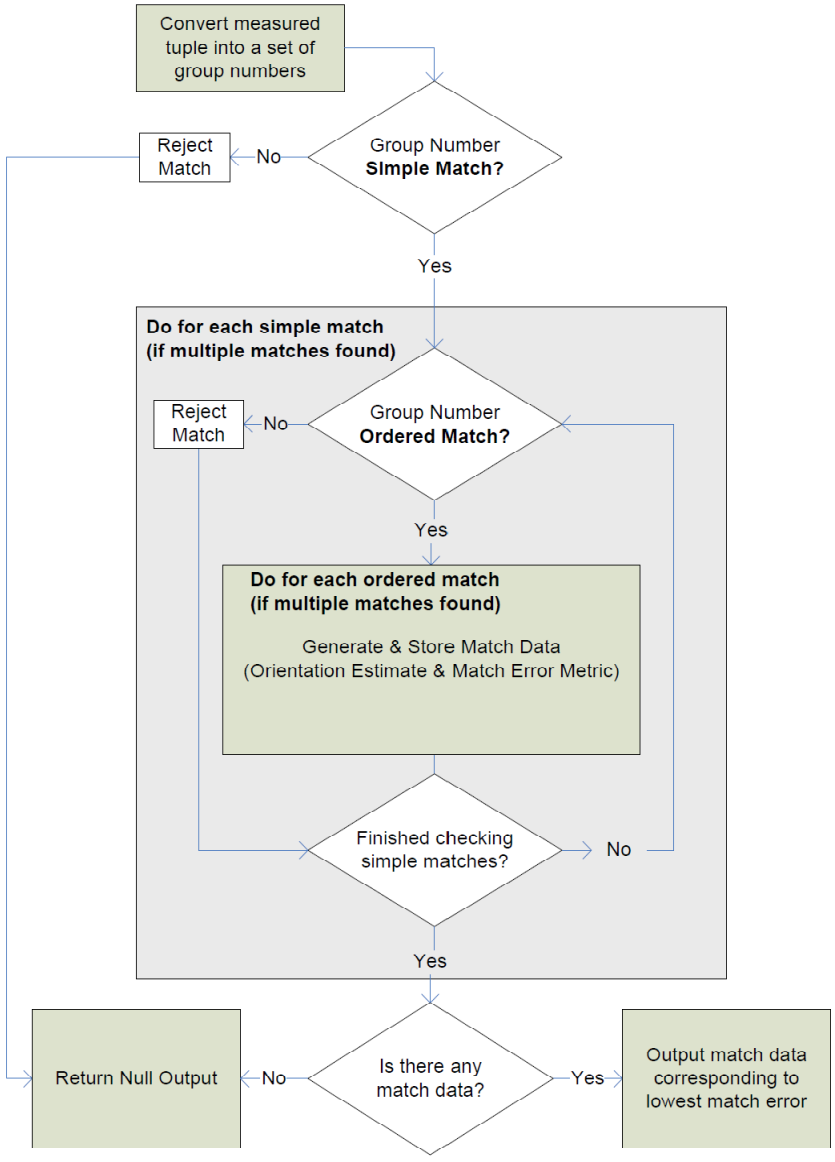

Figure 2.6: Basic catalogue lookup algorithm.

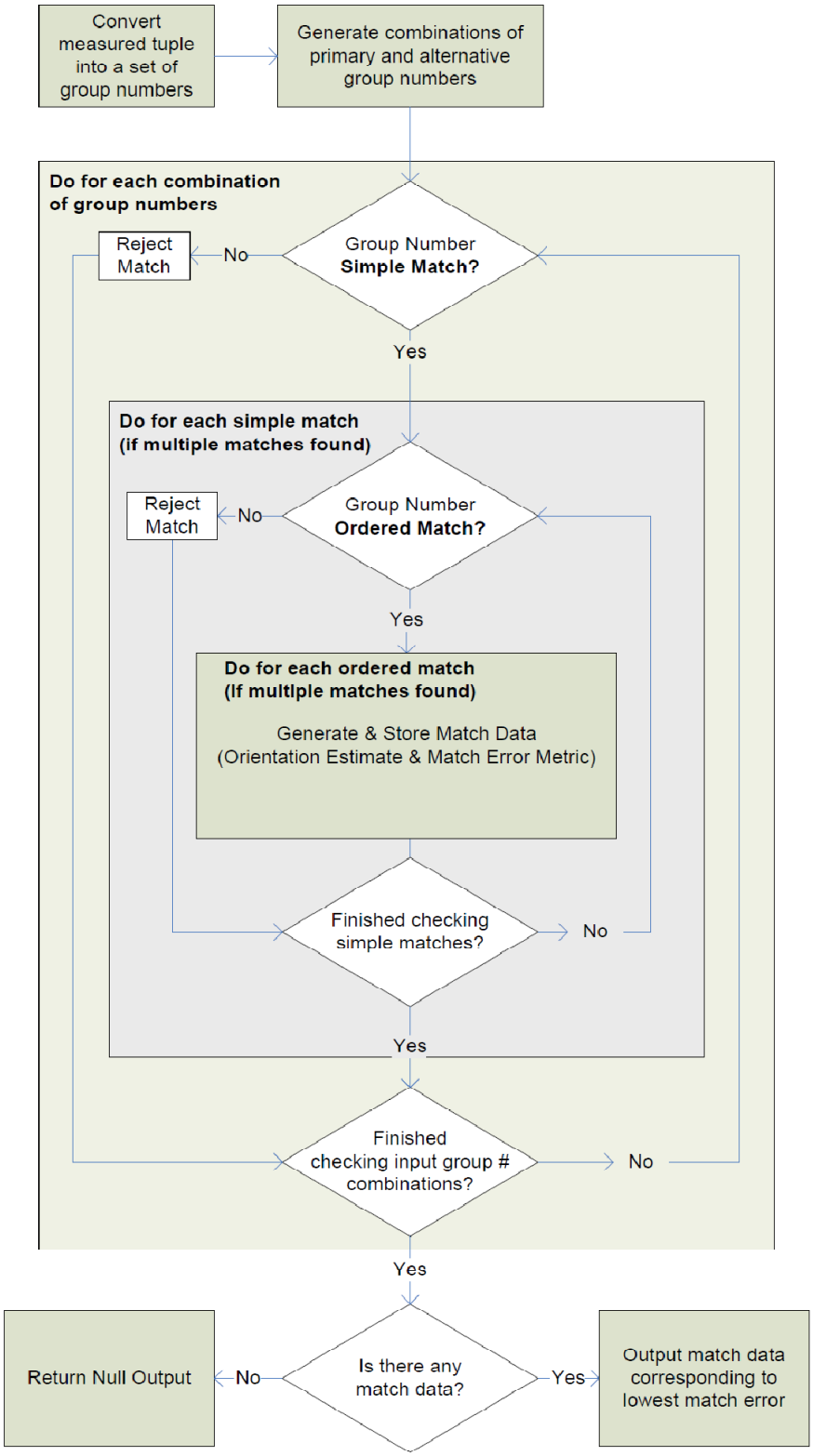

Figure 2.7: Robust configuration of the catalogue lookup algorithm.

(This configuration is the one implemented with all the experiments in this thesis).

2.3 Matching Error

If the measured tuples are associated with only one ordered match, the identity refinement process simply generates an orientation estimate for that match (the error metric is still generated, but it is not used). If there are multiple ordered matches, orientation estimates are calculated for each one, and used to generate associated match errors. The minimum match error is used to select the correct tuple, and its corresponding orientation estimate.

2.3.1 Match Refinement

The error metric generated by the catalogue lookup algorithm is used to determine the entry in the reference catalogue that best matches the input data[4]. It is calculated in two parts. The first measures the difference between the measured and reference angles:

$$e_1 = \sqrt{\left(\theta_a - \theta_a^{'}\right)^2 + \left(\theta_b - \theta_b^{'}\right)^2 + \left(\theta_c - \theta_c^{'}\right)^2} \quad \text{where} \quad \begin{matrix} \theta = \text{reference value} \\ \theta^{'} = \text{measured value} \end{matrix} \tag{2}$$

ESOQ2: Second Estimator of the Optimal Quaternion

ESOQ2 [16], is one of many possible algorithms for solving the Wahba problem, any of which could have been used. The ESOQ2 algorithm is a computationally efficient way of calculating q_{opt} as the eigenvector associated with the greatest eigenvalue of the matrix K:

$$Kq_{opt} = \lambda_{max} q_{opt} \tag{3}$$

where

$$K = \begin{bmatrix} B + B^T - tr[B] I_{3\times 3} & z \\ z^T & tr[B] \end{bmatrix} \tag{4}$$

B is the attitude profile matrix, $I_{3\times 3}$ is the 3×3 unit matrix, and z is the vector:

$$z = \{B(2,3) - B(3,2)\,, \quad B(3,1) - B(1,3)\,, \quad B(1,2) - B(2,1)\}^T \tag{5}$$

The ESOQ2 algorithm computes λ_{max} using the solution of the quartic algebraic equation associated with the characteristic polynomial of the K matrix:

$$\lambda^4 + a\lambda^3 + b\lambda^2 + c\lambda + d = 0 \tag{6}$$

where a = tr[K], b = -2(tr[B])+tr[adj(B+B^T)]-z^Tz, c = -tr[adj(K)], and d = det(K). The complete

[4] Note that an older version of this match error metric was used when conducting the FNPE1 experimental simulation. The older metric is included in the appendix.

solution can be found in [16].

ESOQ2 is used to find a best-fit rotation that maps the reference vectors onto their measured counterparts. After the rotation is applied, the angular differences between each measured normal $[\boldsymbol{n}_a' \quad \boldsymbol{n}_b' \quad \boldsymbol{n}_c']$ and its corresponding reference normal $[\boldsymbol{n}_a \quad \boldsymbol{n}_b \quad \boldsymbol{n}_c]$ are calculated. The second error metric is then defined as follows (note that all the normals are unit normals):

$$e_2 = \sqrt{(\Delta\theta_a)^2 + (\Delta\theta_b)^2 + (\Delta\theta_c)^2} \quad \text{where} \quad \begin{aligned} \Delta\theta_a &= \arccos(\boldsymbol{n}_a \cdot \boldsymbol{n}_a^T) \\ \Delta\theta_b &= \arccos(\boldsymbol{n}_b \cdot \boldsymbol{n}_b^T) \\ \Delta\theta_c &= \arccos(\boldsymbol{n}_c \cdot \boldsymbol{n}_c^T) \end{aligned} \tag{7}$$

The first error is based purely on angle mismatches between measured and reference tuples. The second is an evaluation of the effectiveness of the orientation estimate. The sum of these two errors is the metric used to determine the overall match quality:

$$e_3 = e_1 + e_2 \tag{8}$$

The smallest value of e_3 is used to identify the matching reference tuple, and formulate the orientation estimate for a given view.

2.4 Accommodating the Deformed Geometry of the Physical Model

During testing, it was noted that the geometry of the physical model does not perfectly match the theoretical model. The part was fabricated in a 3D printer, and it is believed that the vertices of the shape became skewed when the data were entered into the printer. Since the physical shape is not obviously dissimilar to the theoretical model, the manufacturing error was not discovered until after the test sessions using the Neptec LCS. The project schedule did not permit a repeat of the tests using a corrected model. In the present work, it should be noted that the numeric data presented in Chapter 2 (except for the purely demonstrative examples), as well as the point cloud data used in the FNPE1 experimental simulation (section 5.1), are based upon the parameters of the undeformed theoretical model. Upon discovery of the error, the catalogue lookup algorithm was modified to better represent the shape of the physical model. A combination of repeated physical measurement and trial and error was used to find approximate rotations and stretching parameters so as to approximate the physical model as closely as possible. Rather than the uniform scaling factor of 250 [mm] used previously, the shape was first rotated about the x-axis by 127.1074 [deg], before being stretched by the following factors:

$$\begin{bmatrix} X-Axis: & 243.9946 \\ Y-Axis: & 259.4240 \\ Z-Axis: & 228.9920 \end{bmatrix}$$

This resulted in the modified tuples shown in Table 2.2:

Table 2.2: Modified tuples for the physical model of the RAC

Central Face	θ A	θ B	θ C
1	48.2	14.9	40.5
2	72.0	40.5	35.2
3	72.0	35.2	40.5
4	55.1	40.5	14.9
5	83.7	9.5	56.8
6	64.4	55.1	9.5
7	41.8	64.7	56.8
8	32.3	41.8	41.8
9	32.3	14.9	14.9
10	62.5	72.0	64.4
11	70.2	83.7	80.1
12	9.6	50.3	48.2
13	76.8	9.6	57.6
14	41.8	50.3	64.7
15	72.0	62.5	57.6
16	70.2	62.5	27.3
17	70.2	27.3	62.5
18	80.1	38.0	64.7
19	80.1	64.7	38.0
20	70.2	80.1	76.8

As a result of the altered grouping of the angles, they were sorted into 8 groups instead of 9. The minimum separation between two groups changed from 4.48° to 4.88°. As a result of the new bounds on the angle ranges, tuple error tolerances are more widely varied than before. Error tolerance for some tuples increased to 8°, while the most error-sensitive tuple saw its maximum error tolerance fall from 2.74° to 2.52°. All of the experiments relating to FNPE2 (Sections 5.2 and 5.3) were performed using these updated parameters.

Chapter 3 Normal Extraction Methods

Two methods were investigated to extract normal vector measurements from the input point cloud. The first is referred to as FNPE1. Due to some inherent limitations of the method (it is not robust to noisy data), it was later replaced by a different method, referred to as FNPE2. The conclusions drawn from the results of the FNPE1 experimental simulation led to the development of FNPE2 (this is discussed in Section 5.1.5). The FNPE2 method is more robust to noise and outliers, and is more computationally efficient. FNPE1 has been included here because it is functional (if only for low-noise point clouds), and it provides context for the development of FNPE2.

3.1 FNPE1: Measurement of Face Normals Using Local Surface Reconstruction

When reconstructing a model from a point cloud, the data points must be converted into surfaces. One method of doing this is to select a point, and use its neighbours to infer the normal vector of the surface represented by that point. This process, repeated for every point in the point cloud, will result in an array of "point normals" associated with the array of points. For a shape consisting of flat surfaces such as the RAC, neighbouring points which have similar normals are assumed to belong to the same surface. If surfaces are found in this way, then the points that remain are identified as edges (or outliers, if real data are used).

Figure 3.8 shows a general overview of the data processing chain that was used in the FNPE1 algorithm.

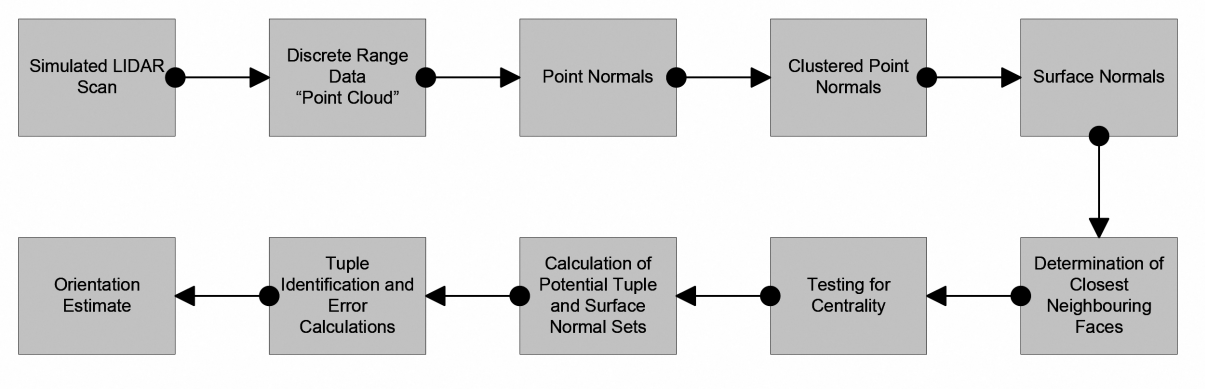

Figure 3.8: Data processing chain.

Before the FNPE1 algorithm can be applied, the input point cloud must be processed to extract appropriate sets of surface normal vectors, and their corresponding angles. Point normals, $\boldsymbol{n}_p$, are estimated at each point using a variation of the algorithm developed by OuYang and Feng [18]. This method is based upon reconstruction of the local geometry at each point, and as such requires that the Voronoi neighbours of each point be known. Matlab has a built-in function for calculating the Voronoi vertices and cells of the 3D Voronoi diagram of a set of input points. After applying this function, each point in the point cloud is in the centre of a Voronoi cell. These cells are defined by a set of Voronoi vertices. Given a point in a Voronoi cell, the neighbours of that point are simply the points in the adjacent Voronoi cells. Adjacent cells are here defined as cells which share at least three Voronoi vertices (i.e. they share at least one facet). To find these neighbours, a technique from graph theory was used, much as described in section 3.2.4, except using a connectivity of three instead of two (i.e. the faces referred to in the example are now Voronoi cells, and the edges are the common facets). The neighbours of each point in the point cloud are then determined. These Voronoi neighbours are referred to henceforth as "global neighbours" since the nature of the Voronoi diagram can result in a point having a Voronoi "neighbour" that is in fact located on the other side of the point cloud (in addition to the neighbours in the immediate vicinity).

With the global neighbours found, the following procedure is used to find the point normals. This procedure is taken directly from [18], as is Figure 3.9 and Figure 3.10.

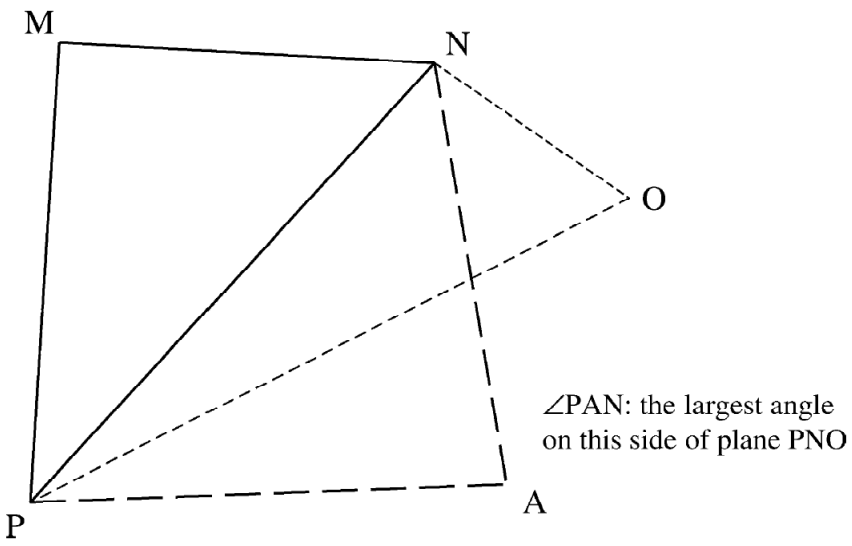

Figure 3.9: The best local mesh growing point for a triangle's edge.

1. Find the point N in GVN that is nearest to P (N is considered as the most credible local neighbour of P).
2. Use PN as the diameter and pivot a size-growing ball around PN until the ball hits another point M by evaluating all the points in the common GVN of P and N (M is considered as the second most credible local neighbour of P). Given the triangle NPM, identify the best local mesh growing point A for PN in the common GVN of P and N according to the following two criteria (Figure 3.9): (1) A and M must be on different sides of the plane PNO, where O is the first pole ball center of P (the farthest vertex of the Voronoi cell of P); and (2) PAN has to be the largest in order to have the smallest ball enclosing the points P, N and A. The best local mesh growing point B for PM is found with the same criteria.
3. Repeat the identification of the best local mesh growing points for the two newly added edges PA and PB and continue until the local mesh is connected or overlapped. All the local mesh points are then taken as the valid neighbouring points of P for the estimation of its normal vector.

Having found the local neighbours, the normal is found by fitting sets of quadric curves to the local points (see Figure 3.10) as follows: First, find the corresponding point P_j of each local

Voronoi mesh neighbour P_i with the largest angle through P_0:

$$\angle P_i P_0 P_j > \angle P_i P_0 P_m, \quad 1 \le m \le K, \ m \ne j \tag{9}$$

Where the constant K denotes the number of triangles in the local mesh. Next, fit a quadric curve $\boldsymbol{P}(u)$ through P_i, P_0 and P_j:

$$\begin{gathered} \boldsymbol{P}(u) = \boldsymbol{a}_0 + \boldsymbol{a}_1 u + \boldsymbol{a}_2 u^2, \ u \in [0,1] \\ \boldsymbol{P}(0) = P_i \ ; \ \boldsymbol{P}(1) = P_j \ ; \ \boldsymbol{P}(u_0) = P_0 \\ u_0 = \frac{\overline{P_i P_0}}{\overline{P_i P_0} + \overline{P_0 P_j}} \end{gathered} \tag{10}$$

The directional unit tangent vector $\boldsymbol{v}_i$ is then derived at P_0 from the fitted quadric curve:

$$\boldsymbol{v}_i = \left(\frac{\boldsymbol{a}_1 + 2\boldsymbol{a}_2 u_0}{|\boldsymbol{a}_1 + 2\boldsymbol{a}_2 u_0|} \right) \tag{11}$$

The normal vector $\boldsymbol{n}$ is found at P_0 by minimising the variances s^2 of the dot products of $\boldsymbol{n}$ and the K directional tangent vectors

$$s^2 = \frac{\sum_{i=1}^{K} \left(D_i - \overline{D} \right)^2}{K - 1} \tag{12}$$

where D_i and $\overline{D}$ are defined as follows:

$$\begin{gathered} D_i = \boldsymbol{n} \cdot \boldsymbol{v_i} \\ \overline{D} = \frac{\sum_{i=1}^{K} D_i}{K} \end{gathered} \tag{13}$$

Equation (12) can be rewritten as

$$s^2 = \frac{\sum_{i=1}^{K} [\boldsymbol{n} \cdot (\boldsymbol{v}_i - \overline{\boldsymbol{v}})]^2}{K - 1} \tag{14}$$

where

$$\overline{\boldsymbol{v}} = \frac{\sum_{i=1}^{K} \boldsymbol{v}_i}{K} \tag{15}$$

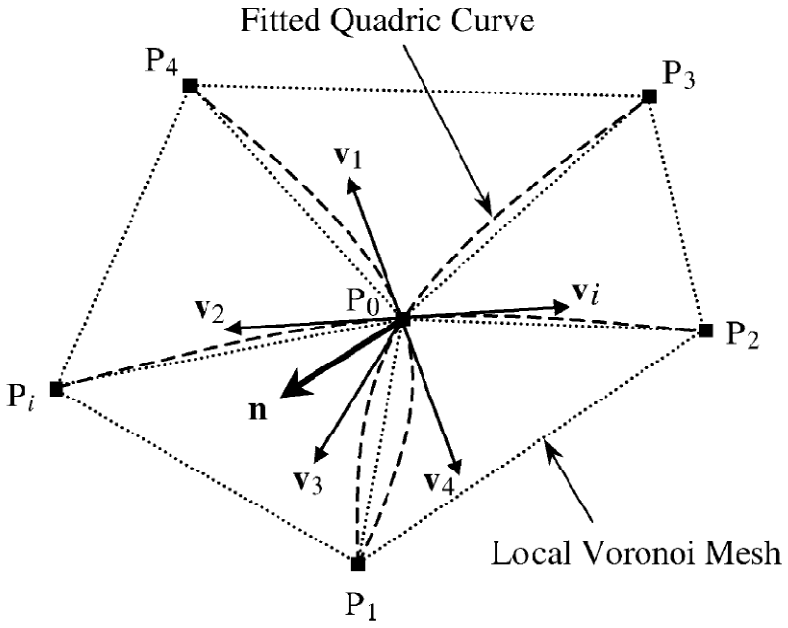

Figure 3.10: Finding the normal vector n at a specified point P_0.

Let the $\boldsymbol{v}_i - \overline{\boldsymbol{v}}$ vectors be the K row vectors of a $K \times 3$ matrix $\boldsymbol{A}$. Using singular value decomposition, matrix $\boldsymbol{A}$ can be formed as:

$$\boldsymbol{A} = \boldsymbol{W} \begin{pmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{pmatrix} \boldsymbol{N}^T \tag{16}$$

where $\boldsymbol{W}$ is a $K \times 3$ matrix having orthogonal column vectors and $\boldsymbol{N}$ is a $K \times 3$ matrix in which the three column vectors are eigenvectors with λ_1, λ_2 and λ_3 as the three eigenvalues. The eigenvector corresponding to the smallest eigenvalue is the normal vector that minimises the variance formulated in equation (14).

Although OuYang and Feng proceed to describe a method of determining the inward or outward direction of the point normals, is was found to be unnecessarily complicated for the present application. Instead, a vector was defined pointing from the centroid of the point cloud to the origin (the camera). The dot product of this vector and any correctly oriented normal vector should be positive, since the camera can only register surfaces oriented towards it. This observation is used to "correct" any point normals deemed to be pointing the wrong way.

Normal vectors for each visible face are found by clustering the resultant point normal vectors. If all of the (unit) point normals are plotted at the origin, then similar normals will appear as clusters around a unit sphere (see Figure 3.11, left). The unit vector pointing to the centre of each cluster is the average of all the normals in the cluster. A built-in Matlab clustering algorithm was used to identify these clusters, and normal vectors associated with them. Each point normal plotted on this unit sphere is associated with a point in the point cloud. A cluster of point normals can therefore be used to identify a subset of the point cloud, consisting of points which have similar point normals. If these points are then clustered spatially, there should be only one large cluster, corresponding to one of the surfaces scanned. Any other points can be discarded as outliers from one of the other faces. The centroid of these points is then taken to be an approximation of the geometric centroid of the scanned face (see Figure 3.11, right). This process is then repeated for each cluster of point normals.

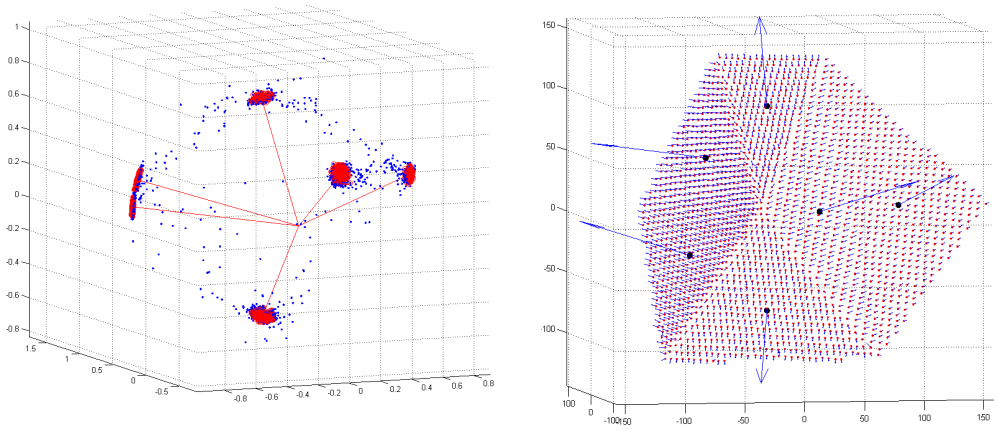

Figure 3.11: Clusters of normal vectors (left), and average normal vectors from each cluster (right).

The catalogue lookup step must determine which of the detected face normals represents the central face, and which are neighbours. Each visible face must be tested to see if it can form an appropriate pattern with those nearby. For a face to be eligible, it must be central (call it face A), with three faces arrayed around it (call them B, C, and D) as in the arrangement shown in Figure 2.4. (Note that the A,B,C,D notation used to describe centrality is arbitrary, whereas that used in Figure 2.4 is not. The process of identifying the required pattern of faces is separate from the process of ordering angles and normals into the specific sequence used in the reference

catalogue). The FNPE1 algorithm identifies appropriate patterns by means of an elimination process followed by a brute-force permutation of the remaining faces to look for matches.

The Euclidian distance from the centroid of face A to the nearest points in all visible faces is calculated. The minimum three distances should correspond to the surrounding faces B, C, and D. This requires that all the points corresponding to the visible faces are known. The clustering approach (as implemented here) neglects some points, and so combinations of the nearest four faces must sometimes be used to ensure that faces B, C, and D are found. Every such combination is subjected to the tests described below.

Having found the three faces closest to face A, face A must then be tested for centrality (i.e., every neighbouring face must share a side with the central triangle). The test is performed as follows: project the centroid of face A onto the plane formed by the centroids of faces B, C and D. If the projection of centroid A falls outside of triangle BCD, it is not considered a central face. Faces A B C and D are each tested in this way. Any combination passing this test proceeds to the catalogue lookup step as an unordered tuple/normal set.

3.2 FNPE2: Measurement of Face Normals Using Plane-Fitting

As the catalogue lookup step requires inputs based on surface normal vectors, planar surfaces must be extracted from the noisy data (point cloud). In a point cloud consisting of more than 7000 - 10000 points, point-by-point iterative methods become inefficient. Although the scans from the LCS have error characteristics that prevented them from being used with the FNPE1 algorithm, they are nevertheless good scans that show the target surface features with high resolution. When observing the 3D point cloud with the naked eye, the surfaces are clearly visible despite the edge effects and waves in the data. In this situation, the eye is in fact fitting known shapes to the data points, in this case planes. A random sample consensus approach (RANSAC) for plane-fitting was therefore investigated [10], to extract geometric information from the point cloud while minimising computations.

RANSAC is a robust heuristic for finding consistent data in the presence of large numbers of

outliers. A random sample is taken from the data and tested according to given criteria, and the results stored. This process is repeated many times. Whenever a sample is found which is a better fit to the criteria, that sample is kept and the old one discarded. In the present work, RANSAC is used to find triangular planes (i.e. shape facets) in point cloud data. The algorithm works by selecting three (unique) points at random from the data and using them to define a plane. As the algorithm iterates, any planes having more desirable characteristics (i.e. a better fit to the data) are kept and the previous one discarded. After a predetermined number of iterations, a plane in the data is assumed to have been found. The number of iterations required for this assumption to be valid depends on the number of points in the point cloud, and the number of planes represented in the data. Upon finding a plane, all the points that were "fitted" to it are recorded and removed from the point cloud, and the procedure repeated to find subsequent planes. A very basic representation of the algorithm is shown below:

Input: raw (but cropped) point cloud
Output: target pose estimate
while new triangles meet given *criteria*
    find a best-fit plane in *pointlist*
    store the plane parameters
    find a best-fit triangle for the points on that plane
    check fitted point / triangle data against given *criteria*
    if *criteria* have been met
        store the triangle vertices, and the points that match that triangle
    else
        end while
    end if
    remove the fitted points from *pointlist*
end while

Figure 3.12: RANSAC plane-fitting algorithm

The need for triangle fitting

Given the geometry of the RAC, this plane fitting process is insufficient. Well-fitted planes will frequently cut into other parts of the point cloud. When points associated with these planes are removed, some of the data representing the remaining surfaces are lost. As more planes are found this data loss continues, occasionally removing entire surfaces from the point cloud. To prevent this, points must be returned to the point cloud that fit the plane but do not represent the face that the plane is modelling. As all the faces of the RAC are triangular, the algorithm takes all the points associated with a given solution plane, and uses a second RANSAC procedure to search for a best-fit triangle. A well-fitted triangle will have a large number of points in a small planar area. Any points that do not fall within the best-fit triangle are returned to the point cloud. Note that this approach works best if the entire target is scanned.

The need for face vertices

In order to determine the relative positioning of the faces, all the detectable triangular faces are stitched together to form a continuous faceted surface. Since some of the vertices of such a surface are common to more than one face, an incidence matrix can be constructed that shows the relationships between the faces and the vertices. The incidence matrix can then be used to find the connection matrix, from which the relative locations of the faces can be extracted. With this information, experimental tuples can be easily calculated and fed into the catalogue lookup algorithm. The general processing chain is shown in Figure 3.13.

3.2.1 The Plane-Fitting Process

In order to reduce the number of iterations required, a restriction is placed on the random point selection. Rather than picking the three points from the complete dataset, a shell with inner radius $\mathbf{r}_1$ and outer radius $\mathbf{r}_2$ is defined around a random seed point. The three randomly selected points are then taken from this region, increasing the chance that they will define a plane that matches the surrounding data. The outer radius $\mathbf{r}_2$ of the shell was chosen to be slightly smaller than the inradius of the smallest face of the RAC that is visible in the LCS scans. This increases the chances that the three random points will all be from the same face, even when matching the smaller faces. The number of iterations used in the simulation and in the experiment is $N_1 = 750$

(this is also equal to the number of iterations used for triangle fitting, $N_2 = 750$. See section 3.2.2). This number is somewhat arbitrary, as alteration of the operating parameters of the algorithm will change the number of iterations required to get a given accuracy within a given number of iterations[5].

Once the three random points have been selected, they are used to define a plane. The quality of the fit is determined by the number of points found within a given threshold ($\mathbf{t}_1$) of the plane, as well as the standard deviation of the distances of those points. The plane parameters and the fit quality are saved, and the procedure repeated N_1 times. As new candidate planes are found, the fit quality is compared to the saved values. If a new candidate plane can be fitted to more points, or if the plane fits the same number of points but with a lower standard deviation, the new values overwrite the old ones.

The threshold value $\mathbf{t}_1$ is typically very narrow (+/- 0.35mm). Although this increases the accuracy of the solution plane, it also tends to result in multiple planes being found for a given face, due to the outliers remaining beyond the threshold (see Figure 3.14). It also results in relatively few points being associated with the solution plane itself, which is problematic when finding the triangular boundaries of the plane. Fitting the plane using a wider margin, however, encourages cutting across two planes, if they meet at a shallow angle as shown in Figure 3.15.

As the determination of normal vectors is a critical part of this method of pose estimation, planes are found using the narrow margin. The outliers are incorporated into the set of fitted points during the triangle-fitting portion of the algorithm. It is important to do this rather than allowing duplicate planes to be found, due the method used to identify adjacent faces described in section 3.2.4.

[5] The number 750 was chosen simply to provide a particularly high accuracy, given the operating parameters chosen. A lower value could possibly have produced very similar results at greater speed, but since the algorithm was not being specifically optimised for speed, the selection of the number of iterations was not examined in detail.

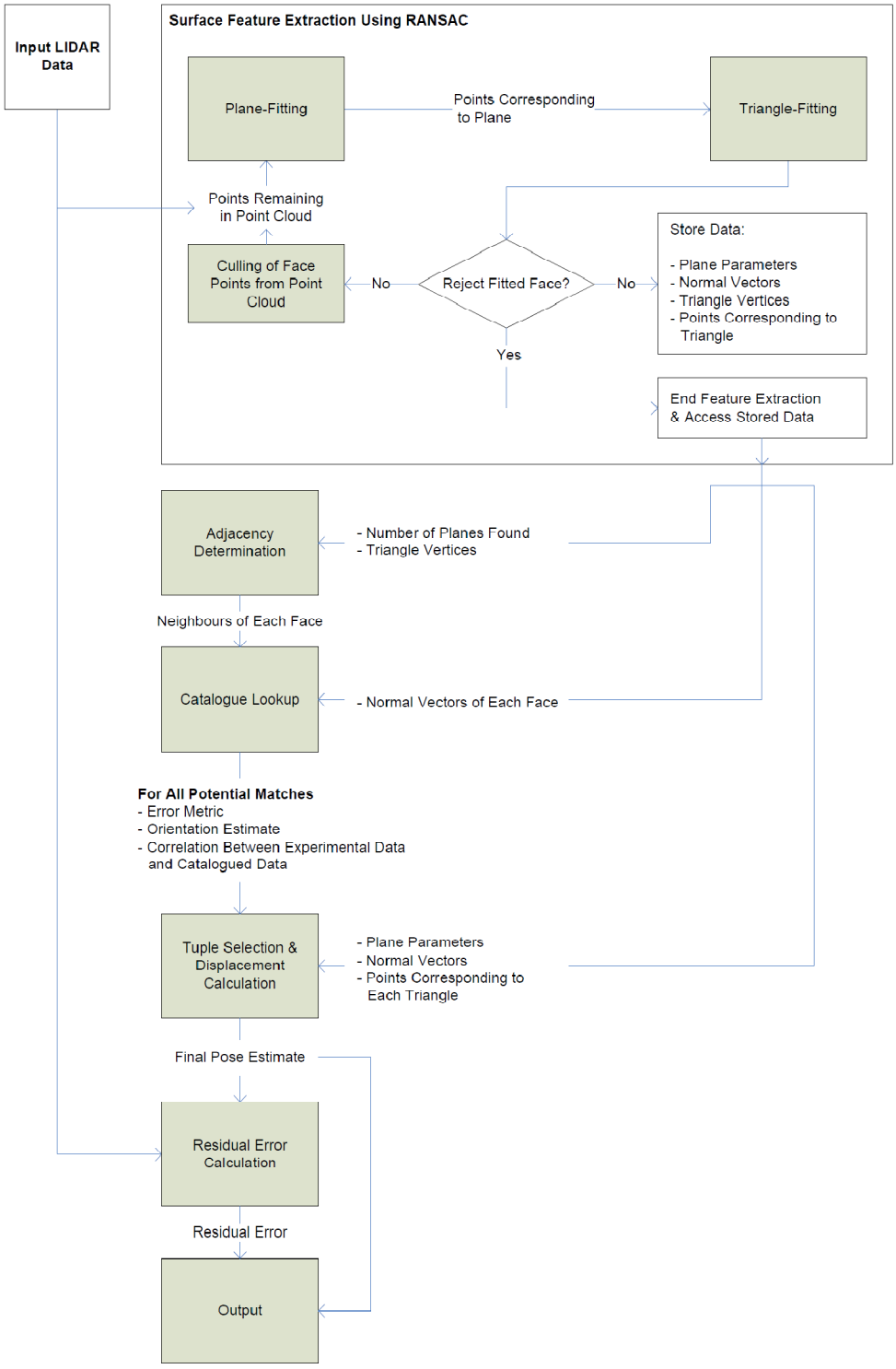

Figure 3.13: FNPE processing chain.

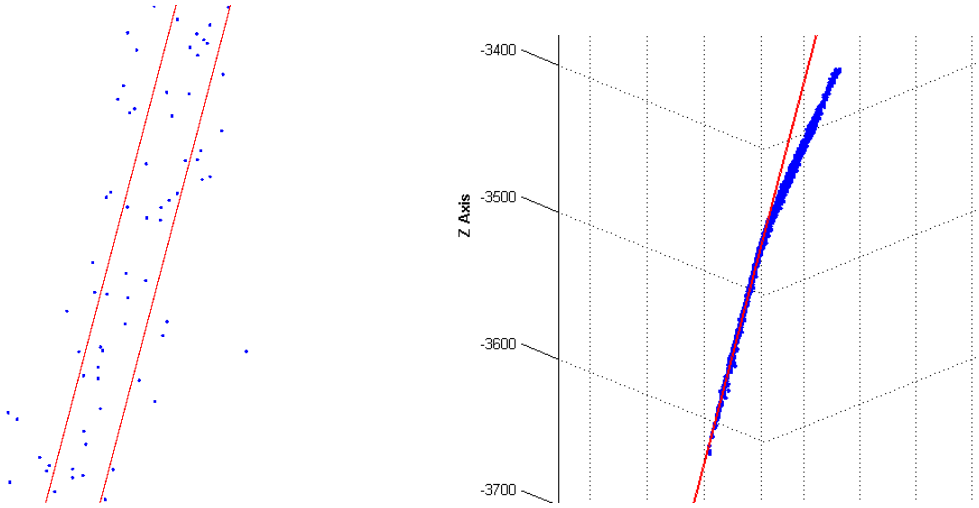

Figure 3.14: Effect of a narrow plane-fitting threshold.

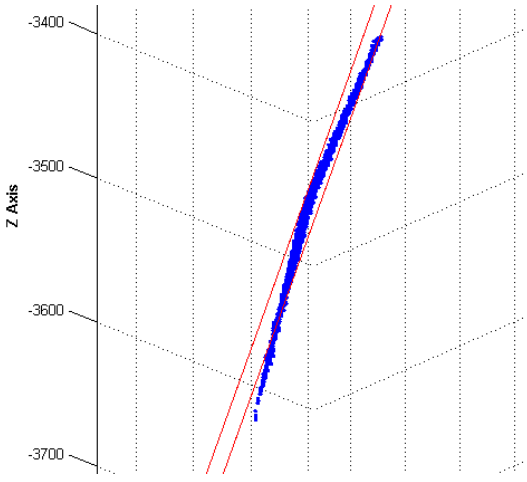

Figure 3.15: Effect of overly wide plane-fitting threshold.

The general plane-fitting procedure is summarised below.

1. Pick a random seed point
2. Determine a shell (using r_1 and r_2) around that point
3. Pick three random (but unique) points from within that shell, determine the corresponding plane
4. The number of points within t_1 of the plane is calculated, as well as the standard deviation of their distances to the plane. High numbers of points with low standard deviation are desired.
5. Keep the best plane found after a given number of iterations (N_1 iterations, or (number of outliers)/15 iterations, whichever is greater)

3.2.2 The Triangle-Fitting Process

Given the nature of the shape, an infinite solution plane that matches one surface may very well cut through another part of the target (see Figure 3.16). Since all the faces on the target are triangular, and it is assumed that the entire target was imaged, the desired triangular solution plane should encompass the greatest number of points with the highest point density. A RANSAC-based algorithm is used to find the best-fit triangle.

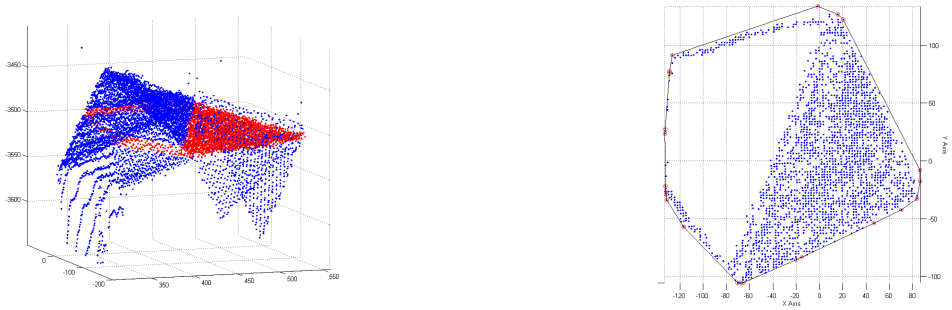

Figure 3.16: A plane identified in a point cloud (left), and an overhead view of the fitted points (right). The connected points in the right-hand figure represent the (2D) convex hull.

Once a plane has been fitted to a set of points, several things need to happen. First, the plane must be collapsed into a triangle. Second, the outlying points (in the perpendicular direction) must be incorporated into the set of fitted points, so as to avoid finding duplicate planes. In order

to collapse the plane into a triangle, the fitted points are projected into the plane, and the convex hull calculated. The algorithm then picks random sets of three points from within the convex hull, and uses them as test vertices to construct a triangle. Better-fitted triangles will contain a larger number of points, with a higher point density (see Figure 3.17).

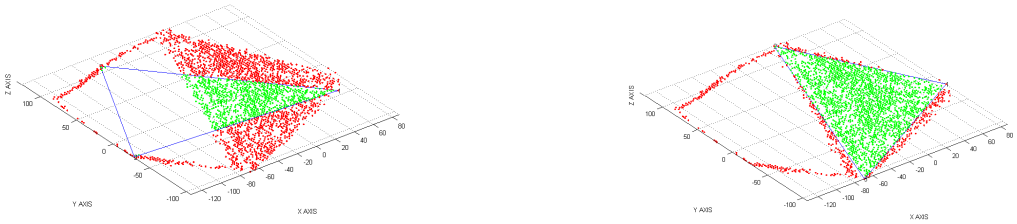

Figure 3.17: A badly-fitted triangle (left), and a well-fitted triangle (right).

Ideally, the edges of the triangle will coincide with the actual locations of the edges of the face. Since the narrow fitting threshold t_1 can sometimes exclude points from the immediate neighbourhood of the true vertices of the face (see Figure 3.18 (left)), points within threshold $\mathbf{t}_2$ = 2.0 [mm] of the solution plane are added to the set of fitted points. This increases the chances that points near the correct locations will be found (see Figure 3.18 (right)). Note that the magenta boundary in the figure represents the true boundary of the face.

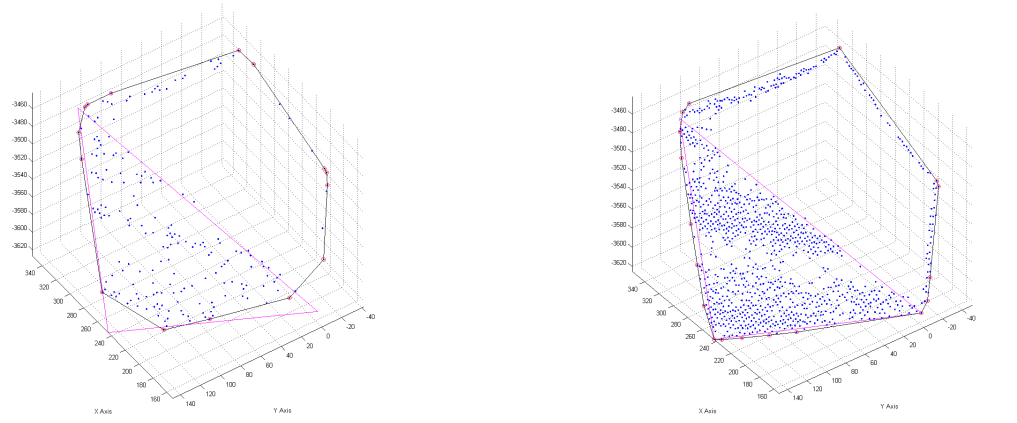

Figure 3.18: Points fitted to the solution plane only (left), and the effect of adding points from a wider margin (right).

It is important not to make $\mathbf{t}_2$ too large, as this will include larger numbers of points from other faces (if the faces are connected by a shallow angle). In this situation, a triangle that has been positioned to include the largest number of points with the highest point density may in fact be in the wrong orientation. The following figures show the effect of different inclusion thresholds on the triangle-fitting process. The thresholds represented in the figures are for t = 0.35 , 2 , 5 , and 20 [mm], respectively.

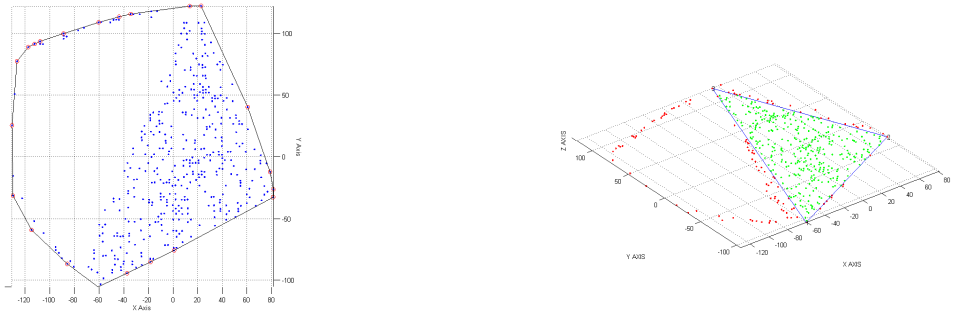

Figure 3.19: Triangle fitting after addition of points within t_2 = 0.35[mm] of the solution plane (i.e. no extra points added).

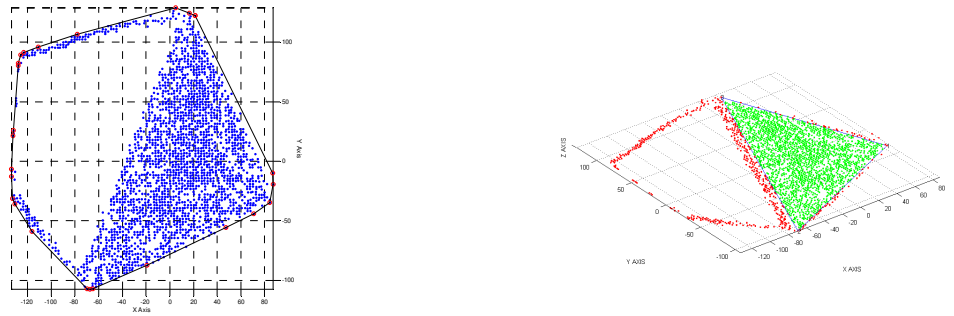

Figure 3.20: Triangle fitting after addition of points within t_2 = 2.00[mm] of the solution plane. (The bounds of this triangle are correct).

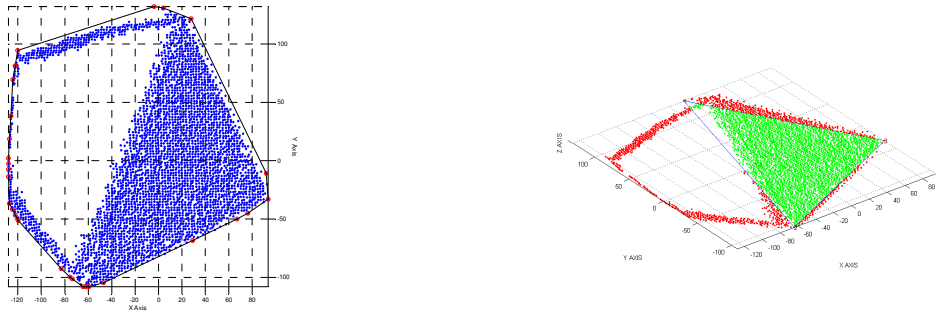

Figure 3.21: Triangle fitting after addition of points within t_2 = 5.00[mm] of the solution plane.

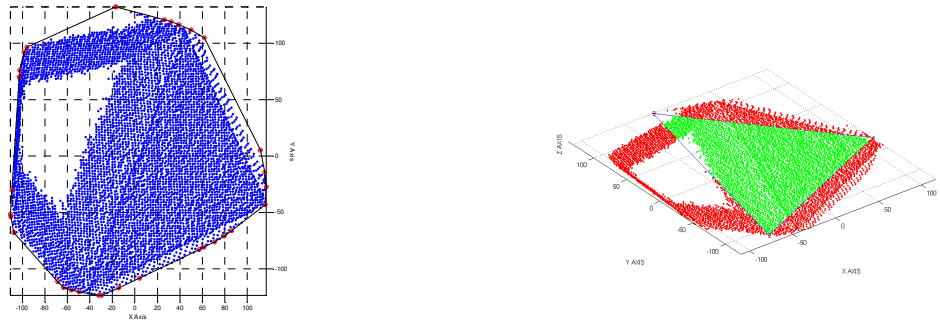

Figure 3.22: Triangle fitting after addition of points within t_2 = 20.00[mm] of the solution plane.

Once the vertices of the triangle have been chosen, the rest of the outlying points up to threshold distance $\mathbf{t}_3$ ($\mathbf{t}_3$ = 5.0 [mm]), are tested to see if their projections into the plane fall within the triangle. If they do, they are included, otherwise, they are returned to the point cloud. This helps ensure that almost all points from a given face become associated with the fitted triangle, while preventing the triangle projection threshold $\mathbf{t}_3$ from cutting too far into other faces (in the event that the triangle is not perfectly oriented).

A summary of the triangle fitting procedure is shown below.

1. Perform triangle search with the point cloud subset that "matches" the infinite solution plane found previously.

2. Project the points into the plane.
3. Calculate the convex hull of the planar points. The convex hull of the planar points forms the search space for the best-fit triangle.
4. Select three random (unique) points to define a triangle.
5. Calculate the number of points that fall within this triangle[6], as well as the point density.
6. Iterate N_2 times, or (number of outliers)/15 times, whichever is greater.
7. Remove all points that are contained in (or "represented by") the resulting triangle, from the point cloud.

The plane-finding and triangle-finding process are repeated until the best-fit triangle has a point density less than *d*, or a number of points less than *num*. Given the noise in the data, the best-fit plane for a given surface/face may leave a significant number of (low point density) leftover points in the point cloud, that were too far away from the solution plane to be included (see Figure 3.23). The leftover points will be too widely scattered, or will be too few in number to generate reliable surface normal vectors. (i.e. widely distributed higher-noise fringe data, or lines of points excluded by the triangle-fitting process)

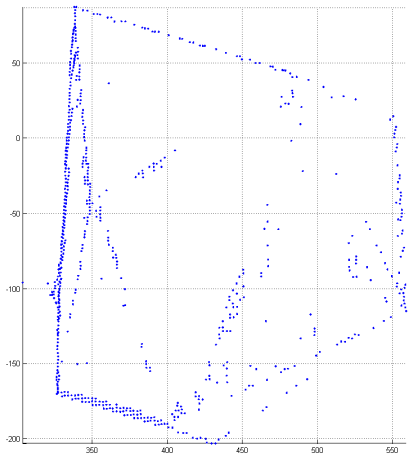

Figure 3.23: Leftover fringe data.

[6] This procedure is described in the appendix

Planes fitted to these points will be the result of higher-noise fringe data, and the corresponding normal vectors will likely be of lower accuracy than those found from lower-noise points, or they will represent entirely fictitious surfaces.

3.2.3 Determining Face Adjacency

The input to the catalogue lookup step cannot be determined unless some information is known about the adjacency of the triangular faces. Before this adjacency information can be determined, the individual triangles must be stitched together. The triangles found from the two RANSAC processes will, as such, not contain any shared vertices (i.e. vertices common to more than one triangle). For the purpose of identifying adjacency (i.e. NOT for normal calculation), vertices within *x-dist y-dist* and *z-dist* of each other are averaged (spatially) and are shared between the corresponding triangles. Since each triangle defines a face, graph theory can be used (as described in section 3.2.4) to determine face adjacency using the graph's connection matrix. Once the relative arrangement of the faces is known the input to the catalogue lookup algorithm can be calculated. The main steps in the procedure are summarised below:

1. Collect all triangle vertices into an *n*x*3* matrix
2. Begin at the first vertex, and find all other vertices within the required distance
3. Average these vertices, and replace each original vertex with the averaged one.
4. Do this for all the vertices
5. Generate the connection matrix as previously described

   The threshold for connectivity = 2 (i.e. if triangles share two vertices, they share an edge, and are therefore “adjacent”).

3.2.4 Adjacency Determination Using Graph Theory

FNPE2 generates a face-vertex[7] representation of the observed surface of the RAC, using point cloud data. In order to identify a unique set of faces with the catalogue lookup algorithm, a face must be found that shares an edge with three adjacent faces. This can be easily done using graph theory.

---

[7] The vertices are stored as a list of coordinates, and each face is defined as a set of indices into the vertex list. These indices are not stored in any particular order.

A graph is a set of linked nodes. In general, the nodes and links can represent various things (depending on the application), but in the present circumstances the nodes are vertices and the links are the edges of triangular faces. An example of a graph ***G = (V,F)*** is shown in Figure 3.24.

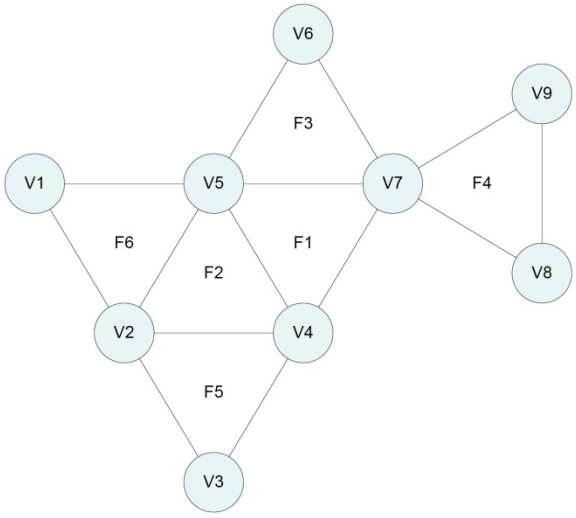

Figure 3.24: An example of a graph G = (V,F).

The faces and vertices need not be labelled in any particular order. The incidence matrix of this graph (see Figure 3.25) is a |V| x |F| matrix $\boldsymbol{B} = (b_{ij})$ defined such that

$$b_{ij} = \begin{cases} 1 & \text{if vertex } i \text{ is connected to face } j \\ 0 & \text{otherwise} \end{cases}$$

V \ F	1	2	3	4	5	6
1	0	0	0	0	0	1
2	0	1	0	0	1	1
3	0	0	0	0	1	0
4	1	1	0	0	1	0
5	1	1	1	0	0	1
6	0	0	1	0	0	0
7	1	0	1	1	0	0
8	0	0	0	1	0	0
9	0	0	0	1	0	0

Figure 3.25: The incidence matrix B of graph G

The connection matrix $\boldsymbol{C = B^T B}$ is a symmetric |F| x |F| matrix. $\boldsymbol{C} = (c_{ij})$ shows the number of vertices common to faces *i* and *j* (see Figure 3.26).

F	1	2	3	4	5	6
1	3	2	2	1	1	1
2	2	3	1	0	2	2
3	2	1	3	1	0	1
4	1	0	1	3	0	0
5	1	2	0	0	3	1
6	1	2	1	0	1	3

Figure 3.26: The connection matrix C of graph G

Adjacent faces share an edge, and so will share two vertices. For FNPE2, the required pattern of faces is therefore present if a face shares two vertices with three different neighbours, as with face number 2 in Figure 3.24. The diagonal elements are ignored, as any face will share three vertices with itself.

3.2.5 Processing data into and out of the lookup table

Central faces can now be defined as any triangle with a connectivity of three or more (although in an ideal situation the number is exactly 3, extra planes that failed to be weeded out can raise the connectivity of a correctly-fitted face past 3). Plane-to-plane angles are then calculated for every face that has *at least* three neighbours. If the face has more than three neighbours, angles are generated for every combination. All of these angular sets, together with their corresponding face normals, are evaluated by the catalogue lookup algorithm. The output with the lowest (internally evaluated) error is used as the orientation estimate.

3.2.6 Least-Squares Displacement Estimate

The normal vectors that were used to generate the (lowest error) orientation estimate correspond to the surfaces found previously. As such, they are also associated with the distinct sets of points that were fitted by the RANSAC portion of the algorithm. Since the catalogue lookup determines, by necessity, the correspondence between the input surface normals and the tabulated normals, each set of planar points from the point cloud can be associated with a specific reference plane from the model. The orientation estimate is used to rotate the reference

model to (approximately) match the orientation of the point cloud. Using a least squares approach, a single displacement vector is then found that simultaneously minimises the distances between each set of points and its corresponding reference plane (see Figure 3.27).

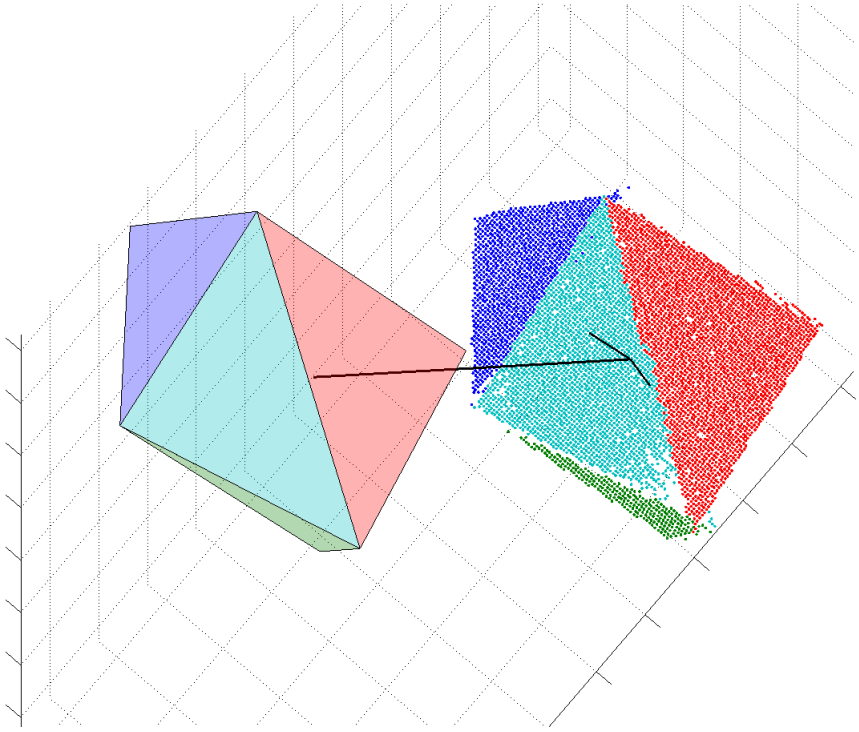

Figure 3.27: Translation to fit rotated reference planes to corresponding point cloud subsets.

Given a plane of the form $ax + by + cz + d = 0$ with normal vector $N = [a\hat{\boldsymbol{\imath}} \quad b\hat{\boldsymbol{\jmath}} \quad c\hat{\boldsymbol{k}}]$ and a set of n points $P = [x_i \quad y_i \quad z_i]$ (where $1 \leq i \leq n$) to be fitted to the plane, the following holds true only if the points in P lie directly on the plane: $ax_i + by_i + cz_i + d = 0, \forall i$. If the points do not all lie directly on the plane, they can be fitted by finding a translation vector that minimises the point to plane distances in a linear least squares sense. The scalar response vector of the linear regression model for the k^{th} plane and it's corresponding point set is

$$e_k = N_k \cdot P_k + d_k \tag{17}$$

where:

- n_k is the number of points associated with the k^{th} plane
- N_k is an $n_k \times 3$ matrix, whose rows each consist of the normal vector of the k^{th} plane
- P_k is an $n_k \times 3$ matrix, whose rows are the position vectors of each point associated with

the k^{th} plane

- d_k is an $n_k \times 1$ matrix, whose rows each consist of the plane parameter d of the k^{th} plane

the design matrix for the k^{th} plane is simply

$$A_k = N_k \tag{18}$$

resulting in a total scalar response vector of

$$e = \begin{bmatrix} e_1 \\ \vdots \\ e_k \end{bmatrix} = \begin{bmatrix} N_1 \\ \vdots \\ N_k \end{bmatrix} \cdot \begin{bmatrix} P_1 \\ \vdots \\ P_k \end{bmatrix} + \begin{bmatrix} d_1 \\ \vdots \\ d_k \end{bmatrix} \tag{19}$$

with a total design matrix of

$$A = \begin{bmatrix} N_1 \\ \vdots \\ N_k \end{bmatrix} \tag{20}$$

The translation vector that minimises the point to plane distances for all k planes is then found from

$$r = [A^T A]^{-1}[e] \tag{21}$$

If the geometry of the faces has been accurately reconstructed, then when the model is displaced by this vector, it will match the point cloud very closely. Using the original input point cloud, the residual pose error is then taken to be the root-mean-square of the minimum point-to-model distances.

3.2.7 Irregular least-squares improvement

Typically, only 4 sets of points (i.e. the points associated with the faces needed to find one solution) are available for the least-squares fitting process. However, if two sets of faces are found, and if the catalogue lookup error metric for the second set is below an acceptable value (an empirically determined value of 10 was found to be effective), then any faces from the second set not found in the first are added to the least-squares fitting process.

Chapter 4 Data Sources and Experimental Overview

This chapter introduces (briefly) the different sets of point cloud data used in this thesis, and outlines any noteworthy attributes or characteristics.

4.1 Data Sources

Note that the following information is intended as an overview. Detailed descriptions are provided in Chapter 5.

4.1.1 Simulated Point Clouds used with FNPE1

The point clouds used in the first simulation were generated by projecting an array of rays from a given camera position, towards a computer model of the RAC. The intersections of the rays with the model determine the location of the error-free point cloud (see Figure 4.28). The model of the RAC was scaled to approximately match the size of the physical model (using the uniform scale factor of 250 [mm]. The range of the "camera" (the origin of the projected rays), as well as the number and angular spacing of the rays, was selected such that point clouds of 1700 to 2300 points were generated. Point clouds were generated from 2000 positions around the RAC, evenly distributed in viewing angle. Low intensity uniformly distributed noise (with an amplitude of ± 0.5 [mm]) was then generated and added to the coordinates of each point.

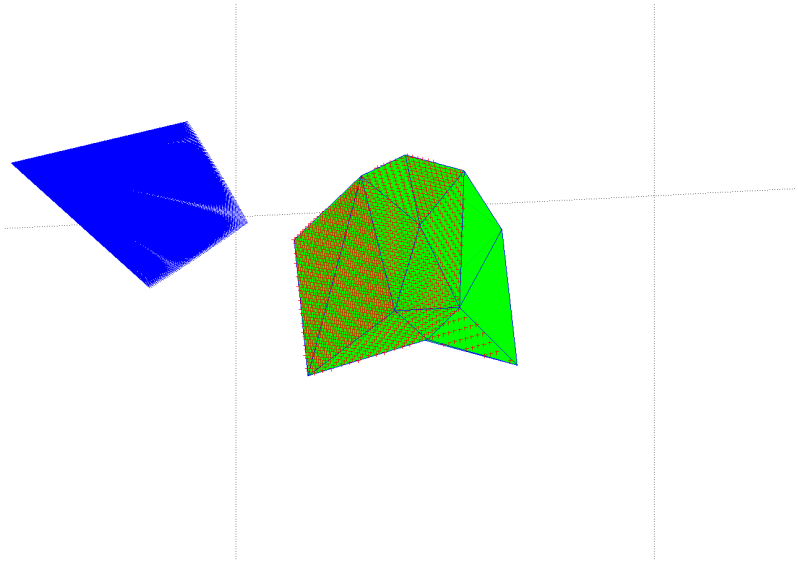

Figure 4.28: Projection of camera rays(the blue lines) onto the RAC, and the resulting ray-model intersection points.

4.1.2 Simulated Point Clouds used with FNPE2

For the second simulation, 2000 point clouds were generated from evenly distributed viewing angles, as in the first simulation. The model of the RAC was scaled differently in the X, Y, and Z axes, so as to match the geometry of the physical model as closely as possible. Both the range of the "camera" and scan density were chosen to approximate the range and scan density of the point clouds generated by Neptec's LCS. This resulted in point clouds of approximately 6-10 thousand points, all generated from a range of 3.6 meters. At this range, the majority of the points in the point cloud are near to 3.5 [m] distant from the camera. Since LIDAR range error is significantly more intense than lateral error, its effects will dominate the error characteristics of the point cloud. For this reason, as well as for simplicity, only range error was added to the point cloud. Gaussian error was added along the vectors from the origin of the camera to the error-free point coordinates. The standard deviation of the range error for the LCS (according to data from Neptec [17], and for a 3.5 [m] range) was used to generate the simulated error (3.3125 mm).

4.1.3 The Neptec LCS

The RAC was previously taken to Neptec and scanned from multiple perspectives with their LCS. As Neptec's LCS is designed specifically for use in spacecraft pose estimation (such as for orbital rendezvous and docking procedures), the LCS scans of the RAC are the most appropriate

source of data with which to test the pose estimation algorithm that has been developed.

The data that were received from Neptec are point cloud representations of the entire room in which the RAC was mounted. The point clouds were cropped in order to isolate the RAC from its surroundings, but the data were not cleaned or edited in any other way before being used in the pose estimation algorithms described later. Each view of the RAC consists of 6-10 thousand points, at an average point-cloud-to-camera range of approximately 3.5 meters.

The LIDAR data have some problematic attributes that make surface normal estimation difficult. Due to the nature of the LIDAR technology used, data points on or near acute angles can demonstrate range biases with respect to the surrounding points. These "edge effects" are seen in the data as walls of points projected either towards or away from the camera. In addition to this, surfaces that are in reality flat, are reproduced in the point cloud with a certain degree of undulation. These "waves" in the data are present, to varying degrees, in most views of the RAC. These artefacts are distortions of the range estimate of a given point, and as such tend not to be perceptible when viewing the data from the perspective of the camera. An example is shown in Figure 4.29.

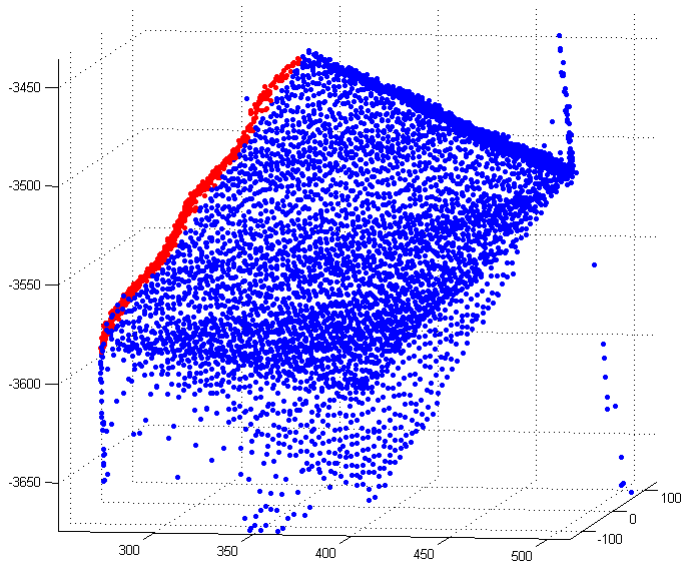

Figure 4.29: Profile view of LCS point cloud showing waves (shown in red) and edge effects.

4.2 Experimental Overview

In addition to a functioning catalogue lookup algorithm, a method for extracting normal vectors from LIDAR data is needed. Of particular importance is the measurement of angular tuples. As mentioned in section 2.1, a tuple measurement requires the identification of a central face and its three adjacent faces. The tuple itself is simply the set of three angles between the central normal and the normals of the adjacent faces. FNPE1 (Section 3.1) was designed to calculate the required normal vectors, and identify the arrangement in terms of central and adjacent faces. FNPE1 was used mainly to test the catalogue lookup algorithm, and to explore possible methods of extracting not only the normal vectors, but also the required geometric relationships between them. FNPE2 (Section 3.2) represents a more developed technique, intended for use with real LIDAR data. It has been tested using simulations, as well as with data from Neptec's LCS. The following is a brief summary of the two methods used, and an overview of the experiments performed using each method.

4.2.1 FNPE1 - Point-by-Point Surface Reconstruction

The immediate neighbours of every point in the point cloud are identified. The neighbours of a given point are then used to calculate a unit normal vector at that point. The endpoints of all of these vectors are then considered to be data points on the surface of a unit sphere. A Matlab clustering algorithm[8] is used to identify groups of normal vectors from the clustered points. These clusters signify a consistent planar surface, and the average of the normal vectors in the cluster is taken to be the normal vector of the surface. The vectors identified in the cluster each correspond to points in the point cloud, meaning that each surface normal (i.e. the mean of the point normals in a given cluster) can be associated with a specific point cloud subset. The relative positions of these "face points" are then used to identify central and adjacent normals. Tuples of angles are then calculated for any set of normals having the required arrangement, and the results are sent to the catalogue lookup algorithm. The orientation with the lowest catalogue output match error is then taken to be the best estimate of the orientation of the point cloud. The

[8] The Matlab function is called "subclust" . It identifies clusters one at a time, and stops when the established clustering conditions are no longer found (among the unidentified points that remain, if any). Further information can be obtained from the Matlab help file (version R2009b was used).

displacement estimate is only very roughly estimated, and is taken to be the position vector of the centroid of the point cloud, moved towards the camera by 2/5 of the maximum width of the model. The adjustment towards the camera is necessary because if the centroid of the computer model is coincident with the centroid of the point cloud, the point cloud will (more often than not) be inside the model. This causes ICP to behave unpredictably. This normal extraction technique is the basis of FNPE1.

Testing of FNPE1 with Simulated Data

Simulated point clouds from 2000 viewpoints around the RAC are processed using the FNPE1 algorithm to determine a pose estimate. This estimate is then used as the initial guess in an ICP algorithm, which is allowed to run for no more than 50 iterations. The number of iterations is limited, as an accurate initial guess should converge fairly quickly. Although a rough position estimate is generated for use as an initial guess in ICP, it is the orientation estimate that is being considered for this experiment. Both the direct orientation estimate and the results from ICP are compared to the known orientation of the simulated camera, and the results discussed.

4.2.2 FNPE2 - Plane-Fitting

Planes are fitted to the point cloud data, using an algorithm based on a Random Sample Consensus (RANSAC) [10]. Another RANSAC-type approach is used to isolate the points from a specific (triangular) face from any outlying points matching that plane. The triangular surfaces found are then stitched together into a continuous surface defined by vertices, with each face being identified as a set of indices into those vertices. Using a technique from Graph Theory, a connection matrix is constructed that is used to identify faces from which tuples of angles can be correctly calculated. The tuples are calculated for any such faces, and the results sent to the catalogue lookup algorithm. The orientation estimate with the lowest catalogue match error is then taken to be the best estimate of the orientation of the point cloud. This orientation estimate is also used to correlate points from specific faces identified in the point cloud, to individual planes from the reference catalogue. The reference planes are then rotated according to the orientation estimate, and the points fitted to their corresponding planes using a least-squares minimisation. This gives the displacement estimate that, together with the orientation, gives the final pose estimate.

Testing of FNPE2 with Simulated Data

Point clouds were generated from 2000 positions around the RAC, evenly distributed in viewing angle. The FNPE2 algorithm was then used to generate a pose estimate for each point cloud. The FNPE2 pose estimates were used as the initial estimates in an ICP algorithm, and a refined pose estimate was generated. The absolute pose error as well as the RMSE of the pre and post-ICP estimates were then compared.

Testing of FNPE2 with the LCS

Fifteen LCS-generated point cloud representations of the RAC were used in this experiment, each one depicting a different view of the RAC. The FNPE2 algorithm was used to generate 50 "correct" pose estimates of each point cloud, meaning that failed estimates were rejected, as well as any estimates wherein the reconstructed geometry was mistaken for that from a different region of the RAC (this second situation did not arise for the point clouds used, but such mistakes are technically possible and have occasionally been observed). Each estimate was used as the initial estimate in an ICP algorithm, from which a refined pose estimate was generated. Since no truth data were available for these datasets, the mean ICP pose for each point cloud was used as truth data. The relative (to the ICP pose) position and orientation error of the FNPE2 estimates was examined in relation to the pre and post-ICP RMS error. The results from these experiments were then compared to those from the simulation and discussed.

Chapter 5 Experiments

This chapter contains all the experiments that were conducted, using the previously described methods. FNPE1 was tested with simulated data. FNPE2 was tested with simulated data, as well as with data from Neptec's LCS.

5.1 Testing of FNPE1 with Simulated Data

This section describes the setup, procedure, evaluation, results and conclusions drawn from the FNPE1 simulation.

5.1.1 Setup and Point Cloud Error Characteristics

The point clouds used here are generated by projecting rays from a "camera" at a given range, towards a computer model of the RAC (see Figure 4.28). The rays originate from the same point, but are projected in a rectangular fashion: 70 rays across the top and side, with a ray separation (in both dimensions) of 0.2 [deg]. This yields point clouds of approximately 1700 to 2300 points for each view (This is similar to the number of points expected in a close-range scan of the RAC using a SwissRanger 4000 flash LIDAR). The intersections between the rays and the RAC were used as the initial error-free point cloud, to which noise was added. A rotation was then introduced resulting in a range that is always expressed along the same axis (in this case the x-axis) as would be the case for actual measurements.

It should be noted that the model of the RAC has maximum dimensions of 1x1x1 units. These units, as such, have no actual measurement units attached to them (m, cm, mm, etc…). For this simulation, the model has been increased in size by a scale factor of 250. If the scale factor is assumed to be in [mm], this makes the scaled shape roughly the same size as the physical model used in later experiments. The simulated point clouds were generated from a range of 1500 units to the centroid of the object. The error added to the point clouds is uniformly distributed, with an amplitude of +/- 0.5 units (This low noise level was selected so as to better test the theoretical functionality of the FNPE1 algorithm. More realistic noise levels are considered in the second

experiment). The results presented here will be valid for a model of any size, so long as the scan density, point cloud error and observation range are scaled accordingly.

5.1.2 Procedure

Point clouds of the RAC were generated from 2000 camera positions around the object, evenly distributed in viewing angle, with no boresight roll (see Figure 5.30). Each point cloud was processed by the FNPE1 algorithm, and the orientation estimate recorded. To validate this approach, this orientation estimate was used as an initial guess for ICP. The ICP algorithm was implemented using Horn's method [19] and was limited to 50 iterations. The centroid of the point cloud was used as the initial guess for the position of the object, adjusted away from the observer by 2/5 of the model scale factor. All orientations are calculated as quaternions.

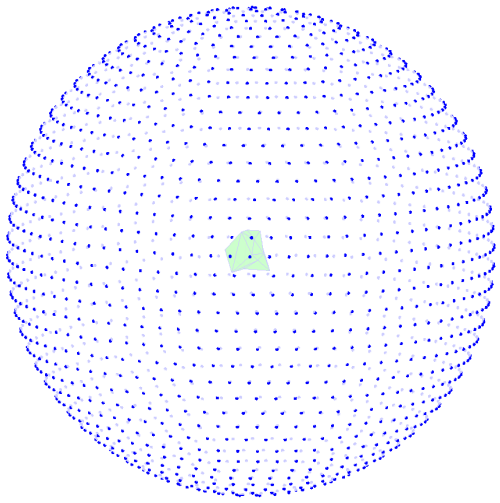

Figure 5.30: Distribution of viewing angles used to generate point clouds.

5.1.3 Orientation Error Metric

A four element unit quaternion **q,** used in attitude estimation, can be decomposed into a scalar component q_s, and a three-element vector component $\mathbf{q_v}$:

$$\mathbf{q} = \begin{bmatrix} q_s \\ \mathbf{q}_v \end{bmatrix}$$

The inverse quaternion $\mathbf{q}^{-1}$ can be found by negating the vector component of the original

quaternion:

$$\mathbf{q}^{-1} = \begin{bmatrix} q_s \\ -\mathbf{q}_v \end{bmatrix}$$

Successive rotations can be composed in quaternion notation:

$$\mathbf{q}_{\mathrm{c}} = \mathbf{q}_a \otimes \mathbf{q}_{\mathrm{b}}$$

where

$$q_{s,c} = q_{s,a} q_{s,b} - \mathbf{q}_{v,a}^{T} \mathbf{q}_{\mathrm{v,b}}$$

$$\mathbf{q}_{\mathrm{v,c}} = q_{s,b} \mathbf{q}_{\mathrm{v,a}} + q_{s,a} \mathbf{q}_{\mathrm{v,a}} + \mathbf{q}_{v,a}^{\times} \mathbf{q}_{\mathrm{v,b}}$$

The magnitude of the angular rotation that the quaternion expresses, can be easily extracted from the scalar component:

$$\phi = 2\cos^{-1} q_s$$

In the present work, absolute error is based on comparisons between estimated orientation quaternions, and true orientation quaternions. Comparisons between quaternions, say $\mathbf{q}_{\mathrm{a}}$ and $\mathbf{q}_{\mathrm{b}}$, were made by calculating the error quaternion $\mathbf{q}_{\mathrm{c}}$ required to rotate from $\mathbf{q}_{\mathrm{a}}$ to $\mathbf{q}_{\mathrm{b}}$. The angular displacement component of $\mathbf{q}_{\mathrm{c}}$ is the value used for comparison. Both the direct estimate and the final ICP estimate were compared to the known quaternion orientation of the model.

When estimating the orientation of the target, it is necessary to have an error margin within which an estimate can be said to be correct. This error margin was determined as follows: Using the target reference values, calculations were made of the rotations required to match one set of normal vectors onto any other set, as closely as possible. This is the type of mismatch that would occur if an orientation estimate was attempted for an incorrectly identified set of normals. The minimum magnitude of any such rotation was found to be approximately 22 [deg]. Note that this presupposes perfect measurement of the normal vectors. Since a certain degree of error is inevitable, the 22 [deg] margin should by no means be thought of as an absolute threshold for success or failure. For this reason, a more conservative threshold was used. Orientation errors within 10 percent of this "minimum orientation match error" (2.2 [deg]) were taken to be the result of a correct estimate.

5.1.4 Results

A summary is included in

Table 5.3. The table shows the number of successful and unsuccessful direct and ICP estimates and the correlation between them (top), as well as the corresponding absolute orientation error for each case (bottom). Of interest is the fact that every correct FNPE1 estimate resulted in a correct ICP estimate. ICP was even able to determine a correct orientation from a small number of incorrect FNPE1 estimates. As is evident from Figure 5.33, ICP sometimes resulted in a higher error than FNPE1.

Table 5.3: Summary of FNPE1 results

	Direct Estimate		Sucessful ICP		Unsucessful ICP	
	Matches	Percent	Matches	Percent	Matches	Percent
Successful Direct Estimate:	1741	87.1	1741	87.1	0	0.0
Unsuccessful Direct Estimate:	117	5.9	7	0.4	110	5.5
Failed to Make Direct Estimate:	142	7.1	-	-	-	-
Total:	2000	100				

	Direct Estimate			Sucessful ICP			Unsucessful ICP		
Absolute ϕ Error (deg)	Min	Mean	Max	Min	Mean	Max	Min	Mean	Max
Successful Direct Estimate:	0.0	0.4	1.3	0.1	0.4	0.9	-	-	-
Unsuccessful Direct Estimate:	13.9	145.4	179.4	0.3	0.6	1.0	2.6	157.1	179.9
Failed to Make Direct Estimate:	-	-	-	-	-	-	-	-	-

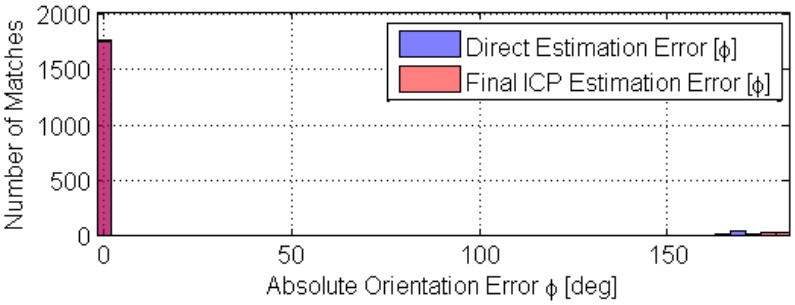

Figure 5.31: Absolute orientation error histogram (no scaling).

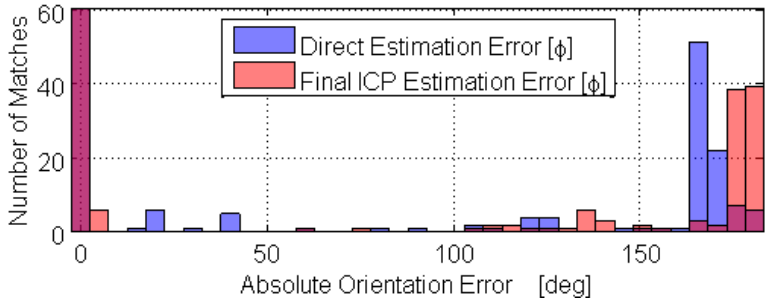

Figure 5.32: Absolute orientation error histogram,
scaled (vertically) to enhance visibility of unsuccessful matches.

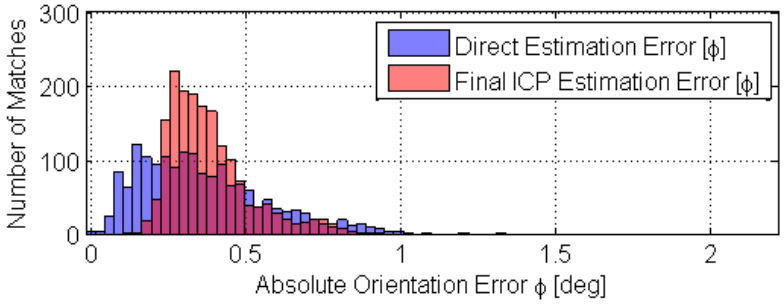

Figure 5.33: Absolute orientation error histogram for values within the solution region.

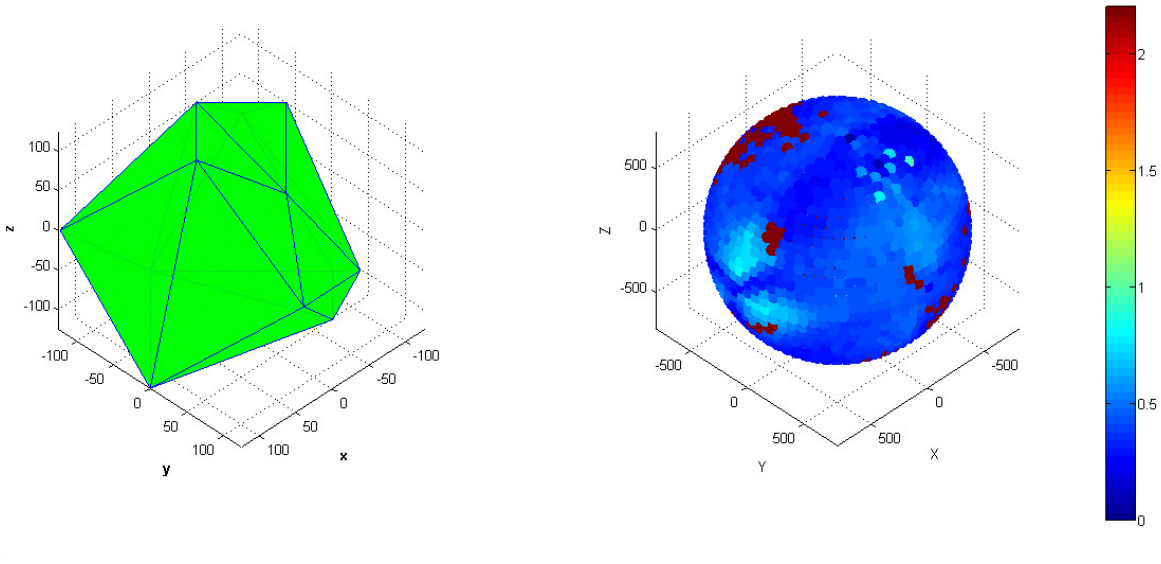

Figure 5.34: RAC (left), and the associated sphere of absolute rotation errors after ICP (right) (Colour bar is in degrees).

The sphere shown in the right hand side of Figure 5.34 is the "sphere of errors" after ICP has been performed. The sphere is a plot of all 2000 viewing directions, with each point coloured according to the absolute orientation error. The dark red portions are the viewpoints from which no pose estimate was possible, as well as those which resulted in high error.

There are certain portions of the shape from which a direct pose estimate is not possible with the FNPE1 algorithm (see Figure 5.35). In these regions, however, the shape remains unambiguous and ICP will converge on the correct solution if an appropriate initial guess can be supplied. It is only the direct pose estimate that is not possible. It should be noted that the catalogue lookup algorithm is itself capable of providing a pose estimate in the problematic regions described above. The issue is simply one of measurement, in that the required input cannot be extracted from a single view of the object. This problem can be overcome by integrating views from multiple perspectives, although that is beyond the scope of the current work.

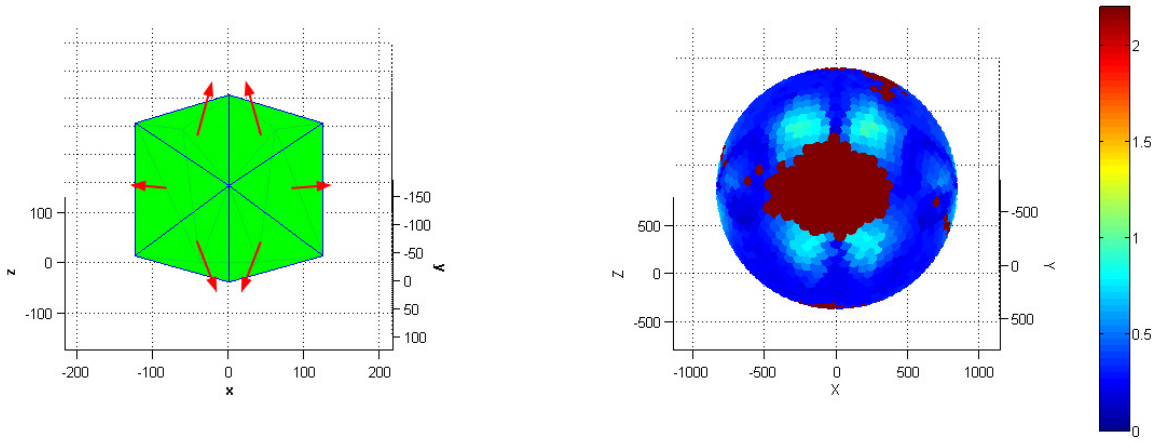

Figure 5.35: Cannot find orientation estimate for this type of configuration (Colour bar is in degrees).

FNPE1 provides reasonable accuracy even without subsequent ICP refinement. Problematic viewpoints are localised, and comprise only 13 percent of the viewing directions. The errors encountered in the other regions can be greatly reduced either by developing a more robust clustering algorithm, or by calculating the point normals in a more effective way (or both).

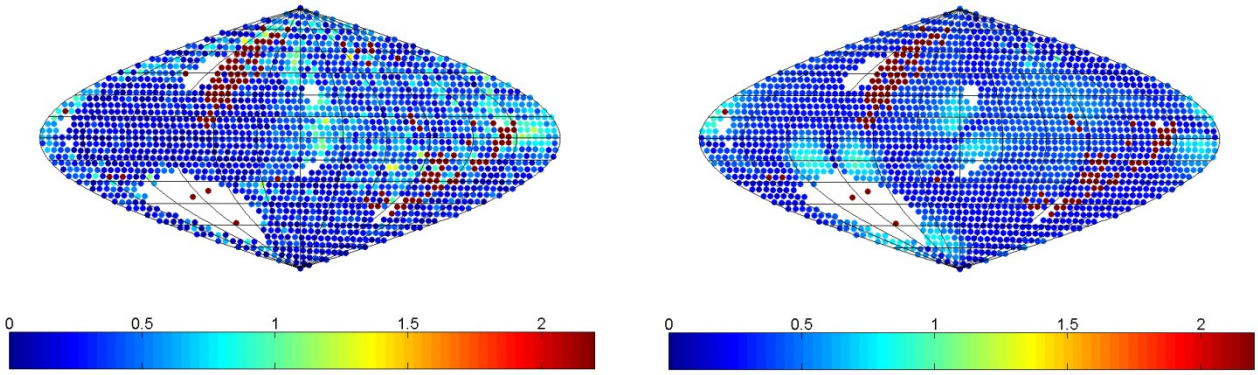

Figure 5.36: Sinusoidal projection of the sphere of errors for FNPE1 (left) and ICP (right). (Blank spaces are areas where direct estimates could not be made. Colour bar units are in degrees).

Although it works for a large majority of views, the test for centrality of a face can sometimes reject an ideal face arrangement, or accept an incorrect one, even when the normal vector estimates are good. This is a result of incorrect estimation of the face centroid positions. Centroid estimation error is affected by error in the point cloud, point loss due to clustering, and

asymmetric point distribution because of oblique viewing angles.

5.1.5 Conclusions

Although some fairly good performance was observed from the simulation, this method was not robust to the noise present in the scan data from the LCS. The reasons for the breakdown formed the basis for the alternate approach described in 3.2.

This method functions best when using scan data with very low noise. When high resolution data with higher noise levels (as with the LCS) are used, the point normals become scattered such that accurate surface detection through clustering becomes impossible. A range error that is significantly higher than the scan resolution means that the immediate neighbours of each point cannot be used to infer the surface normal at that point. Point normals will often be significantly different from the normals of the underlying surfaces. This effect can be somewhat reduced if rings of more distant neighbouring points are used instead, but this has the effect of increasing the size of the edge region. This, in turn, has the effect of reducing the density of the clusters of point normals, such that normals either cannot be found (due to the dispersion of the clusters), or have too high a deviation to be used in the catalogue lookup algorithm. Lastly, the algorithm is not computationally efficient for large numbers of points (i.e. for LCS scans) if the operation is desired in real-time.

In short, although this method can be made to work under a very specific set of (simulated) circumstances, it is not robust or efficient enough to be used with the LCS data.

5.2 Testing of FNPE2 with Simulated Data

Since the algorithm employs a random sampling approach, there is some variability in the pose estimate it generates. Although it is desirable to quantify this variability, the primary purpose of the simulation is to evaluate the integrity and basic functionality of the algorithm, while providing data on its accuracy. The variability will be examined more closely when the FNPE2 algorithm is applied to the LCS data. Also, since the RAC has some geometric regions where the required arrangement of faces cannot be seen, the simulation gives some idea of the locality and distribution of errors or impediments associated with these regions.

5.2.1 Setup / Methodology

Point clouds were generated from 2000 positions around the RAC, equally distributed in viewing angle, with no boresight roll (as with the first simulation). Gaussian range noise was added to each point cloud, and a rotation introduced to so that the range is always expressed along the same axis (in this case the z-axis) as would be the case for actual measurements. A constant camera range of 3600 [mm] was used (yielding an average point cloud range of approximately 3500 [mm]). This distance was chosen to match (approximately) the range of the point clouds from the LCS. The point density was set so as to generate point clouds with roughly the same number of points as with the LCS data. Although range noise was taken into account, the edge effects and waves seen in the LCS data were not modeled. The intensity of the Gaussian range error (standard deviation of 3.3125 for a range of 3500 mm) was based on LCS performance data received from Neptec [17]. The range error was added along vectors from the “camera” to the (error-free) points. No other error was added to the simulated point cloud. Each point cloud was processed by the FNPE2 algorithm, and the pose estimate compared to the true pose. Any views which failed to produce an estimate, or which produced an obviously incorrect orientation estimate ($\varphi > 20$ [deg]) were re-evaluated (up to 10 times), so as to distinguish viewing angles that generate consistently poor results, from those that generate only "fluke" failures. Such fluke failures could be compensated for in future more refined versions of the algorithm, and are not indicative of a fundamental flaw in the approach.

As with the first simulation, the accuracy of the orientation estimates is based on the magnitude

of the angular component of the error quaternion between the estimated orientation and the true orientation, φ, expressed in [deg]. Similarly, the accuracy of the position estimate is based on the magnitude of the error vector between the true position and the estimated position. The minimum rotation required to map one set of normal vectors onto another is approximately 20 [deg][9]. Although it would be difficult to draw any conclusions from the fact that an orientation estimate has an error less than this value, it is reasonable to expect that errors larger than this value are indicative of some part of the FNPE2 algorithm functioning in an unintended way.

5.2.2 Results

All the FNPE2 estimates account for 95.55% of the total number of scans (1911 estimates from 2000 scans). The remaining 4.45% failed to generate a pose estimate. The results of the FNPE2 estimates are shown in Figure 5.37. Colours represent position error in [mm]. Together with Figure 5.38, this gives a good impression of the general performance of the FNPE2 algorithm. Note that as RMS values rise above 6 [mm], there are some estimates with a position error significantly higher than the surrounding ones, despite having a comparable orientation error. This tends to be the result of a pose estimate based on an inaccurate geometric reconstruction. These anomalies are clustered around the region of the RAC where the faces are smallest, and have very shallow angles separating them from their neighbours. In these regions, the FNPE2 algorithm has difficulty distinguishing one face from another, due to the noise in the point cloud[10]. The result is reconstructed surfaces which tend to cut across more than one face. Pose estimates under these circumstances are more variable than with the larger, better-distinguished faces. There can be redundant faces cutting across each other in this region of smaller faces, and typically, some of them will be fairly close to correct in terms or orientation, if not in terms of position. Under these circumstances a fairly good orientation estimate can sometimes be generated, but since the position estimate requires that the data points be associated with

[9] Surprisingly, the exact number (21.73) is very similar the one for the original unskewed RAC (22.07), as used in the FNPE1 simulation

[10] While it may be possible to alter the fitting parameters of the FNPE2 algorithm to correct for this, it could result in degraded performance in other regions. The parameters that were used were intended mainly for the larger, better differentiated surfaces, since the LCS scans of the physical model are of those regions. In the live experiments, the region containing the small shallow faces was used as a mounting surface, and so was not scanned.

correctly reconstructed surfaces, the position estimate can be greatly degraded. Fortunately, as seen in Figure 5.38, 88.25 percent of the views (92.36 percent of the estimates made) are associated with RMS errors of less than 6 [mm], and grossly incorrect orientation estimates ($\varphi >$ 20 [deg]) do not begin to appear until RMS errors rise above approximately 6.5 [mm]. This observation is particularly useful when truth data are not available for the orientation and position errors, as with the experiments using Neptec's LCS. Under such circumstances, the RMS error can be used to assess the likelihood of a pose estimate that is accurate in terms of both position and orientation.

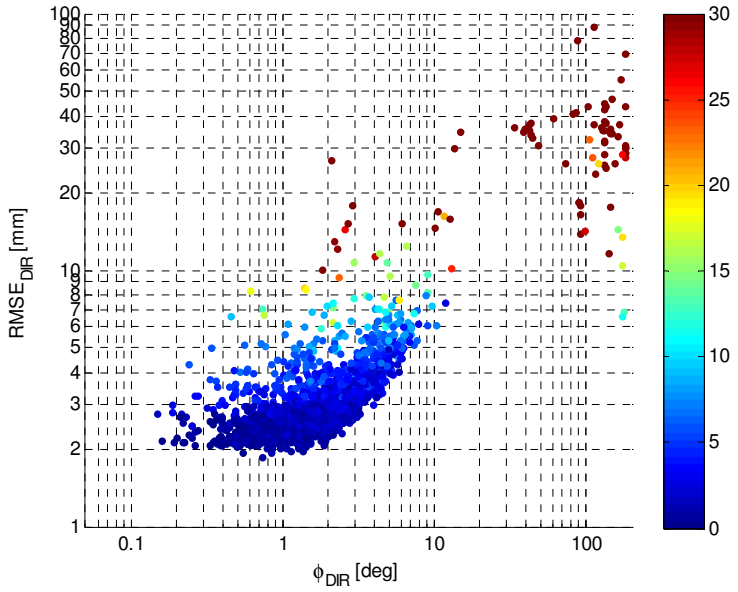

Figure 5.37: FNPE2 RMSE versus FNPE2 phi error (colour indicates displacement error, units are in [mm]).

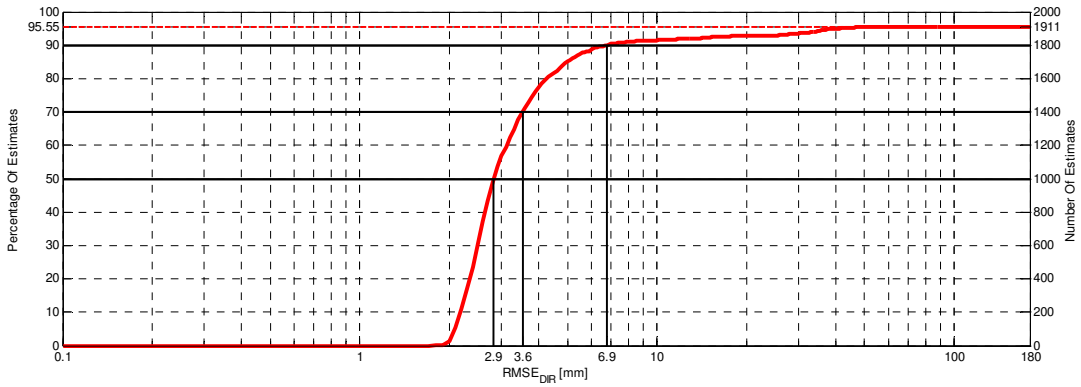

Figure 5.38: Cumulative FNPE2 RMS error.

When used as an initialiser for ICP, it was found (as expected) that it is the amount of orientation error in the initial estimate, rather than position error, that has the greatest influence on the probability of success with ICP. When the pre and post ICP orientation errors are plotted against each other, one can see clearly that there is a distinct threshold region beyond which ICP will not converge on the correct solution (see Figure 5.39). As previously stated, the minimum error expected when a set of faces is incorrectly identified is approximately 20 [deg], and so good ICP results were not expected from initial estimates with orientation errors exceeding this value. After manual verification, FNPE2 estimates with orientation errors greater than 20 [deg] were found to be the result of incorrect reconstruction of the requisite surfaces. All initial estimates with orientation errors of less than 20 [deg] led to ICP convergence on the correct solution. Surprisingly, good ICP results were obtained from initial estimates with orientation errors as high as 47 [deg], although only orientation errors lower than 43 [deg] resulted in consistently good ICP performance. ICP performance results are shown in Figure 5.40.

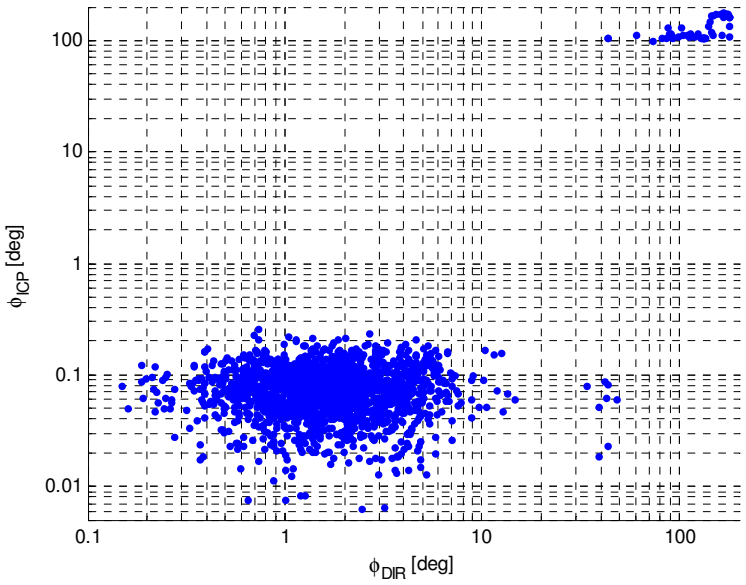

Figure 5.39: Orientation errors for ICP versus those for FNPE2.

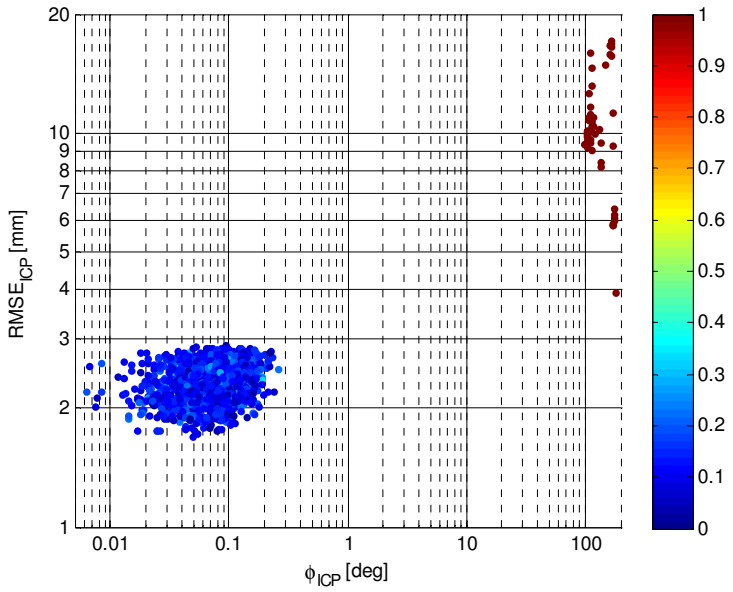

Figure 5.40: ICP RMSE versus ICP phi error (colour indicates displacement error, units are in [mm]).

It is clear from Figure 5.40 that correct post-ICP estimates are characterised by post-ICP RMSE values of less than 3[mm]. This is entirely reasonable, since the Gaussian range error added to the point cloud has a standard deviation of $\sigma = 3.3125$ [mm]. This information can therefore be used to determine whether or not ICP has converged on the correct solution, even if no truth data are available. From Figure 5.41, one can see the correlation between pre and post ICP RMSE. The colour of the points indicates the number of iterations ICP took to converge. This suggests that the pre-ICP RMSE of the pose estimate can be used not only to predict the likelihood of accurate ICP convergence, but also the maximum number of iterations it is likely to require. More detailed information is provided in Table 5.4.

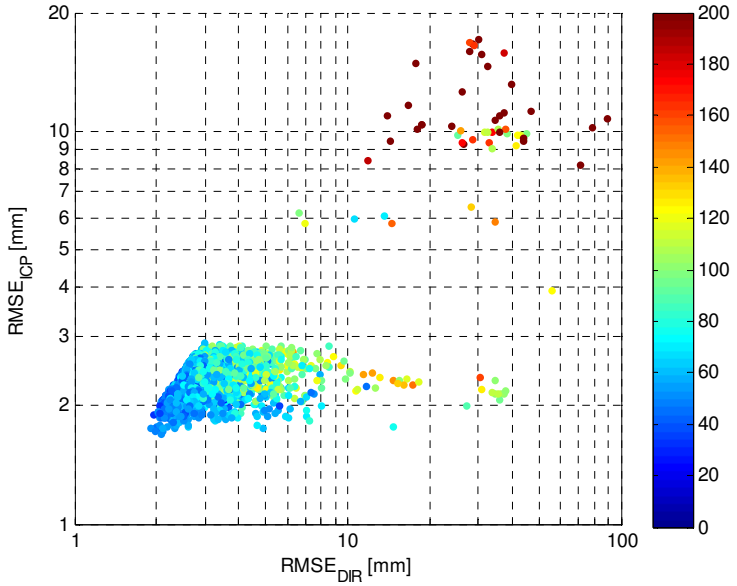

Figure 5.41: RMSE(FNPE2) versus RMSE(ICP). Colours show number of iterations required.

Table 5.4: Pre-ICP RMSE as a predictor of ICP performance

RMSE Range (Pre-ICP)	Percent of Total Scans	Percent Good Post-ICP (phi < 1)	Iterations μ	Iterations σ
0 < RMSE <= 2	0.25	100.00	46	8
2 < RMSE <= 3	53.55	100.00	59	17
3 < RMSE <= 4	22.30	100.00	71	18
4 < RMSE <= 6	12.15	100.00	82	22
6 < RMSE <= 20	4.60	85.87	108	36
20 < RMSE <= 100	2.70	20.37	151	43
0 < RMSE <= 100	95.55	97.07	70	27
No Estimate:	4.45	-	-	-

It is clear from Table 5.4 that the pre-ICP RMSE values less than 6 [mm] are the ones that can be relied upon to deliver good ICP performance, and to converge on the correct solution. If ICP refinement is not used, it is more important that the initial estimate be accurate in and of itself, as opposed to merely leading to good ICP performance. Fortunately, low RMS error values are strongly associated with low orientation errors, and a very large proportion of the total number of estimates falls into this range. This is illustrated in Figure 5.42, which shows the cumulative orientation and RMS error distributions associated with the error correlation plot originally introduced as Figure 5.37.

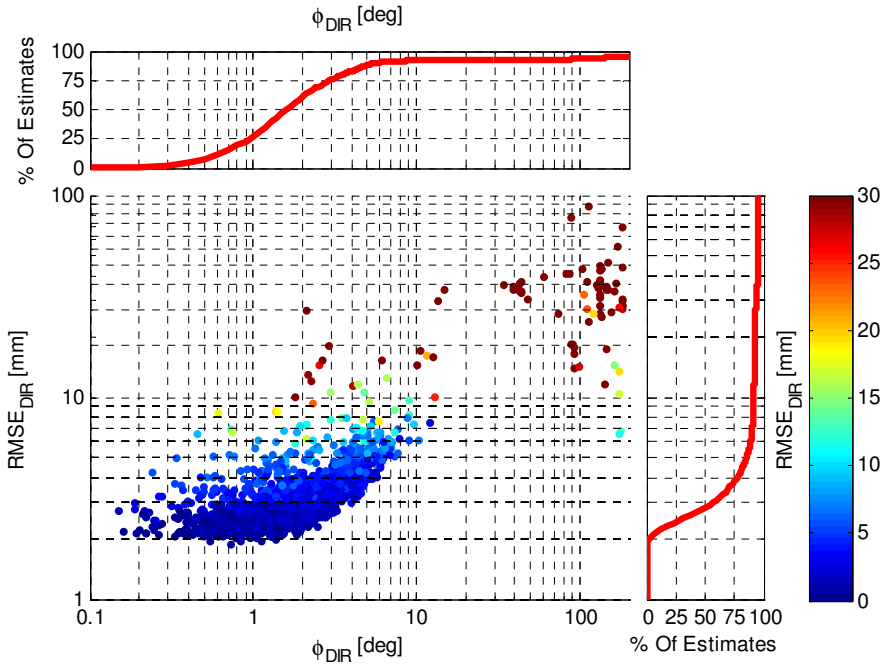

Figure 5.42: Cumulative error distributions in the pre-ICP correlated error plot (colour bar shows displacement error, units are in [mm]).

If the pose estimates are correct for post-ICP RMSE values less than three [mm], then one would expect that these values would be localised, and would vary with shape geometry. Furthermore, the pre-ICP RMSE values less than 6 [mm] (with which they are correlated) should be localised in the same pattern. The upper and lower portions of Figure 5.43 show sinusoidal projections of all 2000 camera positions, colour coded according to pre and post ICP RMSE.

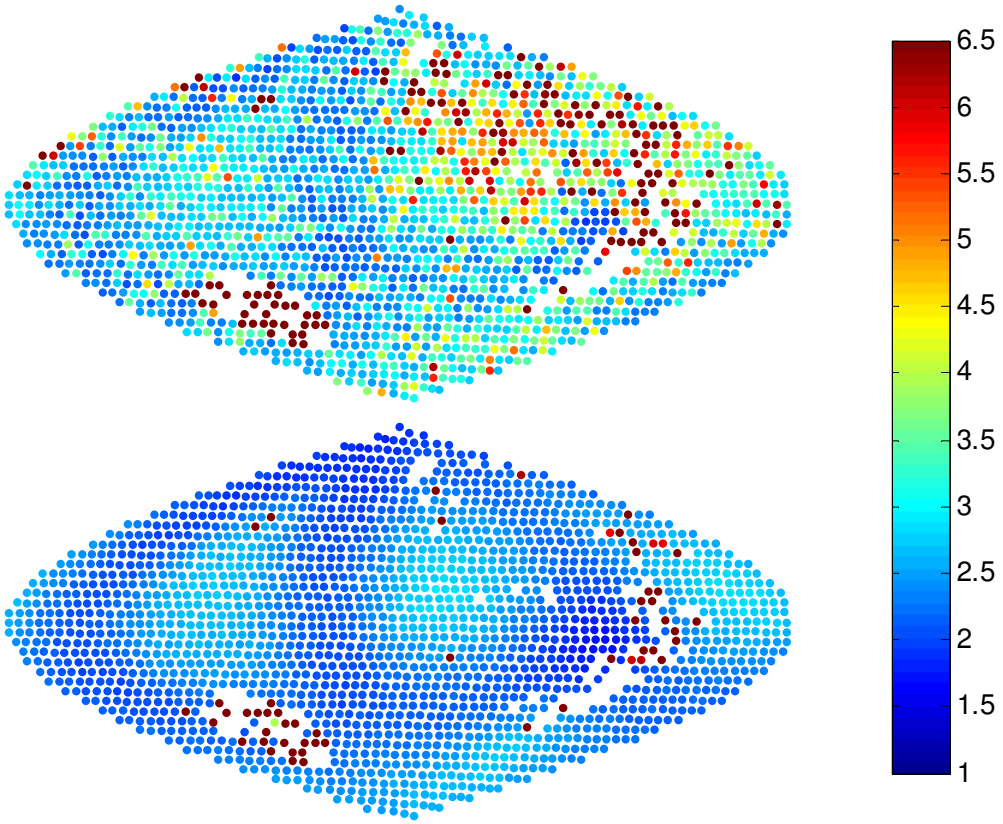

Figure 5.43: Sinusoidal projection of all 2000 view angles,
coloured according to the RMSE (expressed in [mm]) pre-ICP (top) and post-ICP (bottom).

It is clear that although the pre-ICP RMSE values have higher error and demonstrate greater variability, both datasets demonstrate similar regional distributions, except in one particular area. The regions where the FNPE2 algorithm failed to generate an estimate occur mostly where they were expected, i.e. where the required pattern of faces is not visible. The other regions where FNPE2 estimation failed (or resulted in large errors) are associated with small and/or undersampled faces. It is possible that planes matching such faces are not being found, due to the size of the spherical shell from which the point cloud is sampled during the plane-fitting procedure. If the shell is large enough to approach (or surpass) the size of the incircle of a given face, the probability of sampling points from that face will be correspondingly low. If the size of the shell is too small, the increased chance of sampling points from the face is offset somewhat by the larger numbers of iterations required to fit the plane. The size of the shell used here is better suited to the large faces, since they are instrumental in finding the pose estimate in a

higher percentage of the views. In the event that the algorithm manages to fit a plane to one of the small faces, it may be classified as "outliers" and rejected if it does not contain enough points. This rejection criterion is needed, however, in order to prevent redundant planes being found on the larger faces. In practice, the problems associated with the small or undersampled faces are not considered critical, since (as mentioned previously) these surfaces were not scanned during the LCS experiments.

Since the amount of range noise added to the simulated point clouds is of the same order as that expected from real scans, these results indicate that the FNPE2 algorithm should be useful when applied to the LCS scans, provided that unmodeled noise (scanning artefacts) does not feature prominently in the point clouds. The post-ICP RMSE is consistent when ICP has converged on the correct solution, and is in a range that is distinct from the incorrect post-ICP solutions. The post-ICP RMSE from the LCS experiments is therefore expected to fall into a similar range.

5.3 Testing of FNPE2 with LCS Data

In order for FNPE2 to be considered useful, it must be able to generate reasonably accurate pose estimates using real LIDAR data. To this end, the plane-fitting technique was applied to LCS scans of the RAC. Unfortunately, truth data were not available for these scans. For the tests described here, truth data are approximated by performing ICP on a version of the point cloud that has very few outliers or edge effects. This point cloud is extracted from the output of the plane-fitting algorithm. The algorithm was tested to determine the variability of the pose estimate, the registration error associated with the pose estimate, and the relative error compared to the ICP “truth” pose estimate. Note that “correct direct estimates” refer to pose estimates based on the correct identification of the faces of the RAC represented in the point cloud, in contrast to “incorrect direct estimates”.

5.3.1 Variability

When generating direct pose estimates from the LCS point clouds, the algorithm occasionally fails to produce an estimate, and on very rare occasions produces an incorrect one. The degree to which this occurs varies from point cloud to point cloud. These occurrences are due in part to the

way the algorithm has been implemented in Matlab[11], and in part to the error margins permitted by the catalogue lookup algorithm, which in turn are the result of the sensitive geometry of the RAC. Re-optimization of the RAC and higher-level refinement of the Matlab implementation could significantly reduce these occurrences while continuing to use the plane-fitting approach that has been previously described. As it is the approach itself that is of interest, failed and incorrect estimates (if any) are excluded from the dataset used in the evaluation of the plane-fitting algorithm (FNPE2).

A total of 50 correct estimates were used to establish the variability of the pose estimate for a given point cloud. The variability was examined in terms of the displacement vector, as well as the orientation quaternion. For the orientation, each estimate was compared to the mean of the 50 orientation estimates. The mean quaternion was determined using the method described in [20]. Given the matrix M:

$$M \equiv \sum_{i=1}^{n} \mathbf{q}_i \mathbf{q}_i^T \tag{22}$$

The average quaternion is the eigenvector of M corresponding to the maximum eigenvalue. Having found the mean orientation, an angular deviation from the mean is calculated for each of the 50 estimates. For each estimate, an error quaternion is determined (with ESOQ2 [16]) using the mean orientation as a reference. The variation of each estimate from the mean is then quantified by the magnitude of the rotation which the error quaternion expresses. The standard deviation of these magnitudes is used to describe the variability of the orientation estimates for the given point cloud.

The variability of the displacement estimate is also described by the magnitude of the deviation from the mean. A vector is defined from the position of the mean estimate, to each of the individual position estimates. The standard deviation of the magnitudes of these vectors is then used to describe the variability of the position estimate.

---

[11] In addition to the actual implementation, there are also some parameters whose values have been chosen based on empirical testing. Modification of these parameters can significantly affect the performance of the algorithm. While values were chosen to provide good results from the LCS data, globally optimal values were not determined.

5.3.2 Accuracy

Since truth data are not available, ICP was used to establish a "true" position and orientation for the point cloud. Since the raw point cloud contains outliers and edge effects, a reduced point cloud was used which was mostly free of these artefacts. Each FNPE2 pose estimate is associated with a reduced point cloud, made up of the points that were deemed to "belong" to each of the faces found during that particular evaluation of the raw point cloud. Each of those 50 reduced point clouds generated by the FNPE2 algorithm contain different combinations of points from the original point cloud. The "common" point cloud is used for ICP. It consists of the points that are common to all 50 FNPE2-produced point clouds. Since outliers and edge effects are less likely to be consistently included by the FNPE2 algorithm, this common point cloud is typically free of any such artefacts.

Each of the 50 FNPE2 pose estimates were then used as initial guesses in the ICP algorithm. The variability of the ICP pose estimate was determined using the previously described technique. The low variability of the ICP estimate was taken to be indicative of an accurate ICP pose estimate (standard deviation of the angular component of the orientation deviation $\sigma_{orientation} < 0.02\,[deg]$, standard deviation of the magnitude of the position deviation $\sigma_{position} < 0.003\,[mm]$). The accuracy of the FNPE2 estimate for a given point cloud is then established by comparing the mean positions and orientations before and after ICP. The angular deviation between the two mean orientations is simply the magnitude of the angular component of the error quaternion between them. The displacement error is the magnitude of the vector between the mean FNPE2 position estimate, and the mean ICP position estimate.

5.3.3 Residual Error

The residual error is used to assess the precision of the fitting of the model to the point cloud, as opposed to the precision (or accuracy) of the FNPE2 pose estimate relative to the pose estimate obtained from ICP. A given pose estimate provides a displacement and an orientation with which the point cloud and model may be aligned to one another. Once they have been so aligned, the residual point-to-model distances are calculated (the reduced point cloud used in ICP is used here). The distances are then used to calculate the residual root mean squared error (RMSE):

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2} \tag{23}$$

Where n is the number of points in the point cloud, and x is the distance from point i to the closest corresponding point on the model. The units of the RMSE are the same as the units of x. The RMSE for a given pose will show how far from zero the fit error is. This measure of error is particularly useful as it includes error due to the pose estimation algorithm, as well as LIDAR scanning error, and it can be calculated using only a single pose estimate. Using the RMSE, the improvement afforded by the ICP algorithm can be put into context, since the "truth pose" found from ICP has a certain amount of fitting error associated with it as well.

5.3.4 Results

In total, fifteen LCS point clouds were used to assess the performance of the FNPE2 algorithm. For each one, 50 (correct) estimates were generated. Each FNPE2 estimate was used as the initial conditions for ICP, which produced a refined estimate.

Residuals

The RMSE is used to assess the fit quality of both the direct and ICP pose estimates. When calculating these errors the reduced point clouds are used, which contain anywhere from 56 to 80 percent of the original points. Many of the complete point clouds contain outliers and edge effects that will increase the RMSE, even if the pose of the point cloud is correct. The process by which the reduced point clouds are calculated removes most, if not all, of the artefacts in the data. Although the number of points removed varies from point cloud to point cloud (from 20 to 44 percent), this does not seem to be significantly correlated with the magnitude of the RMSE. This is shown in Figure 5.44 (note that the lines in the figure connect the mean estimates for each point cloud, and the RMSE shown is in units of [mm]).

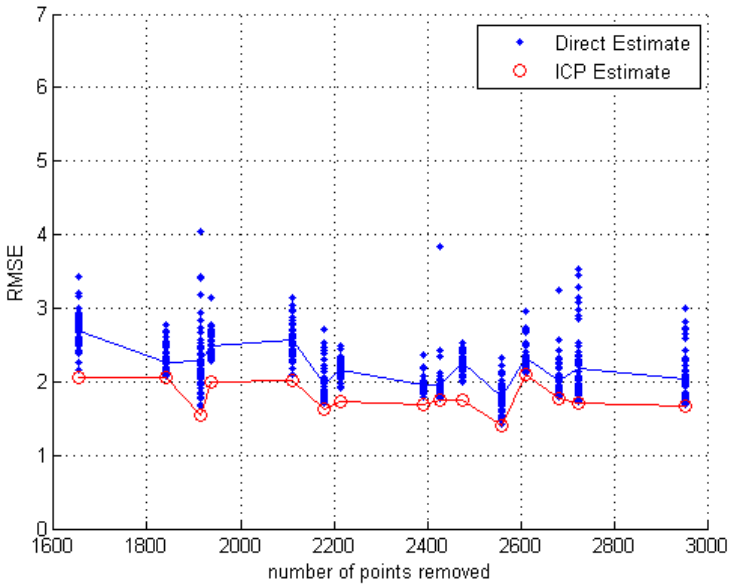

Figure 5.44: RMSE versus point cloud reduction.

It can be seen from Figure 5.44 that the use of the reduced point cloud does not significantly skew the relative accuracy of the RMSE. It is therefore reasonable to use the reduced point cloud to measure the fit quality of all the pose estimates. The reduced point cloud is also used when calculating the ICP refinement of the FNPE2 pose estimate, for the same reasons as for the calculation of the RMSE.

When comparing the RMSE before and after ICP, it is plain that the initial FNPE2 estimate is comparable in accuracy to the refinement from ICP. The mean reduction in residual error afforded by ICP was only 0.4030 [mm], from the mean FNPE2 value of 2.1947 [mm] to the mean ICP value of 1.7917 [mm]. The FNPE2 residuals are also highly consistent, with a standard deviation of only 0.1489 [mm]. A histogram of the residual error for all 750 estimates is shown in Figure 5.45.

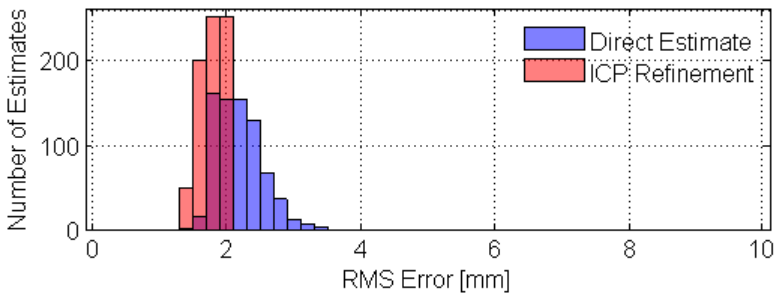

Figure 5.45: LCS RMS error histogram.

A plot of the FNPE2 results, in the same format as for the FNPE2 simulation, is shown in Figure 5.46. As mentioned earlier, the orientation and position errors are calculated relative to the mean ICP values. Although the pre-ICP RMSE cannot effectively be used to predict the exact magnitude of the orientation error, it is nevertheless clear that the vast majority (>95%) of the results have pre-ICP RMSE values smaller than three [mm], and are also associated with orientation errors of less than three [deg].

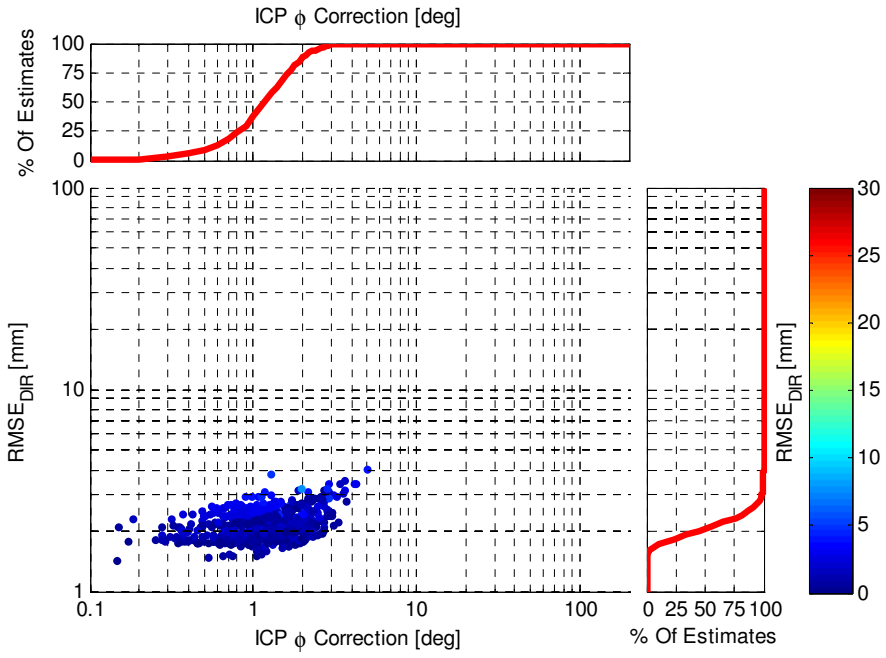

Figure 5.46: Relative orientation error (phi) versus pre-ICP RMS error (colours indicate relative position error in [mm]).

Pose Error

For correct estimates, the FNPE2 algorithm was shown to generate pose estimates with a mean standard deviation of 0.4238 [deg] and 0.6257 [mm]. The ICP results were seen to be highly consistent from estimate to estimate, having pose estimates with a mean standard deviation of 0.0176 [deg] and 0.0011 [mm]. Using the ICP estimates as truth data, the mean FNPE2 pose error was found to be 1.0301 [deg] and 1.0799 [mm].

Table 5.5 shows a summary of the results for each of the fifteen LCS point clouds. Note that the error columns represent the difference between the mean FNPE2 and mean ICP poses.

Table 5.5: Standard deviation and relative error of FNPE2 and ICP pose estimates

	FNPE2 standard deviation		Error		ICP standard deviation	
Point Cloud	**orientation [deg]**	**position [mm]**	**orientation [deg]**	**position [mm]**	**orientation [deg]**	**position [mm]**
1	0.37	0.54	0.68	1.03	0.0173	0.0020
2	0.88	0.91	0.24	0.51	0.0169	0.0015
3	0.47	0.46	1.15	0.94	0.0191	0.0004
4	0.35	0.58	0.95	0.61	0.0168	0.0016
5	0.50	1.08	0.59	1.13	0.0173	0.0015
6	0.32	0.34	0.95	2.69	0.0163	0.0001
7	0.38	0.48	1.71	0.47	0.0184	0.0009
8	0.45	0.40	1.58	0.94	0.0178	0.0008
9	0.59	0.84	2.24	0.44	0.0157	0.0012
10	0.38	1.22	0.98	1.55	0.0166	0.0020
11	0.37	0.88	0.40	0.26	0.0176	0.0008
12	0.31	0.82	0.47	1.08	0.0152	0.0005
13	0.26	0.49	0.73	2.26	0.0199	0.0002
14	0.40	0.21	1.63	1.15	0.0186	0.0020
15	0.33	0.14	1.15	1.16	0.0199	0.0015
Mean:	0.42	0.63	1.03	1.08	0.0176	0.0011

The histograms in Figure 5.47 and Figure 5.48 show the distribution of the FNPE2 orientation and position estimation errors, respectively. These histograms show the error for all 750 individual FNPE2 estimates (across all 15 point clouds). To calculate the error, each FNPE2 estimate was compared to the mean ICP estimate for the corresponding LCS point cloud.

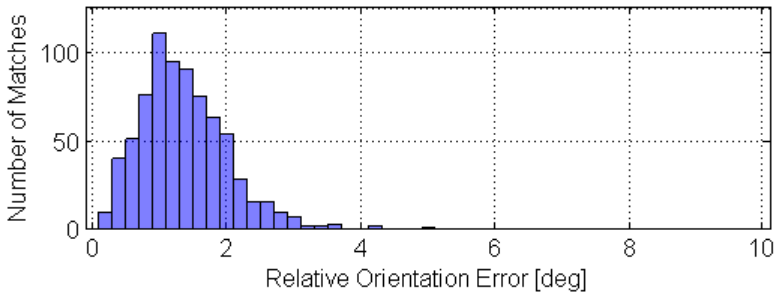

Figure 5.47: LCS Orientation Error Histogram.

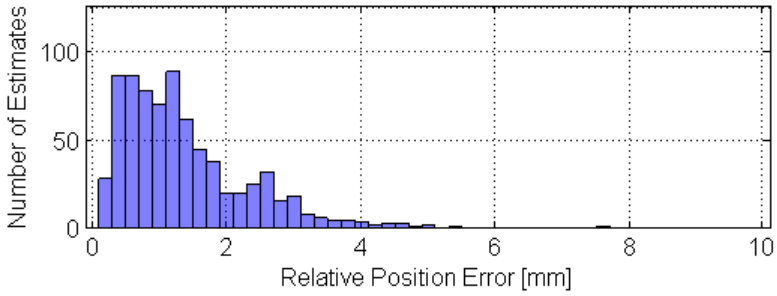

Figure 5.48: LCS Position Error Histogram.

The histograms show a clear tendency towards a low pose error, and are consistent with the results seen in the FNPE2 simulation (further comparisons are made in Chapter 6).

Although the overall results are fairly consistent, it is interesting to note the results from some of the individual LCS point clouds. The consistency of the pose error varies from point cloud to point cloud. Some, such as those shown in Figure 5.49 and Figure 5.50, resulted in consistently low pose and RMS error, while others tended to vary in only one (Figure 5.51), or both (Figure 5.52), of the pose error components (position and orientation).

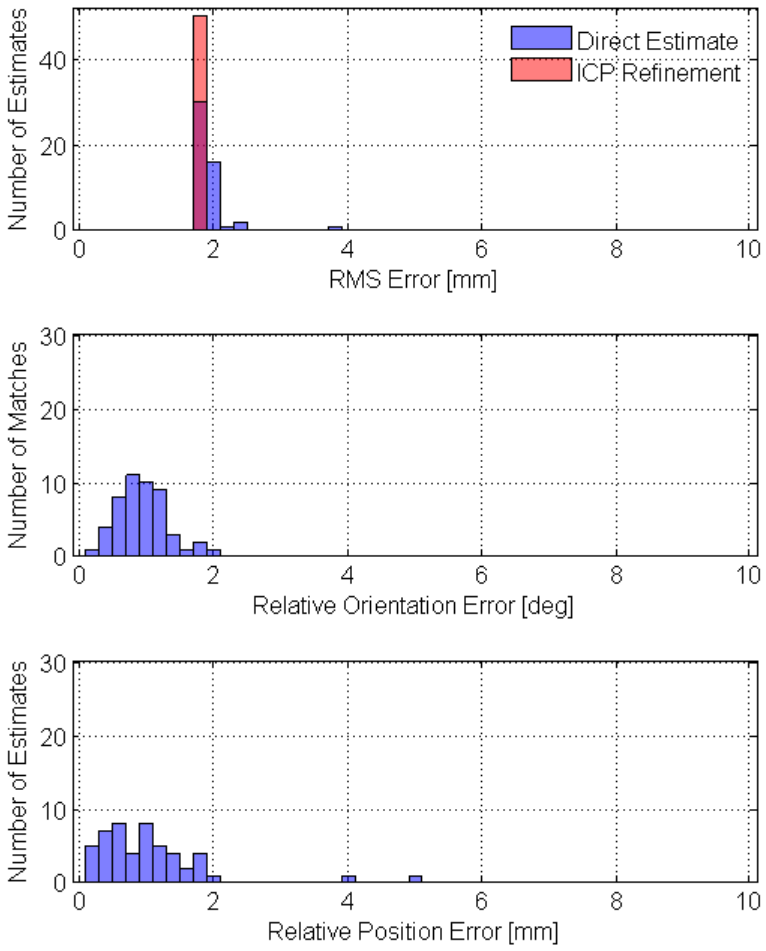

Figure 5.49: Results for LCS point cloud 13.

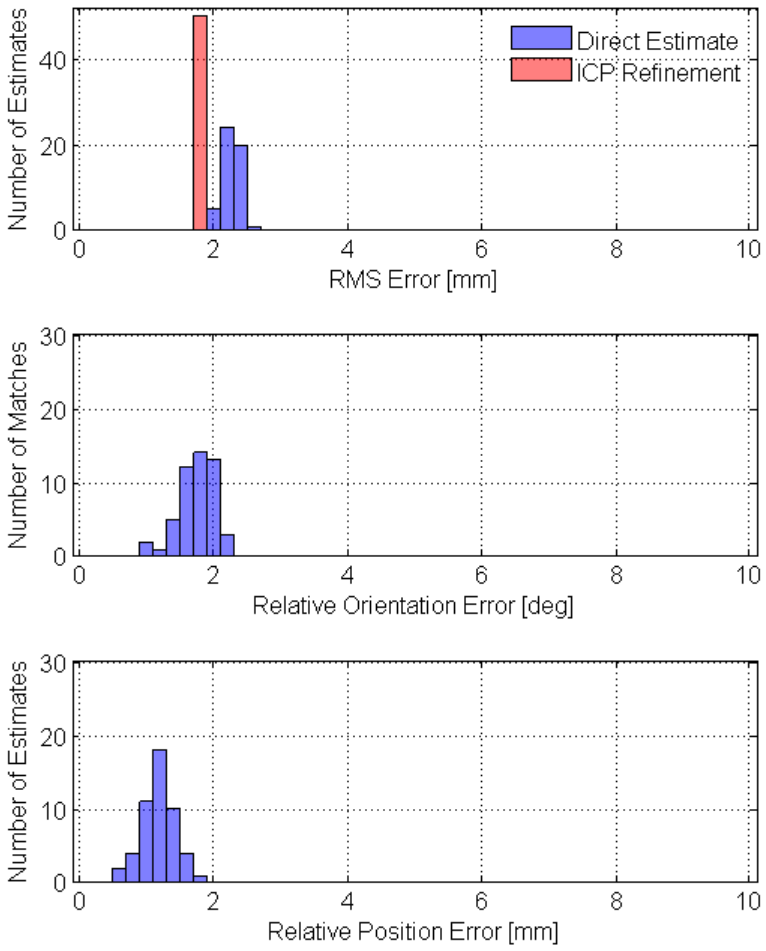

Figure 5.50: Results for LCS point cloud 16.

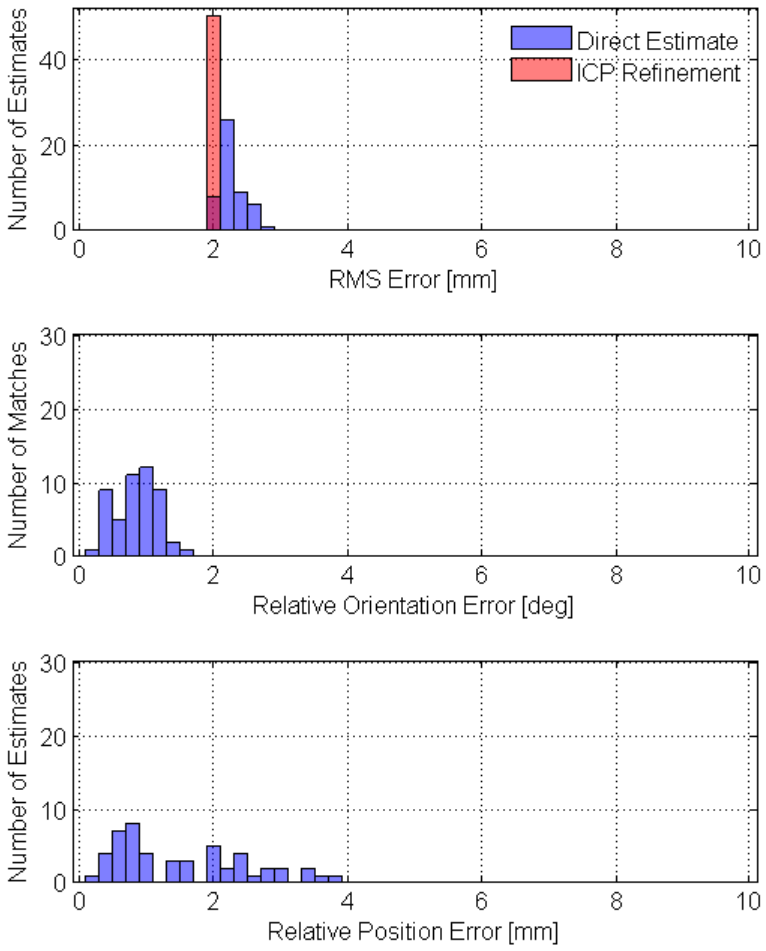

Figure 5.51: Results for LCS point cloud 14.

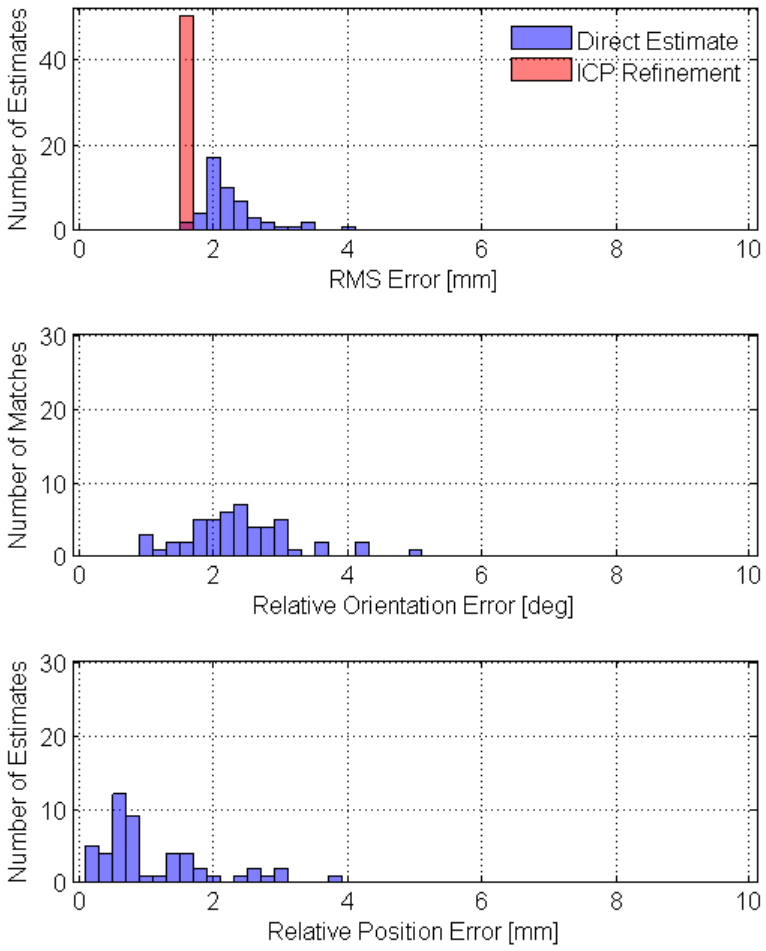

Figure 5.52: Results for LCS point cloud 11.

Chapter 6 General Discussion

When comparing the experimental results to those from the simulation, one must account for the fact that the physical model was only scanned from a limited number of perspectives. It has been established that the performance of the FNPE2 algorithm is in part dependent upon the region of the RAC depicted in a given scan. The simulated results from the "mounting bracket" portion of the RAC should not be included in the comparison, in part due to the regionality of the algorithm, and in part because the execution parameters of the algorithm were tuned to the regions scanned by the LCS, and not the "mounting bracket region". A "LCS subset" was therefore extracted from the complete simulation results. This was done by comparing the set numbers (i.e. the angular set numbers) of the pose estimates from the LCS experiment, with those from the simulation. Recall that the "set number" of a pose estimate is the numeric identifier of the angular set chosen by the FNPE2 algorithm as the best match to the input data. Due to the random sampling used in the plane-fitting process, multiple re-evaluations of the same point cloud will not always result in the same set of faces being chosen as the best ones from which to form the pose estimate. A given view of the RAC (or even a given point cloud) is not necessarily associated with a single set number. To form a correspondence between the simulation and the experiment, a list was compiled of all the set numbers found in the LCS experiment, from which duplicate values were culled. The "LCS subset" of the simulation results is then any pose estimate associated with a set number that matches an entry on this list. This effectively identifies any situation in which the simulation found faces that were also found in the LCS data. The results of the LCS experiment are shown in Figure 6.53 (note that the orientation and displacement errors are relative to the ICP pose). Figure 6.57 and Figure 6.58 show the LCS subset of the simulation and full simulation results, respectively.

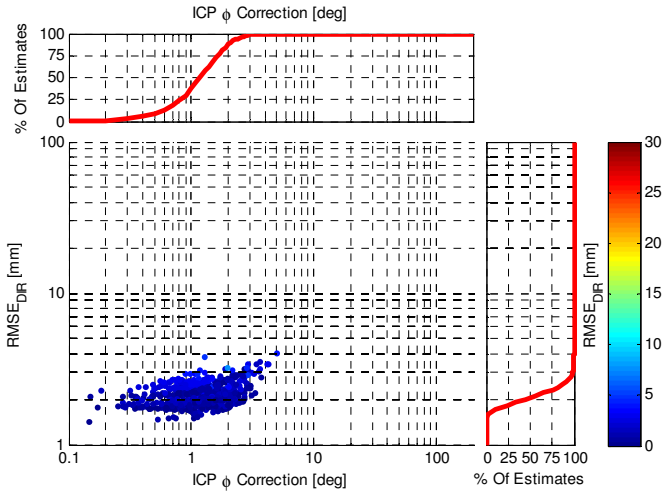

Figure 6.53: FNPE2 LCS experiment.

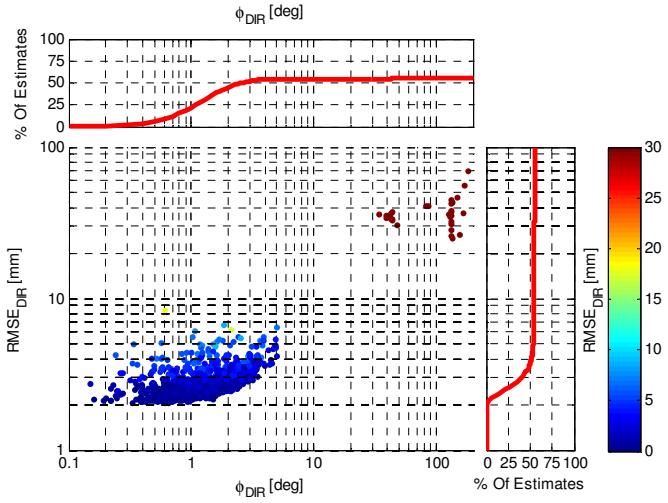

Figure 6.54: LCS subset of FNPE2 simulation.

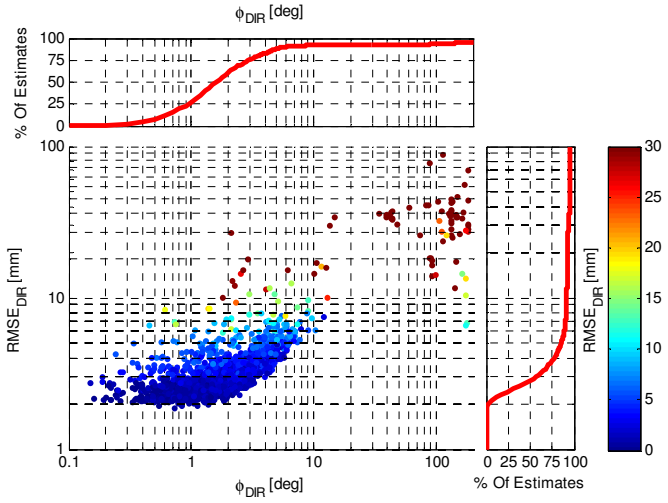

Figure 6.55: Full FNPE2 simulation.

The LCS experimental results demonstrated a better FNPE2 fit quality (RMSE) than predicted by the simulation. Each LCS point cloud was made to generate repeated pose estimates of a single point cloud, whereas in the simulation, many different perspectives were used, each with a slightly different point cloud, from a slightly different perspective. This means that any variability in the LCS results (for a single point cloud) are not due to estimates from varying geometry. The variability of the simulation results, however, derives from a combination of continuously varied viewing direction, the unique Gaussian error added to each point cloud, and the inherent variability of the FNPE2 algorithm. As such, it is reasonable to see more dispersed results in the simulation results. The lower values of RMSE in the LCS data, however, suggest either that the standard deviation of the range noise in the LIDAR unit was less than the value that was quoted (i.e. the quoted literature was out of date), or else the unmodeled error characteristics of the LCS (combined with the modeled range noise) serve to preserve within the point cloud more of the geometric characteristics of the RAC than the simplified error model was able to account for. A combination of these two scenarios is also possible. It should be noted that the tuning parameters used in the simulation were slightly different than those used in the LCS experiment. Although still tuned for the larger faces, the parameters in the simulation were adjusted to be more tolerant of the smaller faces. Although this difference could account for the slightly different ranges of RMSE resulting from the FNPE2 pose estimate, this would not account for the correspondingly lower RMSE seen in the post-ICP pose estimate. The correlation between the pre and post ICP RMSE values are shown in Figure 6.56, Figure 6.57 and Figure 6.58 for the LCS experiment, the LCS subset of the simulation, and the full simulation, respectively. Regardless of the reason for the smaller RMSE values seen in the LCS experiment, it is clear that the FNPE2 algorithm is capable of high-accuracy pose estimates using real scan data.

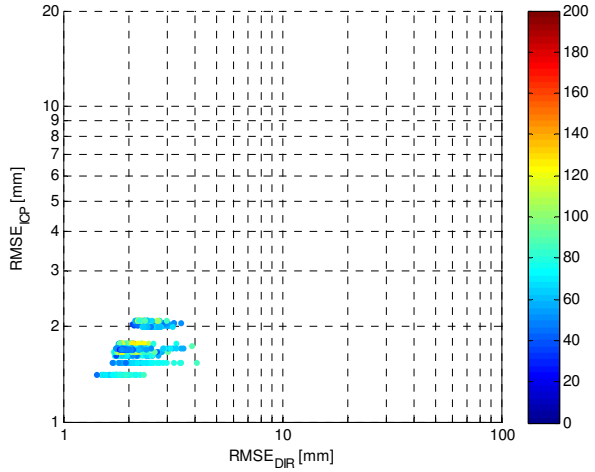

Figure 6.56: FNPE2 LCS experiment.

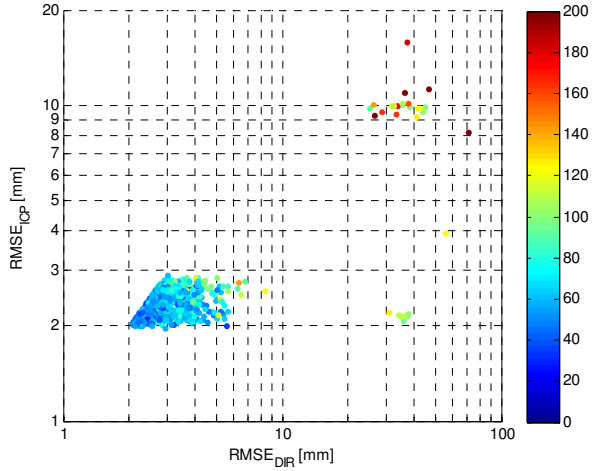

Figure 6.57: LCS subset of FNPE2 simulation.

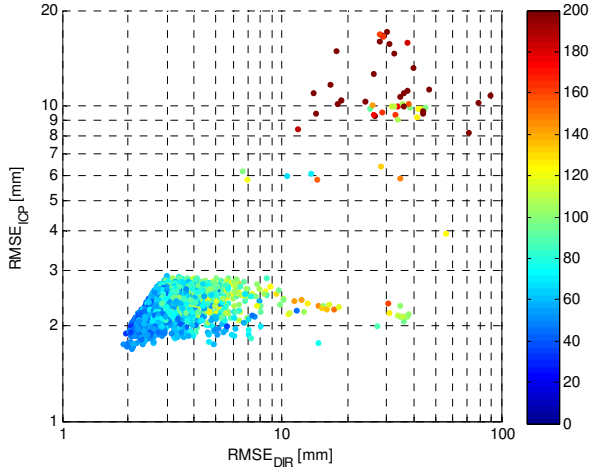

Figure 6.58: Full FNPE2 simulation.

Interestingly, despite the difference in RMSE values between the experimental LCS results and the simulation, the predicted orientation error distribution matches the experimental LCS results extremely well. Figure 6.59 Figure 6.60 and Figure 6.61 show the orientation error distribution histograms for the LCS experiment, the LCS subset of the simulation, and the full simulation, respectively.

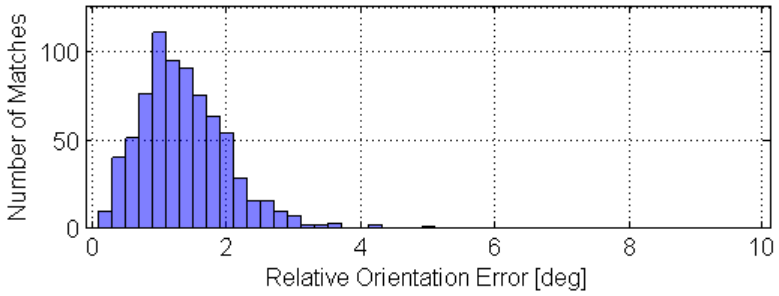

Figure 6.59: LCS orientation error histogram.

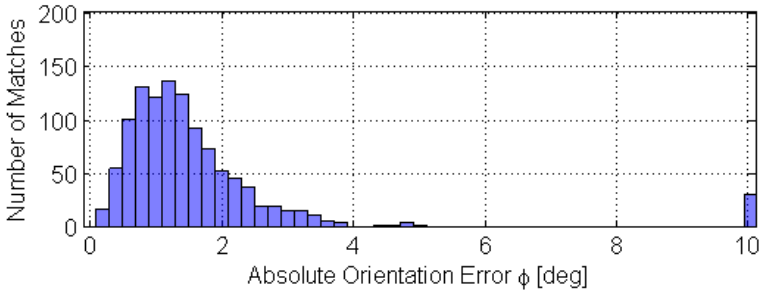

Figure 6.60: LCS subset of FNPE2 simulation.

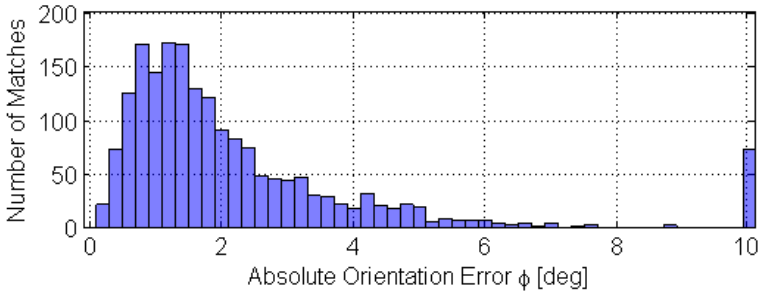

Figure 6.61: FNPE2 simulation orientation error histogram.

This high degree of conformity between the LCS experiment and the simulation is not as pronounced in terms of the position error (see Figure 6.62, Figure 6.63, and Figure 6.64). The location of the peak of the distribution matches, but the simulation predicts a wider range of position errors than were seen in the physical experiment. This could be due to the varying point density on certain faces, caused by the different view angles. The orientation estimate does not depend on the point density of a given face so much as it does on the orientation of the plane that was fitted to that face. The displacement estimate, however, identifies points as belonging to specific faces, and fits those points to their respective faces. There is, in effect, a weighting element to the position estimation based on the number of points found on each face of the set. Faces with more points are in effect given a heavier weighting when determining the displacement. For this reason, correctly oriented but poorly sampled faces will probably result in a position estimate that is of lower quality than the orientation estimate. For the better-sampled faces, however, the large numbers of points can help to average out the noise in the data, thus increasing accuracy. None of the LCS point clouds necessitated the use of poorly sampled faces, and so the position and orientation estimates were seen to be of comparable quality. The distribution of the LCS estimates does not stray too far from this scenario, since the data were spawned from repeated evaluation of only 15 scans.

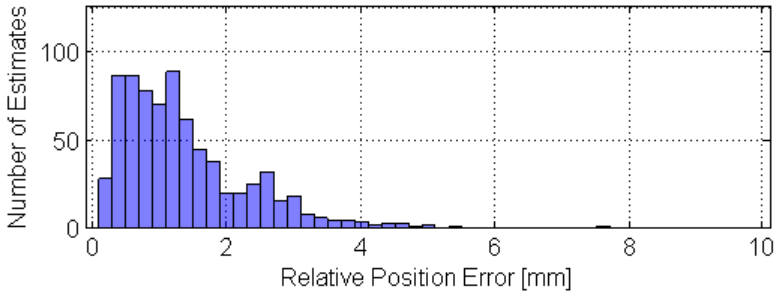

Figure 6.62: Position error histogram – FNPE2 LCS experiment.

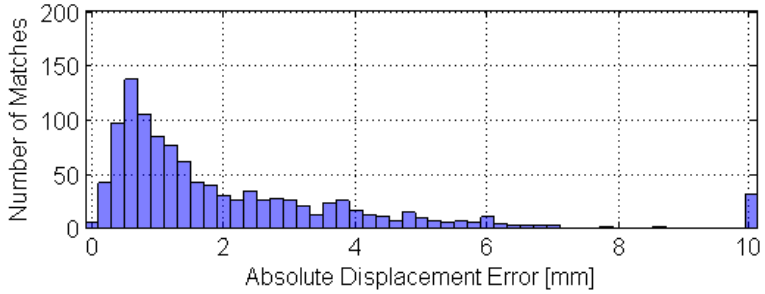

Figure 6.63: Position error histogram -- LCS subset of FNPE2 simulation.

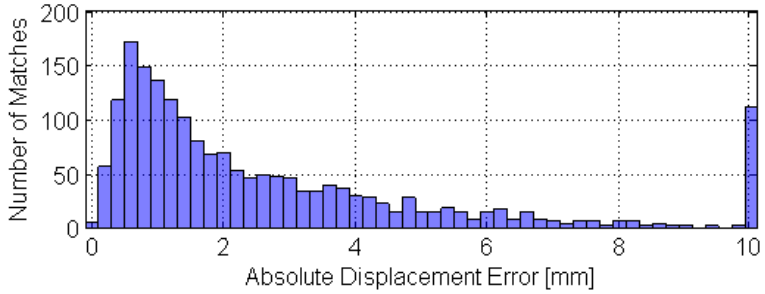

Figure 6.64: Position error histogram -- FNPE2 simulation.

Given the greater variety of views found in the simulation, it is reasonable to see a greater distribution of position estimation errors, even if the orientation errors do not demonstrate this dispersion. In fact, one would expect to see more of this kind of disparity in situations where the pose estimate was generated using faces of disparate surface areas (or surface areas projected along the camera view direction). This explains why the distribution of displacement errors shows some high-error clusters with no analogue in the orientation error distribution (see Figure 6.65). The only region that shows a clear correlation is the "mounting bracket" region (in the

upper right hand portion of the map), where the angular sets are based on faces of particularly dissimilar surface area.

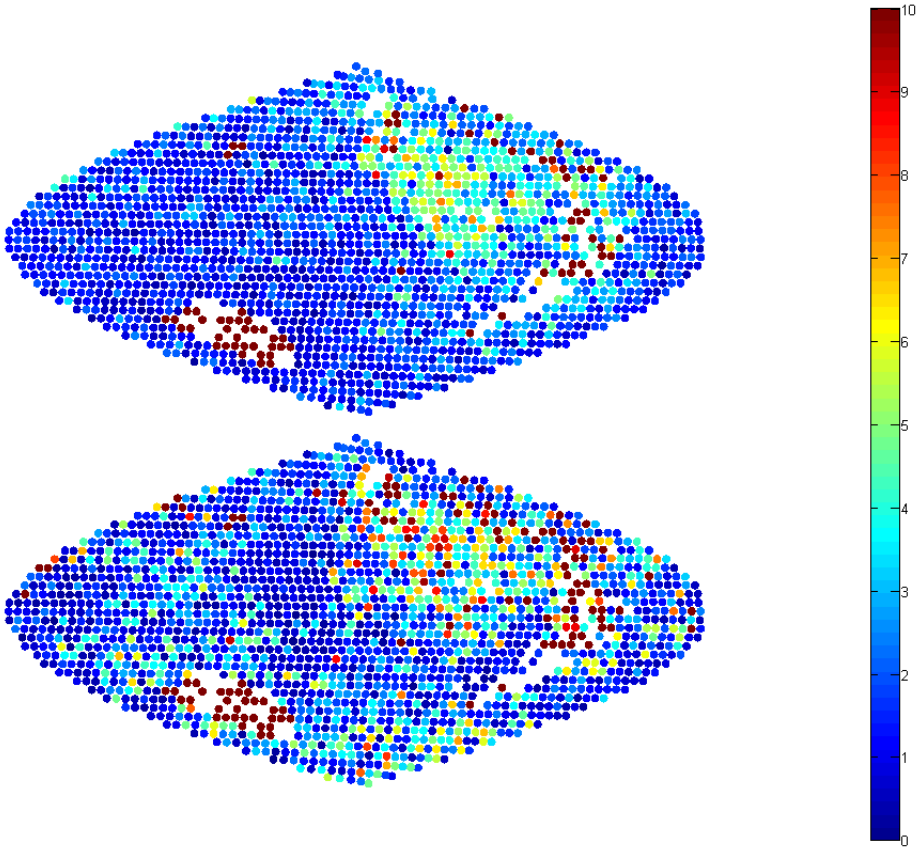

Figure 6.65: Sinusoidal projection of FNPE2 simulation view positions, coloured by orientation error in degrees (top) and displacement error in mm (bottom).

The reason for the clustering of orientation error in this location was discussed earlier. Note, however, that the problems with the displacement estimate manifest sooner than the problems with the orientation estimate. This is reasonable, since even a single undersampled face can cause the displacement estimate to be skewed in favour of the other three faces, thus degrading the position estimate. The orientation estimate is not affected because the FNPE2 algorithm tends to find the faces with the largest numbers of points first (The high point density on these faces will cause badly-fitted planes to contain significantly fewer points, and so even fairly shallow faces can be distinguished from one another). Having found those faces, it removes them from the point cloud before searching again. Once the three large faces have been found and removed,

the remaining face should be more or less isolated, and easy to reconstruct. In those regions where multiple small faces are visible (and are the only ones available for the formulation of a pose estimate), especially when connected with shallow angles, well-fitted and badly-fitted planes may contain a similar number of points, and the algorithm will have difficulty separating one face from another. In these situations the incorrectly reconstructed geometry will degrade the orientation estimate. The flawed geometric reconstruction will obviously cause significant degradation to the position estimate as well, hence the correspondence between the two error distributions in this region.

Chapter 7 Conclusions

The present work has shown that a unique target shape such as the RAC, combined with a pose estimation algorithm such as FNPE2, form a potentially viable basis for LIDAR-based pose estimation. Simulation results have shown that FNPE2 works well as an independent means of pose estimation. Both the simulation and the LCS experiments show that FNPE2 facilitates high accuracy ICP estimates, typically requiring fewer than about 120 iterations of the ICP algorithm. The simulation shows that this is possible for over 95% of the viewing directions. The FNPE2 algorithm has demonstrated the ability to generate pose estimates from raw LCS point clouds, having an average error (relative to the ICP refinement) of only 1.03 [deg] and 1.08 [mm], with standard deviations of 0.56 [deg] and 0.67 [mm] respectively. These results are better than those predicted by the simulation, and the simulation suggests that a large and contiguous portion of the RAC viewing directions can be used to generate pose estimates of similar accuracy. Although the combined use of FNPE2 and the RAC have demonstrated the distinct potential for use as LIDAR-based pose estimation tools, further research and development would be required before practical implementation in a dynamic setting could be realistically considered.

7.1 Future Work

There are several ways in which the present work could be expanded upon. The current assessment of performance would have been more complete had there been truth data concerning the relative pose between the model and the LCS, and so future assessments should include this data, if possible. Since the LCS data available were for a consistent set of target ranges, it would be worthwhile to examine the effect of varying range on the accuracy of the estimation process, as well as the effect of varying scan patterns (i.e. Lissajous, etc…). A method for determining the optimal operational parameters (diameter and thickness of point selection region, plane-fitting threshold distances, etc…) must also be developed. Manual adjustment of these parameters for a specific range, scan density, and point cloud error is not feasible when considering practical use in a dynamic setting, such as in an orbital rendezvous and docking scenario. Finally, to achieve real-time functionality the algorithm must be reconfigured to maximise speed and efficiency (i.e. general streamlining, use of parallel processing, etc…). For such real-time operation to be of

practical use, the consistency of the estimates must also be improved.

It should be noted that practical considerations associated with surface normal calculation were not a part of the original set of optimisation parameters used to determine the geometry of the RAC. With precise data concerning the effect of the RAC's various geometric properties on a refined, more robust version of the FNPE2 algorithm, the target could be redesigned to improve performance (For a given set of target ranges and viewing geometries, it is even conceivable that simultaneous optimisation could be performed on the FNPE2 operational parameters and the RAC geometry). The next step after such optimisation would be the repetition of some form of static assessment of the pose estimation algorithm using the new geometric design, followed by a set of dynamic experiments, both simulated and physical. For the simulated experiments, the development of a realistic LIDAR point cloud error simulator (including the addition of edge effects and other such characteristics) could improve the accuracy of simulation-based predictions, concerning performance in physical testing situations. The increased predictive capacity of such simulations could also prove to be useful in the design and optimisation of future 3D target shapes. Dynamic experiments could include the study of the effects of motion blur on individual pose estimates, methods to mitigate the impact of these effects, the identification and isolation of the target from a wide scan, and the integration of a Kalman filter to facilitate dynamic pose prediction.

Appendices

Appendix A: Background on the Iterative Closest Point (ICP) Algorithm

The ICP algorithm is an iterative method of minimizing the mean-squared distance error between two sets of data. Within the context of this thesis, it is a method for the registration of 3D shapes, based on a computer model of the shape, and a given set of 3D points. The points are a representation of the shape, based on range data from a sensor (either real or simulated). The computer model consists of a closed polygon with triangular faces, defined by a list of vertices. Faces are defined as indices into the list of vertices.

The algorithm is based on a correspondence between two sets of data. The first is the given sensor data, and the second is a set of closest points on the model which is re-evaluated with every iteration. Initial translation and rotation estimates are not technically required for this implementation of ICP [2][19] to converge, but for the reasons outlined earlier, they are provided to the algorithm[12]. With each iteration, incremental rigid rotations and translations are determined (non-iteratively) which reduce the mean squared error between the correspondence pairs. The increments are used to update the initial estimates, and the process is repeated. The iterations stop when either the current mean squared error, or the difference between the mean squared errors of the previous two iterations, drops below a given threshold. The final translation and rotation represent the absolute pose of the input point cloud.

Brief Mathematical Summary

The following is a very brief description of the determination of the incremental translations and rotations used, and is taken from [2]. The optimal quaternion rotation is described in further detail in [19].

For the following description, $P = \{\overrightarrow{p_i}\}$ is a measured point set of N_p points, describing a "data"

[12] This implementation of ICP can function with an initial guess of d = [0,0,0] for the translation, and q = [1,0,0,0] for the rotation.

shape that is translated and rotated so as to align with a "model" shape defined by the point set $X = \{\overrightarrow{x_i}\}$ (consisting of N_x points). Each point $\overrightarrow{p_i}$ corresponds to the point in $\overrightarrow{x_i}$ with the same index, and $N_p = N_x$. The registration state vector is denoted $\vec{q} = [\overrightarrow{q_R}|\overrightarrow{q_T}]$ where $\overrightarrow{q_R}$ is a unit quaternion rotation, and $\overrightarrow{q_T}$ is a translation vector. The mean square objective function is shown in equation (24).

$$f(\vec{q}) = \frac{1}{N_p}\sum_{i=1}^{N_p} ||\overrightarrow{x_i} - \boldsymbol{R}(\overrightarrow{q_R})\overrightarrow{p_i} - \overrightarrow{q_T}||^2 \tag{24}$$

The centroids of the point sets P and X are denoted $\overrightarrow{\mu_p}$ and $\overrightarrow{\mu_x}$ respectively. The cross-covariance matrix $\sum px$ for the sets P and X is given by (25).

$$\begin{aligned}\sum px &= \frac{1}{N_p}\sum_{i=1}^{N_p}\left[(\overrightarrow{p_i} - \overrightarrow{\mu_p})(\overrightarrow{x_i} - \overrightarrow{\mu_x})^T\right] \\ &= \frac{1}{N_p}\sum_{i=1}^{N_p}[\overrightarrow{p_i}\overrightarrow{x_i}] - \overrightarrow{\mu_p}\overrightarrow{\mu_x}^t\end{aligned} \tag{25}$$

The cyclic components of the anti-symmetric matrix $A_{ij} = \left(\sum px - \sum px^T\right)_{ij}$ are used to form the column vector $\Delta = [A_{23} \quad A_{31} \quad A_{12}]^T$, which is used to form the symmetric 4×4 matrix $Q(\sum px)$, as shown in equation (26).

$$Q\left(\sum px\right) = \begin{bmatrix} tr\left(\sum px\right) & \Delta^T \\ \Delta & \sum px + \sum px^T - tr\left(\sum px\right)\boldsymbol{I}_3 \end{bmatrix} \tag{26}$$

Where $\boldsymbol{I_3}$ is the 3×3 identity matrix. The unit eigenvector corresponding to the maximum eigenvalue of the matrix $Q(\sum px)$ is selected as the optimal rotation. The optimal translation is given by

$$\overrightarrow{q_T} = \overrightarrow{\mu_x} - \boldsymbol{R}(\overrightarrow{q_R})\overrightarrow{\mu_p} \tag{27}$$

The registration vector $\vec{q}$ is then used to transform the point set P, whereupon the mean squared registration error is recalculated, and the process repeated until the stop conditions of the algorithm are reached.

Appendix B: Old Catalogue Lookup Error Metric

The error metric generated by the catalogue lookup algorithm is used to determine the entry in the reference catalogue that best matches the input data. It is calculated in two parts. The first measures the difference between the measured and reference angles:

$$e_1 = \sqrt{\left(\theta_a - \theta_a^{'}\right)^2 + \left(\theta_b - \theta_b^{'}\right)^2 + \left(\theta_c - \theta_c^{'}\right)^2} \quad \text{where} \quad \begin{matrix} \theta = \text{reference value} \\ \theta^{'} = \text{measured value} \end{matrix} \tag{28}$$

The smallest value of e_1 is used to identify the matching reference tuple. Using ESOQ2 (the Second Estimator of the Optimal Quaternion) [16], a best-fit rotation is then found that maps the reference vectors onto their measured counterparts. After the rotation is applied, the reference unit normals $[\boldsymbol{n}_a \quad \boldsymbol{n}_b \quad \boldsymbol{n}_c]$ are compared to the measured unit normals $[\boldsymbol{n}_a^{'} \quad \boldsymbol{n}_b^{'} \quad \boldsymbol{n}_c^{'}]$ as follows:

$$e_2 = \sqrt{\left(1 - \mathbf{n}_a \cdot \mathbf{n}_a^{'}\right)^2 + \left(1 - \mathbf{n}_b \cdot \mathbf{n}_b^{'}\right)^2 + \left(1 - \mathbf{n}_c \cdot \mathbf{n}_c^{'}\right)^2} \quad \text{where} \quad \begin{matrix} \mathbf{n} = \text{reference value} \\ \mathbf{n}' = \text{measured value} \end{matrix} \tag{29}$$

The first error is based purely on angle mismatches. The second is based on the effectiveness of the orientation estimate. A combination of these two errors is used to formulate the direct pose estimate for a given view. It should be noted that this implementation may appear somewhat arbitrary, and the subsequent combination of the second metric with the first is not entirely justified from a mathematical perspective. This implementation was arrived at somewhat empirically, and was used only because it generated satisfactory preliminary results. The implementation and use of the error metrics was later revised (see section 2.3.1).

Appendix C: Points in Triangles

Are the points in the triangle? (note: points being tested are in the plane of the triangle)

1. The triangle has three vertices t_1, t_2, t_3
2. Generate unit vectors between vertex t_1 and t_2 (t_{12}), between vertex t_1 and t_3 (t_{13}), and between vertex t_1 and t_x (t_{1x}) where t_x is the point being tested
3. Compare the following cross products: t_{12}, $\times$ t_{13} and t_{12}, $\times$ t_{1x}. If they point in the same direction, then the point t_x is on the same side of the vector t_{12} as the point t_3
4. Repeat this for $t_{23} \times t_{21}$ and $t_{23} \times t_{2x}$
5. And again for $t_{31} \times t_{32}$ and $t_{31} \times t_{3x}$
6. If each of the three pairs of cross products point in the same direction, then the point t_x is inside the triangle defined by the vertices t_1 t_2 and t_3

   Note: Using matrix notation, this procedure is performed non-iteratively for all the points being tested.
7. The area of the triangle is determined using Heron's formula:

$$A = \sqrt{s(s-a)(s-b)(s-c)} \tag{30}$$

   where s is the triangle's semiperimiter, and is calculated from:

$$s = \frac{a+b+c}{2} \tag{31}$$

Appendix D: Extra Figures from the FNPE2 Simulation Results

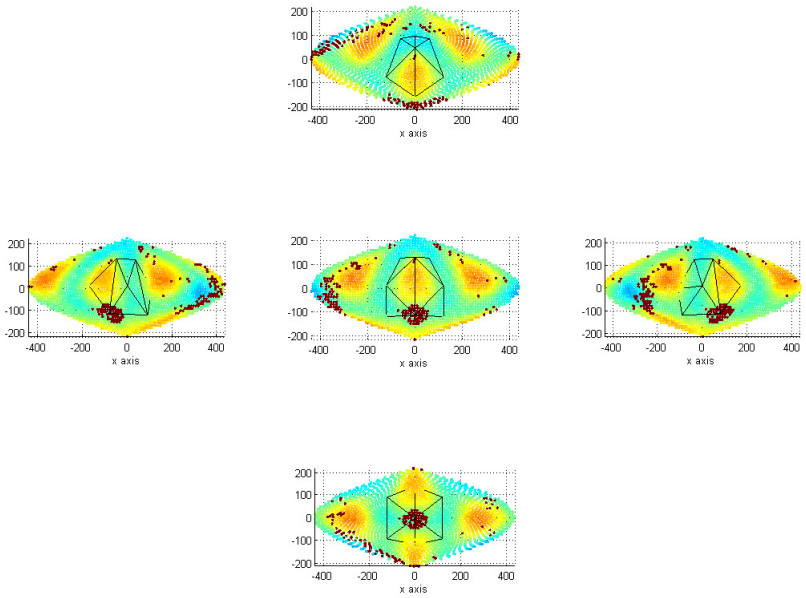

Figure 7.66: Regional Distribution of Post-ICP RMSE (FNPE2).

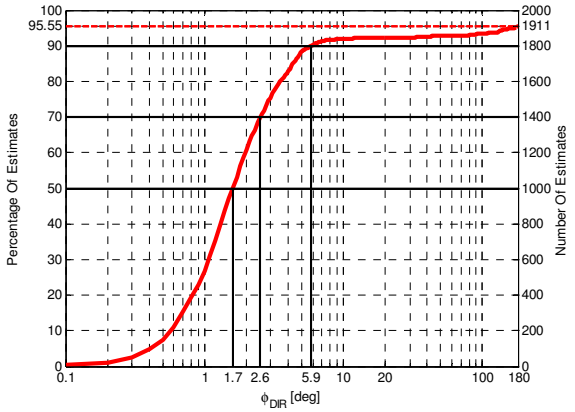

Figure 7.67: Cumulative Orientation Error (FNPE2 Simulation).

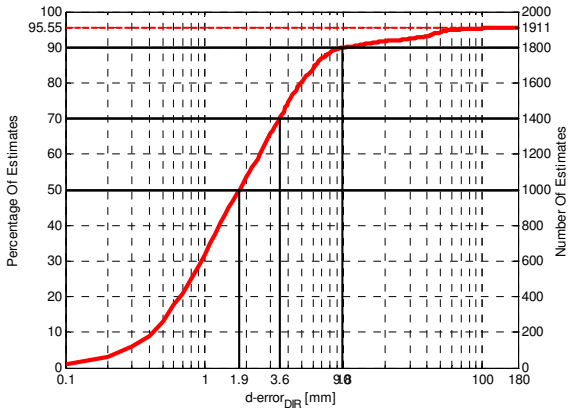

Figure 7.68: Cumulative Position Error (FNPE2 Simulation).

Appendix E: Extra LCS-to-FNPE2 Comparison Data

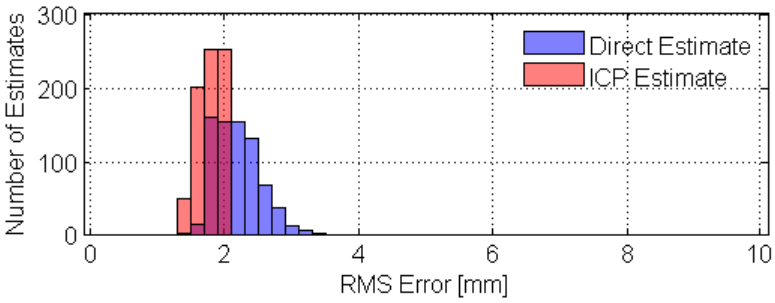

Figure 7.69: LCS RMS Error Histogram.

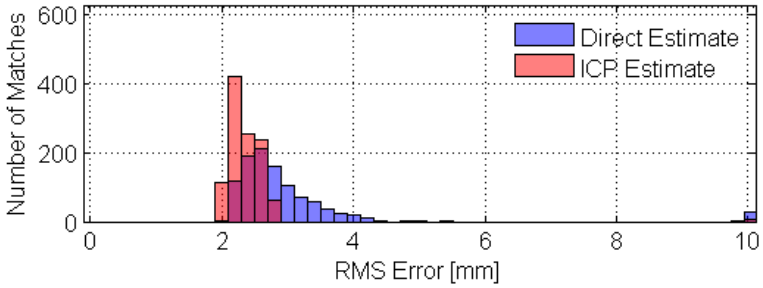

Figure 7.70: LCS Subset of FNPE2 Simulation.

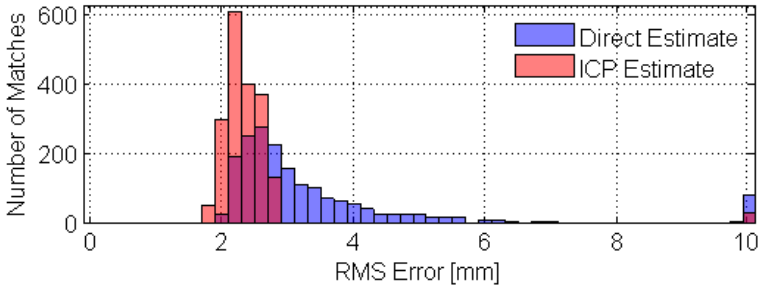

Figure 7.71: FNPE2 Simulation RMS Error Histogram.

References

[1] M. Bondy, R. Krishnasamy, D. Crymble, and P. Jasiobedzki, "Space Vision Marker System (SVMS)," presented at the AIAA SPACE 2007 Conference & Exposition, Long Beach, California, 2007.

[2] P. Besl and H. McKay, "A method for registration of 3-D shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239-256, Feb. 1992.

[3] S. Ruel, T. Luu, M. Anctil, and S. Gagnon, "Target Localization from 3D data for On-Orbit Autonomous Rendezvous & Docking," in *2008 IEEE Aerospace Conference*, pp. 1-11, 2008.

[4] S. Granger and X. Pennec, "Multi-Scale EM-ICP: A Fast and Robust Approach for Surface Registration," *Lecture Notes in Computer Science*, pp. 418-432, 2002.

[5] D. McTavish, G. Okouneva, and A. Choudhuri, "CSCA-Based Expectivity Indices for LIDAR Computer Vision," in *Proc. of WSEAS Conf. on Applied Computing (ACC '09)*, pp. 54-62, 2009.

[6] A. Choudhuri, "Target Design for LIDAR-Based ICP Pose Estimation for Space Vision Tasks," Ryerson University, 2009.

[7] D. McTavish and G. Okouneva, "A New Approach to Geometrical Feature Assessment for ICP-Based Pose Measurement: Continuum Shape Constraint Analysis," presented at the International Machine Vision and Image Processing Conference, 2007.

[8] G. Okouneva, D. McTavish, and O. Okunev, "Selection of Regions on a 3D surface for Efficient LIDAR-based Pose estimation," presented at the Vision, Modeling, and Visualization Workshop (VMV'09), Germany, 2009.

[9] R. Schnabel, R. Wahl, R. Wessel, and R. Klein, *Shape Recognition in 3D Point Clouds*. Bonn, Germany: Institut für Informatik II, Universität Bonn, 2007.

[10] M. Y. Yang and W. Förstner, *Plane Detection in Point Cloud Data*. Institute of Geodesy and Geoinformation, Department of Photogrammetry, University of Bonn, 2010.

[11] T. K. Dey, G. Li, and J. Sun, "Normal Estimation for Point Clouds: A Comparison Study for a Voronoi Based Method," *Eurographics Symposium on Point-Based Graphics (2005)*, 2005.

[12] J. Rocapardinas, H. Lorenzo, P. Arias, and J. Armesto, "From laser point clouds to surfaces: Statistical nonparametric methods for three-dimensional reconstruction," *Computer-Aided Design*, vol. 40, no. 5, pp. 646-652, May. 2008.

[13] H. Song and H. Feng, "A progressive point cloud simplification algorithm with preserved sharp edge data," *International Journal of Advanced Manufacturing Technology*, 2007.

[14] Y. Kenmochi, L. Buzer, A. Sugimoto, and I. Shimizu, "Discrete plane segmentation and estimation from a point cloud using local geometric patterns," *International Journal of Automation and Computing*, vol. 5, no. 3, pp. 246-256, 2008.

[15] J. Jiang, J. Cheng, and X. Chen, "Registration for 3-D point cloud using angular-invariant feature," *Neurocomputing*, vol. 72, no. 16, pp. 3839-3844, 2009.

[16] D. Mortari, "Second estimator of the optimal quaternion," *Journal of Guidance, Control, and Dynamics*, vol. 23, no. 5, p. 885, 2000.

[17] Neptec Design Group, "Neptec Internal LCS Operators Overview," Aug-2003.

[18] D. OuYang and H. Feng, "On the normal vector estimation for point cloud data from smooth surfaces," *Computer-Aided Design*, vol. 37, no. 10, pp. 1071-1079, Sep. 2005.

[19] B. K. P. Horn, "Closed-form solution of absolute orientation using unit quaternions," *Journal of the Optical Society of America A*, vol. 4, no. 4, p. 629, Apr. 1987.

[20] F. L. Markley, Y. Cheng, J. L. Crassidis, and Y. Oshman, "Averaging Quaternions," *Journal of Guidance, Control, and Dynamics*, vol. 30, no. 4, pp. 1193-1197, Jul. 2007.