B17132216

TA
1634
.S54
2006

INTELLIGENT SELECTION OF SCANNING AREA FOR ICP BASED POSE ESTIMATION OF SPACE STRUCTURES

by

Kamran Shahid, B.Eng

Ryerson University, 2004

A thesis

presented to Ryerson University

in partial fulfillment of the requirements for the degree of

Master of Applied Science

In the program of

Mechanical Engineering

Toronto, Ontario, Canada 2006

AUTHOR'S DECLARATION

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

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

I further authorize Ryerson University to reproduce this thesis 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

DETERMINATION OF OPTIMAL SCANNING AREA FOR ICP BASED POSE ESTIMATION OF SPACE STRUCTURES

Master of Applied Science 2006, Kamran Shahid

School of Graduate Studies, Ryerson University

Future autonomous satellite repair missions would benefit from higher accuracy pose estimates of target satellites. Constraint analysis provides a sensetivity index which can be used as a registration accuracy predictor. It was shown that point cloud configurations with higher values of this index returned more accurate pose estimates than unstable configurations with lower index values. Registration tests were conducted on four satellite geometries using synthetic range data. These results elucidate a means of determining the optimal scanning area of a given satellite for registration with the Iterative Closest Point (ICP) algorithm to return a highly accurate pose estimate.

ACKNOWLEDGMENTS

I would like to thank my thesis supervisor Dr. Galina Okouneva for granting me the opportunity to work on this topic.

I also give special recognition to Dr. John Enright and Dr. Don McTavish who helped me to tackle a few hurdles during the course of my research.

Special thanks to Piotr Jasiobedzki and Chris Langley at MD Robotics.

My fellow rocket scientists and friends Szymon "Radar" Buhajczuk, Damian "Pimp" Rogers, Sandun "Pablo" Weerasinge and Nathan "Gizmo" Eng. My best friend Nasim, his wife Jolie and children Kamran, Anysia and Abrar have always believed in me. Thank you all.

The long hours in the lab were made bearable by my friend Godard (yes...only one name). Although we worked on different topics, he was always there to discuss concepts and helped me to further develop my ideas. He also kept me sane during the more difficult moments. Thank you, my friend.

Finally, to my darling Michelle. Thank you for your love and support.

DEDICATION

This thesis is dedicated to my first teacher and professor:

my father Dr. Mohammed Sharif Shahid.

TABLE OF CONTENTS

Author's Declaration . . . . **iii**

Abstract . . . . **v**

Acknowledgments . . . . **vii**

Dedication . . . . **ix**

Table of Contents . . . . **xi**

List of Figures . . . . **xv**

List of Tables . . . . **xvii**

Nomenclature . . . . **xix**

Chapter 1. Introduction **1**

1.1 Space Vision System 2
1.1.1 Satellite Capture 2
1.1.2 Why Vision? 2
1.1.3 Space Visual Environment 3
1.1.4 Vision System Range 4

1.2 Computer Vision 4
1.2.1 Processing Methods 4
1.2.2 Laser Range Scanners 5

1.3 Current Projects 6
1.3.1 Orbital Express 6
1.3.2 Rendezvous Laser Vision System 6

1.4 Thesis Framework 7
1.4.1 Motivation 7
1.4.2 Solution Approach 7
1.4.3 Objectives 8
1.4.4 Outline 10

Chapter 2. Range Data Synthesis **11**

2.1 Introduction 11

2.2 Raytracing Preliminaries 12
2.2.1 Model Simplification 12
2.2.2 Bounding Box 12
2.3 Open GL Viewing Volume 14
2.4 Raytracing Algorithms 14
2.4.1 Sunday Algorithm 16
2.4.2 Badouel Algorithm 17
2.4.3 Moller-Trumbore Algorithm 18
2.4.4 Held Algorithm 19
2.4.5 Segura Algorithm 20
2.4.6 Raytracing Method Comparison 21
2.5 Voxel Raytracing 22
2.5.1 Voxelize Model 22
2.5.2 kd Trees 25
2.5.3 Voxel Traversal 27
2.5.4 Voxel Size 29
2.5.5 Performance 30
2.6 Summary 30

Chapter 3. Lidar Model 33
3.1 Introduction 33
3.2 Model Equations 33
3.3 Baseline Range Scanner 36

Chapter 4. Range Image Registration 39
4.1 Introduction 39
4.2 Overview 40
4.3 ICP Background 42
4.3.1 Closest Point Matching 44
4.3.2 Registration Error 47
4.3.3 Initial Pose Estimate 48
4.3.4 The ICP Algorithm 49
4.4 Experimental Method 50
4.4.1 Satellite Subsection Scanning 50
4.4.2 ICP Registration 51
4.5 Summary 53

Chapter 5. Constraint Analysis **55**
5.1 Introduction 55
5.2 Sensitivity Vector 56
5.3 Scale Dependence 61
5.3.1 Unstable Configurations 62
5.4 Measure of Stability 63
5.4.1 Unfit Conditioning Measures 66
5.4.2 Noise Amplification Index 67
5.5 Ideal NAI and Registration 69
5.6 Summary 71

Chapter 6. Results & Conclusions **73**
6.1 Ideal Region 73
6.2 Improved ICP Performance 78
6.3 NAI Implementation 78
6.3.1 Predictive Scanner 78
6.3.2 Look Up Tables 80
6.4 Conclusions 80
6.5 Future Work 81

References **83**

Appendix A. Raytrace Algorithm Comparison **87**

Appendix B. Point to Plane Derivation **89**

Appendix C. Eigenvalue Constraint Derivations **91**

LIST OF FIGURES

Figure 1.1 Example of extreme space lighting conditions - full solar Illumination and Earth albedo from [Blais et al, 2000] 3

Figure 1.2 Solution Approach Block Diagram 9

Figure 2.2 Raytrace Bounding Box 13

Figure 2.3 Raytracing Camera and Target Frames 13

Figure 2.1 Radarsat Docking Ring Mesh Reduction 13

Figure 2.4 Open GL Perspective Viewing Volume 15

Figure 2.5 Ray-Triangle Intersection using Barycentric Coordinates 17

Figure 2.6 Comparison of Raytracing Methods on Stardust Satellite Model 23

Figure 2.7 Open GL Indexed Mesh Array Structure 24

Figure 2.8 2D Conceptual kd Tree 25

Figure 2.9 2D Voxel Traversal 27

Figure 2.10 Basic 2D Voxel Traversal Loop 28

Figure 2.11 Point in Voxel Condition 29

Figure 2.12 Raytracing Time vs Voxel Size 30

Figure 2.13 Held vs Voxel Raytracing Comparison 31

Figure 2.14 Satellite Models With Their Respective Point Clouds 32

Figure 3.1 Triangulation Geometry from [Beraldin et al, 1993] 34

Figure 3.2 Astigmatism Between X & Y Axes and Simplified Geometrical Model from [Blais et al, 2000] 35

Figure 3.3 35

Figure 4.1 Normal Point to Plane Projection 44

Figure 4.2 Triangle Parametric Regions 45

Figure 4.3 Project Point onto line segment 46

Figure 4.4 Registration Error Transformation 47

Figure 4.5 Window Scanning Parameters 51

Figure 4.6 Window Scanning Flowchart 52

Figure 4.7 ICP Registration Flowchart 52

Figure 5.1 Scale Effects on Rotation and Translation 61

Figure 5.2 Constraint Analysis Shape Configurations 1 64
Figure 5.3 Constraint Analysis Shape Configurations 2 65
Figure 5.4 Uncertainty Hyperellipsoid 66
Figure 5.5 Test Model 70
Figure 5.6 Reduced Rotation & Translation Error with Increased NAI 70
Figure 6.1 Radarsat Optimal Scanning Location Sample Result 74
Figure 6.2 Deep Space 1 Optimal Scanning Location Sample Result 75
Figure 6.3 Stardust Optimal Scanning Location Sample Result 76
Figure 6.4 Shuttle Optimal Scanning Location Sample Result 77
Figure 6.6 Predictive Scanner Implementation 79
Figure 6.5 NAI Predictive Scanner Output 79
Figure 6.7 Varying Optimal Window Size 80
Figure A.1 Radarsat Raytrace Algorithm Comparison 87
Figure A.2 Deep Space 1 Raytrace Algorithm Comparison 88
Figure A.3 Shuttle Raytrace Algorithm Comparison 88
Figure B.1 Point to Plane Correspondence 89

LIST OF TABLES

TABLE 1.1 Computer Vision Ranges for Satellite Capture 4

TABLE 2.1 Triangle count of satellite models before and after simplification. . . 12

TABLE 2.2 Raytrace Algorithm Comparison 21

TABLE 3.1 Lidar RELAVIS Instrument Specifications 36

TABLE 4.1 Qualitative comparison of algorithm performance (1=best, 4=worst). N-Noise, ID-No Noise, IN-Isotropic Noise, AN-Anisotropic Noise. . . 43

TABLE 5.1 Scalar Measures of Matrix Stability 66

TABLE 6.1 Impact of NAI on number of ICP Iterations Required for Convergence 78

NOMENCLATURE

A	Triangle area (m^2)
D_g	Lidar axis separation distance (m)
ΔD	Registration Residuals
d	Lidar triangulation base (m)
F	Distance to far plane (m)
f_o	Effective focal length (m)
h	Voxel size (m)
H	Near plane height (m)
$\boldsymbol{M}$	Covariance Matrix
$\vec{n}$	Plane normal (m)
N	Distance to near plane (m)
$nCols$	Width of image screen (pixels)
$nRows$	Height of image screen (pixels)
$\{m_i\}$, $\{\vec{p}_i\}$	Point cloud sets
$\bar{m}, \bar{p}$	Point cloud centroids (m)
$\vec{P}_0$, $\vec{P}_1$	Raytrace ray coordinates (xyz m)
q_i	Eigenvectors
Δp	Laser detector spot size (m)
$\vec{p}_s$	Surface point
$\boldsymbol{R}$	Rotation matrix
Δt	Pose vector parameter error
S_{xx}	Covariance matrix element
u_c	World screen x coordinate (m)
v_r	World screen y coordinate (m)
t	Parametric value of ray
$\boldsymbol{T}$	Transformation matrix
$U \Lambda V^T$	Singular value decomposition
$(voxX, voxY)$	Voxel coordinates (m)
W	Near plane width (m)

Greek Symbols

α	Vertical field of view (rad)
λ, μ	Barycentric triangle coordinates
θ, φ	Lidar deflection angles (rad)
τ	Lidar flight time (s)
Ψ_P	Scatter Matrix

Chapter 1

INTRODUCTION

Manned spaceflight repair missions are far too costly and are only reserved for the most expensive spacecraft, such as Hubble and the ISS, and restricted in range to low earth orbit. There are many communications satellites that are in a geosynchronous orbit and beyond the reach of the Space Shuttle. The servicing of these satellites using manned missions is not possible. Also, the communications lag and limited bandwidth between ground and on-orbit satellites make teleoperated servicing vehicles unreliable. Geosynchronous communications satellites are a vital link in the terrestrial communication chain. In the future, the ability to reach, repair and return problematic satellites back to active service in a short turn around time will be highly advantageous. The solution is the development of maintenance satellites that will be capable of executing these complex repair missions. Consequently, repair satellites will be required to possess a high degree of autonomy and be able to conduct the required servicing operations with minimal assistance. Future satellites will be designed for servicing by these autonomous servicing systems by including the proper interfaces for docking, berthing, fluid transfer and installation of new computer hardware and software. These modifications will come at an added cost. However, this cost is offset by the ability to correct any future operational issues. This added functionality will extend the life of satellites, reduce launching, production and operating costs. Also impaired satellites that would ordinarily be lost will be able to be recovered.

1.1 Space Vision System

1.1.1 Satellite Capture

Satellite capture is preformed using a robotic arm capture and berthing technique. The lag time between satellite and ground station is generally several seconds [Inaba & Oda, 2000] rendering a closed loop control that includes a ground station unstable. Hence, an on-board autonomous machine vision system is necessary to conduct satellite capture. This calls for the development of a means to estimate satellite motion using on-board sensors combined with a closed loop control of the manipulator. Satellite capture is a difficult challenge as it involves a full six degree problem in a zero gravity environment. This task is further compounded since the motion of a manipulator capture arm will induce motion in the servicing satellite. A failed satellite capture could cause a spin in the target satellite. If this state cannot be corrected through stabilization, the satellite may be permanently lost.

1.1.2 Why Vision?

To conduct rendezvous, docking and repair missions, a high degree of accuracy is required for estimating the relative translation and orientation (pose) of target satellites. Active systems such as radar and differential GPS systems can be used for relative position, but these systems give little information regarding the rotational orientation of the target satellite. For a servicing satellite to have the required level of autonomy, it needs to be able to replicate the human ability to recognize complex objects in arbitrary orientations over a range of distances and lighting conditions [Hollander, 2000]. Currently, it is not possible to grant a machine the full abilities of human vision. However, it is possible to bestow some basic vision functionality, such as depth and range, to a semi-automated system. Most robotic operations are inherently risky thus driving the need for considerable amounts of sensing data to improve reliability. The ever changing space environment can cause conditions to change unexpectedly requiring a reassessment of operational plans. Visual processing will allow ground controllers to quickly react to these changing conditions.

1.1.3 Space Visual Environment

The space environment poses many challenges for an operational orbital vision system. For thermal protection, spacecraft are usually covered with a thermal blanket or a loosely attached reflective foil. When illuminated, this foil causes shadowing and reflectivity that pose problems for imaging systems. The range of illumination can vary from full solar exposure to complete eclipse. The effects of solar illumination and Earth albedo[1] can cause some satellite geometric features to be undetectable in images. This degrades the imaging accuracy and reliability. Variables affected include imaging dynamic range, poor signal quality, saturation and shadowing. Compensation for these effects require extended dynamic range and sophisticated processing algorithms.

Camera vision systems are affected by both over illumination and the lack of ambient light. Lasers scanners are robust to lighting conditions and are effectively 100% operational regardless of sun interference, saturation, shadowing and insufficient ambient light. Figure 1.1 shows the effect of extreme illumination and Earth albedo. Also, examples of artificial targets can be seen on the satellite body.

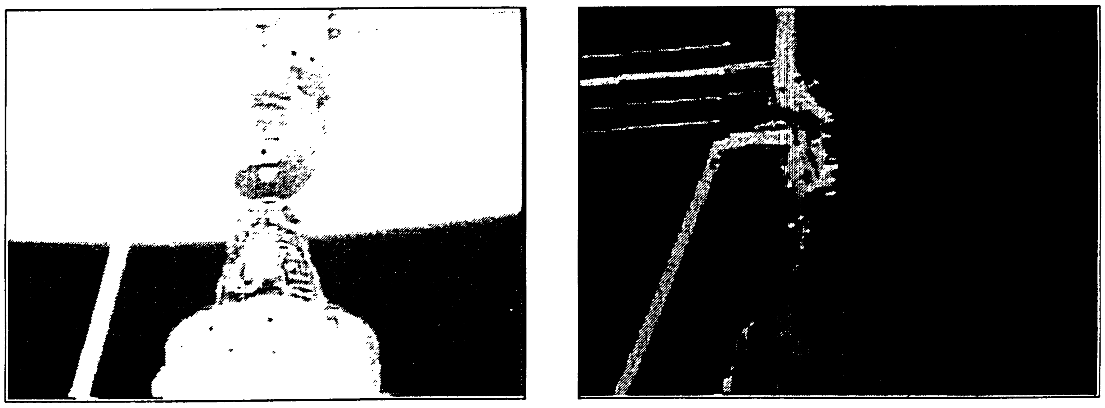


Figure 1.1 Example of extreme space lighting conditions - full solar Illumination and Earth albedo from [Blais et al, 2000].

1. the fraction of incoming sunlight reflected back into space

1.1.4 Vision System Range

[Jasiobedzki et al, 2001] outlined the ideal operational range of a space vision system. The operational range of a vision system for satellite proximity operations is from 100m to contact. This distance is further subdivided into three categories: long, medium and short. The operational requirements for these ranges are outlined in Table 1.1.

TABLE 1.1 Computer Vision Ranges for Satellite Capture

Range	Distance	Function
Long	100-20m	determine satellite bearing and distance
Medium	20-2m	accurately determine satellite pose
Short	2m to contact	track docking interface

Based on these range designations, a vision system must be capable of returning accurate pose estimates for the medium and short ranges.

1.2 Computer Vision

1.2.1 Processing Methods

The three main methods of determining the position and orientation of one spacecraft relative to another differ in the type of features utilized. These are:

1. artificial targets on spacecraft [Thorsley et al, 2004]
2. matching of 2D/3D features to 3D models [Gelfand et al, 2005]
3. matching of 3D surface data to 3D models [Besl & McKay, 1992]

Single or stereo cameras can be used to detect specially designed, high contrast fiducial marks affixed to the target satellite body. Vision processing algorithms are designed to detect these artificial targets in the satellite image and determine the pose of the given satellite. The use of artificial markers is limited by the distance and viewing angle. All flight operational steps including tracking and motion have to be designated at the pre-mission design stage. This method is not robust to unexpected events requiring additional tasks. Also, an unforeseen occlusion of the targets by a satellite appendage or a sudden change in lighting conditions can render this type of vision system inoperable.

Detecting natural 2D/3D features eliminates the need for the use of artificial targets. This format matches the detected feature with a known CAD model of the satellite to establish its pose. However, correct feature correspondence is difficult to establish under certain conditions and may lead to an inaccurate pose estimate. This method is best suited for short range distances.

The most promising approach are systems that will match the satellite 3D surface geometry with a known 3D CAD model. This method makes use of all of the geometric information available from all visible surfaces and eliminates the need for maintaining correspondence between a small number of geometric feature elements. This method is less sensitive to shadowing, illumination conditions and surface reflectivity and is best suited for medium range operations. 3D systems are scale and rotation invariant unlike 2D perspective projection methods. LIght Detection And Ranging (LIDAR) imaging systems are now capable of acquiring data at speeds equivalent to video rates making range imagery a viable option for space robotics.

1.2.2 Laser Range Scanners

A laser range scanner acquires the three dimensional geometry of an object by projecting rays of laser light and determining the intersection with a target body. The data produced is a set of 3D coordinates of the intersection points. A range scanner operates by emitting a beam of laser light onto a target, collecting and projecting the reflected beam onto a linear detector[1]. Two high speed galvanometers are used to steer the laser beam onto a target within the field of view of the camera. This process returns two angular positions of the reflectors and the location of the laser spot on the detector. These values can be used to determine the *xyz*

1. a photo diode array or a charged couple device

Cartesian coordinates of the intersection point in reference to the camera frame. The range of the target can be obtained by using triangulation based on the geometric arrangement of the internal laser components, or by measuring the time of flight of the laser beam.

1.3 Current NASA Projects

1.3.1 Orbital Express

The Boeing Company is developing the Orbital Express system aimed at demonstrating a fully autonomous on-orbit satellite servicing system. The Orbital Express system consists of two satellites: the Autonomous Space Transport Operations (ASTRO) servicing spacecraft and the NextSat modular serviceable target satellite. The system aims to demonstrate a fully functional autonomous rendezvous system operating at a distance of 7m to contact. Proximity operations such as soft capture, station keeping, on-orbit refuelling and component replacement will be demonstrated. The Baseline Integration Systems Test (BIST) of the ASTRO spacecraft was recently completed. This test included the operation of all major spacecraft subsystems and software individually and collectively for the first time. The Orbital Express launch is scheduled for October 2006.[1]

1.3.2 Rendezvous Laser Vision System

Optech and MD Robotics have developed the Rendezvous Laser Vision System (RELAVIS). The primary function of this integrated laser-based vision system is to support autonomous satellite rendezvous and docking operations. RELAVIS is a time-of-flight laser scanner that has the ability to produce highly accurate data over a range of 0.5 metres to 3 kilometres [Martin et al, 2004]. More specifically, it provides the bearing and range data of a target spacecraft which can be process to provide very accurate pose estimates of a target satellite.

1. Aerospace Daily and Defense Report www.aviationnow.com

1.4 Thesis Framework

1.4.1 Motivation

The broad focus of this thesis is to improve the accuracy of pose estimation for the purpose of satellite rendezvous. This aim could be realized in many different ways by suggesting changes in overall system design, hardware components or processing algorithms. It is the latter method of on-board processing that will be examined. It is hoped that with minor changes, a more efficient LIDAR scanning system will result.

Current registration[1] techniques make use of the total scanning area. However, little work has been done which examines if this strategy is best. This leads to posing the following questions:

- How does model geometry affect registration?
- Is it possible to predict the registration accuracy of a given data set?
- Are some scanning regions better suited for registration than others?

Particular attention is placed on the last question. When looking at a given satellite geometry, one can intuitively select a good scanning area. However, this intuitive human ability to select a scanning location is based on the local geometry of candidate scanning regions. In order to pass this intelligent selection ability on to a LIDAR system, a mathematical basis for quantifying regional geometry is needed.

1.4.2 Solution Approach

This section gives an overview of the testing scheme developed in this thesis in order to meet the objectives outlined in the next section. Figure 1.2 shows the solution approach in the form of a block diagram. The three major blocks that need to be developed are highlighted. The

1. the process of transforming different sets of data into one coordinate system thereby determining the transformation between the starting frames.

LIDAR scanner will be the first system to be developed. This block intakes the specified inputs and outputs a 3D range image. These input parameters will also specify a particular scanning region. Once the registration block has been formulated, the generated point cloud data can be processed by the registration algorithm to determine the pose accuracy.

The final block to be developed will be the intelligent region selection block. At first this stem will be linked to the registration block in order to demonstrate the utility of this method, but this link will be severed once this relationship has been established. The purpose of this system will be to estimate the registration accuracy of a point cloud configuration without requiring the use of the registration algorithm. This will allow the bypass of the registration system to reach the accuracy block. The utility of this block is the focus of this thesis.

There will also be a clear dividing line that will separate the simulation zone from the processing zone. The zones cannot be completely isolated from each other since both zones require the use of the same satellite model. Unfortunately, there is no current solution that can totally separate these zones. A possible future solution would be to use a physical LIDAR scanner with a physical scaled model of a satellite. For now, the only available option is to represent the satellite model in two different formats.

1.4.3 Objectives

The primary objective of this thesis is to establish a heuristic for finding the ideal scanning area of a given satellite geometry using a LIDAR scanner. A particular region is designated as ideal if it returns a highly accurate pose estimate over all candidate regions when processed with a registration[1] algorithm. Before registration tests can be conducted, a suitable registration algorithm will be established based on the different forms currently used. Also a

1. the process of transforming different sets of data into one coordinate system thereby determining the transformation between the starting frames.

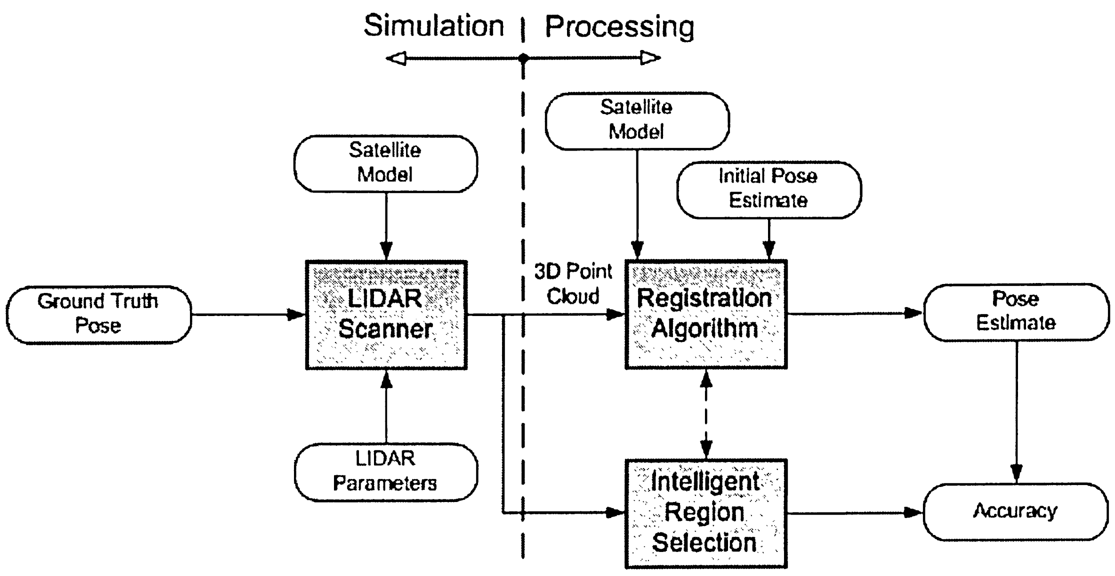


Figure 1.2 Solution Approach Block Diagram

ranking measure will be established that will determine how suitable a given geometric region is for registration purposes.

To ensure that any results are not model specific, the *Stardust*, *Deep Space 1*[1] and *Space Shuttle*[2] models are included along with the *Radarsat* model.

To accomplish the main objective, a method of generating synthetic LIDAR data will be developed. Differing raytracing techniques will be investigated to determine an efficient means to extract point cloud data from the satellite mesh structures. Noise will then be added to these ideal data sets in accordance with an actual LIDAR instrument.

The emphasis of this study is to develop a method that can be used to improve the accuracy of satellite pose estimates for autonomous operations.

1. Caltech Laboratory for Spacecraft and Mission Design http://www.lsmd.caltech.edu/tools/drawcraft/spacecraft.htm
2. NASA Liftoff to Space Exploration http://liftoff.msfc.nasa.gov/vr/vr.html

1.4.4 Outline

The raytracing method used to generate synthetic range data is outlined in Chapter 2. The method used to add noise to this data in accordance with the RELAVIS LIDAR unit is presented in Chapter 3. The Iterative Closest Point algorithm (ICP) is discussed in Chapter 4 along with the experimental environment used to conduct registration trials. Chapter 5 outlines constraint analysis and introduces the noise amplification index used to predict registration accuracy. Finally, results and conclusions are reported in Chapter 6.

Chapter 2

RANGE DATA SYNTHESIS

2.1 Introduction

In order to investigate registration on 3D point clouds[1], a method to derive point cloud LIDAR data was needed. A lack of readily available LIDAR equipment and physical satellite models lead to the development of techniques for generating simulated data. The computer graphics method of raytracing simulates the behaviour of a laser range scanner by projecting virtual rays to intersect with a model composed of a triangular mesh. Such a mesh structure can exactly represent planar features, but is only an approximation to curved surfaces. However, as the mesh is made more fine, the mesh geometry more accurately represents the curvature. Raytracing is a technique for rendering 3D graphics with complex light interactions producing realistic computer images. The basics of raytracing involves projecting a ray of light from a virtual point, through a pixel on an image plane and into the scene to determine object intersections. The pixel is then set to the colour value of the intersection surface. The full functionality of raytracing reflections, refractions and shadowing are not utilized in this thesis since only the ray-model intersection point is required. The raytracing method was used along with the Open GL library [Woo et al, 1997] to produce a program capable of 3D synthetic LIDAR point cloud data. The outline of the development of this program is contained within this chapter.

1. a set of three-dimensional points describing the outline or surface features of an object such as the data produced by a LIDAR scanner

2.2 Raytracing Preliminaries

2.2.1 Model Simplification

The *Radarsat, Deep Space 1, Stardust* and *Shuttle* models were all originally available in VRML 1.0 format. The mesh geometry as originally defined in VRML format was not suitable for raytracing methods used in this thesis. The mesh is overly complex with many internal surfaces that will never be visible. These surfaces were erased manually in order to reduce the overall number of surfaces. As outlined in Section , raytracing involves numerous ray/triangle intersection tests. Three dimensional models with a minimal number of surfaces are preferable to reduce the number of these intersection tests. Once all redundant surfaces were removed, the mesh is further simplified by using a triangle decimation software which reduces the number of triangles in the mesh, but preserves the original object topology. Table 2.1 shows the triangle mesh reduction for each satellite model used in this thesis.

TABLE 2.1 Triangle count of satellite models before and after simplification.

Model	Original	Simplified
Deep Space 1	8852	1976
Radarsat	8566	2046
Stardust	5606	1896
Shuttle	2476	1778

Figure 2.1 Shows the result of mesh reduction on the *Radarsat* docking ring. Mesh reduction and manipulation was done using a combination of *Catia, AutoCad, Lightwave 3D, Rhinoceros, CXC* and *Crossroads.*

2.2.2 Bounding Box

A bounding box that encapsulates the satellite model helps to decrease raytracing time by eliminating the need to test for ray/model intersection if the ray misses the object entirely. The ray is first tested for intersection with the bounding box and further model intersection is only conducted once this test passes. Figure 2.2 illustrates the bounding box on the *Stardust*

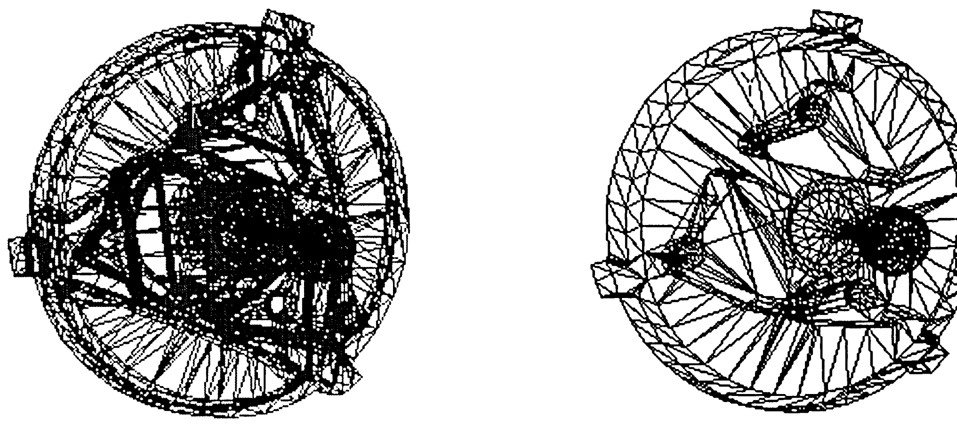


Figure 2.1 Radarsat Docking Ring Mesh Reduction

satellite model. Voxel raytracing (Section 2.5) makes use of the enter and exit locations returned by the bounding box intersection test. The pose estimate of the satellite is always in reference to the camera frame, thus it is natural to consider this frame as the intertial raytracing frame. However, this choice will severely hamper raytracing efficiency.

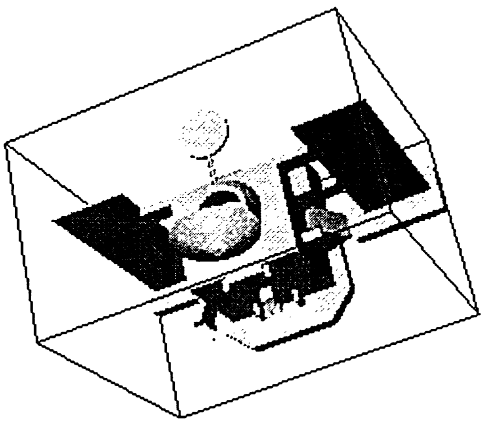


Figure 2.2 Raytrace Bounding Box

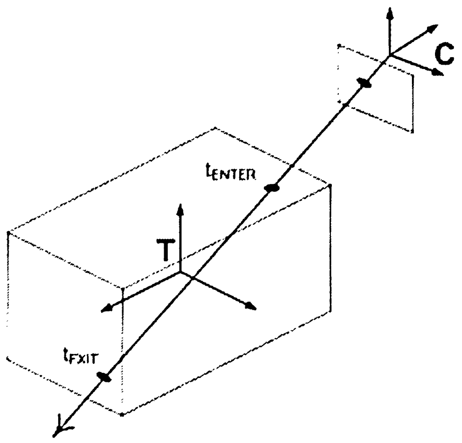


Figure 2.3 Raytracing Camera and Target Frames

The satellite mesh geometry is defined in terms of its own body frame **T** as shown in Figure 2.3. It is more prudent to define the target frame as inertial and consider the camera frame mobile. This eliminates the need to apply a transformation to all of the mesh vertices in order to reference them to the camera frame. There will be two transformations per ray, however this is significantly less than having to transform the entire mesh geometry.

2.3 Open GL Viewing Volume

The rays required for raytracing are generated in accordance with the *Open GL* viewing volume as shown in Figure 2.4. The parameters of the viewing volume define perspective projection matrix used to project the model onto the viewing screen. Only parts of the model that lie within this volume are visible. The image plane lies on the near plane and has a width of $nCols$ and height $nRows$ in pixels. The angle α defines the vertical field of view in the yz camera plane. Specifying the viewing volume also establishes the internal perspective projection matrix within the *Open GL* graphics pipeline. The viewport transformation establishes the relationship between the 3D world space to the 2D pixel space by overlaying a grid of pixels onto the near plane. An image screen is established that extends from $-H$ to H in the x direction and from $-W$ to W in the y direction where:

$$H = N\tan\left(\frac{\alpha}{2}\right) \qquad W = H \cdot \frac{nCols}{nRows} \tag{2.1}$$

The screen has $nCols$ by $nRows$ of pixels on the viewport [Hill, 2001]. The relationship between screen pixel coordinates (c, r) and screen world coordinates (u_c, v_r) is given by:

$$\begin{aligned} u_c &= -W + W\frac{2c}{(nCols-1)} \quad , \quad c = 0, 1 \ldots nCols - 1 \\ v_r &= -H + H\frac{2r}{(nRows-1)} \quad , \quad r = 0, 1 \ldots nRows - 1 \end{aligned} \tag{2.2}$$

2.4 Raytracing Algorithms

The different methods of finding ray/triangle intersections are outlined in this chapter. The two geometric entities are defined as a ray originating at $\vec{P_0}$ passing through $\vec{P_1}$ having the parametric equation:

$$\vec{P} = \vec{P_0} + t(\vec{P_1} - \vec{P_0}) \tag{2.3}$$

and a triangle having vertices $\vec{V_0}$, $\vec{V_1}$ and $\vec{V_2}$ with normal vector $\hat{n}$ (see Figure 2.5). The

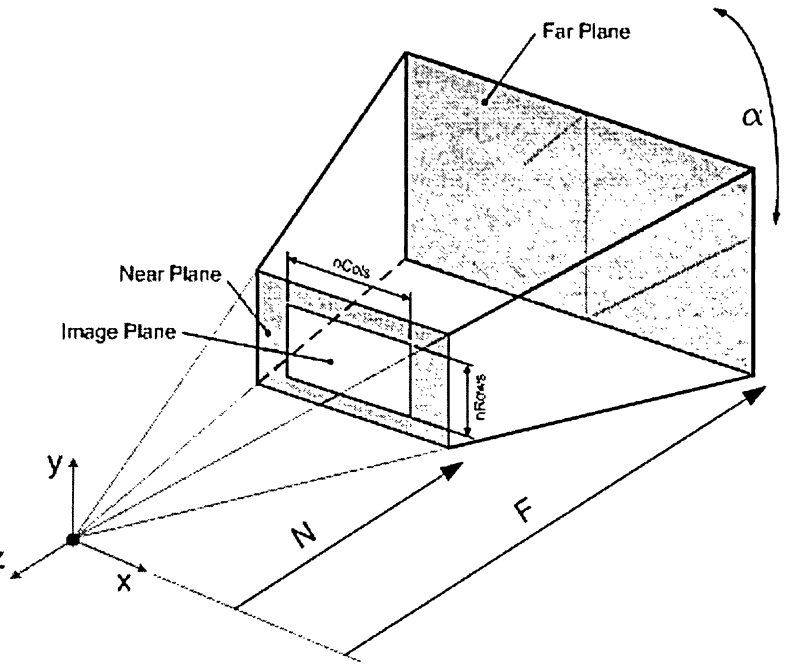


Figure 2.4 Open GL Perspective Viewing Volume

transformation from the camera to the target frame T_{CT} is specified.

The steps involved in raytracing to produce a 3D point cloud data set are:

1. specify the scanning area on the image plane (see Figure 2.4) (the full image plane is used for a full view scan)
2. select a pixel from the scanning area and convert the pixel coordinates into world coordinates using Equations (2.1) and (2.2) designating this point as $\vec{P_1}$ and the origin is designated $\vec{P_0}$ as defined in the camera frame
3. both $\vec{P_1}$ and $\vec{P_0}$ are transformed to the target frame using $(T_{CT})^{-1}$
4. the parametric ray equation is specified by using these transformed points using Equation (2.3)
5. ray/triangle intersection tests are conducted for every triangular mesh element
6. the intersection with the smallest parametric value t is designated as the point of intersection

7. the intersection point is defined in the camera frame so use T_{CT} to reference it back to the camera frame
8. repeat steps 2-8 to generate a point cloud data set $\{\vec{p}_i\}$

2.4.1 Sunday Algorithm

The three vertices of a triangle define a plane. The technique proposed by [Sunday, 2001] begins by determining the point of intersection $\vec{P}_i$ between the ray and this plane. A plane with a normal $\hat{n}$ passing through point $\vec{V}_0$ has the equation:

$$\hat{n} \cdot (\vec{P} - \vec{V}_0) = 0 \tag{2.4}$$

Combining Equations () and (2.4) gives t_i the parametric value of the intersection point:

$$t_i = \frac{\hat{n} \cdot (\vec{V}_0 - \vec{P}_0)}{\hat{n} \cdot (\vec{P}_1 - \vec{P}_0)} \tag{2.5}$$

Equation () can be used to determine the actual hit location $\vec{P}_i$. The inclusion of point $\vec{P}_i$ within the triangle boundary now needs to be established. The parametric plane equation of a triangle, see Figure 2.5, is given by:

$$\hat{P} - \vec{V}_0 = \lambda(\vec{V}_1 - \vec{V}_0) + \mu(\vec{V}_2 - \vec{V}_0) = \lambda\hat{u} + \mu\hat{v} \tag{2.6}$$

where λ and μ are the barycentric coordinates.The values λ and μ are obtained using:

$$\lambda = \frac{\hat{w} \cdot (\hat{n} \times \hat{v})}{\hat{u} \cdot (\hat{n} \times \hat{v})} \qquad \mu = \frac{\hat{w} \cdot (\hat{n} \times \hat{u})}{\hat{v} \cdot (\hat{n} \times \hat{u})} \tag{2.7}$$

where $\hat{u} = \vec{V}_1 - \vec{V}_0$, $\hat{v} = \vec{V}_2 - \vec{V}_0$, $\hat{w} = \vec{P}_i - \vec{V}_0$, and $\hat{n}$ the plane normal. A point that is within the triangle must fulfill the conditions:

$$\lambda \geq 0,\ \mu \geq 0 \text{ and } \lambda + \mu \leq 1 \tag{2.8}$$

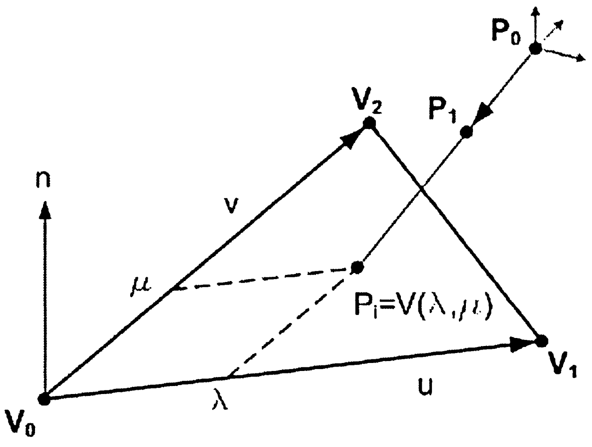


Figure 2.5 Ray-Triangle Intersection using Barycentric Coordinates

Thus, an intersection point $\vec{P_i}$ can be tested for inclusion [Ericson, 2004] within a triangle by comparing its parametric values against the aforementioned conditions.

2.4.2 Badouel Algorithm

[Badouel, 1990] begins by finding the intersection of the ray with the triangle plane in the same manner as [Sunday, 2001] using Equation 2.5. Equation 2.6 is then expanded into a system of linear equations:

$$\begin{aligned} x_P - x_0 &= \lambda(x_1 - x_0) + \mu(x_2 - x_0) \\ y_P - y_0 &= \lambda(y_1 - y_0) + \mu(y_2 - y_0) \\ z_P - z_0 &= \lambda(z_1 - z_0) + \mu(z_2 - z_0) \end{aligned} \tag{2.9}$$

Only two of the equations from (2.9) are needed to obtain λ and μ. This is analogous to projecting the triangle onto one of the primary planes xy, xz or yz. If the triangle is perpendicular to one of the primary planes, its projection will degenerate into a line. To avoid this problem and to select the largest possible projection, the dominant axis of the triangle plane normal is determined. The projection plane is composed of the two remaining axes. Given a plane normal $\hat{n}$, the dominant axis is found using:

$$q_0 = \begin{cases} i & \text{if } |n_x| = max(|n_x|, |n_y|, |n_z|) \quad q_1 = j, q_2 = k \\ j & \text{if } |n_y| = max(|n_x|, |n_y|, |n_z|) \quad q_1 = i, q_2 = k \\ k & \text{if } |n_z| = max(|n_x|, |n_y|, |n_z|) \quad q_1 = i, q_2 = j \end{cases} \tag{2.10}$$

The projection plane is given by q_1 and q_2. Let (u, v) be the two dimensional coordinates of a vector on this plane. Equation (2.9) can be reduced to:

$$\begin{aligned} u_0 &= \lambda u_1 + \mu u_2 \\ v_0 &= \lambda v_1 + \mu v_2 \end{aligned} \tag{2.11}$$

where :

$$\begin{aligned} u_0 &= P_{q1} - V_{0_{q1}} & u_1 &= V_{1_{q1}} - V_{0_{q1}} & u_2 &= V_{2_{q1}} - V_{0_{q1}} \\ v_0 &= P_{q2} - V_{0_{q2}} & v_1 &= V_{1_{q2}} - V_{0_{q2}} & v_2 &= V_{2_{q2}} - V_{0_{q2}} \end{aligned} \tag{2.12}$$

The solutions of λ and μ are:

$$\lambda = \frac{\begin{vmatrix} u_0 & u_2 \\ v_0 & v_2 \end{vmatrix}}{\begin{vmatrix} u_1 & u_2 \\ v_1 & v_2 \end{vmatrix}} \quad \text{and} \quad \mu = \frac{\begin{vmatrix} u_1 & u_0 \\ v_1 & v_0 \end{vmatrix}}{\begin{vmatrix} u_1 & u_2 \\ v_1 & v_2 \end{vmatrix}} \tag{2.13}$$

These values can now be used to test for the inclusion of $\vec{P_i}$ within the triangle using the conditions of Equation (2.8).

2.4.3 Moller-Trumbore Algorithm

[Moller & Trumbore, 1997] use a technique that delays the calculation of the intersection point. Equations () and (2.6) are combined to give:

$$\vec{P_0} + t(\vec{P_1} - \vec{P_0}) = \vec{V_0} + \lambda(\vec{V_1} - \vec{V_0}) + \mu(\vec{V_2} - \vec{V_0}) \tag{2.14}$$

The strategy here is to solve for the barycentric parameters (t, λ, μ) directly without first having to find the point of intersection as do the methods of Sunday and Badouel. Equation (2.14) can be rearranged to give:

$$\begin{bmatrix} -\vec{D} & \vec{E_1} & \vec{E_2} \end{bmatrix} \begin{bmatrix} t \\ \lambda \\ \mu \end{bmatrix} = \vec{T} \tag{2.15}$$

where $\vec{D} = \vec{P_1} - \vec{P_0}$, $\vec{E_1} = \vec{V_1} - \vec{V_0}$, $\vec{E_2} = \vec{V_2} - \vec{V_0}$ and $\vec{T} = \vec{P_0} - \vec{V_0}$
The solution of Equation (2.15) is:

$$\begin{bmatrix} t \\ \lambda \\ \mu \end{bmatrix} = \frac{1}{\vec{A} \cdot \vec{E_1}} \begin{bmatrix} \vec{B} \cdot \vec{E_2} \\ \vec{A} \cdot \vec{T} \\ \vec{B} \cdot \vec{D} \end{bmatrix} \tag{2.16}$$

with $\vec{A} = \vec{D} \times \vec{E_2}$ and $\vec{B} = \vec{T} \times \vec{E_1}$. The use of $\vec{A}$ and $\vec{B}$ increase the efficiency of the algorithm by only evaluating the respective cross products once and reusing the result. The triangle inclusion test using the conditions from Equation (2.8) can now be conducted. The point of intersection needs only be calculated if it is determined that the point does indeed lie within the triangle boundary. Also, the order in which the parametric values are actually calculated are (λ, μ, t). This allows early termination of the intersection test as soon as one of the parameters do not fall within the allowable range of:

$$\lambda \geq 0, \mu \geq 0, t \geq 0 \tag{2.17}$$

2.4.4 Held Algorithm

[Hill, 2001] begins by reducing the dimension of the intersection problem to a planar case as does [Badouel, 1990] (see Section 2.4.2). The inclusion of the point within the triangle boundary is conducted by examining the sign of the determinants $det(v_0, v_1, p_i)$,

$det(v_1, v_2, p_i)$ and $det(v_2, v_0, p_i)$. If a point p_i is contained within a 2D triangle $v_0 v_1 v_2$, the three determinants will have the same sign. This concept is based on the signed area of a triangle:

$$A = \pm\frac{1}{2}\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix} \tag{2.18}$$

2.4.5 Segura Algorithm

[Segura & Feito, 2001] use the definition of the signed volume of a tetrahedron to conduct an intersection test. The signed volume of a tetrahedron[1] with vertices D, A, B and C is given by:

$$[ABCD] = \frac{1}{6}\begin{bmatrix} A_x & A_x & A_x & 1 \\ B_x & B_y & B_z & 1 \\ C_x & C_y & C_z & 1 \\ D_x & D_y & D_z & 1 \end{bmatrix} \tag{2.19}$$

Consider a triangle ABC in R^3 and Q and Q' be two points on opposite sides of the plane ABC. The vertices of ABC need to be ordered in a counter clockwise orientation. This condition is easily accommodated as triangles in Open GL have a consistent counter clockwise or clockwise orientation. A counter clockwise orientaion can be arranged so that the plane normals point outward from the object model. The segment QQ' cuts ABC iff:

$$sign([Q'ABQ]) \geq 0 \quad and \quad sign([Q'BCQ]) \geq 0 \quad and \quad sign([Q'CAQ]) \geq 0 \tag{2.20}$$

These conditions allow for an early termination of the test if at any point one of these conditions is violated. Segura only evaluated this algorithm against those of Badouel and

1. a 3D pyramid-like shape made up of four triangular faces, three of which meet at each vertex

Moller-Trumbore in terms of intersection detection only. This method is included here to evaluate its performance when the intersection point is actually required.

2.4.6 Raytracing Method Comparison

The five different raytracing methods presented were all programmed into a raytracing algorithm in order to determine which method is the fastest. Figure 2.6 shows the results for the *Stardust* satellite model, with the results for the *Radarsat*, *Deep Space 1* and *Space Shuttle* models presented in Appendix A. The five methods can be divided into two categories: those that conduct calculations in 3D space and those that project onto 2D space. The algorithms show different performance due to varying complexities used in their formulation. Some of the algorithms need the triangle plane normal and if not provided have to calculate this vector. Other algorithms need the intersection point of the ray and the triangle plane. Table 2.2

TABLE 2.2 Raytrace Algorithm Comparison

Method	**Algorithm**	**Need ray/plane intersection?**	**Need Normal?**	**Complexity**
3D	Sunday	Yes	Yes	5
	Moller-Trumbore	No	No	2
	Segura	No	No	4
2D	Badouel	Yes	Yes	3
	Held	Yes	Yes	1

compares the five raytracing algorithms. The algorithms were tested by using similar programming implementations in order to provide a common computational overhead. The 2D methods are simpler than the 3D methods due to a reduction in dimension. The Moller-Trumbore and Segura algorithms have the advantage that the ray/plane intersection calculation can be delayed since the test for inclusion within the triangle boundary does not require the actual intersection point. However, the Segura algorithm shows poor performance due to its complexity. The Sunday algorithm fails across all categories and should not be used. When normals are precalculated, the Held algorithm is the fastest method. The satellite meshes all had normals defined thus the Held algorithm was used. These run times are for a C++ implementation of the algorithms on a AMD Opteron Dual Processor 1.79 GHz.

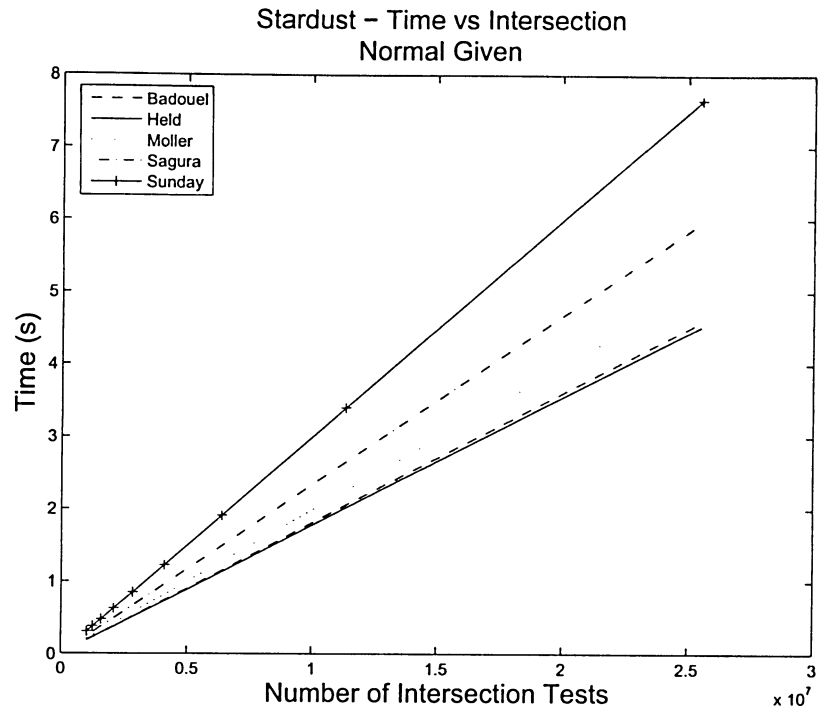
2.5 Voxel Raytracing

The first raytracing method that was developed used a brute force approach to conduct ray/ triangle intersection tests. For each ray, all of the mesh elements were tested to find the closest intersection. To increase efficiency, a reduction in the number of intersection tests was needed. Bounding volumes and space partitioning are the two main methods used for intersection culling. The concept of the first method was presented in Section 2.2.2. Bounding boxes can be placed around complicated model regions to avoid intersection tests with this region if there is no intersection with the bounding volume. The second approach is to partition the model into voxels[1] with each voxel containing a list of triangular mesh elements that intersect or lie within the voxel region of space. This subdivision allows for the prediction of the path that the ray will follow thereby finding the intersection point once the model is reached.

2.5.1 Voxelize Model

This is a pre-processing step which involves subdividing the model into small 3D volumetric cubes at a predetermined size. The size of the voxel is small enough so as to allow the capture of the finer details within the model. Small voxel sizing produces voxels that contain fewer mesh elements. The Open GL structure of indexed arrays (see Figure 2.7) allows for easy storage of the triangle elements that occupy a given voxel space. An example of how to access the first three triangle elements based on this mesh structure is shown in Figure 2.7.

1. 3D volumetric element



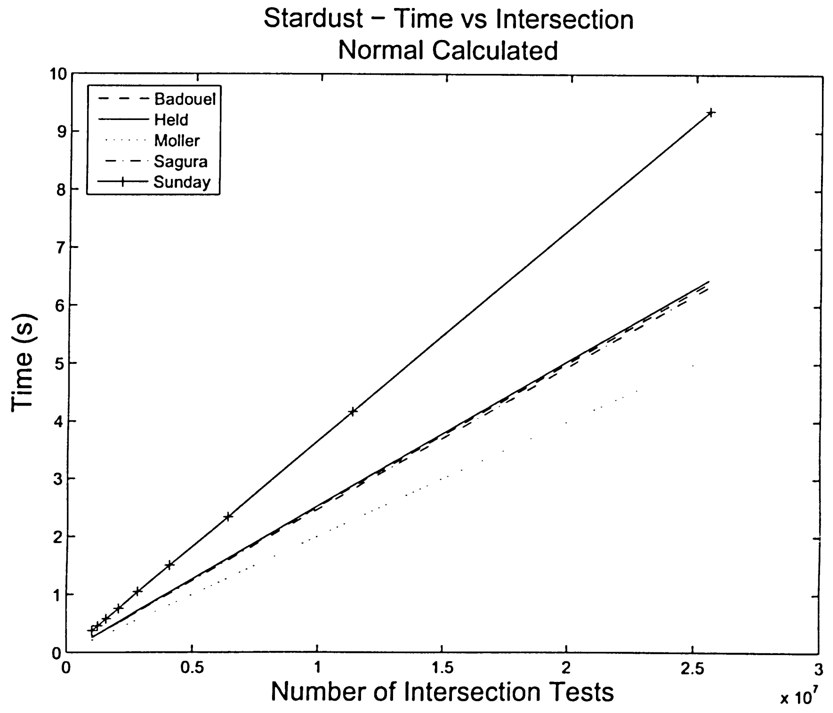
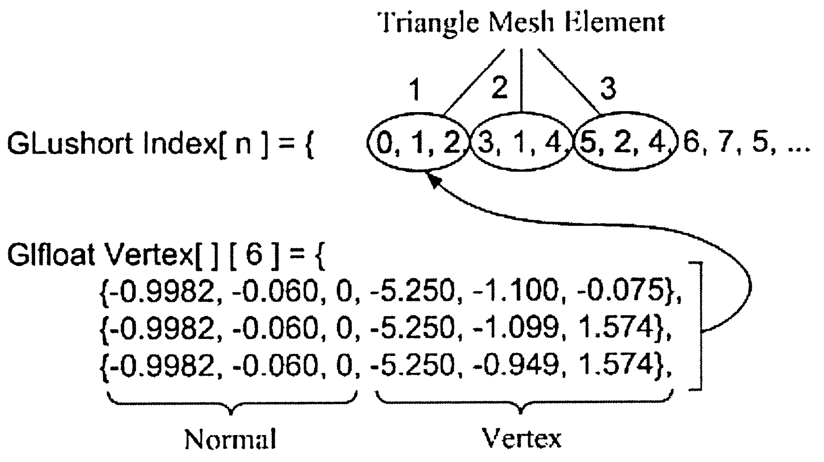


Figure 2.6 Comparison of Raytracing Methods on Stardust Satellite Model

A voxel is designated by its *XYZ* coordinates of its centre along with all of the triangle numbers that intersect or are contained within that particular region of space. The triangle number can be converted to its index elements and then the vertices, along with the normal,



```
GLushort Index[ n ] = {  0, 1, 2, 3, 1, 4, 5, 2, 4, 6, 7, 5, ...

Glfloat Vertex[ ] [ 6 ] = {
    {-0.9982, -0.060, 0, -5.250, -1.100, -0.075},
    {-0.9982, -0.060, 0, -5.250, -1.099, 1.574},
    {-0.9982, -0.060, 0, -5.250, -0.949, 1.574},
```

Triangle 1	**Triangle 2**	**Triangle 3**
Vertex[Index[0]][0...5]	Vertex[Index[3]][0...5]	Vertex[Index[5]][0...5]
Vertex[Index[1]][0...5]	Vertex[Index[1]][0...5]	Vertex[Index[2]][0...5]
Vertex[Index[2]][0...5]	Vertex[Index[4]][0...5]	Vertex[Index[4]][0...5]

Figure 2.7 Open GL Indexed Mesh Array Structure

which make up that particular triangular mesh element can be recovered. Voxel pre-processing of models has two steps:

1. Loop over the entire mesh, bounding each element with voxels and recording the *XYZ* coordinates and triangle number of voxel/element intersection.
2. Process the list and combine duplicate voxel entries.

The second steps ensure that for a given voxel location, all elements that intersect that region of space are accounted for. The triangle/box intersection code of [Moller, 2001] was utilized to conduct the intersection tests. Only those voxels that contain a portion of the satellite geometry are stored on the list.

2.5.2 kd Trees

For the voxel raytracer, a data structure was needed for the model voxel list to be able to quickly determine if a given voxel location exists on the list and what triangle elements are contained within the voxel. A kd tree[1] structure [Friedman et al, 1977] is a spatial subdivision scheme that is an efficient method for conducting nearest neighbour queries. Figure 2.8 shows

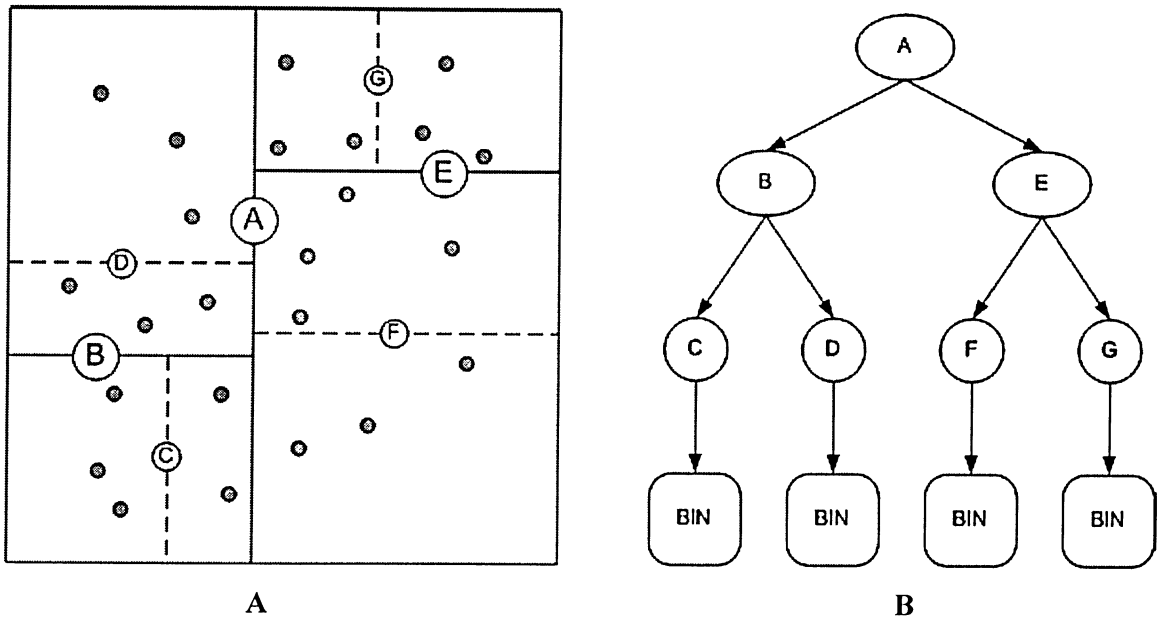


Figure 2.8 2D Conceptual kd Tree

the structure of a simple 2 dimensional kd tree. The dots in Figure 2.8A are points having xy coordinates. The subdivision scheme starts by examining the value $max(|x_{max} - x_{min}|,|y_{max} - y_{min}|)$. In this example, the x direction has the largest range of values. The splitting plane value is the mean value of the coordinates having the largest range. For node A, the x values will be averaged and stored within node A. Points having an x coordinate less than this splitting plane value are defined to be in the left region, while the remaining points are in the right region. This process continues for each sub-region until the number of points in a given region fall below an established threshold (eg. 5 points). The

1. k dimensions - in this case 3D

branch is terminated and is designated as a bin storing the coordinates of the points within that region.

A nearest neighbour search can now be conducted. Given an input query point, the goal is to find the point contained within the kd tree structure that has the closest distance to the query point. At node A, the x coordinate is compared to the x-plane splitting value. Assume that the query point x coordinate is less than the x-plane splitting value. The left branch from node A would be descended to reach node B. Node B stores a y-plane splitting value, thus the y value of the query point would be compared to this value. The left branch is always descended if the query point coordinate is less than the respective splitting value with the right branch descended for values greater than the splitting plane value. This process continues until a bin is reached where a quick check of all points contained within the bin can be made to determine the closest point. It is possible that the true closest point is contained in another bin. If the distance between the query point and the nearest point found is greater than the distance between the query point and a node splitting plane, then that node branch needs to be checked as well. The example outlined here was for a 2D case, but the same concept applies for any k number of dimensions. Each line of the voxel list generated as outlined in Section 2.5.1 has the *xyz* coordinates of the centre of the voxel as well as the triangle numbers (see Figure 2.7) of any mesh elements that occupy or intersect that region of space. The kd tree subdivision method outlined in this section is applied to this voxel list.

Initially a kd tree constructor and traversal code was written, however the *ANN: A Library for Approximate Nearest Neighbour Searching* [Mount & Ayra, 2005] code was used. This kd tree code returns the coordinates of the point closest to the query, the index of the closest point as well as the squared Euclidian distance. The return of the index allows access to other data associated with the closest point.

2.5.3 Voxel Traversal

Once the satellite model has been subdivided into voxels, a method of predicting which voxel a given ray passes through is needed. The technique used in this thesis is similar to the work of [Amanatides & Woo, 1987].

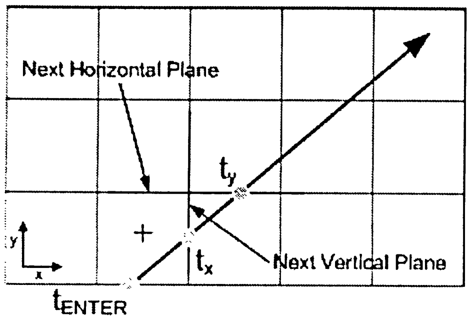


Figure 2.9 2D Voxel Traversal

The method presented here is for the two dimensional case, but has a straightforward extension into three dimensions. The traversal method is initialized by using the entry point of the bounding box, see Figure 2.3, to identify the voxel that the ray first enters. The coordinates of this voxel are designated $(voxX,voxY)$ and the entry parametric value of the ray is stored as the current value $t_{Current} = t_{Enter}$. Given a ray of the form $\hat{u} + t\hat{v} = 0$, and half of the voxel size designated by h, the voxel coordinates will be incremented according to $\Delta voxX = 2h \cdot sign(v_x)$ and $\Delta voxY = 2h \cdot sign(v_y)$. The sign of $\hat{v}$ allows the determination of the direction that the ray will travel and how the voxel will be incremented. The next horizontal and vertical planes that the ray will encounter (see Figure 2.9) are designated as pX and pY with $\Delta pX = h \cdot sign(v_x)$ and $\Delta pY = h \cdot sign(v_y)$. The traversal stage begins by calculating:

$$t_x = \frac{pX - u_x}{v_x} \qquad t_y = \frac{pY - u_y}{v_y} \tag{2.21}$$

The next voxel that the ray passes through is determined by $min(t_x,t_y)$. The basic voxel traversal loop is shown in Figure 2.10. The loop continues until an intersection is encountered,

```
while( nohit & tcurrent<texit)
        tx = (pX-Ux)/Vx
        ty = (pY-Uy)/Vy

        if( tx < ty)
                voxX = voxX + vDeltaX
                pX = voxX + pDeltaX
                tcurrent = tx
        else
                voxY = voxY + vDeltaY
                pY = voxY + pDeltaY
                tcurrent = ty
        endif

        checkVoxel( voxX, voxY )

endloop
```

Figure 2.10 Basic 2D Voxel Traversal Loop

or the ray leaves the bounding box without hitting the satellite. When a voxel is entered, it needs to be established if a part of the satellite geometry lies within this voxel region of space. This is done by using the centre point of the current voxel as a query point to search the kd tree generated as outlined in Section 2.5.2. The size of the voxels is known and the kd tree returns a closest point as well as the distance from the query point to this closest point. If this distance is less than the width of a voxel, the closest point actually equals the query point. This indicates that the current voxel contains a portion of the satellite mesh. The Held algorithm (see Section 2.4.4) is then used to test for ray/triangle intersections with all mesh elements contained with the current voxel to find the true intersection point.

There is one further condition which must be imposed in order to determine the true intersection point. This situation is illustrated in Figure 2.11. When the ray enters voxel B, the existence of triangle 2 will be detected and once the intersection between the ray and the triangle is determined, a false intersection point will be found to be in voxel D. To avoid this situation, any intersection point must reside in the current voxel. Voxels in regions of complex geometry will contain multiple triangle mesh elements. The ray is tested for intersection with all triangles that exist in the voxel and the ray paremetric value t (see Equation (Chapter 2.3)) is used to determine the absolute closest point. The following two conditions need to be satisfied for a point to be considered the true intersection point:

- the lowest parametric value t of all intersections within the voxel
- the intersection point must lie within the current voxel space

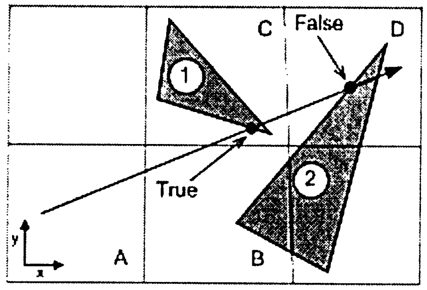


Figure 2.11 Point in Voxel Condition

2.5.4 Voxel Size

There is a tradeoff between the computational cost of voxel traversal and ray/triangle intersection tests. As the voxel size is increased, each voxel element contains more triangle mesh elements within its region. Also, as the voxel size increases, the number of voxel traversals decrease while the number of ray/triangle intersections increase. Figure 2.12 illustrates the relationship between voxel size and raytracing time. As the voxel size is increased the computational cost of voxel traversal is reduced. However, the number of triangles per voxel is increasing. Once the curve flattens, there is no need to keep increasing the voxel size since this will cause more ray-triangle intersection tests and increase computational time.

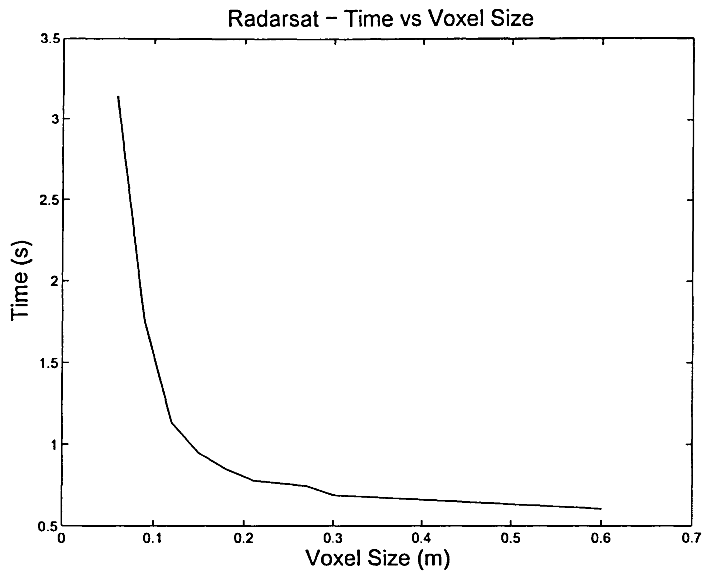


Figure 2.12 Raytracing Time vs Voxel Size

2.5.5 Performance

As expected, the optimized voxel raytracer outperforms the brute force raytracing method as shown in Figure 2.13. The voxel traversal occurs very quickly with a smaller number of computations. This slight addition in computation cost yields a large return in the overall computational time since fewer intersection tests are need to be conducted to determine the ray intersection point. The purpose of developing a high speed raytracing technique is explained in Chapter 6

2.6 Summary

This chapter outlined the techniques used in order to generate synthetic range data of satellite models for use with the ICP algorithm. Five different ray/triangle intersection methods were tested to determine their computational speed. From this test, the Held algorithm was deemed the fastest. The brute force method of raytracing was originally used and was satisfactory in producing a few thousand points in a few seconds. However, as work progressed, the need for a faster method of raytracing developed, hence the development of the voxel raytracer. The

Held algorithm is used in conjunction with the voxel raytracer because triangle intersection tests still occur, but at a significantly reduced rate. Figure 2.14 demonstrates the cloud point generation ability developed for the *Radarsat, Stardust, Deep Space 1*, and *Space Shuttle* models..

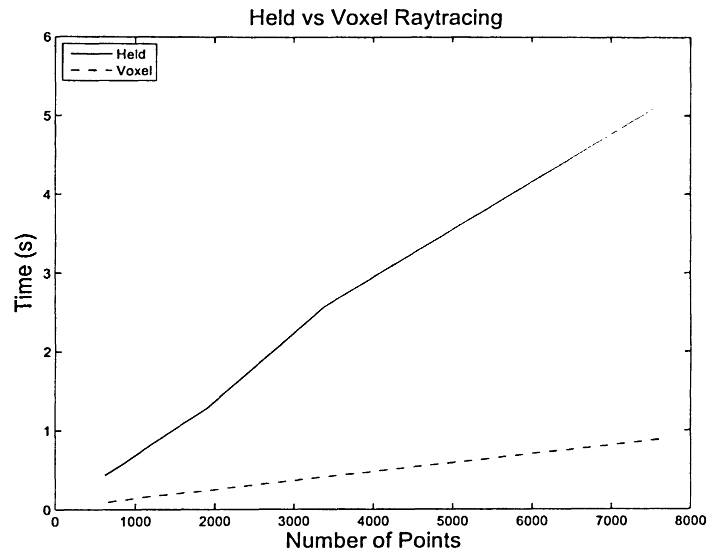


Figure 2.13 Held vs Voxel Raytracing Comparison

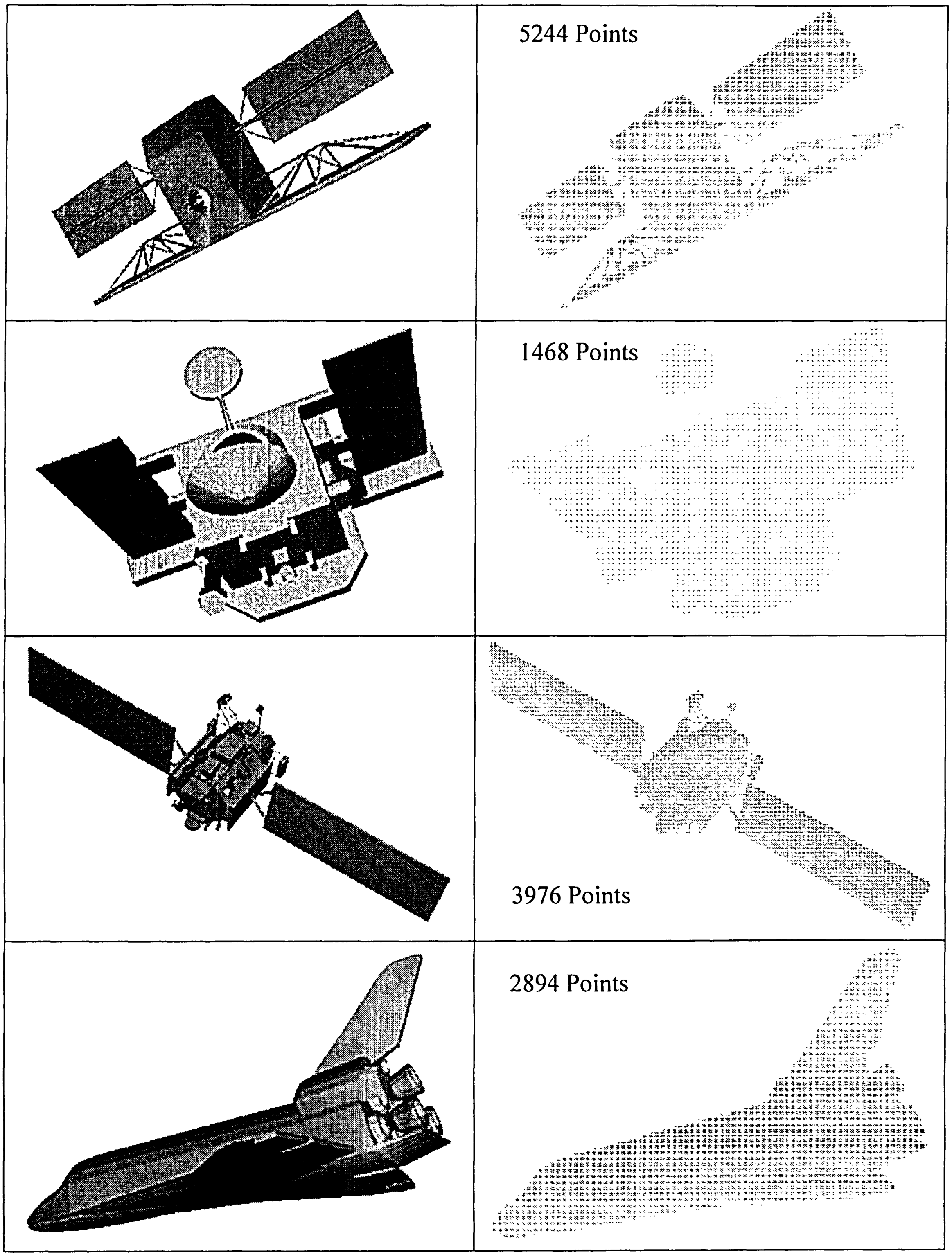


Figure 2.14 Satellite Models With Their Respective Point Clouds

Chapter 3

LIDAR MODEL

3.1 Introduction

The previous chapter outlined the method used to generate 3D cloud point data for a given satellite model. For this data to have value, it needs to be corrupted in a manner that replicates the operational performance of an actual LIDAR system. The instrument used for the noise model baseline is the Rendezvous Laser Vision (RELAVIS) LIDAR system developed by Optech. The goal was not to simulate the exact performance of this instrument, but rather to generate point cloud data that is comparable to the accuracy of RELAVIS.

3.2 Model Equations

Figure 3.1 shows the optical geometry of a laser scanner in the triangulation mode of operation. The light ray produced by the laser is deflected by a mirror and scanned on an object. With some simple trigonometry, the coordinates of the illuminated point can be calculated. The range scanner used in this thesis is based on the work of [Blais et al, 2000].

The range R can be calculated using the triangulation method:

$$R_{Trian} = \frac{f_o \cdot d}{p}\cos\theta + d\sin\theta \qquad (3.1)$$

where p is the image of the target point on the position detector, θ is the deflection angle of the laser beam, d is the triangulation base and f_o is the effective distance between the position detector and the lens. f_o is related to the focal length of the lens.

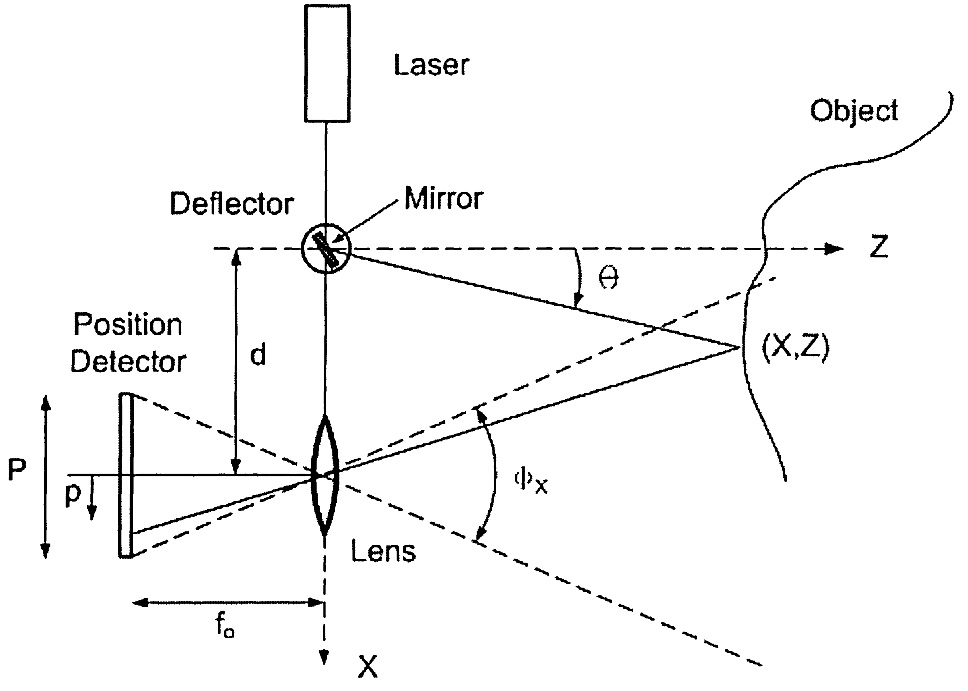


Figure 3.1 Triangulation Geometry from [Beraldin et al, 1993]

For a LIDAR using a time of flight method, the range is a function of the speed of light c and the delay τ of a laser pulse:

$$R_{TOF} = c \cdot \frac{\tau}{2} \tag{3.2}$$

Figure 3.2A shows the astigmatism effect between the X & Y scanning axis. These rotation axes are actually the rotation axes of the galvonometers that steer the laser beam. Due to the spacing among the internal LIDAR components, these axes do not coincide with the position detector array. These component offsets are small as compared to the measured bearing and range to a given target. Therefore, a simplified geometric model is used as shown in Figure 3.2B. The xyz coordinates of a point are:

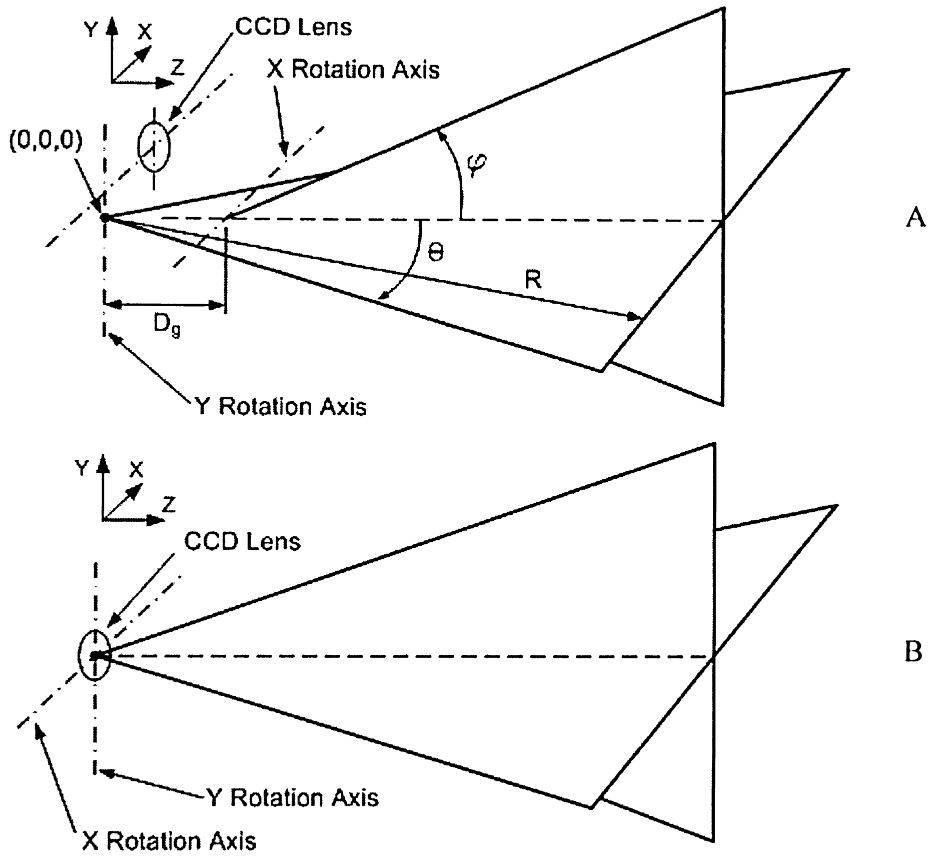


Figure 3.2 Astigmatism Between X & Y Axes and Simplified Geometrical Model **Fig3.2**: from [Blais et al, 2000].

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = R \cdot \begin{bmatrix} \sin\theta \\ (\cos\theta - \psi)\sin\phi \\ (1 - \cos\phi)\psi + \cos\theta\cos\phi \end{bmatrix} \tag{3.3}$$

where θ and φ are the deflection angles and $\psi = D_g/R$ were D_g is the separation between the two scanning axes. Error propagation can be approximated by:

$$\Delta R_{Trian} = \frac{R^2}{f_o \cdot d}\Delta p \tag{3.4}$$

since $D_g \ll R$. Δp is the uncertainty associated with the laser spot measurement and contributes the most to the overall LIDAR error. The overall system distortions and astigmatism caused by D_g are not negligible and induce error in the measurements for pose estimation. However Equation 3.4 is still a good approximation for random noise analysis. The total error of the laser scanner becomes :

$$\begin{bmatrix}\Delta x\\ \Delta y\\ \Delta z\end{bmatrix} = \begin{bmatrix}\sin\theta\\ \cos\theta\cdot\sin\phi\\ \cos\theta\cdot\cos\phi\end{bmatrix}\cdot\Delta R + \begin{bmatrix}\cos\theta\\ -\sin\theta\cdot\sin\phi\\ -\sin\theta\cdot\cos\phi\end{bmatrix}\cdot R\cdot\Delta\theta + \begin{bmatrix}0\\ \cos\phi\\ -\sin\phi\end{bmatrix}\cdot R\cos\theta\cdot\Delta\phi \tag{3.5}$$

Thus given the error in (R, θ, ϕ), the error in the (x, y, z) coordinates can be determined.

3.3 Baseline Range Scanner

The RELAVIS scanner, a time of flight LIDAR instrument, has been developed for the explicit purpose of obtaining the relative position and orientation of a target space vehicle. The specifications for this instrument is shown in Table 3.1. Of most interest are the last two items which show the measurement accuracy of RELAVIS measurements. Traditional lidar scanners have a range accuracy which is proportional to the range magnitude. However RELAVIS is unique with regards to its range accuracy being independant of the range itself.

TABLE 3.1 Lidar RELAVIS Instrument Specifications[a]

Minimum Range	m	0.5
Maximum Range	km	5
Mass	kg	<15
Volume	litres	<9
FOV	+/- deg	10
Power	W	<125
Update Rate	Hz	5
Attitude Accuracy	deg	1
Range Accuracy	cm	1
Bearing Accuracy	mrad	0.35

a. Optech Incorporated http://www.optech.on.ca/prodspace.htm

The errors were simulated as Gaussian noise with zero mean and standard deviations of the bearing and range accuracy of Table 3.1. These values were used in conjunction with Equation 3.5 to generate the *xyz* coordinate perturbations. This method was also used by [Pelletier et al, 2004].

Chapter 4

RANGE IMAGE REGISTRATION

4.1 Introduction

The Iterative Closest Point (ICP) algorithm is a widely used method for the determination of the rigid body transformation between two:

- point sets
- line segments
- implicit curves
- parametric curves
- triangle sets (meshes)
- implicit surfaces
- parametric surfaces

This method iteratively refines a transformation estimate by generating a set of matching points from the models and using these to minimize a given cost function. It is a simple method, but works very effectively when given a good initial estimate. Since this method deals with point clouds, it is the natural choice of processing method for use in conjunction with laser range scanners which return data in the form of point clouds. Also, the use of speed enhancements such as kd trees, closest point caching and iteration acceleration can significantly enhance the speed of the basic algorithm for use in a real time tracking scenario.

The advantages of the ICP algorithm are:

- handles full 6 DOF
- independent of surface representation
- does not require any derivative estimation or feature extraction

The main disadvantage is the requirement of a good initial guess to bring the point clouds into close relative alignment.

There are two distinct problems that can be addressed by using ICP. The scene-scene case arises when multiple scans of the same objects need to aligned in order to generate a full 3D model of the object. The scene-model case arises when an exact model is available and and a scanned scene point cloud needs to be registered. The second problem type is addressed in this thesis.

4.2 Overview

The ICP algorithm was developed by [Besl & McKay, 1992] and is based on minimizing the distance between point pairs. The independent work of [Chen & Medioni, 1992] (see Appendix B) involves a similar iterative technique but use a pairing procedure based on surface normals. Since its development, many ICP variants have been proposed with differing techniques for various stages of the algorithm.

There are several strategies available for the selection of points to form the respective point clouds. [Besl & McKay, 1992] proposed a straightforward method of making use of all available points. However, [Turk, 1994] used a uniform subsampling of the available points while [Masuda et al, 1994] choose a random sampling with a different sampling of points for each iteration. [Rusinkiewicz & Levoy, 2001] tested these three methods and showed only a marginal improvement in performance using normal-space sampling.

Researchers have also turned their attention to the point correspondence problem as this is the most expensive stage of the ICP algorithm. [Besl & McKay, 1992] preferred the closest point to the other geometric entity as a matching criteria. The alternative method of [Chen & Medioni, 1992] is based on the intersection of the ray from a given point in the direction of the point's normal with the other mesh surface. Another interesting method involves using the focal point of the range scanner used to produce a point cloud. [Blais & Levine, 1995] project a ray from the source point through this focal point to find the intersection with the other mesh surface. The original closest point matching is by far the most widely used method and is used in this thesis.

Other researchers have focussed on improving the minimization efficiency of ICP by using alternate transformation representations. [Arun et al, 1987] developed a technique using the singular value decomposition of a covariance matrix to obtain the required transformation. A method that makes use of the orthonormal properties of the rotation matrix is presented by [Horn, 1988]. The basic ICP method uses a unit quaternion representation, but [Walker et al, 1991] expanded this idea and made use of a dual quaternion representation. The choice of transformation method used is outlined in Section 4.3.

Alternative minimization techniques can be applied to the ICP framework since it is an optimization problem. [Mitra et al, 2004] used an approximation to the squared distance function to minimize a cost function using a pure geometric approach. Genetic algorithms for minimization was the choice of [Brunnstrom & Stoddart, 1996]. A hybrid ICP method using simulated annealing was proposed by [Luck et al, 2002]. These alternative formulations are more computationally expensive and are not appropriate for real-time usage.

Most of the above modifications attempt to improve upon robustness, convergence and precision. For this thesis the basic ICP method was used with some minor modifications.

4.3 ICP Background

The ICP algorithm begins with an input point cloud data set $\{\vec{p_i}\}$. For each point in the set $i = 1 \ldots n$, the corresponding closest point is found on the base model producing a second point cloud set $\{\vec{m_i}\}$. These point clouds are related by a by a 3x3 rotation matrix $\boldsymbol{R}$ and a 3D translation vector $\vec{T}$ such that:

$$\vec{m_i} = \boldsymbol{R} \cdot \vec{p_i} + \vec{T} \tag{4.1}$$

The optimal transformation $[\boldsymbol{R}', \vec{T'}]$ is determined by minimizing the mean square error of alignment given by:

$$MSE = \frac{1}{n}\sum_{i}^{n} \left\| \vec{m_i} - \boldsymbol{R}' \cdot \vec{p_i} - \vec{T'} \right\|^2 \tag{4.2}$$

This cost function can be further simplified by introducing the centroids of the point sets:

$$\bar{m} = \frac{1}{n}\sum_{1}^{n} \vec{m_i} \qquad \bar{p} = \frac{1}{n}\sum_{1}^{n} \vec{p_i} \tag{4.3}$$

and using the transformed coordinates:

$$\vec{m_i'} = \vec{m_i} - \bar{m} \qquad \vec{p_i'} = \vec{p_i} - \bar{p} \tag{4.4}$$

reducing to:

$$\begin{aligned} MSE &= \frac{1}{n}\sum_{i}^{n} \left\| \vec{m'_i} - \boldsymbol{R}' \cdot \vec{p'_i} \right\|^2 \\ &= \frac{1}{n}\sum^{n} (\vec{m'_i}^T \vec{m'_i} + \vec{p'_i}^T \vec{p'_i} - 2\vec{m'_i}^T R \vec{p'_i}) \end{aligned} \tag{4.5}$$

The use of the point cloud centroids decouples the rotation and translation components of the transformation. The cost function is minimized when the last term in Equation (4.5) is maximized. This is equivalent to maximizing $Trace(\boldsymbol{R}'\boldsymbol{M})$ [Arun et al, 1987] where:

$$M = \sum_{1}^{n} \overrightarrow{p'}_i \overrightarrow{m'}_i^T = \begin{bmatrix} S_{xx} & S_{xy} & S_{xz} \\ S_{yx} & S_{yy} & S_{yz} \\ S_{xz} & S_{zy} & S_{zz} \end{bmatrix} \tag{4.6}$$

with:

$$S_{xx} = \sum_{1}^{n} x'_{p,i} x'_{m,i} \qquad S_{xy} = \sum_{1}^{n} x'_{p,i} y'_{m,i} \qquad \dots \tag{4.7}$$

In the works of [Besl & McKay, 1992] and [Horn, 1988], a 4x4 covariance matrix was used to obtain the unit quaternion representing the optimal transformation. However, [Arun et al, 1987] used a singular value decomposition technique to obtain the desired rotation. The justification for using the SVD method of [Arun et al, 1987] is derived from the work of [Eggert et al, 1997] and summarized in Table 4.1.

TABLE 4.1 Qualitative comparison of algorithm performance (1=best, 4=worst). N-Noise, ID-No Noise, IN-Isotropic Noise, AN-Anisotropic Noise.

Method	**Accuracy**		**2D Stability**			**1D Stability**			**0D Stability**			**Time**	
	ID	**N**	**ID**	**IN**	**AN**	**ID**	**IN**	**AN**	**ID**	**IN**	**AN**	**small N**	**large N**
SVD	1	1	1	1	1	2	2	2	3	1	1	2	2
Orthonormal Matrix	3	1	4	4	4	1	1	1	1	1	1	1	4
Unit Quaternions	2	1	2	1	1	3	3	3	1	1	1	2	3
Dual Quaternions	4	1	3	1	1	4	4	4	4	4	4	4	1

The SVD technique was selected since it ranks high in all categories and is the simplest method to implement.

The singular value decomposition of M is given by $M = U \Lambda V^T$. Thus the rotation matrix that maximizes $Trace(R'M)$ is given by $R' = VU^T$. The translation vector can be determined using the rotation matrix and the point cloud centroids:

$$\vec{T'} = \bar{m} - R' \cdot \bar{p} \tag{4.8}$$

4.3.1 Closest Point Matching

The ICP algorithm attempts to determine the transformation that will align two point clouds. However, the satellite models used are represented by triangular meshes. Therefore, a method was needed that would extract a set of $i = 1 \ldots n$ points $\{\vec{m_i}\}$ from the model such that the distance to their corresponding points $\{\vec{p_i}\}$ in the input point cloud is minimal. The closest point matching concept developed here stemmed from the use of the voxel raytracing method for producing the input point clouds (see Section 2.5). A kd tree search method along with a

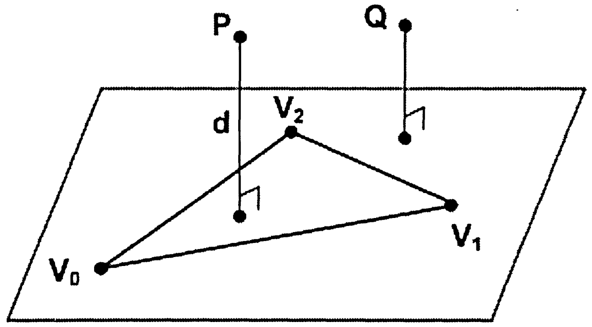


Figure 4.1 Normal Point to Plane Projection

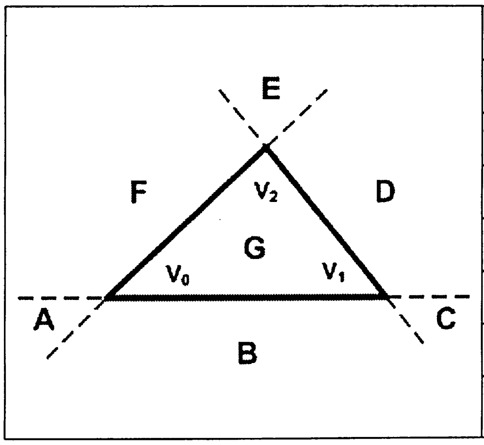
voxelized model was used in order to quickly reach the voxel coordinate that is closest to the query point. The satellite model is voxelized with a reasonably small voxel size. This size cannot be made too small since every time the voxel size is reduced by half, the number of voxels in the model increase by a factor of 8. With a finely discretized voxel space, the voxel coordinate will be a close approximation to the actual closest point. However, the actual closest point lies on the portion of the model surface contained within this closest voxel. Consider a point $\vec{P}$ and a triangle mesh element with vertices $\vec{V_0}$, $\vec{V_1}$, $\vec{V_2}$ with a normal $\hat{n}$. The normal distance to the plane of the triangle is:

$$d = \hat{n} \cdot (\vec{P} - \vec{V_0}) \tag{4.9}$$

which can be used to find the surface point $\vec{p}_s$

$$\vec{p}_s = \vec{p} - d\vec{n} \tag{4.10}$$

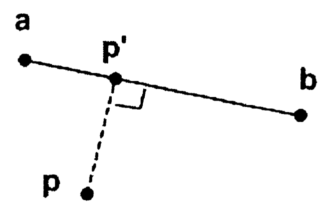
If the projected point lies within the boundaries of the triangle, then this will be the closest point. However, if the projected point lies outside the triangle boundaries (see Figure 4.1), the closest point to the triangle will be either one of the vertices or the projection onto the closest line segment. To determine where the projected point lands, the triangle parametric coordinates (λ,μ) are determined by using the algorithm of [Badouel, 1990]. The parametric



Region	$\lambda < 0$	$\mu < 0$	$\lambda + \mu > 1$	Closest Point
A	Yes	Yes	No	V_0
B	No	Yes	No	Project on V_0V_1
C	No	Yes	Yes	V_0
D	No	No	Yes	Project on V_1V_2
E	Yes	No	Yes	V_0
F	Yes	No	No	Project on V_0V_2
G	No	No	No	p_s

Figure 4.2 Triangle Parametric Regions

coordinates can be used along with Figure 4.2 to determine where the closest point lies [Ericson, 2004]. Figure 4.3 shows the method of projecting a point $\vec{p}$ onto a line segment $\vec{ab}$.For each point $\{\vec{p}_i\}$ in the scanned point cloud, the closest point is found by first using a kd tree method to locate the closest voxel coordinate. The closest point is further refined by searching through all of the triangular mesh elements contained within this voxel in order to determine the actual closest point. This process is repeated for all points $\{\vec{p}_i\}$ to construct the closest point set $\{\vec{m}_i\}$.



$$\alpha = \frac{(\vec{p}-\vec{a})\cdot(\vec{b}-\vec{a})}{(\vec{b}-\vec{a})\cdot(\vec{b}-\vec{a})}$$

$$\alpha' = max(0,min(\alpha,1))$$

$$\vec{p'} = \vec{p}+\alpha'(\vec{b}-\vec{a})$$

Figure 4.3 Project Point $\vec{p}$ onto line segment $\overrightarrow{ab}$

The steps for finding the closest point are as follows:

1. take one point from the point cloud $\{\vec{p_i}\}$ and designate this point as the query point
2. use the kd tree structure to find the closest voxel from the voxelized satellite model list
3. find the closest point to each triangle element contained within the voxel by first projecting the query point onto the plane formed by a given triangle (see Figure 4.1) this gives you the point projecton $\vec{p_s}$
4. use $\vec{p_s}$ along with the Held algorithm (see Section 2.4.4) to calculate the barycentric coordinates (λ,μ) and use these to determine where the candidate closest point lies by using Figure 4.2
5. repeat steps 3 and 4 for all triangles contained in the closest voxel to produce a set of closest candidate points
6. the point from the candidate points that has the smallest distance to the query point is designated as the closest point and is placed in the corresponding point set $\{\vec{m_i}\}$
7. steps 1-6 are repeated for all points in point cloud $\{\vec{p_i}\}$ to generate the corresponding point cloud $\{\vec{m_i}\}$

Admittedly, this point matching scheme is not a very fast technique and would not be appropriate for use in a real-time situation. However, this method was used in this thesis due to its accuracy of returning the closest point on the satellite mesh surface.

4.3.2 Registration Error

The goal of registration is to determine the transformation between the camera frame and the target frame T_{CT}. However, the transformation determined by ICP registration T_{ICP} will have an associated transformation error T_E due to noisy input point cloud data (see Figure 4.4).

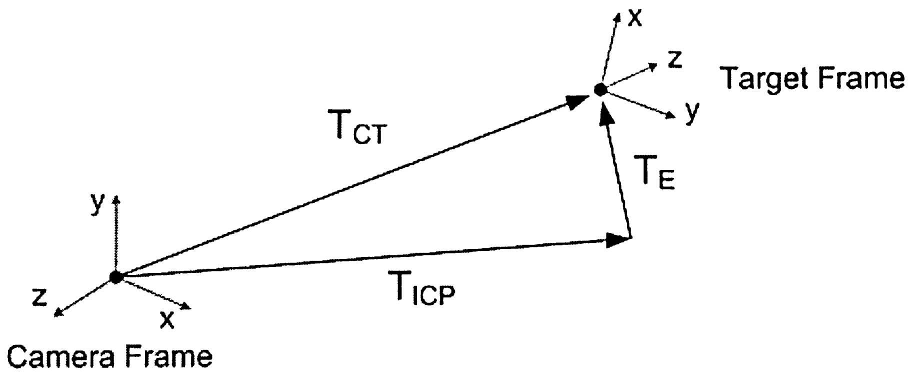


Figure 4.4 Registration Error Transformation

The registration transformation error can be expressed as:

$$T_E = (T_{ICP})^{-1} \cdot T_{CT} \tag{4.11}$$

The rotational registration error is expected to be small allowing the approximations $\cos\theta \approx 1$ and $\sin\theta \approx \theta$ to further simplify and express T_E:

$$T_E = \begin{bmatrix} 1 & -\omega_z & \omega_y & t_x \\ \omega_z & 1 & -\omega_x & t_y \\ -\omega_y & \omega_x & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.12}$$

Where T_E is composed of the rotational and translational pose parameters $\begin{bmatrix} \omega_x & \omega_y & \omega_z & t_x & t_y & t_z \end{bmatrix}$. The error in rotation and translation can be expressed as the norms of the rotation and translation components of Equation (4.12):

$$\|E_t\| = \sqrt{t_x^2 + t_y^2 + t_z^2} \tag{4.13}$$

and:

$$\|E_\omega\| = \sqrt{\omega_x^2 + \omega_y^2 + \omega_z^2} \tag{4.14}$$

4.3.3 Initial Pose Estimate

ICP requires an initial pose estimate that puts the input point cloud in close alignment with the base model. The transformation returned along with the clout point data is the transformation from the camera frame to the target frame. First, the point cloud is transformed using the inverse of the actual transformation $(T)^{-1}$. This puts the point cloud in exact alignment with the model. A second transformation T_{Shift} is applied in order to put the point cloud in a close alignment state. The transformation T_{Shift} is generated by using the axis-angle rotation representation and maximum translation and rotation angle values of τ_{Max} and θ_{Max} respectively. A uniformly distributed random number generator $ran(a, b)$ is used to generate values in the range $[a, b]$. Thus the rotation and translation components are determined as follows:

$$\begin{aligned} t_x &= ran(-\tau_{Max}, \tau_{Max}) \\ t_y &= ran(-\tau_{Max}, \tau_{Max}) \\ t_z &= ran(-\tau_{Max}, \tau_{Max}) \\ \theta &= ran(-\theta_{Max}, \theta_{Max}) \end{aligned} \tag{4.15}$$

The axis-angle rotation representation is used along with the value θ and the rotation axis derived from:

$$\begin{aligned} m_x &= ran(-1,1) \\ m_y &= ran(-1,1) \\ m_z &= ran(-1,1) \end{aligned} \tag{4.16}$$

Normalizing the components of $\vec{m}$ by $\|\vec{m}\|$ gives the unit vector $\hat{n} = \begin{bmatrix} n_x & n_y & n_z \end{bmatrix}^T$ that defines the axis of rotation. The transformation T_{Shift} is now defined as:

$$T = \begin{bmatrix} 1+(1-c\theta)(n^2{}_x-1) & -n_z s\theta + n_x n_y(1-c\theta) & n_y s\theta + n_x n_z(1-c\theta) & t_x \\ n_z s\theta + n_x n_y(1-c\theta) & 1+(1-c\theta)(n^2{}_y-1) & -n_x s\theta + n_y n_z(1-c\theta) & t_y \\ -n_y s\theta + n_x n_z(1-c\theta) & n_x s\theta + n_y n_z(1-c\theta) & 1+(1-c\theta)(n^2{}_z-1) & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.17}$$

With this, the initial pose for the ICP algorithm becomes:

$$T_{Init} = T_{Shift} \cdot (T)^{-1} \tag{4.18}$$

Unless otherwise specified, the value of θ_{Max} was set to 10 degrees while τ_{Max} was set to 10% of the translation magnitude. More care was placed on the initial rotation state since ICP is very sensitive to the initial rotation state, while insensitive to the initial translation state [Besl & McKay, 1992].

4.3.4 The ICP Algorithm

With a given point cloud $\{\vec{p}_i\}$ and a good initial guess T_{Init}, the ICP algorithm can be stated as follows:

- Compute the closest points to $\{\vec{p}_i\}$ to generate a model point cloud $\{\vec{m}_i\}$
- Generate the covariance matrix M
- Use singular value decomposition to M to compute the rotation and translation
- Apply the rotation and translation to the point cloud $\{\vec{p}_i\}$
- Record the rotation and translation

- Iterate until convergence or termination criteria is met
- Determine the registration error

The criteria used to measure convergence was the relative change of the cost function *MSE* (see Equation 4.2):

$$\left\| \frac{MSE_i - MSE_{i-1}}{MSE_i} \right\| < 0.0001 \tag{4.19}$$

This high threshold was used to gain further accuracy from the ICP algorithm. An iteration limit of 400 was also imposed for the case where an unstable point cloud input will not yield a converged ICP solution.

4.4 Experimental Method

This section outlines the experimental method used to demonstrate the relationship between constraint analysis (Chapter 5) and ICP registration.

4.4.1 Satellite Subsection Scanning

The method by which subsections of the satellite models are scanned to produce the input cloud point data for the ICP algorithm is represented by Figure 4.6. The cloud point generation begins by positioning the satellite and establishing the translational portion of the transformation. To be sure that all sides of the satellite are viewed in order to examine all geometric features from different viewpoints, 30 different rotation states were used. The x-axis was rotated so as to point in 30 different directions as shown in Figure 4.5.

The parameters that affect that spacing and sizing of the subsection scanning is shown in Figure 4.5. Windows lie on the image plane and are specified in pixel coordinates (see Figure 2.4). The window spacing is the centre to centre distance between each window and also determines how many windows are generated per given satellite pose. This value is specified such that approximately 50 to 75 windows are generated per pose state. The window size determines the amount of satellite model coverage. This sizing is somewhat arbitrary. The size is picked to be able to cover 20% of the availalbe model area. The resolution establishes the number of points that will be generated per window. This parameter is set so that when a window fully covers the satellite model, about 1200 points will be generated. Point clouds with too few points tend to to cause higher ICP error while point clouds with a very high number of points require longer computation time. The value of 1200 cloud points was selected as a compromise between these two extremes. Windows having point cloud counts of less than 100 were rejected. For each satellite model, 4 translation states along with 30 rotation states were combined to generate 120 pose states with each pose state having 50 to 75 point cloud windows.

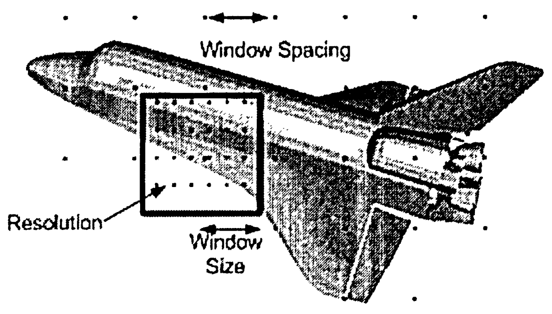
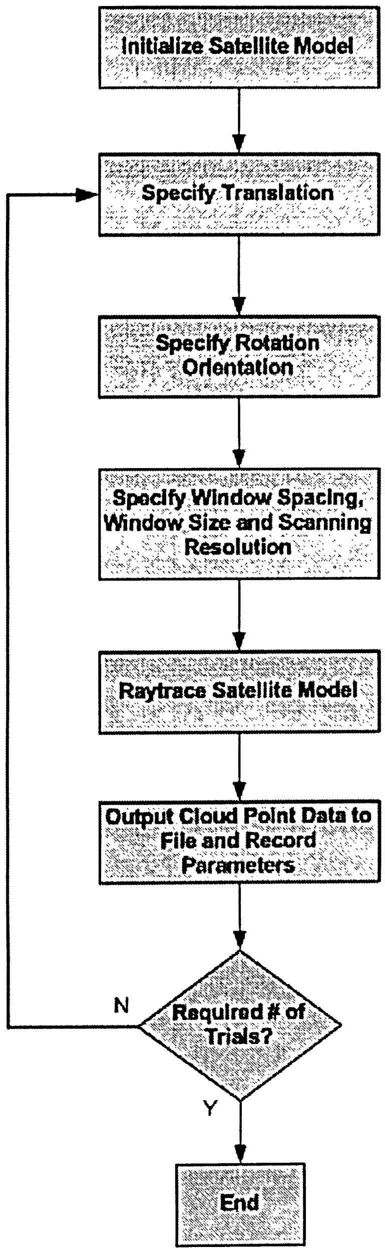


Figure 4.5 Window Scanning Parameters

4.4.2 ICP Registration

The outline of the ICP registration tests are shown in Figure 4.7. Each point cloud data file generated in the previous section was processed using ICP. The noise free data was input and corrupted in the manner as outlined in Chapter 3. An appropriate initial pose estimate was generated and applied to bring the point cloud in close alignment to the satellite model (see Section 4.3.3). The core ICP algorithm ran until convergence, or termination criteria was met. The pose error was determined by using the method described in Section 4.3.2. Each point


Initialize Satellite Model
Specify Translation
Specify Rotation Orientation
Specify Window Spacing, Window Size and Scanning Resolution
Raytrace Satellite Model
Output Cloud Point Data to File and Record Parameters
Required # of Trials?
N
Y
End

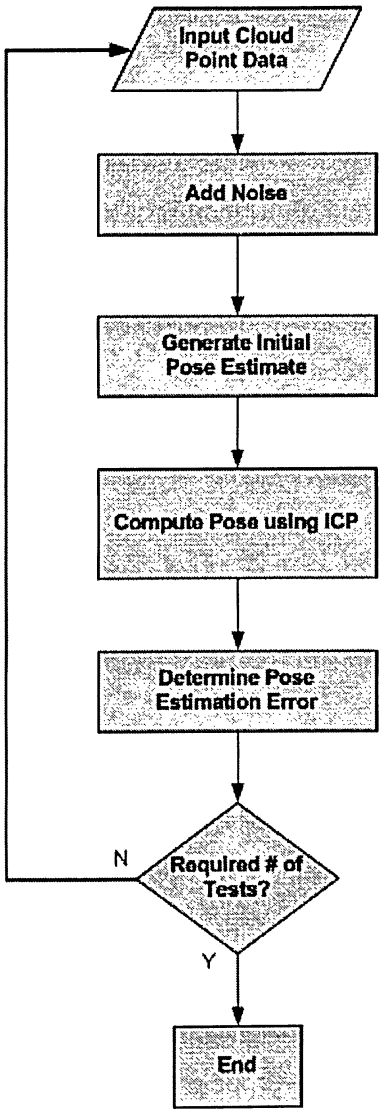


Figure 4.6 Window Scanning Flowchart

Figure 4.7 ICP Registration Flowchart

cloud was processed 25 times using ICP with the results averaged.

4.5 Summary

ICP is a non-linear local search algorithm that will find the nearest local minimum of a mean-squared distance metric. A good initial estimate helps to avoid these local minima and arrive at the global minimum. Other factors affecting ICP convergence include the magnitude of noise in the data and the geometric stability of input cloud point data. Tdoes not require any derivative estimation or feature extraction

ICP was used in its basic form with the small modification of using the SVD method of [Arun et al, 1987] to determine the rotation transformation at each iteration stage. An accurate method of closest point matching was developed which utilized the triangular mesh structure of the satellite models. The rotation and translation norms of the error transformation matrix were used to measure the accuracy of the ICP registration results.

Chapter 5

CONSTRAINT ANALYSIS

5.1 Introduction

The constraint analysis method outlined in this chapter is based on the work of [Simon, 1996]. A sensitivity measure called the Noise Amplification Index (NAI) will be introduced which will be shown to be a good predictor of registration accuracy. There are many different engineering concepts that share the underlying mathematical concepts of constraint analysis. More specifically, there is a similarity between robot dexterity measures of sensitivity and registration sensitivity. The robotic Jacobian relates the rate of change of the end effector position to the rate of change of the joint variables. Similarly, a matrix will be derived that relates the rate of change of a squared distance to the rate of change of the pose estimation variables. Therefore, the scalar measures of sensitivity as used for robotics by [Nahvi & Hollerbach, 1996] can be adapted for use in the context of registration accuracy.

There is another method of estimating pose uncertainty based on the work of [Stoddart et al, 1996]. However, the indicator that they used is embedded within the ICP algorithm and is only available once registration has taken place. This method cannot be decoupled from the ICP algorithm to provide a predictive method of registration accuracy. Hence, it was not used. [Gelfand et al, 2003] formulated a stability measure similar to the constraint analysis method used in this thesis (see Appendix B). However, they used this measure only to determine the contribution of individual points to the overall solution stability.

5.2 Sensitivity Vector

In order to derive the sensitivity vector that will be used for constraint analysis, the orthogonal distance from a given point to a given model surface is required. If the implicit form of the surface is available, the simplest method of obtaining this distance is to substitute the coordinates of the point into the surface equation $f(x, y, z) = 0$ giving the algebraic distance $D = f(x, y, z)$. This approach can work for simple surfaces, but does not return the exact orthogonal distance when d4ealing with higher order surface equations. [Sampson, 1982] showed that normalizing the algebraic distance with respect to the gradient at the surface point better reflects the desired orthogonal distance.

Thus, the distance between a given point $p = (x, y, z)$ and a surface denoted by the implicit equation $f(x, y, z) = 0$ can be approximated as:

$$D(p) = \frac{f(x, y, z)}{\|\nabla f(x, y, z)\|} \tag{5.1}$$

If $f(x, y, z) = 0$ is the exact equation of a plane, then Equation 5.1 becomes the exact equation for the point to plane distance.

Consider a point p_s that lines on the surface such that $D(p_s) = 0$. A differential transformation T applied to the point will cause a perturbation from the surface. The transformation T is a function of the parameters $(t_x, t_y, t_z, \omega_x, \omega_y, \omega_z)$ where $(\omega_x, \omega_y, \omega_z)$ are rotations and (t_x, t_y, t_z) are the translations with respect to the X, Y and Z axes. The six vector of transformation parameters is defined as:

$$t = \begin{bmatrix} t_x \; t_y \; t_z \; \omega_x \; \omega_y \; \omega_z \end{bmatrix}^T \tag{5.2}$$

The corresponding homogeneous transformation matrix is:

$$T = \begin{bmatrix} c\omega_x c\omega_y & -s\omega_x c\omega_z + c\omega_x s\omega_y s\omega_z & s\omega_x s\omega_z + c\omega_x s\omega_y c\omega_z & t_x \\ s\omega_x c\omega_y & c\omega_x c\omega_z + s\omega_x s\omega_y s\omega_z & -c\omega_x s\omega_z + s\omega_x s\omega_y c\omega_z & t_y \\ -s\omega_y & c\omega_y s\omega_z & c\omega_y c\omega_z & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.3}$$

where $c\omega_x = \cos(\omega_x)$, $s\omega_x = \sin(\omega_x)$ and similarily for the y and z components. The variation of the distance D as the point is perturbed is given by the gradient of D about $t = 0$[1]:

$$\frac{\partial}{\partial t}D(T(p_s)) = \left.\begin{bmatrix} \frac{\partial}{\partial t_x} & \frac{\partial}{\partial t_y} & \frac{\partial}{\partial t_z} & \frac{\partial}{\partial \omega_x} & \frac{\partial}{\partial \omega_y} & \frac{\partial}{\partial \omega_z} \end{bmatrix}^T \right|_{t=0} \tag{5.4}$$

Using the quotient rule to take the partial derivatives of:

$$D(T(p_s)) = \frac{f(T(p_s))}{\|\nabla f(T(p_s))\|} \tag{5.5}$$

with respect to each of the parameters of t gives, after simplifying using $f(T(p_s))\big|_{t=0} = 0$:

$$\frac{\partial}{\partial t_i}D(T(p_s)) = \left.\frac{1}{\|\nabla f(T(p_s))\|}\frac{\partial}{\partial t}f(T(p_s))\right|_{t=0} \tag{5.6}$$

Now the chain rule is used to find the partial derivatives with respect to t:

$$\frac{\partial}{\partial t}D(T(p_s)) = \left.\frac{1}{\|\nabla f(T(p_s))\|}\frac{\partial T}{\partial t_i}p_s \cdot \nabla f(T(p_s))\right|_{t=0} \tag{5.7}$$

The unit gradient at a given point on the surface is simply the surface normal at that point:

1. this indicates no rotation or translation thus the local gradient about t=0 represents a small transformation change

$$n_{p_s} = \left.\frac{\nabla f(T(p_s))}{\|\nabla f(T(p_s))\|}\right|_{t=0} \tag{5.8}$$

This simplification allows the final result to be independent of the implicit surface. The rotations involved in the transformation are infinitesimal, thus the expansion of $\left.\frac{\mathrm{d}T}{\mathrm{d}t}\right|_{t=0}$ gives:

$$\frac{\partial T}{\partial t_x} = \begin{bmatrix} 0&0&0&1\\0&0&0&0\\0&0&0&0\\0&0&0&0 \end{bmatrix} \quad \frac{\partial T}{\partial t_y} = \begin{bmatrix} 0&0&0&0\\0&0&0&1\\0&0&0&0\\0&0&0&0 \end{bmatrix} \quad \frac{\partial T}{\partial t_z} = \begin{bmatrix} 0&0&0&0\\0&0&0&0\\0&0&0&1\\0&0&0&0 \end{bmatrix} \tag{5.9}$$

$$\frac{\partial T}{\partial t\omega_x} = \begin{bmatrix} 0&0&0&0\\0&0&-1&0\\0&1&0&0\\0&0&0&0 \end{bmatrix} \quad \frac{\partial T}{\partial t\omega_y} = \begin{bmatrix} 0&0&1&0\\0&0&0&0\\-1&0&0&0\\0&0&0&0 \end{bmatrix} \quad \frac{\partial T}{\partial t\omega_z} = \begin{bmatrix} 0&-1&0&0\\1&0&0&0\\0&0&0&0\\0&0&0&0 \end{bmatrix}$$

The Jacobian of the transformation can now be expressed as:

$$J = \begin{bmatrix} 1&0&0\\0&1&0\\0&0&1\\0&-z&y\\z&0&-x\\-y&x&0 \end{bmatrix} \tag{5.10}$$

Combining Equations (5.7), (5.8) and (5.10) gives:

$$\frac{\partial}{\partial t}D(T(p_s)) = Jn_{p_s} \tag{5.11}$$

This equation can be further simplified and expressed as:

$$V(p_s) = \frac{\partial}{\partial t}D(T(p_s)) = \begin{bmatrix} n_{p_s} \\ p_s \times n_{p_s} \end{bmatrix} \tag{5.12}$$

Equation (5.12) expresses the change in distance of a point on the surface subject to a small change in position. This result does not depend on the implicit surface representation, but rather on the location of the surface point and the corresponding normal at that point. Equation 5.12 can be further simplified by using Equation 5.2 to obtain:

$$dD(T(p_s)) = V^T(p_s)dt \tag{5.13}$$

Equation (5.13) represents the change between a point and a surface under the influence of a small arbitrary transformation. By squaring this equation, the following results:

$$dD^2(T(p_s)) = dt^T V(p_s) V^T(p_s) dt = dt^T M(p_s) dt \tag{5.14}$$

where

$$M(p_s) = V(p_s) V^T(p_s) \tag{5.15}$$

is symmetric, positive semi-definite 6x6 matrix.

Notice the similarity between Equation 5.14 and Equation 4.2. Here lies the connection between ICP and constraint analysis. Equation 5.14 is a local approximation of the ICP cost function. The ICP cost function is summed over all of the point, so similarily Equation 5.15 is summed over the set of surface points P giving:

$$E_P(T(p_s)) = \sum_{p_s \in P} dD^2(T(p_s)) = dt^T \left[\sum_{p_s \in P} M(p_s) \right] dt = dt^T \Psi_P dt \tag{5.16}$$

where E_P is the sum of the squared distance errors between the surface and points P and Ψ_P is the sum of the $M(p_s)$ matrices evaluated for each point. E_P also represents a first order approximation of the ICP registration error. The matrix Ψ_P is a scatter matrix which represents the distribution of $V(p_s)$ over all points P.

Performing principle component analysis on Ψ_P gives:

$$\Psi_P = Q\Lambda Q^T = \begin{bmatrix} q_1 & q_2 & q_3 & q_4 & q_5 & q_6 \end{bmatrix} \begin{bmatrix} \lambda_1 & & & & & \\ & \lambda_2 & & & 0 & \\ & & \lambda_3 & & & \\ & & & \lambda_4 & & \\ & 0 & & & \lambda_5 & \\ & & & & & \lambda_6 \end{bmatrix} \begin{bmatrix} q_1^T \\ q_2^T \\ q_3^T \\ q_4^T \\ q_5^T \\ q_6^T \end{bmatrix} \tag{5.17}$$

where $\lambda_1 > \lambda_2 > \lambda_3 > \lambda_4 > \lambda_5 > \lambda_6$ are the eigenvalues of Ψ_P and q_i are the corresponding unit eigenvectors. Thus Equation (5.16) can be expressed as:

$$E_P(T(p_s)) = \sum_{i=1}^{6} \lambda_i (dt^T q_i)^2 \tag{5.18}$$

The largest eigenvalue λ_1 represents the transformation that will result in the largest possible change in E_P from all possible transformations. This is also the underlying concept used by the ICP algorithm. Finding this maximum transformation is the key step to minimizing the ICP cost function (see Section 4.3).

The smallest eigenvalue λ_6 represents the transformaiton of maximum freedom. This transformation will return the smallest possible change in E_P. It is this eigenvalue that is most problematic for the ICP registration process.

It will be shown that point clouds that have well conditioned Ψ_P matrices tend to result in more accurate ICP registration results.

5.3 Scale Dependence

Constraint analysis attempts to combine rotations and translation into a single representation despite the different units of measure. However, the constraint analysis formulation as described in the previous section is highly dependent upon the scale of the underlying surface geometry. The scale dependency stems from the term $p_s \times n_{p_s}$ contained in Equation (5.12). As shown in Equation 5.12, the displaced distance d induced by a rotation α is dependent upon the distance of the point p_s from the origin, which is the centre of rotation. Translations do not suffer the effects of scale as shown in Figure 5.1.

A normalization is needed that will negate the effects of scale, and equalize the effects of rotation and translation. Consider the situation when $\|d\alpha\| = \|d\tau\|$. The desired result is $D_\alpha(p_s) = D_\tau(p_s)$. This condition will be satisfied only when $\|p_s\| = 1$ which implies that a unit magnitude of translation or rotation will cause the same displacement.

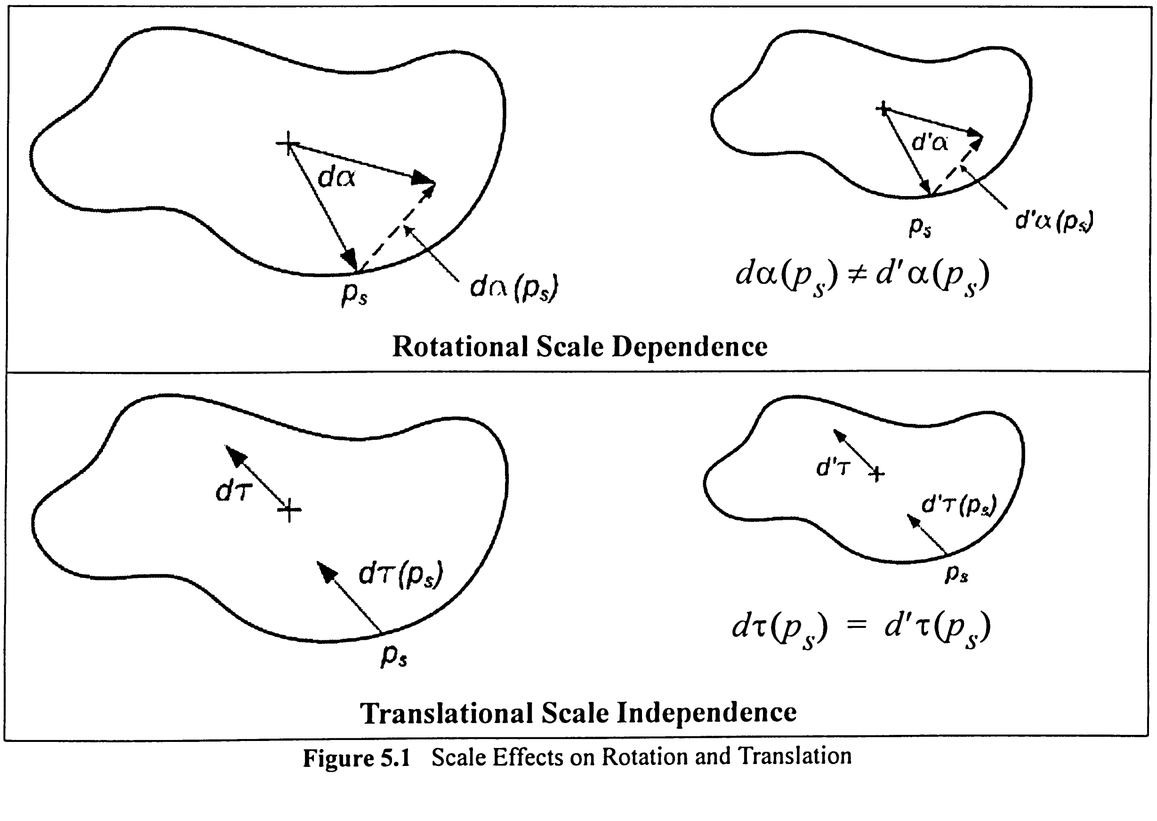

dα
dα (p_s)
p_s
d'α
d'α (p_s)
p_s
$d\alpha(p_s) \neq d'\alpha(p_s)$
Rotational Scale Dependence
dτ
dτ (p_s)
p_s
d'τ
d'τ (p_s)
p_s
$d\tau(p_s) = d'\tau(p_s)$
Translational Scale Independence

Figure 5.1 Scale Effects on Rotation and Translation

The normalization begins by first shifting the centroid of the point cloud to the origin in order to be sure that scaling occurs about the centroid. A scaling factor is applied to all point cloud coordinates such that the average distance between all points and the centroid is one. The normalizing scale factor (NSF) is for n points is:

$$NSF = \frac{n}{\sum^{n} \|p_s\|_i} \tag{5.19}$$

This scaling factor is applied to a given point cloud prior to conducting constraint analysis in order to equalize the contributions of rotation and translation.

5.3.1 Unstable Configurations

In this section the ability of constraint analysis to predict unstable shape configurations is demonstrated. Figure 5.2 and Figure 5.3 show examples of shape configurations that are unstable, except for the stable corner shape in Figure 5.3 which is included for comparison. Any unstable shape will have a Ψ_P matrix which will have small eigenvalues, relative to the largest eigenvalue λ_6, with the corresponding eigenvectors indicating the direction of instability present. This result is shown in Figure 5.2 and Figure 5.3. Where present, small eigenvalues are highlighted with the direction of instability highlighted in the corresponding eigenvector. All of the unstable eigenvalues should actually have values of zero, but a small non-zero value is present due to discretization errors of the triangular mesh. Mesh structures only approximate curved surfaces. Point locations and surface normals will deviate slightly from the true values.

The problem of small eigenvalues can be shown by considering the fundamental problem $Ax = b$. Introducing an error δb into the measured quantities will result in a solution error of δx. The perturbations caused by δb can be determined with $A\delta x = \delta b$. Any vector δb is a combination of the respective eigenvectors $x_1, \ldots, x_n$ and the worst error is in the direction of the eigenvector corresponding to the smallest eigenvalue λ_6. Thus the error magnitude $\|\delta b\|$

is amplified by the factor $1/\lambda_6$ with the amplification being the greatest when λ_6 is close to zero [Strang, 1976]. Thus small eigenvalues are an indication of unstable configurations that will result in amplified pose estimation errors. Appendix B presents a formulation of the covariance matrix such that this arguement using the $Ax = b$ format is justified.

5.4 Measure of Stability

With the Ψ_P matrix now defined for a point cloud, a measurement of stable matrix conditioning needs to be established. Principal component analysis returns the eigenvectors q_i along with the corresponding eigenvalues such that $\lambda_1 > \lambda_2 > \lambda_3 > \lambda_4 > \lambda_5 > \lambda_6$. A six vector p along with the equation:

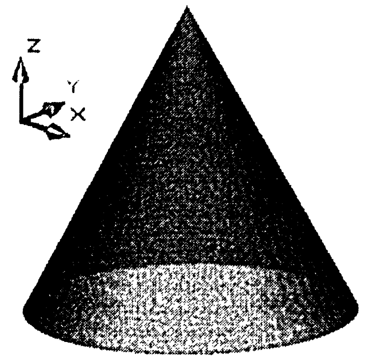
$$p^T \Psi_p p = 1 \tag{5.20}$$

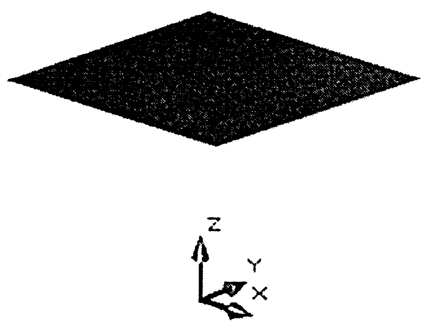
defines a 6D error hyperellipsoid as shown in Figure 5.4 where the principal axes are eigenvectors of Ψ_P lengths are inverse square roots of the respective eigenvalues [Simon, 1996]. The lengths of each axes is a measure of the constraint uncertainty in that direction where a longer axis indicates a greater uncertainty. The two main geometric features that are needed to improve registration accuracy are :

8. The length of the largest hyperellipsoid axis needs to be as small as possible requiring a larger minimum eigenvalue.
9. A smaller hyperellipsoid eccentricy.

The combination of these conditions make the error hyperellipsoid as spherical as possible while minimizing its volume. This indicates that instability increases as the hyperellipsoid major axis length and volume increase.

[Nahvi & Hollerbach, 1996] examined different measures of matrix stability and present a new scalar measure of matrix conditioning called the noise amplification index. Larger values of these measures of matrix conditioning (see Table 5.1) indicate a more stable point cloud configuration leading to reduced registration error.

	λ_i	Eigenvectors					
		Translation			Rotation		
	5.28	0.00	0.00	0.00	0.00	0.00	1.00
	677.81	0.05	0.58	0.00	-0.81	0.07	0.00
	677.81	0.58	-0.05	0.00	0.07	0.81	0.00
	884.43	0.00	0.00	1.00	0.00	0.00	0.00
	2039.80	0.10	0.81	0.00	0.58	-0.07	0.00
Unconstrained: Z Rotation	2039.80	0.81	-0.10	0.00	-0.07	-0.58	0.00

	λ_i	Eigenvectors					
		Translation			Rotation		
	0	1.00	0.00	0.00	0.00	0.00	0.00
	0	0.00	1.00	0.00	0.00	0.00	0.00
	0	0.00	0.00	0.00	0.00	0.00	1.00
	714.64	0.00	0.00	0.00	-0.71	0.71	0.00
	720.20	0.00	0.00	0.00	-0.71	-0.71	0.00
Unconstrained: X & Y Translations, Z Rotation	1536.00	0.00	0.00	1.00	0.00	0.00	0.00

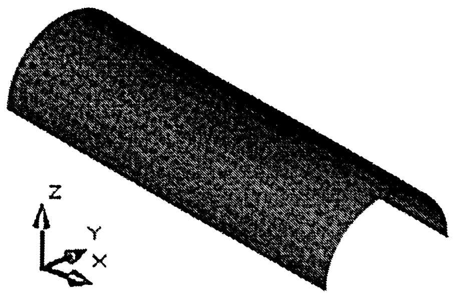
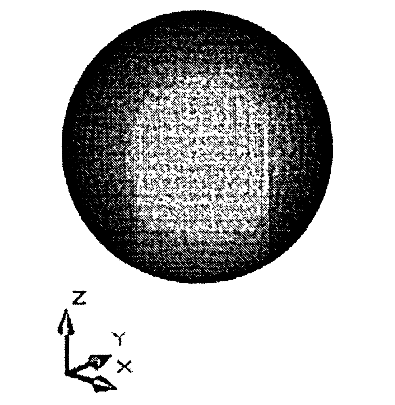
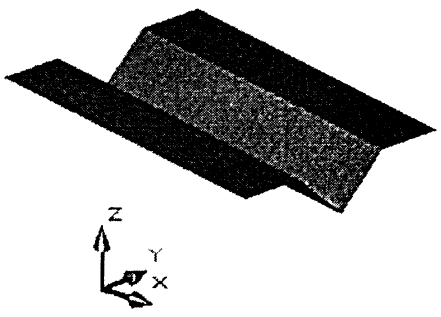
	λ_i	Eigenvectors					
		Translation			Rotation		
	0	1.00	0.00	0.00	0.00	0.00	0.00
	2.48	0.00	0.20	0.00	-0.97	0.00	0.00
	512.2	0.00	0.00	0.00	0.00	0.00	1.00
	512.2	0.00	0.00	0.00	0.00	1.00	0.00
	1620	0.00	0.00	1.00	0.00	0.00	0.00
Unconstrained: X Translation, X Rotation	1688.8	0.00	0.97	0.00	0.20	0.00	0.00

Figure 5.2 Constraint Analysis Shape Configurations 1

	λ_i	Eigenvectors					
		Translation			Rotation		
	0.00	0.00	0.00	0.00	0.00	1.00	0.00
	0.00	0.00	0.00	0.00	1.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	1.00
	4050.1	0.00	1.00	0.00	0.00	0.00	0.00
	4095.0	-1.00	0.00	0.00	0.00	0.00	0.00
Unconstrained: XYZ Rotations	7964.1	0.00	1.00	0.00	0.00	0.00	0.00

	λ_i	Eigenvectors					
		Translation			Rotation		
	0.00	1.00	0.00	0.00	0.00	0.00	0.00
	421.59	0.00	0.00	0.00	0.00	0.00	1.00
	641.81	0.00	-0.12	0.00	-0.99	0.00	0.00
	1281.10	0.00	0.00	0.00	0.00	1.00	0.00
Unconstrained: X Translation	1533.50	0.00	0.99	0.00	0.00	-0.12	0.00
	4622.70	0.00	0.00	1.00	0.00	0.00	0.00

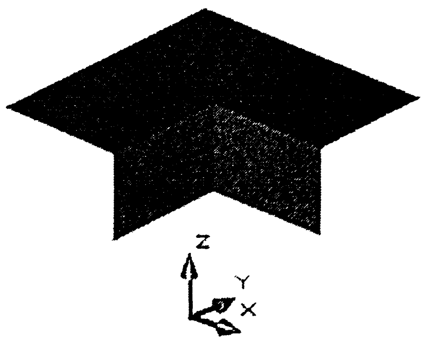
	λ_i	Eigenvectors					
		Translation			Rotation		
	225.89	0.45	-0.48	0.00	0.01	0.01	0.75
	779.54	-0.26	-0.23	0.37	0.51	-0.69	0.01
	1624.70	-0.62	-0.67	-0.19	0.01	0.36	-0.74
	1744.60	-0.37	0.07	-0.07	-0.74	-0.46	0.29
	2721.30	0.45	-0.51	0.00	-0.34	-0.27	0.59
FULLY CONSTRAINED	3526.30	-0.05	-0.03	0.91	-0.26	0.31	0.00

Figure 5.3 Constraint Analysis Shape Configurations 2

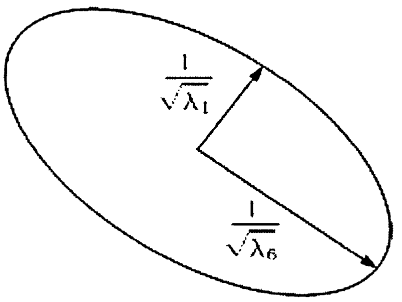


Figure 5.4 Uncertainty Hyperellipsoid

TABLE 5.1 Scalar Measures of Matrix Stability

Name	Equation
Geometric Mean	$\sqrt[6]{\lambda_1 \cdot \lambda_2 \cdot \lambda_3 \cdot \lambda_4 \cdot \lambda_5 \cdot \lambda_6}$ (5.21)
Inverse Condition Number	$\sqrt{\frac{\lambda_6}{\lambda_1}}$ (5.22)
Minimum Eigenvalue	λ_6 (5.23)
Noise Amplification Index	$\frac{\lambda_6}{\sqrt{\lambda_1}}$ (5.24)

5.4.1 Unfit Conditioning Measures

This measure of matrix conditioning was proposed by [Borm & Menq, 1991] and is related to the hyperellipsoid volume. The goal is to make the volume of the error hyperellipsoid as small as possible thereby reducing the effect that a noisy input point cloud will have on the pose estimation parameters. However, this measure does not take into account the eccentricity of the hyperellipsoid. It is possible for a hyperellipsoid to have a small volume, but still have a large major axis. Therefore this measure is not a good indication of point cloud stability for registration purposes.

This measure is the standard matrix conditioning number, but inverted so as to coincide with the other matrix conditioning measures that are to be maximized. The inverse condition number is a good indicator of eccentricity since a decrease in this value leads to a hyperellipsoid that is more spherical. However, this measure does not take into account the volume of the hyperellipsoid and is deemed an inappropriate measure of stability.

The effect of the minimum eigenvalue, as shown in Figure 5.2 and Figure 5.3, leads to the requirement of a larger minimum eigenvalue. However, this measure does not take into account the effects of the hyperellipsoid volume and eccentricity.

5.4.2 Noise Amplification Index

The noise amplification index was proposed by [Nahvi & Hollerback, 1991] in order to remedy the deficiencies of the previous conditioning measures. [Simon, 1996] supported this condition measure and begins his supporting presentation by restructuring Equation (5.16):

$$E_P(T(p_s)) = \sum_{p_s \in P} dD^2(T(p_s)) = \|\Delta D\|^2 = dt^T \Psi_p dt \tag{5.25}$$

where ΔD as a vector associated with residuals of each point. The relative pose error is represented by the vector Δt. This leads to the following relationship [Strang, 1976]:

$$\sqrt{\lambda_6} \le \frac{\|\Delta D\|}{\|\Delta t\|} \le \sqrt{\lambda_1} \tag{5.26}$$

which can be rearranged to become:

$$\|\Delta D\| \ge \sqrt{\lambda_6}\|\Delta t\| \tag{5.27}$$

This states that as the minimum eigenvalue λ_6 increases, $\|\Delta D\|$ more accurately reflects the registration residuals. A more accurate $\|\Delta D\|$ vector will result in greater registration accuracy.

Next, consider the noise associated with the residuals ΔD represented by $\delta\Delta d$ which in turn will result in an error $\delta\Delta t$ in the registration pose components. With these parameters, the following inequality holds [Strang, 1976]:

$$\frac{\|\delta\Delta t\|}{\|\Delta t\|} \le \sqrt{\frac{\lambda_1}{\lambda_6}}\frac{\|\delta\Delta D\|}{\|\Delta d\|} \tag{5.28}$$

Therefore, to minimize the errors of the estimated pose parameters $\|\delta\Delta t\|$, the condition number λ_1/λ_6 should be made as small as possible. Geometrically speaking, this ratio is a measure of the eccentricity of the hyperellipsoid where a smaller condition number reduces the eccentricity of the hyperellipse making it closer to a hypersphere. For accurate registration, the effect that noisy residuals $\delta\Delta d$ will have on the pose parameter errors $\delta\Delta t$ needs to be minimized. However, minimizing the condition number will only lead to a reduced relative error $\delta\Delta t/\Delta t$ as opposed to the desired reduced absolute registration error $\delta\Delta t/\delta\Delta d$. This concern can be rectified by combining Equations (5.27) and (5.28) to give:

$$\frac{\|\delta\Delta t\|}{\|\Delta t\|} \le \sqrt{\frac{\lambda_1}{\lambda_6}}\frac{\|\delta\Delta d\|}{\sqrt{\lambda_6}\|\Delta t\|} \tag{5.29}$$

which can be simplified to:

$$\frac{\|\delta\Delta t\|}{\|\delta\Delta d\|} \le \frac{\sqrt{\lambda_1}}{\lambda_6} \tag{5.30}$$

[Nahvi & Hollerbach, 1996] defined the noise amplification index (NAI) to be:

$$NAI = \frac{\lambda_6}{\sqrt{\lambda_1}} \tag{5.31}$$

It can be understood why this index was coined as the noise amplification index since the NAI represents the maximum amplification between noise in the residuals and registration parameters. Geometrically, a larger NAI forces the minimum eigenvalue to a larger value while also reducing the hyperellipsoid eccentricity.

Therefore the NAI was deemed an acceptable measure of Ψ_p matrix conditioning and was used as in indicator of registration accuracy in this thesis. More specifically, the ideal noise amplification index was used. Equation (5.12) shows the dependency of the NAI on the normal of associated with each point. These normals were generated along with the surface points during the raytracing process. The calculation of normals from a point cloud was attempted using the method of [Mitra & Nguyen, 2003], however the calculated normals resulted in highly inaccurate values of NAI. Point cloud normals are differential quantities and magnifies the effects of error in the xyz point cloud coordinates. Hence, the ideal NAI value was used for comparative purposes.

5.5 Ideal NAI and Registration

To demonstrate the relationship between the ideal noise amplification index and ICP registration, a simple test model was generated from which 1080 scans were generated. Figure 5.5 shows two views of the test model used with a sample scanning area indicated. Each of the 1080 point clouds were given a random initial pose estimate according to Figure 4.3.3, and corrupted with Gaussian noise of zero mean and standard deviation of 8mm. The averaged results of 25 ICP trials for each point cloud is shown in Figure 5.6.

Giving each point cloud a different initial guess is an extreme test of the utility of the noise amplification index since the accuracy of ICP is dependant upon the quality of the initial pose estimate. Even with this harsh scenario, Figure 5.6 shows that the rotation and translation error returned from the ICP trials decreases with increasing NAI. There are a few points not shown on the graphs which have near zero NAI values, but extremely high errors. This observation confirms that low NAI values correspond to highly unstable point cloud configurations that lead to amplified pose estimation error.

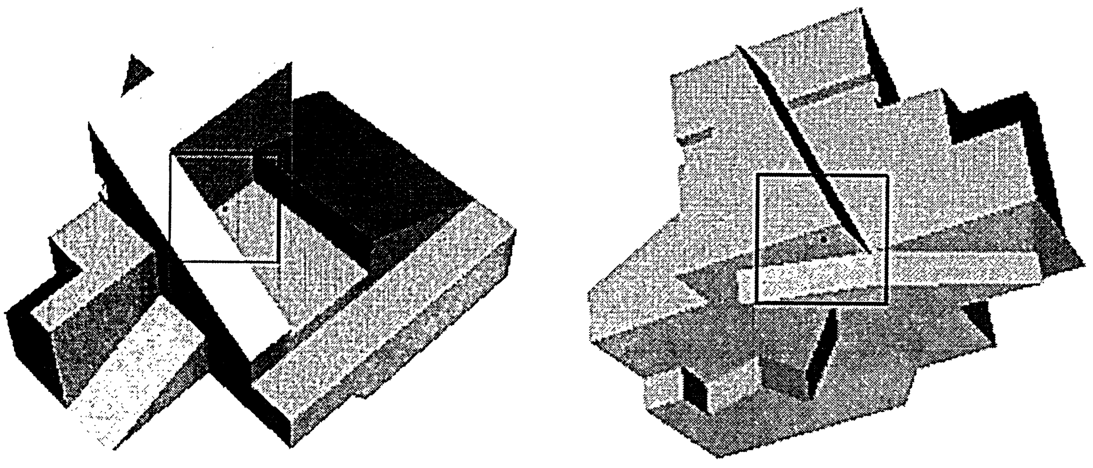


Figure 5.5 Test Model

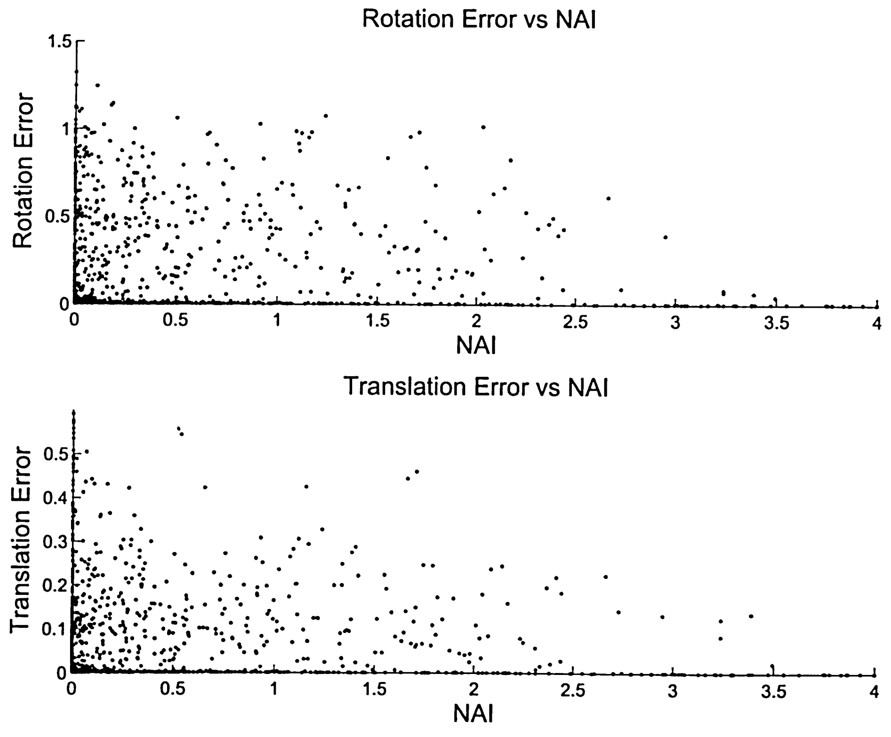


Figure 5.6 Reduced Rotation & Translation Error with Increased NAI

5.6 Summary

The derivation of Point-to-Plane ICP [Chen & Medioni, 1992] is shown in Appendix B. This formulation leads to the same covariance matrix as was obtained in this chapter using constraint analysis.

The constraint analysis formulation presented here makes use of a local approximation of the cost function which is only valid in the vicinity of the global minimum. There is no relationship between registration error and NAI values when ICP converges to a local minimum. Constraint analysis does not provide a means of determining the absolute registration error. However, the utility of constraint analysis is the ability to determine the relative registration error from different point cloud configurations [Simon, 1996].

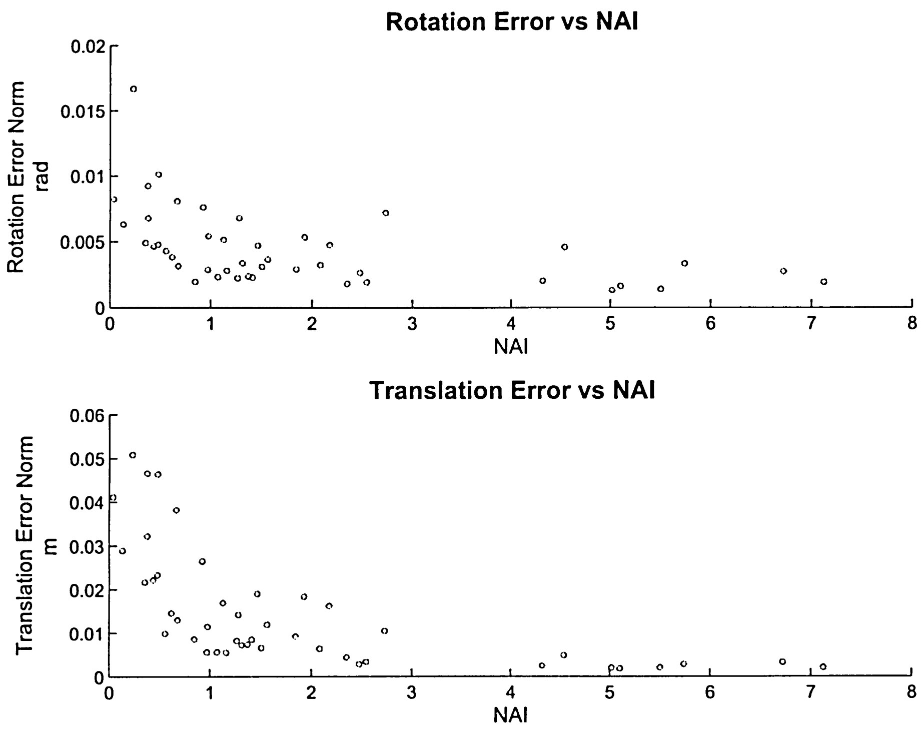
Chapter 6

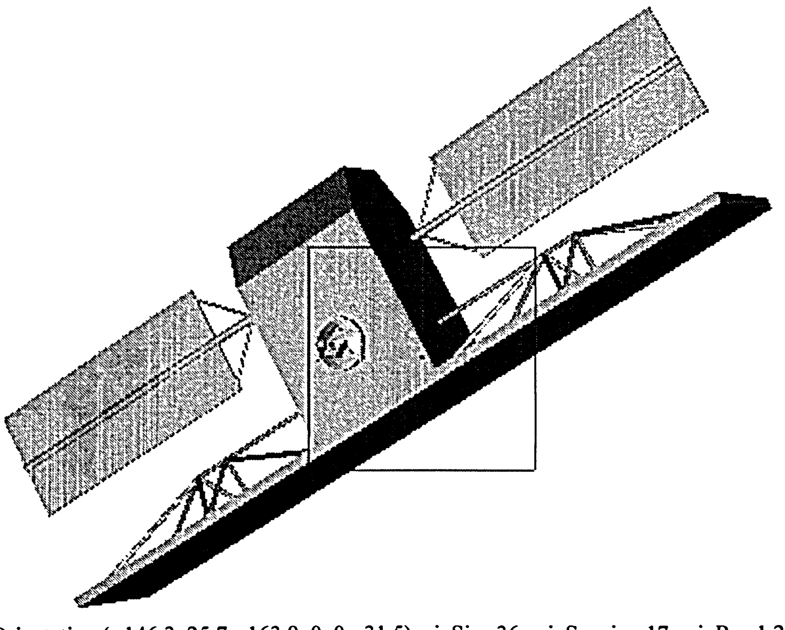
Results & Conclusions

6.1 Ideal Region

Section 5.5 showed how rotation and translation error diminish with increasing NAI values. From this result, the ideal scanning area is defined as the region that has the highest NAI value. The advantage of the NAI value is that it decouples registration accuracy from the ICP algorithm and provides a tool that can be used to determine how suitable a point cloud configuration is for registration.

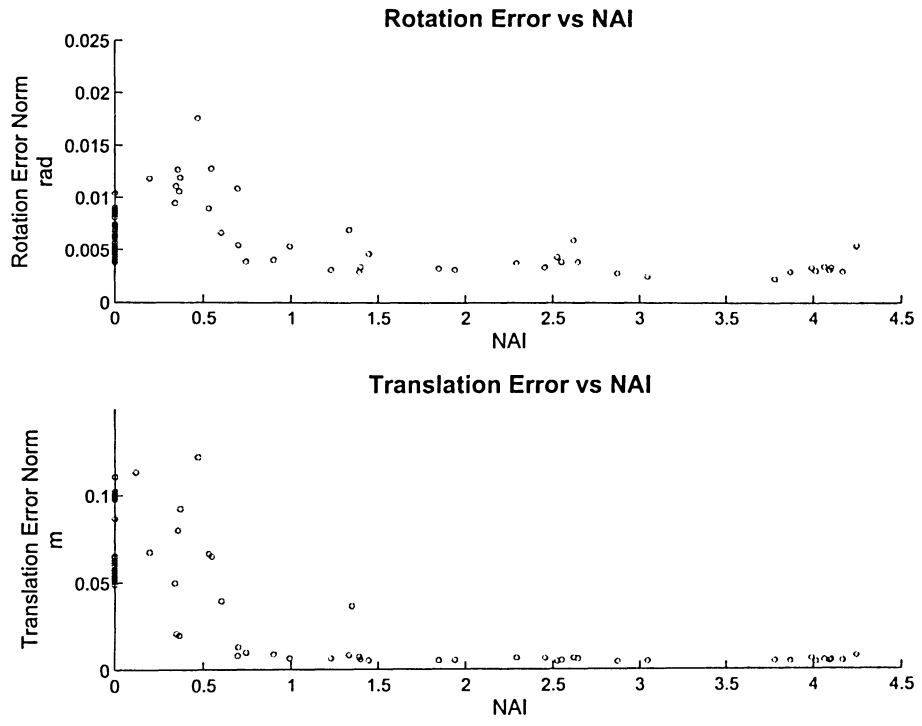
This result is demonstrated experimentally with 100 registration tests on the *Radarsat, Deep Space 1, Stardust* and *Shuttle* models. Figure 6.1 through Figure 6.4 show sample indicative registration results from these tests. The graphs show the ICP registration accuracy of different subregions on the satellites. As expected, the region with the highest NAI value returns the most accurate pose estimate. Granted that the highest NAI value does not always return the most accurate pose estimate as seen by Figure 6.2. However, the trend of decreasing registration error with increasing NAI is evident. It is certainly clear that the constraint singularities that occur for point cloud configurations with low NAI values are to be avoided. The quality of the alignment returned by the ICP algorithm is directly related to the stability of the input point clouds. Data scanned from geometrically unstable regions will cause slower convergence, higher registration error or even total divergence.

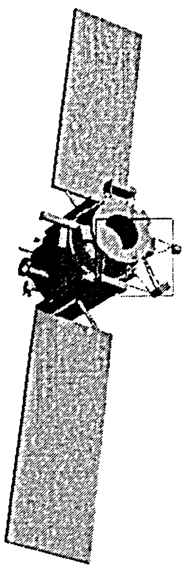




Orientation (-146.3, 25.7, -163.9, 0, 0, -31.5) winSize 36, winSpacing 17, winRes 1.2

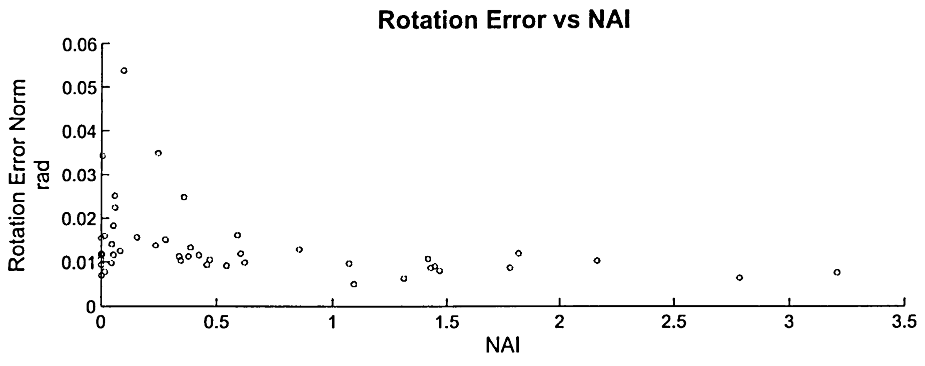
Figure 6.1 Radarsat Optimal Scanning Location Sample Result

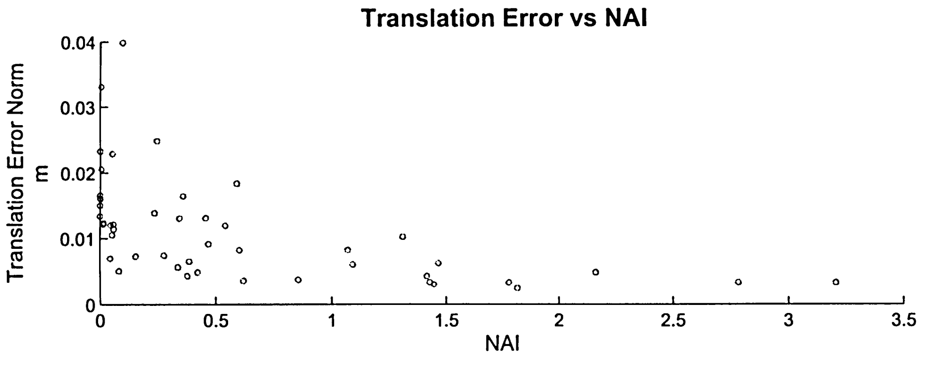


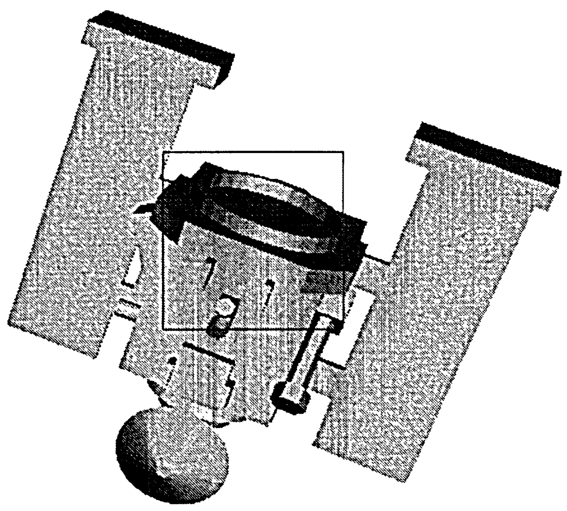


Orientation (-18.4, 14.5, -116.6, 4.0, -0.5, -25.0) winSize 15, winSpacing 25, winRes 1.5

Figure 6.2 Deep Space 1 Optimal Scanning Location Sample Result

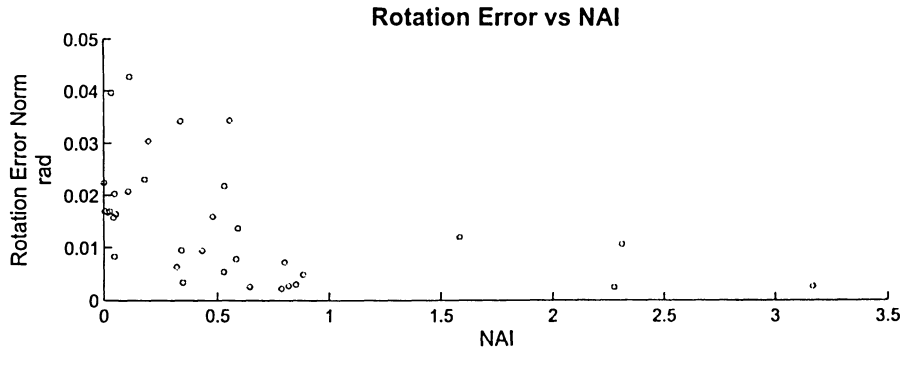


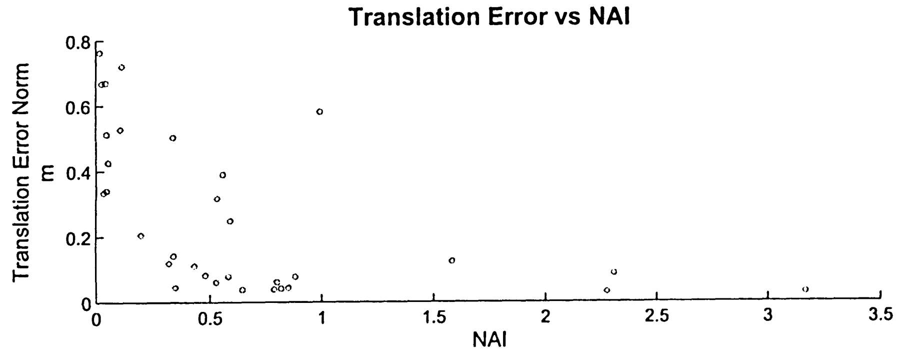


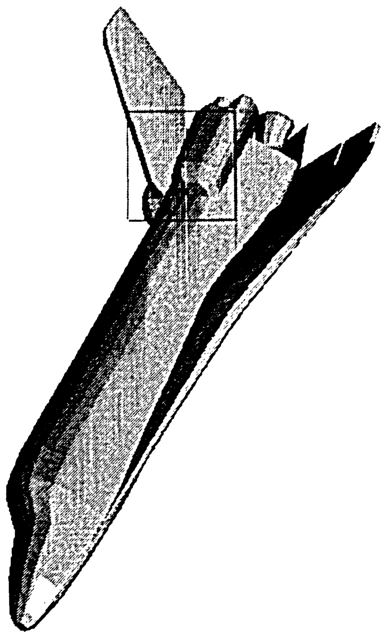


Orientation (-23.0, -12.2, -62.4, 1.5, 0.5, -8.5) winSize 24, winSpacing 40, winRes 2.5

Figure 6.3 Stardust Optimal Scanning Location Sample Result







Orientation (45, -60, 80, 12.5, 3.5, -93) winSize 20, winSpacing 25, winRes 1.8

Figure 6.4 Shuttle Optimal Scanning Location Sample Result

6.2 Improved ICP Performance

The performance derived from the ICP algorithm is directly related to the quality of the input point cloud data. It has been shown in Section 5.5 that using geometrically stable point cloud configurations yields higher accuracy in the pose estimate. Since ICP responds favourably to these stable configurations, it would be expected that a higher rate of convergence would also be realized. To demonstrate this effect, 4 point cloud configurations having different NAI values were extracted from the *Radarsat* model. Table 6.1 shows the results of performing 100 ICP trials each on of these data sets with focus on the number of iterations required for convergence.

TABLE 6.1 Impact of NAI on number of ICP Iterations Required for Convergence

Set	Point Count	NAI	Mean	Std
C1	1210	5.24	45.96	3.53
C2	1288	2.62	78.28	10.20
C3	1151	1.05	179.58	22.16
C4	898	0.15	216.72	37.60

6.3 NAI Implementation

6.3.1 Predictive Scanner

Using the raytracing methods of Chapter 2 and the constraint analysis formulation of Chapter 5, a program was developed that determines and shows the optimal scanning area. In a real-time ICP tracker, the pose of the target satellite can be estimated at a rate of 2Hz using a point cloud of 1000 [Jasiobedzki et al, 2002]. The pose estimate of the satellite position can be used to determine the optimal scanning location for use in the next time step.

For a given pose, a full point cloud scan of the model is generated. This data is then subdivided into window regions by using a kd tree structure to segment the point cloud. The NAI value is calculated for each window and the region with the highest NAI value is deemed the optimal scanning area. The purpose of developing the voxel raytracer in Section 2.5 was to

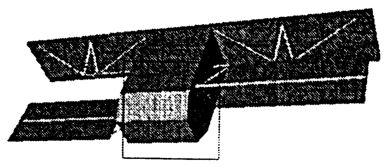
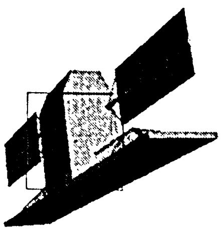
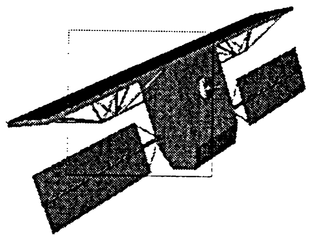
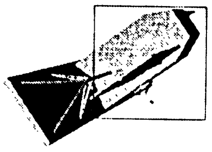
	1404 Points 2.51 NAI 0.2309s 2.93 Full NAI		1010 Points 2.20 NAI 0.2008s 1.69 Full NAI
	1034 Points 2.84 NAI 0.2845s 1.67 Full NAI		1157 Points 2.94 NAI 0.3372s 3.79 Full NAI

Figure 6.5 NAI Predictive Scanner Output

make this method fast enough for possible real-time applications. Figure 6.5 shows four orientations of the *Radarsat* satellite with the optimal scanning area, the NAI value and the run time. The NAI value of the entire point cloud is also included since there are situations where the full point cloud will have a higher NAI value than the best subregion. With further optimization, this program can be made to run even faster and integrated into a real-time ICP satellite tracker. Figure 6.6 shows a block diagram for implementing an intelligent NAI regional region selection method with a LIDAR scanner.

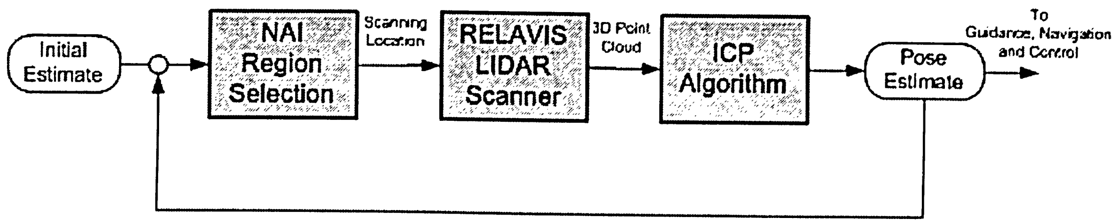


Figure 6.6 Predictive Scanner Implementation

6.3.2 Look Up Tables

An alternate, and probably more realistic, method of designating the optimal scanning area is to generate off-line lookup tables which define the optimal scanning location for a given orientation. This option is derived from an observation that is shown in Figure 6.7. As the window size is increased, the centre point of the window does not shift significantly. The average location of the centre point can be designated as the optimal scanning location for a range of window sizes. The translational component of the satellite position can be used to determine a reasonable window size. The look up table would contain a set of rotation states with an *xyz* coordinate that indicates the centre point of the scanning region.

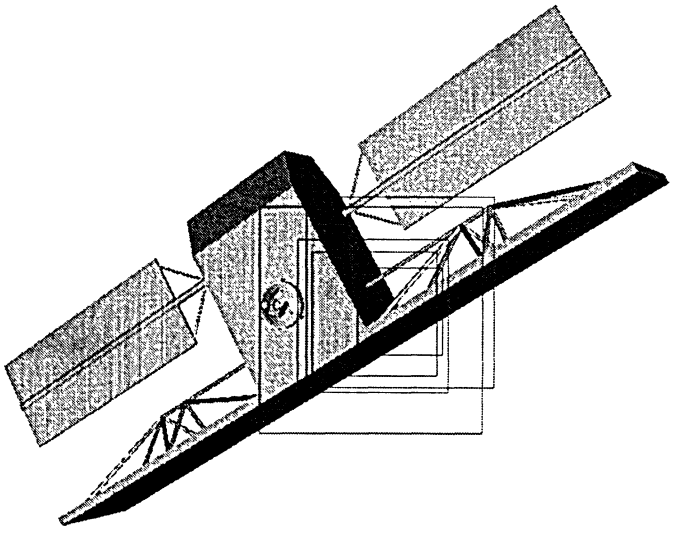


Figure 6.7 Varying Optimal Window Size

6.4 Conclusions

Constraint analysis provides a means of detecting and avoiding geometrically unstable regions. The contribution of small eigenvalues in the covariance matrix Ψ_P from these rather featureless areas can amplify the noise present in the data coordinates to return highly inaccurate pose estimates. The goal of determining an ideal scanning area for ICP based registration was realized with the introduction of the noise amplification index as a measure of point cloud stability. It was demonstrated that point cloud configurations with higher values of NAI returned more accurate pose estimates than those configurations with lower NAI values. With this result, the NAI was put to use as a predictor of the optimal scanning area on a given satellite.

Incorporating the NAI registration predictor into a LIDAR based satellite tracking system will significantly enhance its performance and reduce error due to instabilities. The goal of developing an autonomous repair satellite capable of tracking, docking with and repairing imaired satellites can only be realized with a precise pose estimate of the target satellite.

6.5 Future Work

Synthetic data is useful only up to a certain point. To fully test a method's peformance and limits of operation, experimental testing in a realistic environment is needed. While numerically the methods outlined in this thesis will increase ICP performance, other mitigating factors may negate or perhaps amplify this performance effect.

REFERENCES

[Amanatides & Woo, 1987] Amanatides, J. & Woo, A., *A Fast Voxel Traversal Algorithm for Ray Tracing*, Eurographics Conference Proceedings, pp. 3-10, 1987.

[Arun et al, 1987] Arun, K.S., Huang, T.S., Bolstein, S.D., *Least-squares Fitting of Two 3D Point Sets*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 9(5): 698-700, September 1987.

[Badouel, 1990] Badouel, F., *An Efficient Ray-Polygon Intersection*, Graphic Gems, Academic Press pp:390-393, 1990.

[Besl & McKay, 1992] Besl, P.J.. and McKay, N.D., *A Method for Registration of 3D Shapes*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 14(2):239-256, 1992.

[Beraldin et al, 1993] Beraldin, A., El-Hakim, S.F. & Cournoyer, L., Practical Range Camera Calibration, SPIE Vol 2067:21-31, 1993.

[Blais & Levine, 1995] Blais, G. & Levine, M. Registering Multiview Range Data to Create 3D Computer Objects, Trans. PAMI, 17:(8), 1995.

[Blais et al, 2000] Blais, F., Beraldin, A. & El-Hakim, S.F., *Range Error Analysis of an Integrated Time-of-Flight, Triangulation and Photogrammetry 3D Laser Scanning System*, Proc. SPIE AeroSense, 2000.

[Borm & Menq, 1991] Borm, J.H. & Menq, C.H., Determination of Optimal Measurement Configurations for Robot Calibration Based on Observability Measure, International Journal of Robotics Research, 10(1):51-63, 1991.

[Chen & Medioni, 1992] Chen, Y., Medioni, G., Object Modeling by Registration of Multiple Range Images, Image and Vision Computing, 10(3):145-155, 1992.

[Eggert et al, 1997] Eggert, D.W., Lorusso, A., Fisher, R.B., *Estimating 3D Rigid Body Transformations: A Comparison of Four Major Algorithms*, Machine Vision and Applications, 9:272-290, 1997.

[Ericson, 2004] Ericson, C., *Real-Time Collision Detection*, Morgan Kaufmann, San Francisco USA, 2004.

[Friedman et al, 1977] Friedman, J.H., Bentley, J.L. & Finkel, R.A., *An Algorithm for Finding Best Matches in Logarithmic Expected Time*, ACM Transactions on

Mathematical Software, 3(3):209-226, 1977.

[Gelfand et al, 2003] Gelfand, N., Ikemoto, L. & Rusinkiewciz, S, *Geometrically Stable Sampling for the ICP Algorithm*, Proc. International Conference on 3D Digital Imaging and Modeling, pp. 260-267, 2003.

[Gelfand et al, 2005] Gelfand, N., Mitra, N., Guibas, L.J. & Pottmann, H., Robust Global Registration, Eurographics Symposium on Geometry Processing, 2005.

[Greenspan & Yurick, 2003] Greenspan, M. & Yurick, M., *Approximate kd Tree Search for Efficient ICP*, 3dim Fourth International Conference on 3D Digital Imaging and Modeling, 2003.

[Held, 1997] Held, M., *ERIT - A Collection of Efficient and Reliable Intersection Tests*, Journal of Graphic Tools, 2(4):25-44, 1997.

[Hill, 2001] Hill, F.S. Jr., *Computer Graphics Using Open GL*, Prentice Hall, 2001.

[Horn, 1988] Horn, B.K.P., *Closed form Solution of Absolute Orientation using Unit Quaternions*, Journal of the Optical Society of America, 4(4):629-642, 1987.

[Hollander, 2000] Hollander, S., *Autonomous Space Robotics: Enabling Technologies for Advanced Space Platforms*, AIAA Space Conference, 2000.

[Inaba & Oda, 2000] Inaba, N. & Oda, M., *Autonomous Satellite Capture by a Space Robot*, Proc. of IEEE International Conference on Robotics and Automation, San Francisco, 2000.

[Jasiobedzki et al, 2001] Jasiobedzki, P., Greenspan, M. & Roth, G., *Pose Determination and Tracking for Autonomous Satellite Capture*, Proc 6th i-SAIRAS, 2001.

[Jasiobedzki et al, 2002] Jasiobedzki, P., Greenspan, M. Roth, G., Ng, H. & Witcomb, N., *Video-Based System for Satellite Proximity Operations*, 7th ASTRA Workshop 2002.

[Kim & Khosla, 1991] Kim, J. and Khosla, P.K., Dexterity Measures for Design and Control of Manipulators, Proceedings of the International Workshop on Intelligent Robots and Systems, pp. 758-763, Osaka, Japan, 1991.

[Luck et al, 2000] Luck, J., Little, C. & Hoff, W., *Registration of Range Data Using a Hybrid Simulated Annealing and Iterative Closest Point Algorithm*, Proc. of IEEE International Conference on Robotics and Automation, San Francisco, 2000.

[Martin et al, 2004] Martin, E., Maharaj, D., Richards, R. & Tripp., J., *RELAVIS: The Development of a 4D Laser Vision System for Spacecraft Rendezvous and Docking*

Operations, Proc. SPIE Vol 5418, 2004.

[Masuda et al, 1994] Masuda, T., Sakaue, K. & Yokoya, N., *Registration and Integration of Multiple Range Images for 3D Model Construction*, Proc CVPR 1996.

[Mitra & Nguyen, 2003] Mitra, N.J. & Nguyen, A., *Estimating Surface Normals in Noisy Point Cloud Data*, Proc. 19th Annual Symposium on Computational Geometry, pp. 322-328, 2003.

[Mitra et al, 2004] Mitra, N., Gelfand, N., Pottmann, H. & Guibas, L., *Registration of Point Cloud Data from a Geometric Optimization Perspective*, Eurographics Symposium on Geometry Processing, 2004.

[Moller, 2001] Moller, T., *Fast 3D Triangle-Box Overlap Testing*, Journal of Graphic Tools, 6(1):29-33, 2001.

[Moller & Trumbore, 1997] Moller, T. and Trumbore, B., *Fast, Minimum Storage Ray-Triangle Intersection*, Journal of Graphics Tools, 2(1):21-28, 1997.

[Mount & Ayra, 2005] Mount, D.M., and Arya, S., *ANN: A Library for Approximate Nearest Neighbour Searching*, University of Maryland, www.cs.umd.edu/~mount/ANN, 2005.

[Nahvi & Hollerbach, 1996] Nahvi, A. & Hollerbach, J., The Noise Amplification Index for Optimal Pose Selection in Robot Calibration, Proceedings of IEEE International Conference on Robotics and Automation, pp. 647-654, Minneapolis USA, 1996.

[Pelletier et al, 2004] Pelletier, F.J., Golla, D.F. & Allen, A.C.M., *Lidar-Based Rendezvous Navigation for MSR*, AIAA/ASS Astrodynamics Specialist Conference and Exhibit, 2004.

[Rusinkiewicz & Levoy, 2001] Rusinkiewicz, R. & Levoy, M., Efficient Variants of the ICP Algorithm, Third International Conference on 3D Digital Imaging and Modeling (3DIM), 2001.

[Segura & Feito, 2001] Segura, R.J., Feito, F., *Algorithms to Test Ray-Triangle Intersection: A Comparitive Study*, Journal of Winter School of Computer Graphics, 9(3), 2001.

[Sharp et al, 2002] Sharp, G.C., Sang, W.L. & Wehe, D.K., *ICP Registration Using Invariant Features*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 24(1):90-102, 2002.

[Silva et al,] Silva, L., Bellon, O. & Boyer, K., Enhanced Robust Genetic Algorithms for Multiview Range Image Registration, 3dim Fourth International Conference on 3D Digital Imaging and Modeling, 2003.

[Simon, 1996] Simon, D., *Fast and Accurate Shape-Based Registration*, Ph.D Dissertation, Carnegie Mellon University, 1996.

[Strang, 1976] Strang, G., *Linear Algebra and its Applications*, New York, Academic Press, 1976.

[Stoddart et al, 1996] Stoddart, A.J., Lemke, S., Hilton, A. & Renn, T., *Estimating Pose Uncertainty for Surface Registration*, British Machine Vision Conference, 1996.

[Sunday, 2001] Sunday, D., *Intersections of Ray Segments, Planes and Triangles in 3D*, *www.geometryalgorithms.com*, 2001.

[Turk, 1994] Turk, G. & Levoy., M., *Zipper Polygon Meshes from Range Images*, Proceedings SIGGRAPH, 1994.

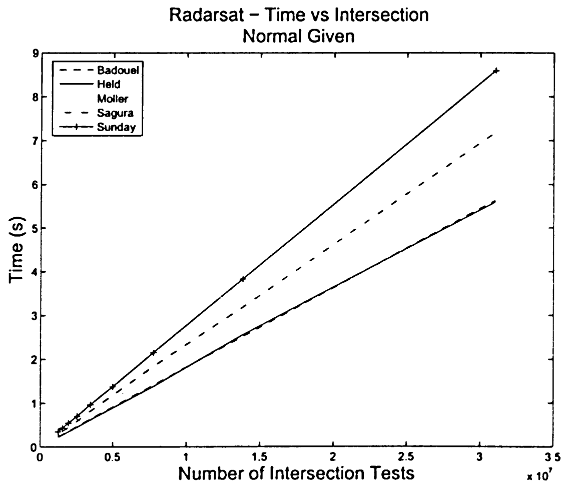
[Thorsley et al, 2004] Thorsley, M., Okouneva, G. & Karpynczyk, J, Stereo Vision Algorithms for Robotic Assembly Operations, Proc. 1st Canadian Conference on Computer and Robot Vision, pp. 361, 2004.

[Walker et al, 1991] Walker, M.W., Shao, L., Volz, R.A., Estimating 3D Location Parameters Using Dual Quaternions, CVGIP: Image Understanding, 54:358-367, 1991.

[Woo et al, 1997] Woo, M., Neider, J., Davis, T., Shreiner, D., *Open GL Programming Guide: 3rd Edition*, Addison-Wesley, 1997.

Appendix A

Raytrace Algorithm Comparison



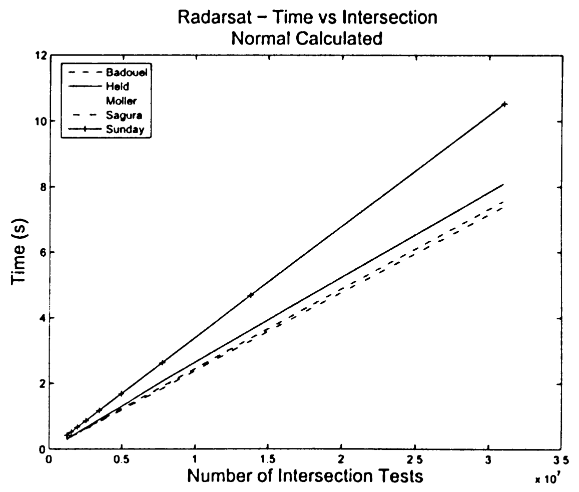
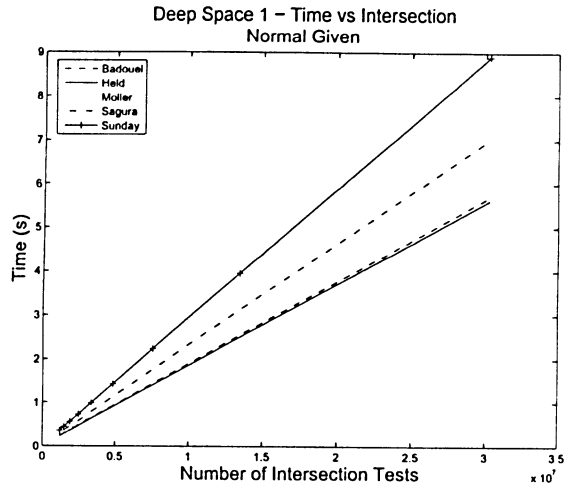


Figure A.1 Radarsat Raytrace Algorithm Comparison



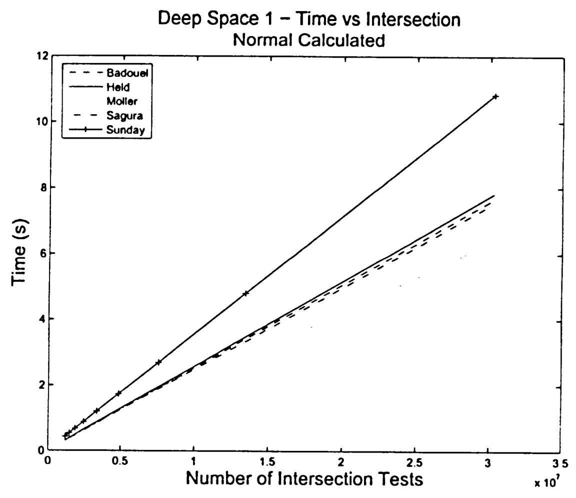
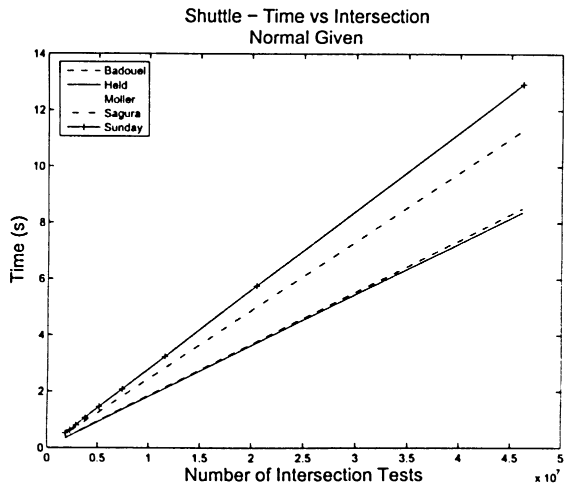


Figure A.2 Deep Space 1 Raytrace Algorithm Comparison



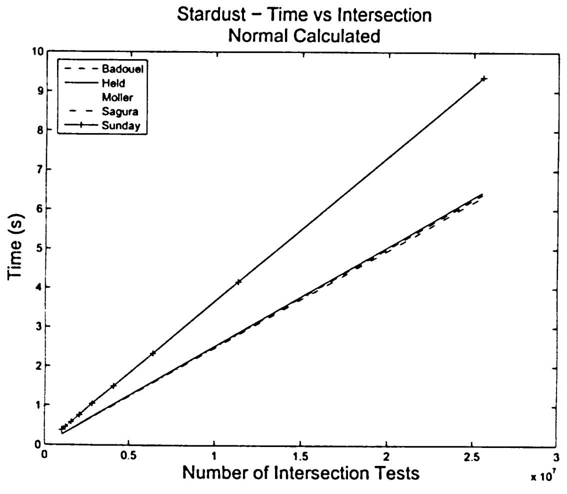


Figure A.3 Shuttle Raytrace Algorithm Comparison

Appendix B

Point to Plane Derivation

This method of registration is based on the work of [Chen & Medioni, 1992]. The constraint analysis formulation was put forth by [Gelfand et al, 2003].

The formulation of the cost function of this method is different than that for the Point-to-Point ICP version as it esplicitly uses point cloud normals in the cost function. The point correspondences are built using a graphical method as shown in Figure B.1. Starting from the cloud point $\vec{p}$, the normal $\vec{n}$ is followed until it intersects the model at point $\vec{t}$. The tangent line at point $\vec{t}$ is determined. The point $\vec{q}$ is found such that it is perpendicular to the tangent line and passes through the point $\vec{p}$.

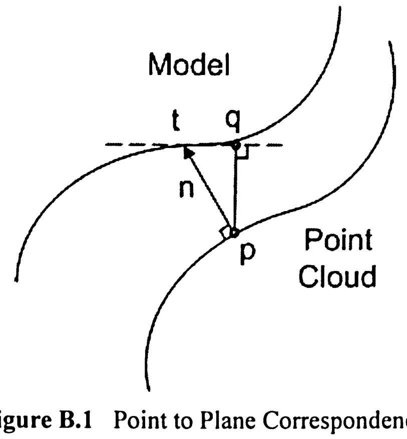


Figure B.1 Point to Plane Correspondence

The cost function that will be minized for this formulation is:

$$E = \sum_{i=1}^{n} ((R\vec{p}_i + \vec{t} - \vec{q}_i) \cdot \vec{n}_i)^2 \qquad \text{(B.1)}$$

The rotation marix for this method is expressed using Euler angles (ω_x, ω_y, ω_z) to represent rotations about the x, y and z axes respectively. The rotation matrix is non-linear since it utilizes the sines and cosines of the rotation angles. If the rotations are incrementally small, it is possible to linearize the rotation matrix by using the approximations $\cos\theta \approx 1$ and $\sin\theta \approx 0$. The rotation matrix can be expressed as:

$$\boldsymbol{R} = \begin{bmatrix} 1 & -\omega_z & \omega_y \\ \omega_z & 1 & -\omega_x \\ -\omega_y & \omega_x & 1 \end{bmatrix} \qquad \text{(B.2)}$$

which allows the cost function to be simplified to:

$$E = \left(\sum_{1}^{n} ((\vec{p}_i - \vec{q}_i) \cdot \vec{n}_i + \vec{t} \cdot \vec{n}_i + \vec{r} \cdot (\vec{p}_i \times \vec{n}_i))^2 \right) \qquad \vec{r} = [\omega_x\ \omega_y\ \omega_z]^T \qquad \text{(B.3)}$$

The required transformation is found by taking partial derivatives of Equation (B.3) with respect to the transformation parameters. The result is a linear system of equations of the form $Ax = b$ where x is the 6x1 vector of transformation parameters $\left[\omega_x\ \omega_y\ \omega_z\ t_x\ t_y\ t_z\right]$

A is a 6x6 covariance matrix given by

$$A = VV^T = \begin{bmatrix} \vec{n}_1 & \dots & \vec{n}_n \\ \vec{p}_1 \times \vec{n}_1 & \dots & \vec{p}_n \times \vec{n}_n \end{bmatrix} \begin{bmatrix} (\vec{n}_1)^T & (\vec{p}_1 \times \vec{n}_1)^T \\ \dots & \dots \\ (\vec{n}_n)^T & (\vec{p}_n \times \vec{n}_n)^T \end{bmatrix} \qquad \text{(6.1)}$$

Appendix C

Eigenvalue Constraint Derivations

Consider a postive definite mxm matrix A and the standard linear equation $A\vec{x} = \vec{b}$. Expanding matrix A using eigen value decomposition gives:

$$Q\Lambda Q^T\vec{x} = \vec{b} \tag{C.1}$$

where the matrix Q contains the eigenvectors and the matrix Λ contains the eigenvalues. The solution vector $\vec{x}$ can be expanded to give:

$$Q\Lambda Q^T\hat{x}\|\vec{x}\| = \vec{b} \tag{C.2}$$

The maximum and minimum values of $\|\vec{b}\|$ occur when $\ddot{x}$ points in the direction of the largest and smallest eigenvalues respectively. When this happens, $Q\Lambda Q^T\hat{x}\|\vec{x}\|$ returns a vector whose magnitude is equal to the eigenvalue associated with the given direction. If the smallest eigenvalue is represented by λ_6 while the largest eivenvalue is λ_1, the following equations:

$$\lambda_6 = \frac{\|\vec{b}\|}{\|\vec{x}\|} \qquad \lambda_1 = \frac{\|\vec{b}\|}{\|\vec{x}\|} \tag{C.3}$$

can be collapsed into the following inequality:

$$\lambda_6 \le \frac{\|\vec{b}\|}{\|\vec{x}\|} \le \lambda_1 \tag{C.4}$$

since the largest and smallest eigenvalues represents upper and lower bounds on $\|\vec{b}\|/\|\vec{x}\|$.

Once again, consider a positive definite mxm matrix A and the standard linear equation $A\vec{x} = \vec{b}$ but also include the error $A\overrightarrow{\delta x} = \overrightarrow{\delta b}$. The goal is now to compare the relative error of the input parameters $\|\overrightarrow{\delta b}\|/\|\vec{b}\|$ with the induced relative error of the solution $\|\overrightarrow{\delta x}\|/\|\vec{x}\|$. The worst case is when a small error in the input parameters cause a large error in the solution. This happens when the numerator $\|\overrightarrow{\delta x}\|$ is large while the denominator $\|\vec{x}\|$ is small.

Simlar to the previous derivation, the standard linear equation and the error equation can be written as:

$$Q\Lambda Q\hat{x}\|\vec{x}\| = \vec{b} \qquad Q\Lambda Q\ddot{\delta x}\|\overrightarrow{\delta x}\| = \overrightarrow{\delta b} \tag{C.5}$$

$\|\vec{x}\|$ has the smallest value when $\vec{x}$ points in the direction of the largest eigenvalue giving:

$$\|\vec{x}\| \geq \frac{\|\vec{b}\|}{\lambda_1} \tag{C.6}$$

$\|\overrightarrow{\delta x}\|$ has the largest value when $\ddot{\delta x}$ points in the direction of the smallest eigenvalue giving:

$$\|\overrightarrow{\delta x}\| \leq \frac{\|\overrightarrow{\delta b}\|}{\lambda_6} \tag{C.7}$$

Combining these two results yields:

$$\frac{\|\overrightarrow{\delta x}\|}{\|\vec{x}\|} \leq \frac{\lambda_1}{\lambda_6}\frac{\|\overrightarrow{\delta b}\|}{\|\vec{b}\|} \tag{C.8}$$

where the quantity $c = \lambda_1/\lambda_6$ is the condition number of matrix A.