Geometric Surface Registration for 3D Model Building

Simon James Cunnington

Submitted for the Degree of Doctor of Philosophy from the University of Surrey

Centre for Vision, Speech and Signal Processing
School of Electronics and Physical Sciences
University of Surrey
Guildford
Surrey
GU2 7XH
United Kingdom

January 2003

Copyright © 2003 by Simon James Cunnington, UniS
Summary

This thesis concerns geometric surface registration, a vital part of automatic 3D model building. The work is centred on the iterative closest point (ICP) algorithm and a study is made of how the choice of closest point method affects the accuracy, stability and speed of the algorithm. A comparative study of n view point set alignment methods is also presented. It is shown how the ICP algorithm can be extended to use the n view point set alignment methods to register multiple surfaces. The value of robustness checks in improving registration is demonstrated, especially when registering multiple surfaces. Finally a post-processing self-calibration technique is presented for data acquired using the ModelMaker Reality Capture System, a laser sensor on a coordinate-measuring arm.

Key words: Surface Registration, Self-Calibration, 2-View, N-View, Closest Point Methods
Acknowledgements

I would like to thank my supervisors Professor John Illingworth and Dr. Andrew Stoddart for the help which they have given me during this research. I would like to thank again Dr. Andrew Stoddart and Dr. Adrian Hilton for the software they provided that contributed to this research.

I would also like to thank a wealth of friends who I have had the privilege to meet while being here at Surrey for their support and friendship. Extra thanks goes to Sami, Daniel Davies and Peter Yuen for their support during various stages of this work. When working towards a PhD, emotional support plays a vital role, and so a special thanks goes to Helen for providing such support and encouragement. I would also like to thank Emily for the extra support given in the final stage of writing this thesis.

Finally, a thank you to my parents for putting up with me in the final stages.

This research was supported by the Copernicus project VIRTUOUS and the Engineering and Physical Sciences Research Council (EPSRC).

Further information:

http://www.ee.surrey.ac.uk/Personal/S.Cunnington/
Email S.Cunnington@eim.surrey.ac.uk
Contents

Summary iii

Acknowledgements iv

List of Figures xi

List of Tables xv

1 Introduction 1
  1.1 Background ................................................. 5
  1.2 Aim Of Work .................................................. 8
    1.2.1 Achievements ........................................... 10
  1.3 Overview Of Thesis ......................................... 12

2 Literature Review 15
  2.1 Registration Of Non-Surface Data ......................... 15
  2.2 Non-Rigid Registration ..................................... 16
  2.3 Registration Of Two Geometric Surfaces ................. 17
    2.3.1 Genetic Algorithms .................................... 17
    2.3.2 Feature Matching ...................................... 18
    2.3.3 Closest Point Approaches ............................. 21
    2.3.4 Spin Images ............................................. 26
    2.3.5 Reverse Calibration .................................... 26
    2.3.6 Planar Segmentation ................................... 27
    2.3.7 Registration Aids ........................................ 27
    2.3.8 Fuzzy Clustering ........................................ 28
2.4 Registration Of Multiple Geometric Surfaces ........................................... 29
  2.4.1 Surface Matching ................................................................................. 29
  2.4.2 Closest Point Approaches ................................................................... 30
  2.4.3 Contours ............................................................................................... 32
  2.4.4 Multiple Error Modelling .................................................................... 33
  2.4.5 Error Distribution via Graphs Across Frame Space ............................. 34
2.5 Luminance, Colour And Texture Models .................................................... 34
2.6 Mesh Optimisation ..................................................................................... 36

3 2-View Point Set Alignment ........................................................................... 39
  3.1 Geometry ................................................................................................. 40
    3.1.1 2D Geometry, Shapes And Transforms ............................................ 40
    3.1.2 Rotation .............................................................................................. 42
    3.1.3 3 Dimensional Geometry And Object Models .................................. 44
    3.1.4 Encapsulated Rigid Body Transforms ............................................. 45
    3.1.5 Different Methods of Representing Rotation ..................................... 46
  3.2 Least Squares .......................................................................................... 52
    3.2.1 Singular Value Decomposition ........................................................ 57
    3.2.2 The Influence of Errors ..................................................................... 57
    3.2.3 Introducing Robustness .................................................................... 59
  3.3 2-View Point Set Alignment ..................................................................... 64
  3.4 Other Methods Of Point Set Alignment .................................................. 69

4 Determining The Closest Point ...................................................................... 73
  4.1 Surfaces ................................................................................................... 74
    4.1.1 Polygon Meshes ................................................................................. 75
    4.1.2 Spline, Bezier And B-Spline Patches .............................................. 77
  4.2 Closest Point Methods ............................................................................. 80
    4.2.1 Vertex Based Search ......................................................................... 81
    4.2.2 Surface Based Search ....................................................................... 83
    4.2.3 Voxel Search ..................................................................................... 88
    4.2.4 Range Image Search ........................................................................ 92
  4.3 Summary ................................................................................................... 97
Contents

5  View Surface Registration
  5.1 Overview Of The Iterative Closest Point Algorithm
  5.2 Measuring Error And Incorporating Robustness
    5.2.1 Mean Square Error
    5.2.2 Median Absolute Deviation Estimator
  5.3 The Registration Termination Criterion
  5.4 Degeneracy, Locking Features And Local Minimum
  5.5 Extra Checks For Improvement Of Registration
    5.5.1 Boundary Check
    5.5.2 Normal Angle Check
    5.5.3 Distance Check
  5.6 The Modified Iterative Closest Point Algorithm
  5.7 The Accelerated ICP Algorithm
  5.8 Results
    5.8.1 Data Used
    5.8.2 General Approach
    5.8.3 The Experiments
  5.9 Summary

6  N-View Point Set Alignment: A Comparison
  6.1 N-View Alignment
    6.1.1 Problem definition
    6.1.2 Pennec
    6.1.3 Stoddart and Hilton
    6.1.4 Benjemaa And Schmitt
  6.2 Results
    6.2.1 Implementation
    6.2.2 Creation of Synthetic Data Sets
    6.2.3 Quantitative Measures Used
    6.2.4 Convergence
    6.2.5 Highly Nonspherical Models
    6.2.6 Conclusion
  6.3 Summary
A.2.3 Range Image Search .............................................. 243
A.3 2 View Surface Registration ............................... 243
   A.3.1 Accelerated ICP ............................................. 244
   A.3.2 Results ......................................................... 245
A.4 N-View Point Set Alignment ............................... 245
   A.4.1 Pennec .......................................................... 246
   A.4.2 Stoddart And Hilton ........................................ 246
   A.4.3 Results ......................................................... 246
A.5 N View Surface Registration ............................... 247
A.6 Self-Calibrating Surface Reconstruction For ModelMaker Data . . . 247

B Published Papers .............................................. 249

Bibliography ......................................................... 273
List of Figures

1.1 The Model Building Process. ........................................... 2
1.2 A Simple Overview Of The Iterative Closest Point Algorithm. 3

3.1 Simple Geometrical Objects. ........................................ 40
3.2 Simple Representation Of Rotation. ............................. 43
3.3 Right-Handed And Left-Handed Coordinate Systems. ...... 45
3.4 Axis Angle: Rotation Of A Point. .................................. 49
3.5 Possible Straight Line Fits. .......................................... 53
3.6 Perpendicular Distances Are Used To Determine The Best Fit. 54
3.7 Analytical Line Fitting. .............................................. 55
3.8 Affects Of An Outlier. ............................................... 59
3.9 How Outliers Affect The Measure Of Central Tendency. ..... 60
3.10 Kernel And Influence Function For Least Squares. .......... 62
3.11 Use Of A Weight Function To Create A New Kernel And Influence Function. ................................. 63
3.12 Solving For The Rotation Between 2 Corresponding Point Sets. 66

4.1 The True And Approximate Closest (Corresponding) Point. 74
4.2 Polygon Mesh. ..................................................... 75
4.3 Different Polygon Mesh Representations. ........................ 76
4.4 Curves: Spline, Bezier And B-Spline. ......................... 78
4.5 Bezier Patch. ................................................... 79
4.6 Vertex based Search: Hidden Point Case (Side On View Of Surface). 83
4.7 Surface Based Search: Encoding. ............................... 84
4.8 Surface Based Search: Determining The Closest Point. ........ 86
4.9 Volumetric Based Search: Encoding. ......................... 90
List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.10</td>
<td>Range Square Triangles.</td>
<td>93</td>
</tr>
<tr>
<td>4.11</td>
<td>Range Image Search: Determining Nearest Point.</td>
<td>95</td>
</tr>
<tr>
<td>5.1</td>
<td>Flowchart Overview Of 2 View Surface Registration.</td>
<td>102</td>
</tr>
<tr>
<td>5.2</td>
<td>ICP And Local Minimum.</td>
<td>107</td>
</tr>
<tr>
<td>5.3</td>
<td>Illustration Of Incorrect Corresponding Points For Non Overlapping Region.</td>
<td>110</td>
</tr>
<tr>
<td>5.4</td>
<td>Boundary Check Weight Function.</td>
<td>111</td>
</tr>
<tr>
<td>5.5</td>
<td>Bad Correspondence Pairs Shown By Difference In Surface Normals.</td>
<td>112</td>
</tr>
<tr>
<td>5.6</td>
<td>Normal Angle Check Weight Function.</td>
<td>113</td>
</tr>
<tr>
<td>5.7</td>
<td>Three Bad Correspondence Pairs Shown By A Large Difference In Distances Compared To The Average.</td>
<td>114</td>
</tr>
<tr>
<td>5.8</td>
<td>Accelerated ICP: Exploiting Properties Of The Convergence Curve.</td>
<td>118</td>
</tr>
<tr>
<td>5.9</td>
<td>Accelerated ICP Curve With Mathematical Notation.</td>
<td>120</td>
</tr>
<tr>
<td>5.10</td>
<td>Foot Data Set.</td>
<td>125</td>
</tr>
<tr>
<td>5.11</td>
<td>Beethoven Data Set.</td>
<td>126</td>
</tr>
<tr>
<td>5.12</td>
<td>Bounding Box Vector For A 2D Object.</td>
<td>130</td>
</tr>
<tr>
<td>5.13</td>
<td>Series 1: Basic ICP vs Accelerated ICP For Foot.</td>
<td>134</td>
</tr>
<tr>
<td>5.14</td>
<td>Series 1: VoxRegis Direct Lookup vs Iterative Search.</td>
<td>135</td>
</tr>
<tr>
<td>5.15</td>
<td>Series 1: Effects Of Varying Neighbourhood Size For RanRegis Using Foot.</td>
<td>139</td>
</tr>
<tr>
<td>5.16</td>
<td>Series 1: Effects Of Varying Voxel Size For VoxRegis Direct Lookup Using Foot.</td>
<td>141</td>
</tr>
<tr>
<td>5.17</td>
<td>Series 1: Effects Of Varying Voxel Size For VoxRegis Iterative Search Using Foot.</td>
<td>142</td>
</tr>
<tr>
<td>5.18</td>
<td>Series 2: Effects Of Varying The Number Of Points For PolRegis Using Beethoven.</td>
<td>145</td>
</tr>
<tr>
<td>5.19</td>
<td>Series 2: Effects Of Varying The Number Of Points For RanRegis And VoxRegis Direct Lookup Using Beethoven.</td>
<td>146</td>
</tr>
<tr>
<td>5.20</td>
<td>Series 2: Stability At Resolution 38x38 And 50x50 For Foot.</td>
<td>148</td>
</tr>
<tr>
<td>5.21</td>
<td>Series 2: Stability At Resolution 100x100 For Foot.</td>
<td>149</td>
</tr>
<tr>
<td>5.22</td>
<td>Series 3: Core Experiments From Series 1 Re-run With Individual Robustness Checks.</td>
<td>152</td>
</tr>
<tr>
<td>5.23</td>
<td>Series 3: Effect Of Changing Distance Check Ramp Start.</td>
<td>154</td>
</tr>
<tr>
<td>5.24</td>
<td>Series 3: Beethoven From Series 2 Re-Run With Robustness Checks.</td>
<td>155</td>
</tr>
</tbody>
</table>
5.25 Series 3: Foot From Series 2 Re-Run With Robustness Checks. . . . 156
5.26 Series 4: Effects Of Noise And MSE vs MAD Using Boundary, Normal And Distance Check. Re-run Of Series 1 Core. ............................ 158
5.27 Series 4: Effects Of Noise And MSE vs MAD Using Boundary, Normal And Distance Check. Re-run Of Series 2 Core, Foot. ............ 159
5.28 Series 4: Effects Of Noise And MSE vs MAD Using Boundary And Normal Check Only. Re-run Of Series 2 Core, Foot. .......... 160

6.1 Illustration Of Overlap Between Point Set Views. ....................... 167
6.2 Pennec’s Mean Shape. ................................................................. 168
6.3 A Physical System Of Rigid Bodies Connected By Springs. ........ 170
6.4 Correspondence Sets Aligned Using Their Own Local Centroids. .. 171
6.5 Illustration Of The Chosen Views For 2, 3 And 6 Views. ............... 178
6.6 2 Views: e, 50 Points, No Noise. ............................................... 180
6.7 6 Views: e, 200 Points, No Noise. ............................................. 181
6.8 6 Views: 200 Points, 0.5 Noise. .................................................. 182
6.9 18 Views: δθ, 200 Points, 0.5 Noise. ........................................... 182
6.10 Degenerate, 6 Views: e, 200 Points, No Noise. ......................... 183
6.11 Degenerate, 6 Views: e, 200 Points, 0.001 Noise. ..................... 184
6.12 Degenerate, 3, 6 And 18 Views: δθ, 200 Points, 0.001 Noise. ...... 184

7.1 Flowchart Overview Of The N View Surface Registration Scheme. . 190
7.2 Bunny Data Set. ........................................................................... 193
7.3 Beethoven, 2 View. ................................................................. 194
7.4 Beethoven 3 View, 3 Overlaps. ................................................... 196
7.5 Foot 3 View, Centroid. .............................................................. 197
7.6 Foot 3 View, Angle And Centroid. ............................................ 198
7.7 Foot 3 View, 3 Csets. ................................................................. 199
7.8 Foot 3 View, 3 Csets. Boundary Check Only. .......................... 199
7.9 Foot 3 View, Angle And Centroid, 2 Csets. Normal, Boundary And Distance Check. ......................................................... 200
7.10 Foot 3 View: 400 Points. ......................................................... 202
7.11 Foot 3 View: 800 Points. ......................................................... 203
7.12 Bunny 10 View: 100 Points. .................................................... 205
List of Figures

7.13 Bunny 10 View: 200 Points ................................... 206
7.14 Bunny 10 View: 400 Points ................................... 207

8.1 The ModelMaker Sensor ........................................ 214
8.2 A Simple Example Of Surface Refitting ...................... 216
8.3 An Example Of Decimation ....................................... 217
8.4 Illustration Of The Stripes And Data Points Acquired By Striper . 219
8.5 Sphere and Corner ............................................... 225
8.6 Gravy Dish .......................................................... 226
8.7 Toy Man ............................................................. 227
8.8 Fabric Softener Bottle Acquired From Poorly Calibrated Sensor . 229
8.9 Fabric Softener Bottle Acquired From Poorly Calibrated Sensor (cont). 230
8.10 Fabric Softener Bottle Acquired From Poorly Calibrated Sensor (cont). 231
List of Tables

5.1 Series 1: Basic vs Accelerated ICP For Foot ............................................. 136
5.2 Series 1: Effects Of Varying Neighbourhood Size For RanRegis Using Foot. .................................................. 138
5.3 Series 1: Effects Of Varying Voxel Size for VoxRegis Direct Lookup. 140
5.4 Series 2: Effects Of Varying The Number Of Points For Beethoven. 147
5.5 Series 2: Stability At Resolution 50x50. .................................................. 148
5.6 Series 2: Effects Of Varying The Number Of Points For Foot. 150
5.7 Series 4: Foot With 2% Noise Using MSE. ............................................ 161
5.8 Series 4: Foot With 0.5% Noise, MAD And MSE. ............................. 161

6.1 6 Views: 200 Points, No Noise. .................................................. 181
6.2 6 Views: 200 Points, 0.5 Noise. .................................................. 181
6.3 Degenerate, 6 Views: 200 Points, No Noise. ........................................... 183
6.4 Degenerate, 6 Views: 200 Points, 0.001 Noise. ........................................... 183

7.1 Beethoven, 2 View. .................................................. 195
7.2 Beethoven, 3 View. .................................................. 196
7.3 Foot: Affects Of Varying Point Set Size. ............................................. 204
7.4 Bunny: Affects Of Varying Point Set Size. ............................................. 208

8.1 Results From Sphere Refinement. ............................................. 223
8.2 Results From Corner Refinement. ............................................. 224
8.3 Results From Dish Refinement. ............................................. 226
8.4 Results From Bottle Refinement. ............................................. 232
Chapter 1

Introduction

The invention of the computer has revolutionised many aspects of our world. It has enabled the solution of problems that could not previously have been feasibly tackled. Computer modelling has resulted in benefits on many fronts - in terms of speed, accuracy and reliability. However, any computer simulation is only as good as both its input assumptions and its data, and it therefore remains an absolute necessity to get these correct. Computer models often involve geometric aspects of physical objects and therefore it is necessary to measure these accurately. This overall process of going from a physical world object to some model of it, is often termed "reverse engineering". This can be contrasted with the usual manufacturing process (perhaps called forward engineering) that goes from a computer representation or paper drawing to a physical object.

Reverse engineering is one application area of model building. Another area is industrial inspection which is where a manufactured component's geometric data is acquired and the reconstructed 3D model is compared against a 3D CAD model looking for defects. Depending on the complexity of the component being inspected, this process can either be fully or partially automated. Another application area is VR (virtual reality) where real world objects are reconstructed for VR environments such as museums or virtual worlds of heritage sites. An advantage of having computer models of historical objects is that it allows people from all over the world to see those objects at any time of day in a VR world via the Internet. It also allows
people to get physically close to those objects in a virtual world sense, to look at fine
detail which they might not be allowed to do even in a real museum, since it may be
a very fragile or valuable object and therefore is enclosed in a glass casing keeping a
person at a fixed distance from the object. Another application area is customised
clothing such as g-suits for pilots which reduce the gravitation forces acting upon a
pilot helping to prevent black outs. Hence, a custom fit is vital.

Until recently, geometric models of real world objects could only be built by using
a hand modelling process involving a computer aided design package or by the
use of a cumbersome and time consuming contact probe system. Over the last 20
years optical systems have been developed based on viewing objects with cameras.
Such systems can perform automatic geometric model building both accurately and
at great speed. However, they are often limited in their viewing angle and by
the fact that the cameras need to be accurately aligned. These limitations could
potentially be overcome by better algorithms for analysing and manipulating the
gathered geometric data.

This thesis is concerned with developing and evaluating advanced algorithms for
parts of the geometric model building problem. In particular it will consider the
problem of building large geometric models by combining many smaller patches of
surface data. The process of finding matching elements of two or more surfaces and
geoetrically aligning them is known as surface registration and is the main topic
of the thesis.

Surface registration is part of the model building process shown as a simple diagram
in figure 1.1. Several views of the real world object will be acquired by a sensor
of some kind, resulting typically in a cloud of points being obtained for each view. Traditionally, these views are taken two at a time and registered. This is done for all views. The registered views are then integrated, and from the fused data set, a surface mesh can be generated allowing the full 3D model to be rendered (displayed) in a visualisation package.

The iterative closest point (ICP) algorithm is the de facto standard algorithm used to perform registration, and a simple overview of the ICP algorithm is shown in figure 1.2. Given two surfaces, the moveable surface is sampled obtaining a point set. By establishing points on the fixed surface that are closest to the points within that point
set, approximate corresponding points between the two surfaces are determined. Using these two point sets, a transformation can be determined which aligns them. This transformation is then applied to the moveable point set. The correspondences (closest points) on the fixed surface are redetermined and a rigid transformation to align the two point sets is computed and applied. This process continues where the closest points in each iteration approach the true correspondences. This iterative process continues until the surfaces are considered registered.

The ICP algorithm performs reasonably well when the two surfaces being registered have large overlaps and when there is no noise present. However, those regions that do not overlap will cause a registration error since any matches established between those regions will cause the registration solution to diverge away from the true solution. Thus it becomes trapped in what is known as a local minima. The ICP algorithm is known to be susceptible to local minima which is more likely when a less favourable starting positions between the two surfaces are used. Robustness procedures will be introduced to reduce the local minima problem caused by incorrect correspondences. These robustness procedures will also be evaluated to see how the modified ICP algorithm copes with noisy data. The ICP algorithm has two main components. One for aligning point sets and another for determining correspondences. The choice of these components affects the performance of the ICP algorithm. When registering two surfaces the closest point component will likely affect the outcome of the solution obtained, and it is also known to be the most time consuming part of the ICP algorithm. Thus by using different closest point methods their affect on the performance of the modified ICP algorithm will be assessed. Throughout the literature surveyed, all schemes that use the ICP algorithm have a fine tuned version to use their closest point method, and thus are trapped into using one surface type only. In the modified ICP algorithm, the generality of method is to be embraced so that the implementation is flexible to allow different closest point methods to be used, while using the same core code. Only one alignment method will be used during this evaluation although it will be adapted to introduce robustness into the method. The modified ICP algorithm will then be extended to register multiple surfaces and its performance evaluated.
1.1 Background

Several methods exist for acquiring 3D data which describes the surface of an object. The data acquired will vary in richness depending on the method used. Each method tends to have an application area, and these methods will now be discussed.

One method of acquiring surface point information is by using a touch probe. A touch probe is a very high precision (1 micron) mechanical device with a pressure sensitive switch that records the instant when probe touches a surface. By placing the probe on a point of an object the position in space of the endpoint of the device is captured, and therefore the 3D position on the object is determined relative to the origin of the probe connected system. To build high accuracy surface models, the user needs good experience of using the system and the result may depend on factors such as the angle of approach of the probe to the surface. To build models hundreds or thousands of positions on an object need to be sensed and as each data point may take several seconds to acquire, the process is very time consuming and tedious.

There are several methods of acquiring data which are based around optical methods. In general these have the potential benefits of speed, accuracy, automation and acquisition of a dense field of data, although a single individual method may not have all these desirable features. There are two types of acquisition methods, passive and active. Passive methods are where natural energy is absorbed, like sonar where an operator just listens for another submarine, and active methods are where an artificial energy source is used, such as a ping to locate another submarine. An example of passive system in 3D is stereo vision where two different intensity images are acquired to obtain depth information. The problem with this method, is that correspondences are not easy to identify and only a few exist, meaning that the 3D data is sparse. Active methods such as time of flight, moire fringing and laser range finders help overcome these correspondence issues. Correspondences do not need to be determined in a subsequent step. A laser range finder can return a dense field of data. Time of flight techniques produce data which is not as dense. The time of flight, range finder and stereo systems are now discussed.
The laser range finder is one type of automated technique, known as range imaging. This is where each point on the surface of the object that is visible by the sensor is acquired. The way this works is that for each point on the object’s surface that is sensed, its distance from the object to the sensor is computed. The sensors tend to have a grid pattern, and thus the points of a surface are captured for example row by row, or column by column. The precise nature of how this grid or image is built is down to the specific sensor used. Therefore an image of distances (ranges) of points on the object’s surface to the sensor is constructed. Hence the name range imaging. Different techniques exist for measuring the distance from sensor to object. One category is time-of-flight. This is a where a pulse of energy sonic or optical is sent from a radar sensor to the object and depending on how long it takes to come back, determines how far the object is from the sensor. These tend to be used for long distances such as Ladar, laser radar, which can be used to build 3D maps of terrain from the air. Another technique uses structured light. These type of sensors work by using principles of triangulation. This range scanner consists of two parts, the light source and a 2D sensor, where the distance (baseline) between these parts remains fixed. The baseline and the angle of the light source with respect to baseline are determined by calibration, which is done by the manufacturer. The angle of the 2D sensor with respect to the baseline is then computed on the fly. This is done by looking along the scan-line of the laser stripe for a peak in intensity, from which the angle can then be computed. Thus a point of the surface is acquired. Using this type of setup means that the range scanner will have a working zone. Hence, if an object is too close or too far, distances of points of an object will not be able to be computed.

Traditionally only depth information was captured by range scanners. However, recently range scanners have been developed to capture the colour information of an object too. One such system was developed by Soucy et al. [102]. This system works by having three lasers, each one being a different colour. These being red, green and blue. This means that the object has to be sampled completely three times since each range and intensity image is captured separately. Once done, the three separate range and intensity images are then merged together using a semi-
1.1. Background

automated tool to form a single 3D RGB image. To be able to ascertain accurate
colour information for the lighting conditions present when scanning the object, the
colour range scanners need to be calibrated. This is done by using a white diffusing
target which is scanned at various depths in the working volume. During capture
a white surface is kept near the border of the scene. This allows the data captured
to be compared with the white standard, and by using a reflectance model and the
calibration data a final reflectance value is obtained.

A different method of acquiring 3D data is by using a technique known as stereo
vision. This technique typically uses two cameras with a fixed distance between
them. Each camera points at the object of interest, thus each camera sees a slightly
different view of the object. A partial 3D model is then constructed by determining
which parts of the image seen by one camera corresponds to the other image seen by
the second camera. Once correspondences have been determined, triangulation can
then be used to determine the depth values of the object. For this process to work,
the cameras have to calibrated, i.e. a fixed focal length, same orientation etc, and
remain the same distance apart. The difficulty tends to be in determining which
parts or pixels of an image correspond to other pixels present in the other image,
and is termed as the correspondence problem. Possible features which may have
correspondences can be selected by using a corner detector for example. Using a
correlation based stereo system, corresponding points are determined by comparing
small pixel neighbourhoods (windows) around the point of interest (possible cor­
responding point) in both images. These windows move around both images, in
attempt to find matching points. Once the correspondences have been determined,
a partial 3D surface is obtained of the object, which also includes intensity informa­
tion, which may also have colour information depending on the cameras used. This
method however will not produce a dense map of the surface structure compared
with a laser scanner. However, this method does have the advantage of being able
to capture large objects, such as buildings.

When acquiring surface information, only what the sensor can see of the object from
that view is obtained. Hence, multiple views containing surface information need to
be obtained so that the whole surface is fully described. When each view is acquired,
unless the precise position, direction and orientation of that view is known, being
able to fully construct a 3D surface model easily will not be possible. This is because
all transformations between the views need to be known so that they are perfectly
aligned, registered, prior to fusion. If approximate transformations are known, by
using an interactive software application, a user can adjust each view’s position so
that all views become aligned. However, it is likely that errors will be present in
their alignment. Assuming, that they are aligned, they can then be fused into one
surface, obtaining a full 3D model. If surfaces are not precisely aligned prior to fusion, then the resultant surface will
contain errors. Meaning that the surface model will not be an accurate representa­
tion of the real world object. In areas such as reverse engineering or industrial
inspection, accuracy is important. Thus these surfaces need to be as accurately
aligned as possible before being fused. The process of automatically aligning these
surfaces is known as surface registration.

Traditionally surface models are built using a pair wise approach. First, two over­
lapping views are taken and registered. These are then fused. Next, another view
is registered to the resultant surface produced by the fusion process, and then these
become fused. Then, one by one, the remaining views are processed with the regis­
ter and fuse approach, obtaining a full 3D model. Therefore, when registering two
surfaces, if the error in the transformation between two surfaces can be reduced as
far as possible, then a more accurate surface will result.

1.2 Aim Of Work

The aim of this research is to improve 3D model building by considering current de­
ficiencies in surface registration techniques by comparing how the methods perform
using different components as well as introducing methods to improve the robust­
ness of the method. A post processing self-calibration technique is also introduced
to improve the data obtained in the acquisition stage prior to the surface registra­
tion stage. Thereby improving the overall model obtained at the end of the model
building process.
1.2. Aim Of Work

In this thesis, the method of registering surfaces is based on the iterative closest point (ICP) algorithm by Besl and McKay [11]. This consists of two parts, a point set alignment method and a method for determining correspondences between points on the two surfaces. The crucial part of the ICP algorithm is the correspondence determination. Correspondences are achieved by applying an iterative scheme that assumes the closest point on the surface is an approximation to the true one. The true correspondences are obtained by using iteration of the closest point. Therefore if the effect of the choice of the method used to determine a closest point can be assessed with respect to the ICP algorithm, better surface registration may result.

The registration problem ought to be easy when no noise is present and correct correspondences are used. However, this is rarely the case. A typical problem is where mismatches occur, i.e. incorrect correspondences, which result in a registration error. Missing data is also a problem. This is where part of a surface from one view is used in the registration process that has no correspondences on the other view's surface, since it does not contain that data. Thus when surfaces are registered, only overlapping parts should be used. Another problem is spurious data that is contained within a view that was obtained in the acquisition stage. This data does not truly reflect the surface of the object, and is classified as outlier data. If this outlier data is used within the registration process, then results obtained are likely to be skewed from the one sought. Therefore robustness procedures are to be introduced, to help decide in how much confidence the registration method has in which parts of a surface correspond. Thereby possibly improving the reliability of registering two surfaces. The usefulness of these robustness procedures needs to be assessed to determine their value.

The problem with reconstructing models using the pairwise approach, is that errors will accumulate. If the first two views for example are inaccurately registered there will be an error present. Once these two surfaces become fused that error becomes fixed. Then if another surface is registered with an error and fused, and the process continues, the errors from each subsequent stage will propagate. This propagation of errors is undesirable when wanting to build surface models.
Chapter 1. Introduction

The pairwise approach also has another problem, that being that several views are all likely to overlap. Thus the optimal transforms for the views can only be found by registering all surfaces simultaneously. To work towards a multi-view approach, being able to align n point sets needs to be done first. Such methods exist, but how they perform with respect to each other and what advantages a method may have, needs to be obtained.

By using n view point set alignment methods, the ICP algorithm can be extended to register multiple surfaces simultaneously. When registering these surfaces, the affect of the robustness procedures will need to re-evaluated. Also methods of being able to determine which surfaces overlap will be required to improve the accuracy of registering the surfaces.

The problem of building surface models does not just lie solely with registration of surfaces. As previously mentioned, another source of error is at the acquisition stage. When using a range scanner system that knows at all times the position of the laser striper, a larger area of a surface of an object can be captured in one go, compared to a static range scanner. This type of system could use an articulated arm to determine the position of the striper. However, errors in the position of the articulated arm are likely to occur over time. Therefore, if a post processing technique can be developed to correct these errors, a better surface can be obtained. Hence if an object is complex, these multiple views can be corrected prior to registration further improving the accuracy of the reconstructed 3D model.

1.2.1 Achievements

The achievements of the work presented in this thesis to be able to build 3D geometric surface models is summarised below.

- A new comparative study of the modified ICP algorithm using three different closest point methods has been performed. A new closest point method called the range image search was presented allowing the registration of range images to be performed. Robustness procedures were also introduced, one of these being the novel boundary check. This study showed how the modified ICP
1.2. Aim Of Work

algorithm is affected by the choice of the closest point method used. It was also shown that the robustness methods affect the accuracy of the final answer, most notably the novel boundary check.

- A new comparative study of three published n view point set alignment methods was performed. The study looked at ease of implementation, convergence properties, speed and accuracy. Their robustness with respect to noise and degenerate cases were also assessed.

- A novel scheme to extend the ICP algorithm to the multi-view case was implemented allowing multiple surfaces to be registered simultaneously. A new comparative study was performed of the novel n view ICP algorithm using the three published n view point set alignment methods. The study examined how the n view point set alignment methods affected the reliability, accuracy and speed of the n view registration implementation. Therefore only one closest point method was used during this study. The closest point method used being the most reliable method that had been determined in the comparative 2 view study. The robustness procedures previously used in the 2 view case were re-evaluated in the multi-view case establishing that the normal check has a more vital role than it did in the 2 view case. The importance of a n view point set alignment method incorporating robustness was also established.

- A novel post processing technique was developed to correct errors in data obtained at the acquisition stage. The method was shown to significantly reduce errors in the surface produced from the acquisition stage. It was also seen that finer surface detail was brought out using this novel technique. The method was shown to cope well with data from a poorly calibrated system where initial error levels in the data were high and therefore demonstrating the robustness of the technique. This novel technique has the potential to allow good surface models to be ascertained with less accurate systems. Also better full 3D models are likely to be built using these post processed surfaces during the registration stage, as registration errors are likely to be reduced further prior to fusion.
1.3 Overview Of Thesis

In chapter 2 the literature in the field of model building, or more specifically registration is reviewed.

In chapter 3, 2 view point set alignment is covered. To be able to align 3D point sets in 3D space, certain transformations need to be applied to manipulate the data so that they become aligned. Therefore, relevant geometry is reviewed first. Alignment is done by using a least squares data fitting technique. Since alignment will likely be done in the presence of noise, outliers and missing data, methods to make least squares less susceptible to outliers present in data are covered. Finally, a 2 view point set alignment method is discussed where both geometry and regression are used to solve the transformation required to align two point sets.

Next in chapter 4, different methods of surface representation are reviewed. Then the problem of determining a closest point on a surface for a given a point is discussed. The strategies used by three such methods are discussed in detail.

Then in chapter 5, it is shown how a closest point method and a 2 view point set alignment method are used to register two rigid surfaces. An overview of the iterative closest point (ICP) algorithm is given, moving on to how error can be measured therefore helping in determining whether the registration is good. The problem of degeneracy and local minima that the ICP algorithm is susceptible to is discussed. Next, some additional checks that can be used to help reduce the likelihood of the ICP algorithm being trapped in a local minima are introduced. Followed by how prediction can be used to accelerate the process of registration. Finally, results are presented for the modified ICP algorithm showing the impact on its behaviour by using the three different closest point methods. Also the benefits of the additional robustness checks are presented.

The problems of only registering two views is then highlighted in chapter 6. Three methods for doing n view point set alignment are then reviewed. Results are then presented for a comparison of these three methods.

Next in chapter 7, it is shown how the modified ICP algorithm has been extended to
perform registration of multiple surfaces. The n view point set alignment methods presented in the previous chapter can then be used by this method. Results are then presented showing how this method performs showing the affect of strategies and robustness features used, as well as doing a comparison on the affect of the algorithm by choice of the n view point set alignment method.

In chapter 8, a post processing scheme is introduced which shows a strategy for recalibrating data acquired from a ModelMaker Reality Capture System. Results are then presented showing that better surface models are obtained using this technique.

Finally, in chapter 9 the conclusions are given about the modified ICP algorithm and the study in which it was involved. Then, the three n view point set alignment methods are discussed and how these behaved in the extended n view ICP scheme. Next, the conclusions are given about the self-calibration technique and finally the future work for this research is discussed.
Chapter 2

Literature Review

Registration is a key part of model building and can be split into two distinct stages. Firstly determining correspondences between two objects such as surfaces, and secondly determining a transformation that will align these correspondences. This chapter will mainly concentrate on registration of surfaces, but other issues which have relevance such as mesh optimisation will be discussed so that there is an awareness of such ideas and their terminology that may be referred to in later chapters. The literature reviewed in this chapter is up to 2002. However, the work presented in this thesis was done in the period of 1996-2000 and therefore some of these reviewed published works may contain ideas similar to those included in this thesis.

2.1 Registration Of Non-Surface Data

Registration is not restricted purely to 3D surfaces, it can also be used with 2D intensity images. Registration of this type is commonly referred to as mosaicing. In its simplest form, the images taken of a scene are approximately planar, such as aerial images [54] of a landscape or town. These images are aligned by finding the 2D translation and 2D rotation that are optimal to build a bigger more complete aerial photograph. Photogrammetry is the study of means of determining precise
geometry from sets of photographs. Usually correspondences are established manu-
ally. Therefore, determining good correspondences is vital for good registration, and
in Hsieh et al.'s [55] paper, a wavelet transform is used to extract feature points,
from which a small subset of good possible matches is selected. Poorer matches are
ignored. The method also benefits from being faster if a smaller number of matches
are used.

2.2 Non-Rigid Registration

Mosaicing is not just restricted to planar views. Mosaicing can be done to create
panoramic images [105] where a camera on a tripod is fully rotated, and these can
be mosaiced and viewed on a cylindrical viewing surface. This type of viewing has
already been taken advantage of by various websites allowing users to view rooms
inside buildings from a fixed view.

When a sequence of images are captured for example of painting on a non-planar
wall, manipulation of the images has to be done. Jaillon and Montanvert [60] cre­
ate a surface model for the wall and then estimate projection parameters allowing
the images to be flattened out. A multi-resolution mosaicing technique is used to
minimise visibility of the joins between the images.

Feldmar and Ayache [30] presented a technique to perform nonrigid surface regis­
tration. They use a nonrigid surface matching technique to allow surfaces which do
not come from the same object to be registered to the same class of object. To do
this, surfaces are deformed so that point to point correspondences used are brought
nearer. The application of this work is mainly medical where objects are flexible
body parts and/or organs. Thus the need for nonrigid matching. Due to the de­
formations used, this technique is unsuitable for the aim of the work presented in
this thesis. Other works of Feldmar et al. [31, 32] have medical applications and
involve 3D-2D projective registration, i.e. curve/surface to X-Ray/video. Solutions
based on the iterative closest point (ICP) algorithm [11] are used. In [33] Feldmar
et al. also present an algorithm for registration-intensity correction of 3D images
based on the ICP. The application of this method is for registering various slices of anatomy acquired by MRI (magnetic resonance imaging).

Thirion presented work [107] to extract reliable landmarks for registration and recognition. Two types of landmark were extracted from 3D images based on lines and their points of intersection. These points of intersection are used as feature points and a matching process is gone through to determine correspondences between the 3D images. Thirion [108] further developed this method to go on to fully automate registration of two 3D images.

Lavallée and Szeliski [71] use 3D MRI and 2D xray projection to estimate the precise position of a patient undergoing an operation. The 3D data is acquired before the operation while the 2D data is taken in the operating room just prior to the operation itself. Both the 3D and 2D images are segmented to obtain a reference anatomical structure. From these it is possible to extract contours, and then register the 2D and 3D structures, thereby obtaining the relative position of sensor and patient. A lot of work on 2D and 3D registration is currently being done in the medical world. The work of Lavallée and Szeliski is representative of many efforts.

2.3 Registration Of Two Geometric Surfaces

2.3.1 Genetic Algorithms

Genetic algorithms (GAs) are an optimisation technique, that solve problems by starting with multiple start points and opportunistically search through the solution space by combining partial solutions to improve some global criteria. More information on GAs can be found in Sonka et al. [101]. Ahmed, Yamany et al. produced a series of papers [1, 117, 116] presenting a 3D curve and surface registration method based on a genetic algorithm. The implementation is based on Besl and McKay's [11] iterative closest point (ICP) algorithm. Instead of using a standard point set alignment method such as Arun et al. [3] they used a genetic algorithm (GA) to minimise the cost function. Yamany et al.'s method used a new closest point method which determines correspondences between surfaces which they called
the grid closest point (GCP) transform. This method speeded up the determination of correspondences. The space that the surfaces occupied was divided into a voxel structure and the closest points inside each voxel are precomputed. However, this can produce inaccurate correspondences. The emphasis of the GCP/GA method was on speed. It was shown to be significantly faster, by a factor of 12, compared to their pure ICP implementation, while obtaining a similar level of accuracy.

2.3.2 Feature Matching

Bergevin et al. [8] register two range images with no a priori knowledge using a two stage approach. In the first stage, initial estimates are determined. First, the range images are triangulated and various resolutions are stored in a hierarchical triangulation surface structure. Initially only coarse feature matching is used. Each triangle in this structure has a token set which consists of its normal, centroid and connectivity (i.e. those triangles which are connected to it). Using this information a pair of adjacent triangles can be matched in two views if both pairs in each view have a similar distance between their centroids and a similar angular direction. Using this criterion reduces the potentially large number of matches to just a few. From each matching triangle pair, a rigid transformation is estimated to align the two views. This produces a list of possible transformations. By measuring the overlap in 2D (through time intensive projection), the best transformations can be determined. Finally in the second stage the most promising transformations are refined to produce the best one. One of the refinement techniques presented used the multi-resolution approach. Each of the transformations were refined during a tracking step, and the worst transformations were removed. This was repeated until a single best transformation remained. The method tends to be time consuming since both multiple transformation tracking and 2D projection is done at each iteration of the registration process.

Higuchi et al. [41] take a set of range images and build tessellated meshes where the nodes of each mesh are uniformly distributed on each surface. A measure of curvature for each node is computed using the nodes three neighbours. Each node
and its measure of curvature is mapped onto a node of a unit sphere known as the spherical attribute image. Thus each range image (tessellated mesh) has its own spherical attribute image. The range images are registered by matching the curvature of nodes between the spherical images. Rotation is determined by matching between the two spherical images. This is then used with the actual data points to determine the translation. To be able to do the matching between the spherical images, both tessellations of the two views need approximately the same density of nodes in order to compare them. Hence, it is likely that the density of nodes in one of the views will need to be changed. The method needs no initial estimates but it does require that each surface has no topological holes. The resolution of the tessellated mesh used for the registration limits the accuracy of the best transformation obtained.

Chen at al. [15, 16] use a constrained exhaustive search which incorporates the RANSAC (random sampling consensus) scheme to determine an estimation that will align the two range images. Pre-selected control points are used to restrict the search range, and thus the problem is treated as a partial-matching one. Initially reference points are selected and a subset are used as control points. There must be at least three control points. The point set registration method of Arun et al. [3] is used to determine the transformation between a set of corresponding points. The limitation of their approach is that it is not appropriate for dealing with data that is either too noisy, or where the overlaps between two range images is small.

Roth [88] registers two overlapping range images by using interest points that have been triangulated and subsequently matched. When acquiring a range image a corresponding intensity range image is also acquired. Each pixel of an intensity range image represents the amount of light returned by the reflected laser beam measured at the frequency of the laser only. Interest points are found in each intensity range image by determining where there is a large change in local intensity. This may occur at points such as corners, since there will be a gradient change in both directions. Good interest points are determined using this type of process along with a threshold. Therefore the threshold together with the texture of the object determines how many interest points are found. The associated 3D values in the range image of these
feature points are then triangulated. There can be a large number of triangles in each range image and thus an efficient pruning process is used to reduce the search. Each triangle has its edge lengths encoded in a string. These bit strings are sorted from largest to smallest, and those that have the same string are grouped together into a list. This allows triangles with the same bit string to be easily determined in both range images. The matching process between triangles is further pruned by using vertex compatibility tests, e.g. vertex normals should approximately be the same for a matched triangle. A geometric transformation is determined by going through each list of triangles of the same length, determining a match and computing the transform to align the matched triangles. This transform is then applied to all interest points in the first image and a measure of alignment is noted. This is then performed for others groups (lists) of triangles. The final transformation is the one associated with the best alignment. Roth states that the time taken for the matching step will be greater if there are many triangles of almost equal size. The method does not assume any prior knowledge about the transformation between the two range images, but does require a 20% to 30% overlap. However, the main limitation of this method, is that the object being scanned must have texture on it to allow feature points to be found.

Recently Sharp et al. [99] developed a fully automatic registration method based on a maximum likelihood framework to register view pairs that are partially overlapping. The method works by first segmenting the range images into different classes. This is achieved by first assigning a valid or missing state to each pixel in the range image. Using these two states with various rules, further refinements are made to the classification of the pixels within the range images. The registration procedure uses a two stage approach. The first stage uses a randomised search modelled on Masuda and Yokoya's [76] approach and Chen et al.'s [16] RANSAC approach. The second stage uses a Nelder-Mead simplex search [99] which allows optimisation to be done over many discontinuities. A multi-resolution approach is also used to speed up the method. Due to Sharp et al.'s classification approach, the method can deal with occlusions, noise, outliers and missing data. It can also register range images with as little overlap as 10%, and no initial guess is required. However, this method
2.3. Registration Of Two Geometric Surfaces

is restricted to range data and certain specific assumptions are made relating the
effect of shadows from the structured light source.

For speed reasons, feature matching techniques tend to use only a few features to
estimate transformations to align views. Using an exhaustive search method can be
very time consuming but can be reduced by performing partial matching. Limiting
the number of features used during registration increases the likelihood of errors in
the estimated alignment. Errors should be smaller if all of the available information
was used.

2.3.3 Closest Point Approaches

2.3.3.1 Point To Tangent Plane

Chen and Medioni [17] use an iterative process that minimises the distance of points
in one view to tangent planes of surfaces in another view. The method is used to
refine the initial approximate transformations for each view, which in Chen and
Medioni's case, is determined from a range finder setup. A set of control points are
selected for each view and their surface normals are computed. For each control
point, the intersection of the normal line with the surface being registered is found
and from this the tangent plane is computed. Registration is then done by finding
the transform that minimises the distances between the control points and tangent
planes. The problem with this method is that it must be given an initial approximate
transformation and the control points have to be selected by hand. Also, to do global
registration, a model is first built using a pairwise technique, and then each view is
re-registered to this new fused surface.

Gagnon et al. [36] take Chen and Medioni's method [17] and use a network of views
so that the registration error is similar between all views. A star network topology
is used, which is constructed such that the centre node is that view where the path
of nodes to the centre node is smallest. The centre node's view transformation
remains fixed, meaning that there are only $N - 1$ view transforms to determine.
The network of views is iterated over, with each view being transformed into the
reference frames of all other views for the tangent plane determination. Using the
distances from a view to all other views, a transformation is computed using a linear
least squares technique. Convergence is reached when the update transforms for each
view approaches the identity matrix, i.e hardly any movement has occurred.

Bergevin et al. [9] take a set of range images and refine the initial estimates obtained.
The refinement technique is based on Gagnon et al. [36] using a star network of views
creating a meta-view. However, a problem with [36] was that errors would accumu­
late. Since each view added to the meta-view is likely to bring extra information
that could be associated with previously registered views, Bergevin et al. re-register
all views in the meta-view every time a new view is added. This process continues
slowly until all the views become registered, and instead of errors accumulating,
they more evenly diffuse among all of the views.

Dorai et al. [23, 24] extended the method of Chen and Medioni [17] since it could
not produce optimal transforms in the presence of noise. Noise in the z values
affect the surface normals, and thus the estimation of the tangent planes. Thus
Dorai et al.'s method uses a minimum variance estimator and a weighting scheme
to model uncertainties in the z measurements, so that a more reliable estimate can
be obtained. They assume that the noise distribution is well behaved and so their
method is susceptible to outliers.

2.3.3.2 Iterative Closest Point (ICP) and Variants

The iterative closest point (ICP) algorithm by Besl and McKay [11] has been
widely adopted throughout the vision community. The difference between Besl
and McKay's algorithm and [17] is that corresponding points are determined on
the actual surface, and not tangent planes as in Chen and Medioni [17]. The ICP
algorithm is general insofar as it is not restricted to range images, or a surface of
a particular type. Hence it can be used to register different surface types. Like all
registration methods, correspondences need to be determined between the surfaces
to be registered. However, unlike feature matching methods, the ICP algorithm
re-determines these correspondences during each iteration.
Zhang [119] proposed a method to register two sets of 3D curves and surfaces using an iterative point matching technique. Like the ICP algorithm, samples of the surface are taken giving a point set, and again it is assumed that the approximate motion between two surfaces is known. Thus point matching for surface registration is done between two point sets, each representing one surface, and for curves Zhang uses chained points. Again like the ICP algorithm, a least squares technique is used to estimate the motion (transformation) between the two point sets. However, Zhang introduces a statistical technique based on a distance distribution to deal with outliers, and thus subset-subset matching is performed. Zhang’s method however needs a parameter to be set by the user which indicates when the registration is considered good, i.e. the average distance between the points. This parameter is also used to update a distance tolerance. Thus after one iteration, any pairings (correspondences) that are greater than that tolerance are removed. Therefore outliers are removed. During the subsequent iterations, the distance threshold will be reduced according to Zhang’s rules which are related to the mean. Therefore the main difference between Zhang’s method and ICP algorithm is the added robustness.

Masuda and Yokoya [76] presented a method that incorporated the ICP algorithm in a least median of squares (LMedS) approach which uses a random sampling technique. A set of points of a specified size are randomly extracted from one of the range images, which is then used by the ICP algorithm to determine the best rigid transformation. On this first trial this transformation is kept. On subsequent trials, other subsets of available points are extracted from the range image, and passed to the ICP algorithm which uses the best rigid transformation obtained so far, as its initial guess. At the end of each trial a median of squares evaluation is done, and if this new transform is better than the previous one, in the median of squares sense, then it is kept. This continues until the specified number of trials have been done, resulting in the best rigid transform to align the two range images. Segmentation of transformed range images is then done to determine inlier points between two successive range images. The motion (rigid transformation) between each range image pair is then re-estimated using the inlier points only using the LMedS ICP technique. From multiple successive range image views that have been segmented
and re-registered using the inliers, the inlier points are integrated with their new transforms to construct a data set representing the object. The main advantage of this technique is its robustness, and because of the LMedS technique being used it can cope up to 50% outliers. Further information on this method and how closest points are determined can be found in Masuda et al. [75].

Yang and Allen [118] presented two methods to estimate the initial transform between two views to perform registration. The first method estimates the initial transforms by extracting and matching 3D space curves from the different views. Thus points with a high gradient which indicate that they belong to edges or contours of an object are used. In Yang and Allen’s case they only used 15 points per curve. These descriptors are stable with respect to rigid transformations and tolerate partial occlusion due to their local nature, which is a very important attribute. Thus to obtain the initial transformation, curve matching is done, from which a transformation can be estimated. A modified ICP algorithm may then be applied to improve the estimate. The problem with using curves as a salient feature, is that it is object dependent, and thus the accuracy can be affected, since there will be a lack of salient features or none. For such objects Yang and Allen place long thin pins asymmetrically distributed around the object in such a way that a pin can be seen from many viewpoints. From the range images, the pins can be detected by an edge detector. The pins between images are then uniquely matched using their asymmetrical distance measures. The initial estimates are then determined using the above space curve method, and then further improved using the modified ICP algorithm. Yang and Allen’s modified ICP algorithm uses curvature as well as distance to determine good correspondences. This weighted distance function based on the surface curvature is used throughout the ICP run, and a distance threshold is reduced during the run to avoid oscillation that may occur around the convergence point. Using these methods good initial estimates are not required, and fairly accurate initial estimates can be obtained using curvature as a salient feature. The final fit can be improved using a modified ICP algorithm. However both methods including their modified ICP algorithm rely on curvature and this is susceptible to noise.
A recent paper by Rusinkiewicz and Levoy [90] presented a comparative study of convergence performance between ICP variants which focused on the speed of convergence. They introduced a new sampling method which uniformly samples the space of normals. By using this uniform sampling of normals, there will tend to be higher clusters of samples in areas of the scan where there are finer details. This allows better alignment between two surfaces, since the use of these finer details will mean that it is less prone to degeneracy especially if the surface contains large planar regions which have such fine details. However, Rusinkiewicz and Levoy chose random sampling for their fast ICP variant, and used a constant-time closest point method for determining correspondences. Any pairings that lie on mesh boundaries (Turk and Levoy [109]) were ignored. Their method is fast, providing a good initial guess is supplied.

Sharp et al. [95, 97] introduced an ICP variant that used Euclidean invariants for the selection of point correspondences. They call their method ICPIF (ICP using Invariant Features). Using a closest point approach in determining correspondences gives a good initial estimate when the range images are approximately aligned, but does not so when otherwise. Therefore Sharp et al. use a shape descriptor such as volume or principle curvature that is view independent and thus invariant under a linear transformation. The Euclidean invariants used by Sharp et al. were moment invariants for which further information can be found in Sadjadi and Hall [91]. These invariant features improve the likelihood of making correct correspondences and thus accelerates the registration process, as well as reducing the probability of the method converging to a local minimum. However, the moments are susceptible to noise, and so Sharp et al. compensated by using a two stage algorithm. They used ICPIF to get near the goal quickly, and as soon as the error fell below a certain threshold, they switched to classic ICP to improve the final fit.

Although the ICP algorithm is iterative, and can be used to refine transformations of surfaces that are coarsely aligned, it can also be used to register surfaces that are slightly more distant from one another. By introducing robustness techniques such as weightings it can cope better with noisy data, and is less likely to become stuck in a local minima. There are variants of the ICP algorithm or methods that incorporate
Chapter 2. Literature Review

it in some way, and each of these variants tend to work with one surface type only. This ignores its generality, and restricts these variants to a specific surface type. A more general approach would enable registration of surfaces of different types using the same implemented method. All that would be required, would be a small piece of plugin software that deals with a different surface type.

2.3.4 Spin Images

Johnson and Hebert [61, 63] proposed a method that registers two surfaces using oriented point matching via a representation called spin images. From the range data, triangular meshes are generated and point correspondences established using spin images. A spin image is a 2D image which describes the shape of a surface independent of its pose. All points on the fixed view have spin images generated and put into a stack. A tenth of the moveable view is sampled, and spin images are generated for each of these points. Then each of these spin images are compared with the fixed spin image stack, via an image correlation technique typically used in image based matching. This allows correspondences to be found and by using a similarity measure and geometric consistency checks (such as a distance threshold) the matches can be filtered to obtain a good set of matches. Using these point correspondences, a rigid transformation is computed to align the views. This transform is then given to the ICP algorithm and refined by it. The correspondences generated in the spin image stage are used and additional ones are created to have an even spread across the surface mesh. These additional correspondences are generated from the existing correspondences using a distance threshold so that regions are not used where there is no surface overlap. Thus by using a larger number of correspondences a more accurate final transformation is computed using the ICP algorithm. This spin image approach enables meshes of different resolutions to be registered.

2.3.5 Reverse Calibration

Reverse calibration techniques such as the one by Chung et al. [18] rely on using a well calibrated experimental setup to solve for correspondences. If one does not
have access to such a setup, then a method like this cannot be used. During the calibration stage camera and projector parameters are determined. By using the reverse of the equations used in the calibration stage a set of equations is obtained allowing a corresponding point in 3D space to be determined (predicted) from a point in a range image. Chung et al. [18] have a two stage process. In the first step, initial estimates for the transformations between the two range images are determined using the principal axes of each range image. In the second step, the estimated view transformations are refined. Correspondences are determined by using the reverse calibration technique which given one range image point, gives another point in 3D space on the other views' transformed space. The transformations are solved for using a distance minimisation technique and process iterates until convergence which is when the registration error is less than the limit of the acquisition error of the range finder. The advantage of the reverse calibration technique is that no iteration is required to obtain a corresponding point, i.e. direct lookup.

### 2.3.6 Planar Segmentation

Gregor and Whitaker [39] presented an approach which used a time of flight range scanner to reconstruct indoor scenes. The method works by using an image segmentation and planar model fitting preprocessing stage. Once this is done, the user selects which planar patches between views correspond. These are then registered by using a nonlinear least squares method for plane fitting. The application for this particular method is indoor scene reconstruction which relies on objects within the scene having planar regions. The use of this method to construct models of individual objects is inappropriate since it is unlikely that all the real world objects will be planar. Also if such a planar model was to be reconstructed, user intervention would be needed to determine the correspondences of the planar patches.

### 2.3.7 Registration Aids

Pito [82] presented a method of building models by use of a non rotationally symmetric registration aid. By using this aid, range data that did not overlap could
be registered. This was achieved by registering the aid to a CAD model of the aid, thereby obtaining a rigid transform for each scan putting all scans into a common reference frame. The object to be scanned was placed on top of the registration aid which itself sat on a computer controlled turntable. In this particular setup, the scanner was rotated about a fixed line acquiring vertical scans of the surface, and therefore was restricted to one degree of freedom. Also the aid occupied 1/3 or the viewing volume, and for each scan the aid was segmented out. The aid was then transformed by the estimated rotation of the turntable and then registered with the CAD model of the aid. This pre-registration transform was then applied to the range data of the object. Subsequently these range images were then registered to the model of the object that was being built to improve the transformation further for each view using a weighted ICP algorithm. Although the aid improves registration of objects which are difficult to scan, or have large planar regions causing degeneracy, the size of the object to be scanned is ultimately limited by the size of the registration aid, and the complexity of the scanner setup.

2.3.8 Fuzzy Clustering

Tarel and Boujemaa [106] present a surface registration method that exploits fuzzy clustering. The method focuses on robustness with respect to noisy data, outliers, and initialisation, i.e. the starting point between the two surfaces. Their method uses a three step approach. In the first step, rough 3D matching is done. In Tarel and Boujemaa's case, they used planar patches as features. Rigid transformations between these features can then be computed between two subsets of the features extracted from both surfaces. The similarity of two features is determined using a Euclidean transformation since the features will be invariant to such a transformation, and a confidence value can be established between matches. This also allows outliers to be detected since they will consistently give bad matches to various rigid transformations. From this first step, a set of rigid transforms is obtained which each have an associated confidence value. Having these transforms in 6D space (three angles and three translation components), creates cluster points. Using this working space and the confidence values, a fuzzy clustering method is used to min-
2.4. Registration Of Multiple Geometric Surfaces

imise a cost function. For more detail refer to Tarel and Boujemaa [106] and the references within. Several hypotheses are obtained, giving a coarse location of the transformation required to register the two surfaces. In the third stage this coarse transformation is refined using the ICP algorithm. The main problem with this method, is that careful choice of the type of features to be used during the first step of matching is important, otherwise a combinatorial explosion is likely to occur, resulting in high computational cost.

2.4 Registration Of Multiple Geometric Surfaces

2.4.1 Surface Matching

Huber [57, 56] uses a surface matching system to exhaustively register all pairs of views. Both overlaps and the relative poses are unknown. The method consists of two stages, surface matching and model construction. During the surface matching stage, through the use of spin images [64] unconstrained pair-wise registration on all pairs of views is performed. The transformations are then refined by the ICP algorithm. Local consistency tests such as overlap distance are applied to remove incorrect matches. In the model construction stage, a globally consistent graph model is built incrementally and incorrect matches at the global level which were identified at the local level are removed. Once a globally consistent model has been determined, multi-view registration is performed using Neugebauer's method [79] which is provided with view correspondences that enable it to find the optimal absolute transformations. Due to the combinatorial approach of registering all view pairs in the matching stage, this method is not feasible for large scenes which contain many views, and thus view pairs must be selectively registered in the matching stage.
2.4.2 Closest Point Approaches

2.4.2.1 Point To Tangent Plane

Dorai et al. [22, 21] presented a model building system where their registration technique [23] had been incorporated and extended. Their method requires some preprocessing to their images to remove isolated pixels and noise through median filtering. When determining correspondences using Chen and Medioni’s method [17], some of the correspondences could be incorrect due to the surface being noisy or rough. Thus Dorai et al. imposed a distance constraint, where a given pair of control points and their corresponding image points, must have the same approximate difference between them to be considered as valid correspondences. Thus control points have to be grouped as pairs, and the possibility exists that depending on how they are grouped, some pairs might always be invalid meaning that useful correspondence information may be lost. Also, the use of median filtering for noise removal means that some of the surface detail may be lost.

The use of star networks in extending the surface orientation techniques to multi-view registration, still assumes that there are correspondences in all other views. Unless the views highly overlap, the solution obtained will likely be sub-optimal. Even in ideal conditions the star network only distributes errors evenly between views.

Neugebauer [79] presented a point based multi-resolution technique which initially requires some user interaction. Range images are considered one by one and at least 3 corresponding points are manually selected between each pair. Each new image selected has to have matching corresponding points selected in the previous image. Rough transformation estimates are then made between the range images. The second stage of the registration process treats the problem as a global one, and registers the range images simultaneously. The rough estimates from the previous stage are given to the algorithm. All the transformations between each range image are defined in a common world coordinate system. Matching distance constraints between range images are solved globally using a piecewise tangential plane
2.4. Registration Of Multiple Geometric Surfaces

approach. The cost function is solved iteratively where the initially estimated transforms have correction vectors applied until the minima is found. During the second stage registration step a multi-resolution approach is used, starting with a coarse representation. Once the minima is found, the resolution is doubled and again the cost function solved. Then the resolution is doubled again and this continues until full resolution is reached. During each resolution increase, outliers are ignored by using the distance measure between correspondences. Those that are greater than three times the standard deviation obtained from the previous resolution are ignored. After registration, the method continues to fuse the data to build a full 3D model using a visibility criterion to eliminate rogue points. The main drawback with this method is its reliance on initial correspondences between the range images being determined manually.

2.4.2.2 Iterative Closest Point (ICP) And Variants

Eggert et al. [26, 27] presented an improved version of the ICP using force based optimisation that uses range images. This approach is based on physics-based analogy where imaginary springs connect the corresponding point sets. This is similar to Stoddart and Hilton [103]. The motions for each view is determined by the cumulative effect of these spring forces between views. The movement for each corresponding point set between views is computed individually. Totalling up these forces gives the final motion for each view and these are applied simultaneously. Eggert et al. also use a multi-resolution approach increasing the resolution of the views until they reach convergence.

Benjemaa and Schmitt [4, 7] do registration of multiple sampled surfaces using a multi-z-buffer technique. The requirement of this method is that the surfaces are coarsely registered. The overlapping parts of the surfaces are segmented into approximate flat regions prior to registration and the amount of time required to do this is not stated. These regions are then stored in z-buffers which are used for point-to-point determination of correspondences. The multi-view registration approach is similar to Bergevin et al. [9] except that they do not wait until the
next round to apply all the transformations calculated. Thus as soon as a transform has been calculated it is applied. The z-buffer technique will not work properly if the surfaces are not highly uniformly sampled, and thus registration is not always possible.

Pulli [86] presented a method that used a two stage approach based both on the ICP and tangent plane methods. The first stage which is performed once, registers range scans in a pairwise approach using a tangent plane method. Pairwise registration is done between every view and each of its neighbouring views just once. The transformations ascertained at this stage are used in the global stage as constraints, in order that the pairwise registration errors can be diffused evenly. In the second stage the general goal is to move each scan relative to its neighbours as little as possible. To do this, the overlapping regions between views need to be kept well aligned. By just using the overlapping regions which have been uniformly subsampled in the multi-view stage, large data sets can be registered without the need to load all data sets into memory. The order in which these constraints are enforced is important. Thus a view that has the most overlaps with other views is processed first, and its associated connections are registered iteratively using an approach similar to Bergevin et al. [9]. Once the minima is reached, the next remaining view with most connections is selected and the process continues. By adding views one at a time, a local minimum is less likely to occur due to a bad initial alignment. If errors are detected in the alignment of a view(s), good overlaps can be given a higher weighting either explicitly or by having more samples for those overlapping views that there is high confidence. The process can be reiterated. This method relies on user interaction at the beginning, where views are selected as pairs for the first stage. For large data sets this can be very time consuming and susceptible to human error.

2.4.3 Contours

Kamgar-Parsi et al. [67] developed a method based on contours of constant range to allow them to correctly register range maps of the ocean floor. Their method works
2.4. Registration Of Multiple Geometric Surfaces

in two stages. The first stage is local registration, where contours are extracted and pairwise registered from the range images which are assumed to share a common reference surface. Matching contours are then determined between pairs of range images, yielding corresponding points. These are then aligned using a closed form least squares method, so that the two range images become locally registered. For the second stage, global matching is used to enforce consistency among the local matches, thereby reducing errors in the local matches. This global registration is done by basing the problem to be solved on a mechanical system involving springs whose strength indicates the confidence between matches. The minima corresponds to the equilibrium state of the system. This spring based approach was also used by Eggert et al. [26] and Stoddart and Hilton [103]. Using a contour based approach can cause problems. Firstly, if the surface does not vary much, there may not be enough contours to allow successful registration. Also contour matching may break down when noise is present in the range images.

2.4.4 Multiple Error Modelling

Recently, Williams and Bennamoun [112] introduced a method to register multiple surfaces where various error models could be used. The approach uses a modified ICP algorithm to perform the registration of surfaces. A coarse to fine multi-resolution scheme is used to further accelerate the convergence. Their method builds upon their recent point set alignment work [115, 114] specifically [113] which allows multiple point sets using different error models to be aligned. This allows multiple surfaces to be registered, where each point correspondence has its own associated error model. Thus points from one surface are not associated with only one error model. The error models used in their work were estimated from resolution parameters of their 3D sensors. By using these various error models, statistically optimal registration parameters can be computed. However, to be able to obtain optimal results, error models appropriate to the method of acquisition have to be built.
2.4.5 Error Distribution via Graphs Across Frame Space

Unlike other methods which only distribute errors to neighbouring views, Sharp et al. [96] presented a method to distribute these proportionally to the total error present in the graph across all the transformations (view frames). The errors are measured in relative terms of rotation and translation between coordinate frames. The method does not use an expensive cost function, and therefore can perform a fast, globally consistent multi-view registration over 100s of views. This means that it is practical for situations where the starting point is not close to the desired solution. The graph represents the topology of how the views relate to each other. If the views were captured via a turntable then the graph would have a ring topology. However, in this particular method, the graph structure has already been determined before applying their method, through the use of prior knowledge, e.g. operator assistance. Also this method works with only one cycle, i.e. one closed path in a graph. This global error distribution method could be used within a pair wise registration framework. Building upon this work, Sharp et al. [98] extended the method to deal with more than one cycle. Given a graph, the method would break it down into basis cycles and solve each of these cycles. Once solved, the solutions could be integrated together across the graph using an averaging technique. Again the method relies on being given a graph topology. Also it does not support weight error distribution or outlier detection.

2.5 Luminance, Colour And Texture Models

The work in this thesis concentrates on building 3D models from geometric data only. In virtual reality environments and games, having 3D objects just described geometrically is not enough to bring them to life. Therefore other methods exist to build models which include colour and texture information. These fall into two categories: hardware and registration.

To build such a model using a mainly hardware approach, Godin et al. [38] take advantage of range sensors that record intensity of even colour information along with
range information to constrain the registration of range images. To be able to use the intensity information a processing step is gone through to convert this information from viewpoint dependent to viewpoint independent. This is achieved by using a reflectance model to remove the specular components of the image, leaving only the diffuse reflectance properties of the surface. Potential matches between the pair of range images are constrained by the use of the intensity information along with a similarity measure to determine whether the match is reasonable. Thus to determine a match for each point, those points in the other range image that have similar intensity are determined, and from those, the one with the smallest Euclidean distance is chosen as its closest compatible point. To improve the registration further, a distance threshold for whether points are compatible is used when convergence is close. This means that the method must be run twice. First to get it to converge, and secondly running it again with the distance threshold switched on. Using this type of method, means that objects such as spheres can be registered providing that they are coloured in such a way, to allow solely the intensity information to be used when registering, since using geometric information by itself would cause the registration to be degenerate. However, if for such an object, large portions of overlap are of similar colour, degeneracy will still be likely.

Pulli and Shapiro [85] use a stereo camera setup (4 cameras) along with active lighting (slide projector) which sits on a computer controlled turntable. This allows the object's surface geometry and its colour information to be obtained. The range data is acquired in the dark. The colour information is captured from the lit scene, with the advantage that the correspondence problem between the different camera views has been solved using the vertical light strip. This allows a textured range map to be built. The range maps are then registered by aligning both colour and geometry where the scans overlap.

Johnson et al. [65, 66] use a registration approach based on the iterative closest point algorithm which they refer to as the colour ICP algorithm. Johnson and Kang acquire the data using a stereo system. By using colour in the registration process the registration error is shown to decrease, since additional colour information is being used in the determination of correspondences.
Weik [111] also uses a stereo system but determines corresponding points by using an optical flow technique. Lensch et al. [73] use a different type of approach by aligning a 2D image to a 3D model using a silhouette technique. Schütz et al. [94] also presented a multi-feature ICP matching algorithm that used colour and surface orientation information.

2.6 Mesh Optimisation

Mesh optimisation allows better meshes to be obtained from an original mesh which meet some optimisation criterion. Hoppe et al. [48, 46] present an approach to allow such optimisation to be done. Given a set of unorganised points (which may have been sampled from the surface) and that are close to the surface that is to be optimised, they can be used together to produce a better surface mesh. This refinement allows finer detail to show, and also makes sure that edges are crisp. The method of Hoppe et al. [46] also allowed decimation to be performed. Decimation is where the number of triangles within the mesh representing the surface could be reduced with no visible alterations, thereby simplifying the mesh. However, by applying too much decimation the mesh would no longer resemble the original surface. Further work on controlling mesh resolution can be found in Johnson and Hebert [62], Kobbelt et al. [69] and in Schroeder et al. [93]. Sharper features of surfaces can be brought out by using a later method of Hoppe et al. [47].

With today's range scanner technologies, when models are built then tend to have very high triangulation resulting in very big files which are slow to download. Therefore, it is sometimes desirable to trade off that complexity while still keeping a surface model that resembles the original object well. Using Eck et al.'s [25] or Kobbelt [70] multi-resolution approach, this becomes possible. In rendering applications, such as games or a virtual reality environment, reducing complexity of meshes allows smoother rendering performance, with no sudden jumps in frame rate. Hoppe [49] further developed this with progressive meshes. This allows a less detailed model to be rendered in the background, and as the object of interest comes closer to the foreground, the model representation can be switched to a more detailed one. This
idea of a level-of-detail (LOD) representation to have progressive meshes can be found in [50, 51, 83]. More detail on LOD and multi-resolution can also be found in Puppo and Scopigno [87].
Chapter 3

2-View Point Set Alignment

The first stage of building surface models is data acquisition. The data consists of distances to the visible surface of the object of interest from the viewing direction of the acquisition device and is known as range data. A range image is a 2D array with each element containing distances. The spatial sampling distance between each column of the array is usually the same. This is also true for the sampling distance between each row, although the distances between each row and each column do not have to be the same. The closer each element in the array is, i.e. the smaller the distance between the rows and columns, the higher the resolution of the range image since it contains more samples of the scene. Lower resolutions mean that a coarser representation of a particular view of an object is obtained. The data gathered is relative to the scanner's viewing direction and contains information only on unoccluded surfaces, and therefore is view dependent. Thus range image data is often referred to as 2\frac{1}{2}D data. In the overall scheme of surface registration, 2\frac{1}{2}D (partial) surfaces are used to reconstruct the whole object. Once the object is constructed, the data is classified as 3D data since it now contains information about all object points, not just those visible from a particular viewpoint.

Before the process of registering surfaces is considered, the process of aligning point sets will be discussed. This chapter will introduce the the mathematics necessary to understand how points can be manipulated by using various transformations. The process of aligning two point sets involves minimising a measure of mismatch between
the two sets. This is a statistical estimation problem. As with many problems in image analysis, the estimation method must be tolerant of typical image data set characteristics, i.e. missing data, noisy data and additional data. The statistical method of data fitting known as least squares is the most commonly adopted criteria for measuring mismatch and will therefore be described. It will then be shown how it can be modified to make it robust to the common image analysis. Finally, an algorithm for 2 point set alignment based on the work of Arun et al. [3] but incorporating robust estimation will be described.

3.1 Geometry

Geometry is a branch of mathematics that deals with geometrical objects (e.g. a straight line), their measurement (e.g. length) and their relationships (e.g. parallelism).

3.1.1 2D Geometry, Shapes And Transforms

The simplest geometric object is a point. A point is a mathematical abstraction that has an infinitesimally small size and is fully described by its position in space by 2 coordinates for a 2D point and 3 coordinates for a 3D point. By using two points we can define a direction and the next geometric object, a straight line extending to infinity. All points on a straight line are said to be collinear. If we just consider

![Figure 3.1: Simple geometrical objects.](image)
3.1. Geometry

a small portion of a line, then it is referred to as a line segment. Simple shapes can be built from straight line segments. The simplest is a triangle which is built up of three segments, using three non-collinear points \( A, B, C \), which are known as the triangle's vertices. Figure 3.1 illustrates these geometrical objects.

Geometrical objects can be manipulated by applying transformations to them. The simplest transformation is translation. Translation allows a geometrical object to be moved in a direction relative to each axis. Thus a geometrical object can be made to only move in one direction relative to one axis, e.g. the \( x \) axis, or to both axes for a 2D object. To translate an object, a transformation \( T \) must be applied to every point of the object. Translation of a point in 2D space is defined as

\[
\begin{pmatrix}
  x' \\
  y'
\end{pmatrix} = \begin{pmatrix}
  x + a \\
  y + b
\end{pmatrix}.
\]

(3.1)

Once an object has been translated by \( T \) it can be moved back to its original position by applying the inverse translation \( T^{-1} \).

Transformations such as translation which do not change the geometric relationships between points of a geometric object are said to be rigid body transformations. The properties which are not altered by such a transformation are length, parallelism and angle. Two lines are said to be parallel if they never intersect each other and angle is a measure of orientation between lines which is discussed in more detail later in section 3.1.2.

Affine transformations preserve parallelism but not necessarily length and angle. Such an example is the scale transformation in which the object is made uniformly either smaller or larger. Distances change between the vertices of an object but the topological (connectivity) properties of the object remains the same. Another variety of the scale transformation is shear where instead of scaling everything up equally, just one component e.g. the \( x \) component, is scaled. Thus the distance and angle relationships of the object change.

The work in this thesis will be using rigid body transformations since we do not want the overall appearance of our models to be altered, although at times it may
be desirable to use decimation which alters the underlying structure of the surface, see chapter 8.

### 3.1.2 Rotation

As mentioned earlier, an angle is a measurement of orientation between lines, i.e. a measure of their divergence. Rotation $R$ is a circular transformation around a given point or axis, and is a rigid body transformation. The amount that a point rotates is specified by the angle given to the transformation. The positive direction of rotation is that of an anti-clockwise motion.

Rotation $R$ and translation $T$ are both linear transformations (mappings). This simply means that an object can be transformed to another position in a series of steps, or by combining those steps by simple addition and doing it in one go. For example, if four separate rotations of 90° were done a full rotation would be the end result, thus a rotation of 360° could have been done instead of 90° + 90° + 90° + 90°. More formally, the concept of these linear transformations are shown in equations (3.2) and (3.3).

\[
R(\alpha) + R(\beta) = R(\alpha + \beta) \tag{3.2}
\]

\[
T(\alpha) + T(\beta) = T(\alpha + \beta) \tag{3.3}
\]

To perform translation the transformation involves simple addition of coordinates, but what about rotation? We know that rotation does not change distance. So if a point $p$ is rotated fully around the origin, it will travel along the circumference of a circle with radius $r$ which is the distance from the origin to the point $p$. Thus by dropping a perpendicular line from $p$ to the $x$ axis, see figure 3.2(a), a right angled triangle is formed and thus simple trigonometry can be used to represent the points position, as shown in equations (3.4) and (3.5).

\[
x = r \cos \phi \tag{3.4}
\]

\[
y = r \sin \phi \tag{3.5}
\]
3.1. Geometry

Figure 3.2: Simple representation of rotation. (a) Defining a point’s position using trigonometry. (b) Rotating the point by $\theta$ to define rotation.

If we wanted to rotate $p$ to $p'$ by $\theta$, see figure 3.2(b), then $\phi$ and $\theta$ are added together. Thus

$$x' = r \cos(\phi + \theta)$$  \hspace{1cm} (3.6)
$$y' = r \sin(\phi + \theta)$$  \hspace{1cm} (3.7)

Trigonometry identities exist which state that $\cos 2\theta = \cos^2 \theta - \sin^2 \theta$ and $\sin 2\theta = 2 \sin \theta \cos \theta$. These can be expanded to $\cos(\phi + \theta) = \cos \phi \cos \theta - \sin \phi \sin \theta$ and $\sin(\phi + \theta) = \sin \phi \cos \theta + \cos \phi \sin \theta$ which can be substituted into the equations (3.6) and (3.7) to obtain equations (3.8) and (3.9).

$$x' = r \cos \phi \cos \theta - r \sin \phi \sin \theta$$  \hspace{1cm} (3.8)
$$y' = r \sin \phi \cos \theta + r \cos \phi \sin \theta$$  \hspace{1cm} (3.9)

By using equations (3.4) and (3.5) and substituting them into equations (3.8) and (3.9), equations (3.10) and (3.11) are obtained which can be used to rotate a point.

$$x' = x \cos \theta - y \sin \theta$$  \hspace{1cm} (3.10)
$$y' = x \sin \theta + y \cos \theta$$  \hspace{1cm} (3.11)
3.1.3 3 Dimensional Geometry And Object Models

So far illustrations in previous figures have been restricted to 2 dimensions (2D) for simplicity and clarity. However, transformations can be extended to work in n-dimensions. If real world objects need to be represented then obviously a 3D coordinate system is required. There are several ways to represent and construct computer models of 3D objects. One of the simplest is to describe the surfaces of that object and within this class of models, polygonal models are most often used because of their simplicity and consequent computational attractiveness. Polygonal models are made up of vertices connected by a series of lines making a surface (a series of connected planes or just an individual plane). From a theoretical point of view, a plane is simply a flat surface which extends to infinity in all directions. However, in computer models only a finite part of each plane can be represented. Three points are required to define a plane. Thus to build a model of a cube there will be a vertex for each of the 8 corners and each face of the cube can either be made of one square or two triangles. Whether the cube is made up of squares or triangles there will be four vertices on the same plane (face). When viewing surface models using a computer, the models are generally built up of triangular faces. To visualise them as a 3D solid object, they need to be rendered (coloured). The rendering process uses lighting models to make the objects more realistic and to do this the surface normal needs to be calculated which is done by using the cross product\(^1\). For consistency, models need to be constructed carefully making sure that the triangular facets have their vertices specified in a consistent order. If this is not done, then the surface normal which denotes the direction of the object exterior will be pointing in the wrong direction and will be rendered incorrectly.

In a 3D world there are two choices for Cartesian coordinate systems - a right handed or left handed system. In the right-handed system, the z-axis points towards the viewer whereas in the left-handed system, the z-axis points away from the viewer. The left handed system is more intuitive since as an object recedes from the ob-

\(^1\)In 3D space the cross product requires two 3D real vectors. The resultant product is a 3D vector which is perpendicular to the plane formed from the given two vectors, and is known as the surface normal. The cross product can be generalised to n dimensions.
3.1. Geometry

Figure 3.3: (a) Right-handed coordinate system. (b) Left-handed coordinate system.

server the z value increases. However, the right-handed system is the standard mathematical convention used. The two coordinate systems are illustrated in figure 3.3. Whatever type of coordinate system is used, rotation can be employed to manipulate geometric objects.

3.1.4 Encapsulated Rigid Body Transforms

Surface registration is the problem of determining the geometric transformation that will map one surface directly onto another, thereby bringing corresponding points into an alignment. To simplify the complexity of equations, the transformations of rotation $R$ and of translation $T$ can be encapsulated in a unified notation used by Pennec [80]. This general encapsulated rigid body transform which is a more compact notation is shown in equation (3.12).

$$ g = \begin{pmatrix} R \\ T \end{pmatrix} = (R,T) $$

(3.12)

The general equation to describe the relationship between a point $p$ and its corresponding point $p'$ is $p' = Rp + T$. Here we introduce the new notation as $p' = g \ast p$ where $g = (R,T)$ and the $\ast$ operator is defined such that $g \ast p = Rp + T$. 
Composition describes the rules that hold when a set of transformations are applied in sequence. To perform composition of $g_1 = (R^1, T^1)$ and $g_2 = (R^2, T^2)$, the operator $\circ$ is used. Thus, $g = g_2 \circ g_1 = (R_2R_1, R_2T_1 + T_2)$, i.e. the combined rotation is the product of the constituent rotation and the combined translation is equivalent to adding the second translation to a vector that has undergone the first translation and the second rotation.

### 3.1.5 Different Methods of Representing Rotation

When using rotation which involves more than one axis, various representations are available. Each representation has its own advantages and disadvantages, and these are now discussed.

#### 3.1.5.1 Euler Angles

Euler's method represents rotation in 3D space as a sequence of three separate rotations about three mutually orthogonal coordinate axes ($x$, $y$, $z$) fixed in space. The size of the rotation is specified by three angles ($\alpha$, $\beta$, $\gamma$) - so called Euler angles. To rotate a point using the $xyz$ sequence, a point is first rotated around the $x$ axis, then around the $y$ axis and finally around the $z$ axis. However, rotation using Euler angles does not have to be done in a specific order. The previous rotation was done using axes in the sequence of $x$-$y$-$z$, but it could have been done in $z$-$y$-$x$, $x$-$z$-$y$ or any other combination. This means that a point can be rotated to the desired location by using any combination of the angular degrees of freedom: $x$, $y$ and $z$. By using Euler angles, a rotation's representation is not unique, since it can be represented by several different combinations. Hence one problem with using Euler angles, is that there is no uniqueness in its parameterisation of rotation.

To do rotation using Euler angles each rotation around an axis is evaluated independently of the other axes in a set order. A problem arises when the 2nd axis that gets evaluated has a rotational value of 90° or 270°. For the $x$-$y$-$z$ sequence, the first rotation to be performed is around the $x$-axis. Next the $y$-axis rotation is performed. Since the $x$-axis rotation has already been executed, it no longer gets updated in
future rotations around other axes. So if the \( y \)-axis is rotated by 90° then the \( z \)-axis is rotated onto the \( x \)-axis. Hence the \( x \)-axis and \( z \)-axis point in the same direction and so they have become locked. Any subsequent rotation done around the \( z \)-axis is effectively done around the \( x \)-axis. Thus a degree of freedom (DOF) has been lost. This is known as gimbal lock.

The storage requirements for Euler angles are minimal, i.e. just three numbers are stored, and the computational cost is minimal too. The price for simplicity is not having any uniqueness for a specific rotation and also being subject to gimbal lock. However, Euler angles are good for specifying a rotation about a single axis.

### 3.1.5.2 Rotation Matrices

Earlier it was shown that a point in 2D could be rotated by using equations (3.10) and (3.11). These can be simply expressed in a matrix and so a rotated point can be expressed as follows.

\[
\begin{bmatrix}
x' \\
y'
\end{bmatrix} = \begin{bmatrix}
\cos \theta & -\sin \theta \\
\sin \theta & \cos \theta
\end{bmatrix} \cdot \begin{bmatrix}
x \\
y
\end{bmatrix}
\]

Hence \( p' = Rp \) where \( R \) is a 2D rotation matrix. The above rotation is equivalent to a rotation in 3D space about the \( z \)-axis. Hence, the rotation matrix for a \( z \)-axis rotation is:

\[
R_z(\theta) = \begin{bmatrix}
\cos \theta & -\sin \theta & 0 \\
\sin \theta & \cos \theta & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

Similarly the rotation about the \( x \)-axis and \( y \)-axis is as follows.

\[
R_x(\theta) = \begin{bmatrix}
1 & 0 & 0 \\
0 & \cos \theta & -\sin \theta \\
0 & \sin \theta & \cos \theta
\end{bmatrix}, \quad R_y(\theta) = \begin{bmatrix}
\cos \theta & 0 & \sin \theta \\
0 & 1 & 0 \\
-\sin \theta & 0 & \cos \theta
\end{bmatrix}
\]

A rotation matrix is a special form of a matrix. For a rotation matrix to be valid it has to be orthonormal. This means that the rotation matrix is normalised and
orthogonal. For it to be normalised, the sums of the squares of the elements for each row and column must sum to 1, and for it to be orthogonal the dot product of any pair of rows or columns has to be zero. Simply put, the rows and columns are of unit length and are also at right angles to each other. A rotation in 3D space has 3 degrees of freedom (DOF), i.e. only 3 numbers are need to specify it, but a $3 \times 3$ matrix has 9 parameters. Hence 6 of the parameters have to be constrained to get back to the 3 DOF that are wanted. The 6 constraints are met by making sure that the rows are normalised and that they are orthogonal to each other.

To rotate an object around all axes, a full 3D rotation matrix needs to be built. This is achieved by using matrix multiplication to compute together the different rotation matrices representing each axis component rotation. This resultant matrix can then be used to rotate an object. To change the objects orientation again, the existing rotation matrix is multiplied by another fully constructed rotation matrix. If a set of rotation matrices had been constructed, and if two different sequences of rotation matrices were applied to an object that had the same starting position, the resultant position of the object would be different since matrix multiplication is not commutative, i.e. $AB \neq BA$.

After successive rotations a problem with rotation matrices occurs due to round off errors, which means the conditions that make a rotation matrix valid are no longer true. This is called drifting. The effect of drifting is that an object that is being rotated moves off course, instead of remaining on the path of rotation that is desired. Also the object may be sheared and scaled too. This becomes more noticeable after several compositions have been performed as numerical round off errors accumulate. To stop/reduce drifting the rotation matrix has to be re-orthonormalised. However, doing this after every composition of rotation to maintain accuracy of the rotation and the validity of the rotation matrix has a high computational cost. If the effects of drifting are not critical then the re-orthonormalising can be done on a less frequent basis. If rotation is only being performed around one axis then Euler angles might be worth considering since they do not suffer from drifting.

The storage requirements for rotation matrices are 3 times those of Euler angles.
3.1. Geometry

However, each rotation is uniquely specified compared with Euler angles. Having to
do re-orthogonalisation and re-normalise operations on rotation matrices obviously
increases the computational cost of using such a representation. If a lot of com-
position is being done and speed as well as accuracy are important then another
representation may be more appropriate. Rotation matrices do not suffer from sin-
gularities such as gimbal lock, and unlike Euler angles, rotation around the three
axes are interdependent.

3.1.5.3 Axis and Angle

The axis and angle representation consists of a vector for an arbitrary axis and a
scalar to represent the angle of rotation around that axis. This representation over
parameterises by having a vector consisting of 3 elements and a scalar, and so again
there are 4 parameters when only 3 DOF are required. The constraint introduced
to bring it down to 3 DOF is that the axis must be of unit length.

Until now, rotation has occurred around 3 different axes. However, with axis angle,
the rotation occurs around the resultant axis of the original 3 axes. Therefore, to
rotate a point $p$ an angle is specified which rotates it to $p'$, see figure 3.4. Only the
point's orientation with respect to the axis has changed. Its height relative to the
axis remains unchanged. Hence each point is rotated on a plane of rotation which is
perpendicular to the axis. Since each point moves in a plane, the required rotation

![Figure 3.4: Axis angle: rotation of a point.](image)
can be performed by a 2D rotation formula.

The axis angle representation stores only 4 parameters which is significantly less than the 9 parameters for a rotation matrix. Also it has only one constraint, and so maintaining the validity of this representation is easier than that of a rotation matrix. Again because this method over parameterises it does not suffer from gimbal lock.

3.1.5.4 Quaternions

The quaternion was developed by Sir William Hamilton in the 19th century as an extension to complex numbers. A complex number consists of a real component \( w \) and an imaginary component \( i \). The imaginary component \( i \) being the square root of \(-1\). A quaternion still has one real component but it extends the idea by having three imaginary components \( ij, j, k\), all being the square root of \(-1\) and is known as a hypercomplex number. The basic rules for the imaginary components are as follows.

\[
i^2 = j^2 = k^2 = -1
\]

\[
ij = -ji = k, \quad jk = -kj = i, \quad ki = -ik = j
\]

A quaternion \( q \) can simply be represented in complex number notation as \( q = w + xi + yj + zk \), and in vector notation as \( q = [w, v] \) where \( v \) is a 3D vector. Thus each quaternion can be plotted in 4D space (quaternion space). Again, this representation uses 4 parameters and so the constraint of unit length is introduced. This constrained quaternion is known as a unit quaternion. A unit quaternion has the property of \( w^2 + x^2 + y^2 + z^2 = 1 \). Unit quaternions have a magnitude of 1 and form a \( S^3 \) (a 3-sphere\(^2 \) of unit length) of the quaternion space. A valid rotation in 3D space is represented by a quaternion that lies on this unit sphere.

To change an existing orientation, all that needs to be done is to multiply the unit quaternion that represents the current orientation by another unit quaternion.

\(^2\)The concept of unit sphere in 3D space, i.e. \((x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2 = 1\) is extended to a \( N \) dimensional Euclidean space, written as \( S^{n-1} \). Thus \( S^3 \) is a unit sphere in 4D Euclidean space.
which represents the modification in orientation that is required. The result of the multiplication (composition) will be another unit quaternion which represents the new orientation. However, if the order of multiplication is changed the resulting unit quaternion will not be the same, as quaternion multiplication is not commutative. Unit quaternions do not suffer from gimbal lock since a valid rotation in 3D space is represented by a unit quaternion that lies on a unit sphere in quaternion space, and all compositions are performed in 4D space.

Quaternions have been a popular choice for rotation parameterisation for several reasons. In computer animation, animators do more than just rotate objects. Scenes are viewed through a camera, and animators want the viewers to see the world in a particular way. Often in films the camera starts at one position and moves smoothly to another position on a particular path. This could be a simple translation movement or a change in orientation. An animator will specify a start and stop position, and the computer package will generate the various positions for the camera of the arc which will vary in length. This process is known as interpolation.

In modern games, having a camera move around a character in an arc is done quite a lot, and becomes a wanted feature in a lot of games. However, depending on the capabilities of the computer the frame rates of the computer generated scene will vary. When interpolating between rotation matrices, jerkiness can be noticeable and this is unacceptable in todays gaming world. The jerkiness can be reduced by increasing the number of predefined orientations. However, with different frame rates problems can still occur. Also interpolating between rotation matrices has a high computational cost. Using Euler angles to do interpolation is unlikely since each angle has to be interpolated. Both the Euler angles and the axis angle representation cannot perform interpolation consistently. Hence game developers prefer quaternions because their interpolation is smoother than that of rotation matrices. In addition, it is computationally less expensive to add angular velocities to quaternions than to matrices, and also the number of interpolated positions can be dynamically adjusted to correspond to a particular frame rate. If interpolation was done between two quaternions, i.e. on a straight line, the animated rotation would speed up in the middle. To create a smooth motion between the first and second specified quaternion
SLERP (Spherical Linear Interpolation), see Shoemake [100], is used to generate a series of quaternions between two specified unit quaternion orientations. As the name suggests SLERP interpolates on an arc which intersects both quaternions and the origin.

Quaternions only require 4 parameters for storage and have only one constraint meaning that computational costs are relatively low. Composition of quaternions is relatively straightforward, and quaternions certainly have an advantage over the other methods described for interpolation. Again because of the over parameterisation this method does not suffer from singularities such as gimbal lock.

### 3.2 Least Squares

Registration is about determining the best alignment of surfaces by using the differences in distances between the two surfaces. To minimise these distances a good fit needs to be found. Hence the well known data fitting method, least squares is now discussed.

Least squares is a simple method for data fitting, also known as regression in statistics. Suppose that a simple experiment is being run to see how the temperature of a piece of equipment varies during the experiment. Observations are made every 5 seconds and the temperature is recorded. These measurements (observations) are then plotted on a graph, shown in figure 3.5(a). The idea of this experiment is to show that the temperature rises linearly against time. Hence a straight line needs to be fitted to the observations. Several straight lines could be fitted as shown in figures 3.5(b),(c) and (d), but how is the best straight line determined?

The equation for a straight line is

\[ y = mx + c \]  (3.13)

and so values for \( m \) and \( c \) have to be determined which gives the best fit for the mathematical model, the straight line. If the data obtained in the experiment was perfect, then all of the data points would sit exactly on one unique straight line.
3.2. Least Squares

Their distance from the line would be zero. However, generally data is not perfect and will deviate from their theoretically perfect positions. The best line is then the line where in some overall sense the points are closest to the line. The distances of the points to the line have to be measured in a consistent manner and the shortest distance from a point to a line is the normal distance i.e. that of a perpendicular line from the point to the straight line, as shown in figure 3.6.

To find the best overall fit, a straight line needs to be determined which minimises the sum of the perpendicular distances between the noisy points \( p_i \) which have deviated from their theoretically perfect positions \( p_i \) to the line. These deviations are commonly referred to as errors. Thus, it is the errors that are to be minimised. One measurement of error used in statistics is the root mean square (RMS) error \( \sigma \).

\[
\sigma = \sqrt{\frac{1}{N} \sum_{i=1}^{N} \| p_i - p_i' \|^2}
\] (3.14)
Figure 3.6: Perpendicular distances are used to determine the best fit.

Hence the optimal values for \( m \) and \( c \) is when \( \sigma \) is at its minimum.

To obtain optimal values for \( m \) and \( c \) an iterative numerical approximation approach can be used, where the RMS is computed and noted for every straight line possible within the region of data. The procedure can be carried out using two nested loops.

**Algorithm 3.2.1 Least Squares, Iterative Numerical Approximation Approach.**

LeastSquaresIterativeApprox(dataPoints)

\{  
    Initialise \( \text{bestrms} \) to \( \infty \).

    Within region of dataPoints vary \( c \) do
    \{
        Vary \( m \)
        \{
            Determine \( \text{rms} \).

            If \( \text{rms} < \text{bestrms} \)
            \{
                \( \text{bestrms} = \text{rms} \).
                \( \text{bestm} = m \).
                \( \text{bestc} = c \).
            
            \}
        \}

    return \( \text{bestm} \) and \( \text{bestc} \).
\}
In the inner loop \( m \) is varied for a fixed value of \( c \). While in the outer loop, all values of \( c \) tried. This has been summarised in algorithm 3.2.1.

For the simple case of determining a straight line, an analytical approach can be used. This will give precise optimal values for \( m \) and \( c \) and is much quicker at determining the values than an approximation method. The analytical method uses vertical distances instead of perpendicular distances because it makes the analytical solution easier and the distances are still measured in a consistent manner, as illustrated in figure 3.7. The distance \( d_i \) is therefore the difference between the deviated value \( y_i \) and its true (theoretical) value of \( y \), as shown in equation (3.15)

\[
d_i = y_i - y
\]  

(3.15)

which can be re-expressed as

\[
d_i = y_i - (mx_i + c) = y_i - mx_i - c
\]  

(3.16)

What we would ideally like to solve for, is a solution where the sum of \( d \)'s are equal to zero. However, we currently have only one equation with two unknowns with which to solve \( m \) and \( c \). Also the value of \( d \) could be positive or negative depending upon whether the point is above or below the line. The value can be made positive by simply squaring the distance and so we now have equation (3.17) where we would like to minimise our cost function \( E(m, c) \).
and \( c \) when the sum of squared distances is zero.

\[
E(m, c) = \sum_{i=1}^{N} d_i^2 = \sum_{i=1}^{N} (y_i - mx_i - c)^2 \quad (3.17)
\]

To find the minimum of equation (3.17), partial differentiation is used. The minimum is found when the partial derivatives with respect to \( m \) and with respect to \( c \) are both zero.

\[
\frac{\partial E}{\partial m} = \sum_{i=1}^{N} 2(y_i - mx_i - c)(0 - x_i - 0) \quad (3.18)
\]

\[
\frac{\partial E}{\partial c} = \sum_{i=1}^{N} 2(y_i - mx_i - c)(0 - 0 - 1) \quad (3.19)
\]

By setting the partial derivatives to zero and rearranging them, we get equations (3.20) and (3.21) which are referred to as the normal equations.

\[
\sum_{i=1}^{N} x_i^2 m + \sum_{i=1}^{N} x_i c = \sum_{i=1}^{N} y_i x_i \quad (3.20)
\]

\[
(\sum_{i=1}^{N} x_i)m + Nc = \sum_{i=1}^{N} y_i \quad (3.21)
\]

The system of equations can then be solved, giving the optimal values for \( m \) and \( c \). A disadvantage of using vertical distances is that the method is sensitive to the orientation of the coordinate system. A least squares technique for 2D points using perpendicular distances which is numerically stable and rotational invariant can be found in Paeth [2].

When the mathematical model is fairly simple for solving least squares, an analytical approach should be used if possible, since an exact answer will be obtained and computing time will be at a minimum. However, if the problem is complex an analytical solution may not be possible. Thus an iterative numerical approximation method will be the only approach available. The solution obtained will be close to the optimal solution but highly unlikely to be the exact optimal solution. The amount of time will increase significantly for an iterative method the more complex the problem is. As the required accuracy for the final solution increases, so does the number of iterations that will have to be performed which will naturally result in a further time increase. Both analytical and approximation methods will be affected by increases in the number of observations.
3.2 Least Squares

3.2.1 Singular Value Decomposition

As the number of observations are increased, better estimates are obtained because noise is smoothed out. With noisy data, the least squares criterion will not be zero but close to zero. Least squares can be expressed as solving a set of linear equations and we can use any of the standard set of numerical techniques for that problem - these include singular value decomposition, SVD.

SVD is a popular method for solving linear least square problems. SVD is used since other methods can fail to give a satisfactory result because the given set of equations to solve are either singular\(^3\) or very close to singular. SVD can be thought of as factorising a matrix \(H\). It decomposes \(H\) into three distinct matrices, such that \(H = UAV^t\). These are an upper triangular matrix \(U\), a diagonal matrix \(\Lambda\) and a lower triangular matrix \(V^t\). The numerical recipes [84] algorithm for solving SVD is very stable. For more details on other numerical methods for solving least square problems, including SVD, the reader is referred to Lawson and Hanson [72].

3.2.2 The Influence of Errors

As discussed previously, least squares minimises errors between the true value obtained via our mathematical model and the actual observations. In our simple temperature experiment an obvious rogue observation is apparent in figure 3.5(a), that being the measurement to the top left hand side of the graph which is further away from the rest of the measurements. In this simple case it could have occurred by misreading the instruments. This particular measurement doesn’t really belong with the distribution of the rest of our observations and is called an outlier. Errors can occur for several reasons including characteristics of the measuring device used, its precision, or environmental changes. These errors have an associated distribution. A specific error distribution that is often assumed is that of a Gaussian or Normal distribution. This is the classic bell shape curve with zero mean and unit standard deviation, although the mean and standard deviation can be arbitrary.

\(^3\)The set of equations in matrix form does not have an inverse, i.e. the matrix's determinant is equal to zero.
When assumptions are made about the type of error introduced, the accuracy of
final results depend on the validity of these assumptions.

It is often assumed for simplicity that the underlying noise is Gaussian, although
this may not be the case. In ideal conditions the variance of observations for a single
Gaussian distribution will be the same over all regions within the observations. Thus
the deviations from the true values will be statistically the same across all the data.
However, if a different distribution of data is present and the variance differs within
local regions of the statistical population, the least squares method will not be able
to find the optimal fit. It minimises errors for a global variance. In statistics the
reliability of how well a procedure (such as least squares) determines the optimal
estimates (e.g. $m$ and $c$) is referred to as its statistical efficiency. For datasets where
the underlying deviation of the observations is a single Gaussian distribution, it
can be shown that the least squares method yields the optimal solution and is fully
efficient, see Besl et al. [12, 13] and the references therein. However, if there is some
other type of noise distribution present, i.e. the noise is non Gaussian, then this
optimality criterion will not be guaranteed.

An outlier is a measurement within a dataset that is outside the statistical distribu­
tion associated with or assumed for the majority of the observations. Consequently
its value is far from the other observations. If an outlier is far enough away from
the rest of the observations then it will cause a significant deviation in the solution
obtained with least squares compared to the true underlying optimal one. This is be­
because the difference of distances between the actual and theoretical value is squared,
and so it becomes very significant within the minimisation cost function. This is il­
ustrated in figure 3.8. Intuitively, we would expect to obtain a fit something similar
to that of figure 3.8(a). However, the distance associated with the outlier is quite
significant and so in a least squares sense the solution obtained is not optimal. The
distances between the line and observations on average should be the same. Hence
the estimated line that is more likely to be obtained through least squares is shown
in figure 3.8(b). A single outlier can significantly outweigh all other observations
in the least squares cost function because its value is so much larger. How well a
procedure performs in the presence of outliers, or when the observed distribution(s)
3.2. Least Squares

Figure 3.8: Affects of an outlier. (a) Intuitive fit. (b) A possible least squares fit.

differ from the assumed distribution(s), is referred to as its robustness. More details on errors and their propagation with a review of statistical concepts can be found in Mikhail [78] along with least square techniques and fundamental concepts on observations and mathematical models.

3.2.3 Introducing Robustness

Problems in image analysis do not generally obey the simplest data models. There are usually significant amounts of missing data and in a given region several distributions may be overlapping. Thus, we would like to be able to introduce some robustness into the least squares technique so that when there are deviations from what we expect, then the procedure will cope relatively well and gives an answer that is close to the optimal solution. To do this the influences of the outliers on the overall solution has to be minimised as much as possible.

In statistics there are three common methods of determining the measure of central tendency, i.e. the middleness of a distribution. They are the mean, the median and the mode. The mean is the arithmetic average of the distribution, obtained simply by adding up all the samples and dividing by their total number. The median is obtained by sorting the samples, so that they are arranged in order numerically, either ascending or descending, and then picking the middle sample. If the number
of samples is even, then there will be two middle samples and the median is at the mid-point between them. Finally there is the mode which is the most frequently occurring value in the set of observations.

When there is a symmetric distribution such as the Gaussian or Normal distribution, the mean, median and mode are equal, see figure 3.9(a). However, when there is a different type of distribution present this no longer remains the case. The presence of an outlier with a fairly large value will skew a distribution considerably, pulling the mean along with it. The median will be pulled a little, and the mode will remain the same as shown in figure 3.9(b).

![Figure 3.9: How outliers affect the measure of central tendency. (a) Normal distribution. (b) Negatively skewed distribution due to outliers.](image)

However the mode differs from the mean and median, in the sense that it represents the range with the highest frequency when the observations are from a continuous space. If the observations are discrete then the mode represents a specific observation and its frequency. In registration, distances are minimised, thus the observations are all distances, and to use the mode in an error analysis capability in a continuous space is not easy. All the distances are likely to be unique due to numerical/rounding errors during their computation. Hence to calculate the mode, classes (ranges) will be required due to the small differences between the computed distances, and somehow the class intervals are computed or they are entered manually. Also, the mode will return the range that has the highest frequency, and that range which
will have some numerical error associated with it. Whereas the mean and median return a precise value. Hence for these reasons the mode could not be used easily in solving least squares.

A method which can give the correct estimate of a parameter even with up to 50% noise contamination is least median of squares method, LMedS. To determine a mathematical model which has n parameters, for example a line with 2 parameters, the LMedS method selects n random observations. In our example two random observations are selected and a straight line is estimated from them. For all of the other observations, excluding the randomly selected ones, the deviations from the estimated mathematical model are calculated. The median of the squared deviations is then computed. The process is repeated a large number of times, randomly selecting observations, fitting the new estimated mathematical model and computing the median of squared deviations. At the end of the process the guess with the least median of squared deviations is taken as the best estimate of the underlying line. A disadvantage of this method is that it uses a brute force approach, and thus is time consuming. As the number of parameters required to determine the mathematical model increase, so does the computing time. Another disadvantage is that the determined mathematical model has to pass through the randomly selected observations meaning that the true optimal one may not be discovered. This is because the mathematical model is forced to go through n observations, whereas the optimal model may pass through 0 to n − 1 observations. In the case of a straight line, it is forced to go through two randomly selected observations.

The function that returns the measurement for the deviation of each observation from its hypothesised one (theoretical value) is known as a kernel function \( k(d_i) \). In least squares, this is simply the squared difference between the observation and the theoretical value, i.e. \( k(d_i) = (y_i - y)^2 \). Large deviations, i.e. outliers, can have a serious impact on the final result as they have a large influence. The derivative of the kernel function describes how the effect of an infinitesimal change affects the weight or influence that a point has and is known as the influence function. The kernel function and influence function for least squares are shown in figure 3.10.
In robust statistics the ideal kernel function is a continuous one where the significance of the error is severely reduced for outliers. If the kernel function is not continuous, then the influence function will have infinite or large spikes. This means any observations that occur in these zones will contribute as very large errors and this is undesirable. A brief overview of robust statistics can be found in Huber [59]. Other commonly used robust regression methods used within computer vision can be found in Meer et al. [77]. For more in-depth information on robust statistics and influence functions see Huber [58], Hampel et al. [40] and Rousseeuw and Leroy [89].

An alternative to LMedS is to make least squares more robust by using weights which do not alter directly the kernel function. Each observation has a weight associated with it. The purpose of the weight is to show how much confidence we have in that particular observation. If we have 100% confidence in an observation then its weight value will be 1. If we have no confidence in it, then the weight will have the value of 0. The value of a weight can be between 1 and 0. When solving the least square problem of fitting a straight line, the sum of distances between each observation's \( y_i \) value and the mathematical model's theoretical \( y \) value is minimised, as shown in equation (3.22).

\[
\sum_{i=1}^{N} (y_i - y)^2
\]  

(3.22)

This can be rewritten in terms of the kernel function.

\[
\sum_{i=1}^{N} k(d_i)
\]  

(3.23)
By simply adding weights, equation (3.24) is obtained.

$$\sum_{t=1}^{N} w_t k(d_t)$$

(3.24)

By setting the weight associated with the outlier point at the top left in figure 3.8 to zero, the intuitive fit (robust fit) shown in figure 3.8(a) is obtained, instead of what was originally obtained in figure 3.8(b).

The use and determination of weights is problem specific. When the problem is fairly simple, weights can be reliably determined beforehand. However, as the problem becomes more complex, setting initial weight values reliably might not be possible.

Figure 3.11: Use of a weight function to create a new kernel and influence function. (a) Weight function. (b) New kernel function. (c) New influence function.
If an iterative numerical approximation method is being used to solve least squares, re-determining the weights each iteration by using a weighting function, may lead to more accurate results, as well as possibly speeding up the computation since more accurate weighting will narrow the search for the optimal solution. A simple weighting function can use the average distance (the mean square error, MSE) from an observation to the mathematical model, so that weight values are decreased the further an observation is from its theoretical one. Figure 3.11 illustrates the way such a scheme can be used. The weight function has the affect of cutting off extreme outliers. Figure 3.11 shows the weight function, and the resultant kernel and influence functions which use the weighting scheme. The MSE calculation involves the use of weights so that a more accurate error value is computed by ignoring outliers via weights. Initially the weights may not be very reliable and thus neither will the MSE, but after a few iterations the value of the MSE becomes more accurate and as a result so do the weights.

Using a weighted least squares or iterative re-weighted least squares approach is much faster than the least median of squares approach, but less robust. As anticipated, there is a trade off between robustness and speed.

3.3 2-View Point Set Alignment

Point set alignment is the heart of surface registration. It is not easy to mathematically describe how to minimise the distance between two surfaces so that they become aligned. However, it is much easier to mathematically express a minimisation of distances between points that lie on both surfaces. The method used in the implementation determines the solution quickly without the need of iterating.

To be able to do the data fitting of the point sets, the correspondences between the point sets needs be known. Determining correspondences is not easy, and the closest point methods described in chapter 4 are used in establishing the correspondences. However, the acquired correspondences may not be the true correspondences.

Once two point sets with known correspondences have been obtained a least squares
3.3. 2-View Point Set Alignment

technique of minimising a cost function involving the distances between the points can be used. To align two 3D point sets $P_1$ and $P_2$, the cost function will involve a transformation comprising of a rotation $R$ (a $3 \times 3$ matrix) and a translation $T$ ($3 \times 1$ column matrix) which are the two variables that need to be solved.

One such method that is based on least squares fitting of two 3-D point sets was presented by Arun et al. [3]. Their method assumes that both point sets are identical (which means that each point in one point set corresponds to one and only one point in the other point set), and also that one of the point sets has been transformed by a translation and rotation, see figure 3.12(a). This means that the error, the difference in distance between each of the corresponding points, will be zero when the reverse translation and rotation are applied to the transformed point set.

The first step in the method is to translate both point sets to a common origin. This is done by determining the centroid for each point set. Vector differences from the points within the each point set to their own centroids are determined. By working with the vector differences it effectively translates both point sets to the origin, as shown in figure 3.12(b). This means that the sets (vector differences) will be approximately overlaying each other. The next step involves determining a rotation that minimises overall the distances between all of the corresponding points. Figures 3.12(c),(d),(e) shows this being done iteratively which makes for a clearer illustration. Once the rotation is determined, the translation between the original sets (not the vector differences) can be easily calculated by finding the difference between the centroid of one point set and the rotated centroid of the other set. A problem with methods that determine $T$ from $R$, is that they will be subject to error propagation, since errors will have accumulated in solving for $R$.

In this example, both point sets are identical and so the minimal distances will all be zero. However, in reality the point sets are unlikely to be identical and hence the minimal distances will be small but non-zero. Some points will have large distances due to the fact that they are outliers (see section 3.2.2).

Unlike the illustrated example, Arun et al.'s [3] method solves the rotation in one step via a closed form solution. To find a good $R$ and $T$, Arun et al. define a cost
Figure 3.12: Solving for the rotation between 2 corresponding point sets. (a) Two identical point sets, one of which has been transformed. (b) Vector differences determined from centroids, effectively translating each point set to the origin. (c),(d),(e) Rotation solved for to minimise distances between corresponding points.

function $E$ to be minimised as shown in equation (3.25)

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

(3.25)

where it is assumed that $p_i^2$ comes from $p_i^1$ corrupted by some noise $\sigma_i$, i.e. $p_i^2 = R p_i^1 + T + \sigma_i$.

First of all the centroids $p_c^1$ and $p_c^2$ of $P^1$ and $P^2$ are calculated.

$$p_c^1 = \frac{1}{N} \sum_{i=1}^{N} p_i^1$$

(3.26)

$$p_c^2 = \frac{1}{N} \sum_{i=1}^{N} p_i^2$$

(3.27)
This enables two sets \( Q^1 \) and \( Q^2 \) of vector differences from their own corresponding centre of masses to be found.

\[
q_i^1 = p_i^1 - p_c^1 \tag{3.28}
\]

\[
q_i^2 = p_i^2 - p_c^2 \tag{3.29}
\]

In Arun et al. [3] it is shown that the minimisation of the cost function as defined in equation (3.25) can be broken up into a two step problem in which we first minimise \( E(R) \) as defined in equation (3.30).

\[
E(R) = \sum_{i=1}^{N} \| q_i^2 - Rq_i^1 \|^2 \tag{3.30}
\]

Arun et al. expand out equation (3.30) to get

\[
E(R) = \sum_{i=1}^{N} (q_i^2 q_i^T q_i^2 + q_i^2 q_i^T q_i^1 - 2q_i^2 Rq_i^1) \tag{3.31}
\]

Equation (3.31) will be at its minimum when \( 2q_i^2 Rq_i^1 \) is at its largest. Therefore the problem turns into a maximisation problem where equation (3.32) is to be solved.

\[
E(R) = \sum_{i=1}^{N} q_i^2 Rq_i^1 \tag{3.32}
\]

As mentioned earlier Arun et al. use SVD to determine the optimal rotation \( \hat{R} \). As an intermediary step the \( 3 \times 3 \) matrix \( H \) is calculated which will lead to the rotation matrix when used by SVD.

\[
H = \sum_{i=1}^{N} q_i^1 q_i^2 T \tag{3.33}
\]

\( H \) is simply the summation of matrices, where each matrix in the sum is the result of a vector outer product expansion of corresponding points, i.e. the multiplication of \( q_i^1 \) a \( 3 \times 1 \) vector by \( q_i^2 T \) a \( 1 \times 3 \) vector which results in a \( 3 \times 3 \) matrix.

Arun et al.’s algorithm for solving \( \hat{R} \) is outlined below.

1. From \( P^1, P^2 \) calculate \( p_c^1, p_c^2 \), and then calculate \( Q^1 \) and \( Q^2 \).

2. Calculate \( H \).
3. Find the SVD of $H$.

$$H = U\Lambda V^T$$  \hspace{1cm} (3.34)

where $U$ and $V$ are orthogonal matrices (e.g. $V^T = V^{-1}$), and $\Lambda$ is a diagonal matrix with non-negative elements. The SVD of a matrix reveals a lot about its structure and properties. SVD determines the rank of the matrix $H$ which is used as an indicator of how solvable the matrix $H$ is. If $H$ is solvable, SVD will produce three non-zero singular values. It provides an orthonormal (orthogonal and normalised) basis for the range and null space of $H$. It also shows the dimension of the null space (number of zero singular values) of $H$.

4. Then calculate

$$\hat{R} = V \begin{pmatrix} 1 & \vdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \vdots & 1 \end{pmatrix} U^T$$  \hspace{1cm} (3.35)

In the original proposal by Arun et al. [3], an extra test had to be done in the final step of the algorithm because there was a possibility of the algorithm failing when $\det(VU^T) = -1$ since no solution could be found. Kanatani noted that techniques which dealt with minimisation over orthogonal matrices such as Arun et al. where noisy data was involved would result in improper rotations\(^4\). To solve this problem, Kanatani introduced a diagonal matrix between $V$ and $U^T$ in the last step of the algorithm.

Once $\hat{R}$ has been obtained, the estimated optimal translation $\hat{T}$ is calculated by substituting $\hat{R}$ into (3.36).

$$\hat{T} = p_c^2 - \hat{R}p_c^1$$  \hspace{1cm} (3.36)

Since the method uses a least squares technique, it is vulnerable to outliers. However it can be made robust by introducing weights. Thus equation (3.25) would simply be modified to equation (3.37) where $w_i$ is the weight associated with the corresponding points and ranges from 0 to 1.

$$E(R, T) = \sum_{i=1}^{N} w_i \| p_i^2 - (Rp_i^1 + T) \|^2$$  \hspace{1cm} (3.37)

\(^4\)An improper rotation, is a rotation which has a determinant of $-1$. 
3.4 Other Methods Of Point Set Alignment

Arun et al.'s method [3] is not the only closed form solution to align two 3D point sets with known correspondences. Horn [52] developed a technique which involves a unit quaternion to solve for the absolute orientation. Horn's method also includes a scaling factor. A symmetric $4 \times 4$ matrix is built up from products and sums of measurements relative to their centroids. The unit quaternion for the optimal rotation is the eigenvector associated with the most positive eigenvalue of the symmetric matrix. The translation is determined in a similar way to Arun et al. except it involves a scaling factor. In other words the translation is the difference between the centroid of one point set and the rotated and scaled centroid of the other point set.

Going on from Horn's [52] previous paper using unit quaternions, Horn et al. [53] presented a method for which the rotation was solved by using a $3 \times 3$ orthonormal matrix (a rotation matrix, as discussed in section 3.1.5.2) purely because the use of orthonormal matrices was so widespread. The approach of Horn et al. is similar to Arun et al.'s method. A $3 \times 3$ matrix is constructed which contains sums of products of differences from the centroids for which eigenvalue-eigenvector decomposition is performed. Again the translation is determined by the difference between the centroid of one point set and the rotated and scaled centroid of the other point set.

Walker et al. [110] presented a method which used both positional information of the points and their direction, i.e. normals associated with the points. Also the minimisation function incorporated the use of two weight sets. One set of weights was associated with the corresponding points and the other with the corresponding unit normal vectors. Their method uses a dual quaternion for the transformation which represents both the rotation and translation. The transformation simultaneously rotates around and translates along a particular line in 3D space. Walker et al.'s cost function is expressed in terms of the two components from the dual quaternion. The rotation is obtained from the eigenvector that has the largest eigenvalue associated with the derived $4 \times 4$ matrix.
Chapter 3. 2-View Point Set Alignment

Given these various methods it is interesting to ask how they each perform. A comparison between these four major algorithms which estimate 3D rigid body transformations was done by Eggert et al. [28]. The comparison focused on accuracy/robustness, stability and efficiency. For the accuracy comparison they used non-degenerate data sets and added noise of various levels. Arun et al.'s SVD and Horn's unit quaternion method were the most similar in the accuracy tests. There was a slight deviation for the orthonormal matrix method of Horn et al.'s method when using small data sets, and the dual quaternion method of Walker et al.'s came out the worse.

None of the algorithms were designed to handle data points arranged in a linear or singular point relation due to the unconstrained degrees of freedom in the solution. So the stability tests involved data sets as they approached these degenerate forms to see how they performed. Initially a 3D-2D degeneracy stability test was done. The \( z \) component of each point in the point set was steadily reduced down to machine level precision, meaning that they came to lie on the \( x, y \) plane. The dual quaternion method and orthonormal matrix method were less stable than the SVD and unit quaternion methods when no noise was present.

Next 3D-1D degeneracy tests were performed. The \( x \) and \( y \) components of each point in the point sets had their values steadily reduced. The dual quaternion method broke down first, and the unit quaternion method was found to be less stable than the SVD method. The orthonormal matrix method was found to be highly dependent on the data set size. It performed well when using a large data set.

Finally, 3D-0D degeneracy was performed where the data set degenerates into a point. The dual quaternion method broke down the earliest and was found to be less stable. The other methods behaved virtually the same, except for the orthonormal matrix method which would break down earlier if a small data set was used.

Fisher et al. concluded that in the stability tests that the dual quaternion method was the most fragile, and that the other methods were generally the same except that the orthonormal matrix method would breakdown slightly sooner if a small
data set was used. In the efficiency tests, it was found that for small data sets the orthonormal matrix method was the quickest, and for large data sets the SVD, unit quaternion and orthonormal matrix methods were about the same. The dual quaternion method tended to be generally slower, but as the number of points increased above a certain point (dependent on computer cache) the algorithm became the most efficient. Fisher et al. summarised that the accuracy of the algorithms were virtually the same and for there to be any noticeable difference between the algorithms on stability, the amount of degeneracy required would have to be above what would occur in practice. With efficiency, the data size and computer configuration (including software and memory cache) would have to be taken into account.
Chapter 4

Determining The Closest Point

In surface registration, it is important to be able to establish matches between points lying on both surfaces. One of the assumptions made for the alignment of two 3D point sets in chapter 3 was that the correspondences between both point sets are known. In the iterative closest point algorithm it is implicit that matching points are equivalent to closest points, i.e. for a point $p^1$ on the surface $S^1$ the closest point is that point on surface $S^2$ which is nearest (has the smallest distance) to $p^1$. Such a closest point will give a good approximation to the actual corresponding point as illustrated in figure 4.1 where the distances associated with the true closest point $d_1$ and the approximated point $d_2$ are approximately the same, i.e. $d_1 \approx d_2$. Thus to establish a match between two surfaces a closest point method is given a point $p^1$ from the surface $S^1$ and returns the identity of a point lying on the surface $S^2$ that $p^1$ is closest too. Note that it is assumed that actual real correspondences exist.

Prior to the matching stage, the closest point method is given the surface on which all subsequent queries are performed to determine the correspondences (closest points). At the simplest level the other surface may be considered as a collection of points. Thus to establish matches between the surfaces, a query is done for each point in that collection to determine its closest point.

In this chapter different ways of determining a closest point will be considered since this is at the core of the iterative closest point (ICP) algorithm. The closest point
method crucially affects both the computation time and the success or otherwise of the registration. The different closest point methods may have different trade-offs and it is important to understand these so as to maximise the effectiveness of the ICP algorithm.

4.1 Surfaces

What is meant by the term: surface? Well, if a snooker ball was to be modelled, then it could be simply represented as a sphere, with a radius that matched the snooker ball's radius. The boundary of this sphere is known as a surface. In other words, a surface is the boundary of a geometric object. The sphere has a type of surface known as a closed surface. This is because if one is to walk along the surface of the sphere, it is not possible to get into a position where one could walk along the inside of the sphere's surface. To do so would require a hole in the surface. Let us consider a cylinder with no ends. This could be a simple representation of a pipe. As one walks along the outside of this pipe and approaches one of the ends it then becomes possible to start walking along the inside of pipe. Such a surface is known as an open surface. A surface that has no holes in it is a closed surface, i.e. it is not possible to travel along the inside of the surface when starting on the outside, and an open surface has one or more holes allowing the inside of the surface to be
reached. A doughnut shaped surface is also closed, even though there appears to be a hole in the middle of the object, it is not a hole in the actual surface. More complex surfaces can be built using other representations. A few of these surface representations are now considered.

4.1.1 Polygon Meshes

A surface of a real world object is a continuous surface, and to represent such a surface precisely as a computer model would require a large amount of storage. Hence surfaces tend to be discretely sampled and represented in a digital representation so that they approximate well the original surface. One representation that approximates a surface is a polygon mesh. It is the simplest representation of a surface and consists of planar patches. A polygon mesh is a 1st degree piecewise linear approximation to a smooth continuous surface.

The simplest type of planar patch is a triangular plane. In section 3.1 of chapter 3 the concept of a vertex was introduced, and it was shown that 3 vertices could be used to construct a triangle in 2D space. Similarly this can be used in 3D space to represent a triangular plane. By adding a fourth vertex which lies on the same plane as the other three vertices a cubic planar patch is constructed, and by adding more vertices the more complex the patch will become. Generally however, triangular meshes are commonest because of their simplicity and resultant computational efficiency. A polygon mesh of this type is shown in figure 4.2.

![Figure 4.2: Polygon mesh.](image)
Chapter 4. Determining The Closest Point

To be able to render a polygon mesh, an algorithm needs to know two things: the vertex positions and the direction of the surface normal to the face of the polygon. For a triangular mesh, three vertices specify a face. One way to construct a mesh is to use a polygon list, where each face (polygon) is defined by its own vertices. This is a very simple method, but it also means that information on all shared vertices is duplicated in defining connecting faces. Also if a single vertex is manipulated, then the data structure has to be fully searched to find duplicate vertices in other face’s vertex sets so that they are appropriately updated. However, when a transformation such as rotation is applied to the polygon mesh, then it is most likely that roundoff errors will occur. Hence when searching for duplicate vertices there may no longer be matches because the round off errors will affect each duplicated vertex coordinate in different ways. Another way to construct a mesh is to have a structure where each face has pointers which point to vertices contained in a vertex list. With a pointer-to-vertex list representation, there is no duplication, and manipulation of a single vertex is now trivial as only one update is required. Also a considerable saving in storage requirements is made.

The polygon list and point-to-vertex list are used for rendering. They do not make explicit what edges and vertices are shared and by what facet as a rendering algorithm does not need to know this. However, a problem with these representations is that edges get drawn twice. This limitation can be overcome by extending the
4.1. Surfaces

vertex list method to include an edge list. Each element in the edge list consists of two pointers to the vertex list, and one or more pointers to a face (the vertices of the polygon). Now, edges are the first things to be drawn, and everything between these edges is rendered. These three different polygon mesh representations are illustrated in figure 4.3.

Algorithms that want to manipulate a surface and perform an operation, such as decimation (see chapter 8 section 8.2) need to know about the connectivity of a surface. The type of data structure used for a polygon mesh and its complexity will depend on the needs of the application. A further problem with a polygon mesh is that it has no information on surface curvature. Each patch is planar and a new definition of curvature is required.

4.1.2 Spline, Bezier And B-Spline Patches

If a surface with continuous curvature is required then polynomial curves such as a spline can be used. To understand how piecewise smooth surfaces are constructed, the representation of these curves will first be considered.

When technical drawings required a smooth curve, draftsmen used a device which could be flexed into a curve where the slope and curvature was continuous. The natural spline curve also has these two properties. A spline uses a polynomial to fit a curve to two given points, i.e. a curve is interpolated between the two points. Thus given a set of points, a spline curve consists of a set of intervals and a set of polynomials where for each interval the associated polynomial has to meet some smoothness and continuity conditions. Each segment (interval) connects with another segment via a shared point (knot). When creating splines, cubic polynomials tend to be used, i.e. polynomials with degree 3 such as \( a_i x_i^3 + b_i x_i^2 + c_i x_i + d_i \). Polynomials with degree 1 would not be used since these are straight lines and the slope would be discontinuous at the knots. Degree 2 do not have this problem with continuity, but they do not tend to be flexible enough. The problem with higher degree polynomials is that they often introduce unwanted wiggles and also have a higher computational cost. Thus cubic splines are popular. A problem with splines
Figure 4.4: Curves: (a) spline, (b) Bezier and (c) B-spline. The points with dashed lines indicate the control points.

is that if one of the points (knots) is moved, then the whole curve changes. Since only the points can be manipulated, using splines in an interactive application is unwieldy. A spline curve is illustrated in figure 4.4(a).

Bezier curves were invented by the French engineer P. Bezier for use in the construction of pleasing surfaces for car bodies. A basic Bezier curve consists of two endpoints and two control points (Bezier points). A control point is associated with one of the endpoints, and is used to control the tangent vector associated with the endpoint. The curve will pass through both endpoints, but will generally not go through the control points as illustrated in figure 4.4(b). A more sophisticated Bezier curve can have several points that the curve goes through, and for each pair of points that make up a segment, they will have their own control points. Each control point influences the behaviour of the curve in its own segment and thus the effect remains local. However, if a point that the curve goes through is changed, then the curve for the two segments that share that point will change, unlike a spline where the whole curve is recomputed. The disadvantage of a Bezier curve compared to a spline curve is that it is not as smooth.

Another type of curve is the B-spline (or basic spline). The B-spline is similar to a Bezier curve, except that its control points are shared between segments, as illustrated in figure 4.4. So any changes made to points, affect only the shared segments. Hence there is only local change to the curve and this results in low computation costs to realise the change. The advantage of a B-spline is that it has
the same continuity as a spline but less computation is required. More properties on splines, Bezier curves, and B-splines can be found in Gerald and Wheatley [37]. There are several variants of B-splines, where the data points can be spaced evenly (uniformly) or not. Further information on these variant B-splines can be found in Foley et al. [35].

Figure 4.5: Bezier patch with 16 control points.

Earlier it was mentioned that spline curves can be created by using polynomials of degree 3. These can be rewritten in parametric form, i.e. the \(x\) component of the curve would be parameterised as \(x(u) = a_x u^3 + b_x u^2 + c_x u + d_x\). Thus a curve in 3D space would be parameterised by \(x(u), y(u)\) and \(z(u)\). By varying the value of \(u\), a curve will be drawn in 3D space. A surface can be defined by having a family of curves where each member is close to one another. A parametric surface can be defined as \(x = f(u, v), y = g(u, v)\) and \(z = h(u, v)\) where the two parameters \(u\) and \(v\) are varied. Similarly Bezier curves can be used to construct surfaces. Figure 4.5 illustrates a Bezier patch. Bigger surfaces can be constructed by joining patches together along a common edge. More details on surface patches such as Bezier and B-Spline patches can be found in Foley et al. [35].
4.2 Closest Point Methods

As mentioned earlier a closest point (CP) method takes a point \( p \) and a surface (or a point set such as a range image) and determines the point that is closest to \( p \) that lies on the given surface. Two types of CP method exist. The simplest type addresses the point to point set problem, and the other methods solve the point to surface problem. The methods described here are of both types. The majority of CP methods described herein work with a surface which is initially represented as a triangular mesh. Other methods work with a structured point set, such as a range image.

A simple analogy can be made between CP methods and databases. When a piece of data is wanted from a database, a query is performed which results in a search that returns data matching the query. With a CP method the form of a query is a point, and the algorithm will then search the surface to determine the surface point closest to the query point. Depending on the type of use expected from a database, i.e. the most common queries likely to be performed, the database is tuned so that the matching search results will be obtained as quickly as possible. Similarly, CP methods should encode a surface to facilitate the most efficient search for the closest point.

The result of a search is either an explicit or implicit point. An explicit point is one which exists in the data structure representing the surface. For a triangular mesh, an explicit point would be a vertex. In contrast, an implicit point, is a point which is computed from the surface representation properties. For example, with a triangular mesh, each face is planar. Thus the vertices of the triangle can be used to compute the plane, and the closest implicit point returned will be one that sits on the plane between the three vertices. This includes the possibility of the returned point being on a triangle's edge. Thus, in a triangular mesh, only the vertices are explicit points. All other surface points are implicit, i.e. their positions can be computed from knowledge of the vertex positions and under the assumption that the surface is planar. Hence, when determining the closest point on a triangulated surface, the result will most likely be an implicit point. However, when dealing with
4.2. Closest Point Methods

a structured point set, the point returned will always be an explicit point unless the method uses interpolation between the structured points.

The closest point search problem is computationally expensive and so compromises or approximations are sometimes made to improve speed. As a result, when obtaining a closest point from a surface the algorithm may either give an incorrect result due to a special case not being treated correctly or even fail. This could mean that the point returned may not actually be the true closest point, or the algorithm may not be able to find a point at all.

To illustrate the algorithmic cost of a CP method, the simpler closest point problem of point to point set matching will be examined. A crude but effective method would be a brute force technique. To find the closest point to a given query point \( p_j \), the Euclidean distance from \( p_j \) to every point in the other point set \( P^2 \) would be calculated. The point with the smallest Euclidean distance would be returned as the closest point. Obviously, the method works but it is not very efficient. If \( n_v \) is the number of vertices or points in the case of range images, then the cost of finding the closest point is \( O(n_v) \), and for \( n_s \) searches (queries) the cost is \( O(n_s n_v) \). When \( n_v \) is small, the cost of finding the closest point is small. However when \( n_v \) is large, the cost is very high, and so this crude algorithm is just unacceptable.

All of the algorithms presented below have a preprocessing stage, where data in the form of a triangulated surface or a range image is encoded. This organises the data to make geometric calculations easier and faster. The time taken for the encoding stage will vary according to the algorithm used but generally will be proportional to \( n_v \). The search time for finding one closest point using the methods that are about to be described will increase slowly as \( n_v \) increases. The amount of time required for finding all the closest points will be \( n_s t(n_v) \), where the time \( t \) depends on \( n_v \).

4.2.1 Vertex Based Search

A vertex based search is the simplest point to surface CP method. The method is given a query point \( p^1 \) and a search is performed to find the closest vertex contained within the surface mesh to the query point. Once the closest vertex has been found,
the adjacent faces which contain that vertex are determined and then each of those faces is searched for the closest point. The position of the closest point to the query point is then returned.

A brute force method for finding the nearest vertex, is to simply go through all the vertices determining the distance to the query point \( p_j \). For each query, this has a computational cost \( O(n_v) \), and for \( n_s \) searches the cost will be \( O(n_s n_v) \). This grows rapidly for large values of \( n_s \) and \( n_v \) and hence it would be desirable to have a more efficient method.

A more efficient method was suggested by Hilton et al. [45]. Assuming the vertices are approximately uniformly spaced, a simple spatial partitioning scheme of Hoppe et al. [48] can be used to encode the surface. A bounding box is determined for the surface and then that space is divided into cubes (voxels) of equal size \( \Delta b^3 \). All vertices which lie inside a voxel are added to a list which is associated with that voxel. These lists are then accessed by a hash table which is indexed by the voxel indices. To determine the nearest vertex to a query point \( p_j \), the voxel that the query point occupies is first identified. If the nearest vertex is within a distance \( \Delta b \) then it must lie within the query point voxel or one of its neighbours. Searching these voxels will find the closest vertex. If the closest vertex is in the range \( \Delta b \) and \( 2\Delta b \) away, then the next layer of neighbours away must also be searched. This continues until the closest vertex is within the distance of \( n\Delta b \). With this method, the computational cost is proportional to the product of the number of voxels searched and the number of vertices per voxel. When \( n_v \) is large and the voxels are sparsely occupied, the method can be considered to have almost constant cost compared to the brute force technique.

The problem with the closest vertex method of determining the closest point is that it can fail to give the true closest point. Consider the case shown in figure 4.6. A query point is given to the vertex based search method and the nearest vertex is obtained. Next, the connecting vertices A and B are used to determine the closest point on each of their faces. However, in this case the true closest point has been missed, as can be clearly seen in figure 4.6. This is referred to as the hidden
4.2. Closest Point Methods

Figure 4.6: Vertex based search: hidden point case (side on view of surface).

point case. Although this method guarantees finding the closest vertex, it cannot guarantee that the true closest point will be returned.

The vertex based search algorithm is very simple and hence has a low computational cost. The memory cost depends solely on the number of vertices that is associated with the encoded surface, since the method only stores each vertex once. As always there are tradeoffs between computational and memory costs, and in this case there is no guarantee that the closest point returned is the true one.

4.2.2 Surface Based Search

In the hierarchy of geometric entities at the next level above vertices, there is the triangular surface patches that form the surface. The surface based search method of Hilton et al. [45] works with an arbitrary triangulated mesh. The preprocessing stage of this method encodes the surface by orthogonally projecting the triangulated mesh down onto a 2D plane as illustrated in figure 4.7(a). The plane chosen could simply be a $x,y$ plane which makes projection relatively simple. The plane is uniformly divided into squares of size $\Delta b^2$ and each has an associated bucket. A bucket is simply a portion of memory that holds a group of records (triangles) that are accessed by the same index. If one assumes that the mesh is uniform then a reasonable value
Figure 4.7: Surface based search: encoding. (a) Triangulated surface being projected onto 2D plane of buckets. (b) A triangle projects into 1 square and is stored in one bucket. (c) A triangle projects onto several squares and is stored in several buckets.

for $\Delta b$ (which is associated with the squares of the planar grid, $\Delta b \times \Delta b = \Delta b^2$) is the average edge length of the triangles in the mesh. However if the mesh is non-uniform, i.e. there are large and small triangles, the average edge length is not a meaningful measure. Each triangle $t_i$ of the mesh is orthogonally projected down onto the plane. When a triangle intersects a grid’s square, then a reference to the triangle, including information on its 3 vertices and its normal, is stored in the bucket associated with the grid’s square. If a triangle is bigger than the grid size then when it is projected it will intersect several grid squares. Hence, several buckets can hold references to the same triangle. Figure 4.7(b) and (c) show this. The encoding process is summarised in algorithm 4.2.1.

To find the closest surface point for a query point $p_j^1$, it is orthogonally projected
Algorithm 4.2.1: Surface Based Search: Encoding.

SurfaceEncode(surfaceMesh)
{
    Initialise bucketStructure.
    For each triangle $t_i$ in surfaceMesh
    {
        Orthogonally project $t_i$ onto the plane.
        For all grid squares intersected by the projection
        Add the triangle reference to those buckets associated with the
        intersected grid squares.
    }
    Return bucketStructure.
}

down onto the 2D plane. The bucket associated with the square of the grid that
the projected point $p'_i$ occupies is read. For each triangle in the bucket, the closest
point is determined and the distance to $p'_i$ computed. Out of those computed points,
the point which has the shortest distance is noted as the closest point $p'_n$ and its
associated distance stored as distance $d$. The method uses a search termination
criterion $\| p'_1 - p'_n \| \leq n\Delta b = d_{\text{max}}$ which guarantees that the correct solution is
found. This means that the distance $d$ is less than or equal to the distance from the
original projected planar square to the border square of the current search$^1$. The
search termination criterion makes use of the property that for orthogonal projection
the distance between two points $p_1$ and $p_2$ in 3D is greater than or equal to the
distance between their orthogonal projections $p'_1$ and $p'_2$ on a 2D plane: $\| p_1 - p_2 \|
\geq \| p'_1 - p'_2 \|$. If the termination criterion is not met, the surrounding buckets
are searched, keeping note of the closest point and its distance. This is continued
until the distance $d$ of the closest point is less than the current maximum search
distance $d_{\text{max}}$. Due to this criterion several iterations may have to be performed.
This iterative search is illustrated in figure 4.8. When a query point is not close to
the surface, more iterations will have to be performed since its likely that the initial

$^1$Strictly speaking to a border square that is orthogonal to the original square.
Chapter 4. Determining The Closest Point

Figure 4.8: Surface based search: determining the closest point. (a) Projected point onto plane of buckets. (b) Determining closest point in current bucket, by searching through all triangles associated with that bucket. (c) Expanding neighbourhood, and looking for a closer point. Note: The neighbourhood is expanded after each iteration, until the closest point distance is less than the search plane distance.

bucket that is projected onto and its immediate neighbourhood will be empty. Thus more surrounding buckets will have to be searched to determine a closest point. Hence as the set of query points get further from the surface, the time taken to determine the closest points will increase.

It was noted earlier that several buckets can have references to the same triangle. During the iterative search, determining the closest point means that the surrounding buckets will have been searched through, and those that have duplicated references to triangles already searched constitute unnecessary computations. This
4.2. Closest Point Methods

Algorithm 4.2.2 Surface Based Search: Determining The Closest Points.

SurfaceGetClosestPoints($P^1$)
{
  For each point $p_i$ in $P^1$
  {
    $d = \infty$.

    Project $p_i$ onto the bucket plane.

    If the bucket is not empty
    {
      Go through each triangle $t_i$ in turn calculating the closest point.

      Keep the best closest point found $p'_n$ and store its distance in $d$.
    }

    $d_{\text{max}} = \Delta b$.
    While $d > d_{\text{max}}$
    {
      Increase Neighbourhood size $n$.

      $d_{\text{max}} = n \times \Delta b$.

      For each bucket in the neighbourhood
      { Determine the closest point $p'_n$ and store its distance in $d$.
    }

    Add $p'_n$ to list of closestPoints.
  }

  Return closestPoints.
}

inefficient behaviour can be avoided by having a searched flag associated with each triangle. This would be initialised to a false value before performing a search for one query point. Once a triangle is searched, the searched flag is set to true so that on subsequent bucket searches for the same query point, the closest point computation is not performed again for that triangle. Algorithm 4.2.2 shows how the closest points for the point set $P^1$ are determined but omits this efficiency trick for the reason of simplicity.
The performance of the method is affected by the value of $\Delta b$. If $\Delta b$ is too small, there will be a high resolution grid of buckets, meaning that there will be a lot of unnecessary searches since many triangles will be stored in 2 or more buckets. If $\Delta b$ is too large then there will be too few buckets, and therefore there will be a large number of triangles in each bucket. Thus, when evaluating a bucket for the closest point, it will be performing computations for all the triangles in the bucket, a lot of which are likely to be unnecessary. Striking a balance for $\Delta b$ is vital to get the optimum speed performance from this closest point method.

### 4.2.3 Voxel Search

A CP method that uses a volumetric method is now discussed. This method has two modes of operation, one for determining a closest point and another for performing surface fusion. A volumetric multi-resolution approach primarily used for surface fusion was presented by Hilton and Illingworth [42]. Their simpler single resolution technique will now be described in the context of determining closest points.

The volume of space that the surface occupies is uniformly divided into voxels (cubes) of volume $\Delta b^3$. Each facet of the surface is processed and a normal volume is built to a distance of $d_v$ and $-d_v$ using the vertex normals of the triangle as shown in figure 4.9(a). It is likely that several voxels will be inside a normal volume, as illustrated in figure 4.9(b). Thus for every voxel a closest point to the surface is computed. The computed point along with its triangle is stored in the voxel. If the voxel size chosen is small, then a normal volume is likely to encompass several voxels, whereas if the voxel size is large, a single voxel will span several normal volumes. If several triangles and their associated normal volumes occupy a single voxel, then the last normal volume to be be evaluated will have its closest point stored in that voxel, since only one closest point is stored in a voxel. Hence the voxel size should be chosen so that each voxel contains only a single triangle. This encoding is summarised in algorithm 4.2.3.

The voxel representation generally has a high memory cost. However this can be significantly reduced by using a simple compression method such as RLE (run length
4.2. Closest Point Methods

Algorithm 4.2.3 Voxel Search: Encoding.

\texttt{voxelEncoding(surfaceMesh)}
{
    \texttt{Determine volume of surfaceMesh.}
    \texttt{Uniformally divide volume into voxels of size } \Delta b^3.\texttt{ }
    \texttt{For each triangle } t_i \texttt{ in surfaceMesh}
    \{ \texttt{Create normal volume.}
        \texttt{For those voxel centres inside normal volume}
            \texttt{Compute closest point from voxel centre to } t_i \texttt{ and store in voxel}
            \texttt{along with } t_i.\}
    \texttt{Return voxels.}
}\n
encoding). To illustrate how RLE works, a simple example will be considered. Suppose we have a 2D 8 bit image of size 50 x 50. If the first row was all black, then the storage required for this row would be 50 x 8 bits, i.e. 400 bits. However, using RLE the same row of single coloured pixels could be represented by the start position of the row, its column and its length i.e. (1, 1,50). The required storage for this row is then only 3 * 8 bits, i.e. 24 bits. This is a significant saving in storage. A simple description of RLE can be found in Sonka [101].

A RLE scheme can be used with the voxel data structure if an arbitrary 2D plane is used and each grid position in the plane is associated with a RLE stream of voxels. This is illustrated in figure 4.9(c) which shows a 2D array (a x, y plane in this case) of RLE voxels for the 3rd dimension (the z axis). RLE of a voxel data structure was used by Curless et al. [19].

To find the closest point for a query point \( p_i \), the voxel which it resides in is first determined. If the point \( p_i \) lies in a voxel that is non empty, then the closest point is retrieved. This is referred to as direct lookup. However, if the voxel is empty, then an iterative procedure is performed which involves a neighbourhood search on
the plane defined by looking along the RLE axis of voxels. The neighbourhood is expanded at each iteration as long as a closest point is not found or while the distance $d$ associated with the closest point exceeds the current maximum search distance $d_{max}$. This guarantees that the closest point is found. Algorithm 4.2.4 shows how the closest point is determined.

The performance of this method is determined by the value of the voxel dimension $\Delta b$ and the strategy employed by this method. Encoding time may be large since all closest points are precomputed. However, search time will be fast. With respect
Algorithm 4.2.4 Voxel Search: Determine Closest Points.

VoxelGetClosestPoints($P^1$)
{
    For each point $p_i^1$ in $P^1$
    {
        Determine the voxel which $p_i^1$ occupies.
        If the voxel is non empty
            Get the closest point $p'_n$.
        Else
            $d = \infty$.
            While $d > d_{\text{max}}$
            {
                Increase neighbourhood size.
                $d_{\text{max}} = \text{neighbourhood size} \times \Delta b$.
                For each non empty RLE voxel in the neighbourhood
                    Get the nearest point $p'_n$ and store its distance in $d$.
            }
        Add $p'_n$ to list of closestPoints.
    }
    Return closestPoints.
}

To voxel size, if $\Delta b$ is too small, then when an iterative search is performed, a lot of voxels which contain the same information are searched. However, the accuracy of the closest point distance will be better because it is assumed that the query point is precisely in the middle of the voxel, since this is what is used in the encoding stage to precompute the closest points. The likelihood that the query point will be closer to the centre of the voxel will increase as the voxel resolution increases. If $\Delta b$ is too large, then when an iterative search is performed there are a lot less voxels to search, and so search time is extremely fast. However, there is the penalty of loss in accuracy. Assuming a uniform distribution of query points, the average error will be $\frac{\Delta b}{2}$. 
Using a voxel representation method tends to have a high memory cost. As has already been discussed the memory requirements can be reduced by using RLE. A further reduction in memory cost without loss of accuracy can be achieved by using a multi-resolution approach such as the one presented in Hilton and Illingworth [42].

4.2.4 Range Image Search

To allow the capability of registering range images a new closest point method for point to point set was developed, called the range image search. When building 3D models, range scanners are often used in the acquisition stage. The acquired data is in a 2D grid form where each position in the grid contains just depth information. Hence, to exploit the structure of this raw data, the range image search CP method was developed. This method works directly with the range data. The range image search closest point method takes a point set $P_1$ (which may or may not be a range image) and for each point $p_j$ in that set determines the closest point, which is either an explicit or implicit point, from the range image point set $P_2$.

The range image search method takes a simple approach. In the encoding stage the range image is divided up into what have been named range squares, each of which consist of 4 range points. A range square is considered to be valid if it has 4 valid range points, and if the surface normals of the two triangular facets (constructed from the range points) do not exceed a slope threshold of 1.46 radians ($\approx 84^\circ$). The value for the slope threshold was arrived at visually during the development of the cri (create synthetic range images) program which is discussed in more detail in chapter 5, section 5.8.1.1. The use of the slope threshold means that any bad data (usually around surface edges) acquired by a range scanner are ignored. The slope values are determined by constructing two triangles, top left and bottom right as illustrated in figure 4.10, in order to determine their surface normals. Range squares that have adjacent invalid range squares are marked as boundary range squares.

\[2\text{Boundary information is used later in the registration process to improve the robustness of the method. Also surface normal information is returned along with nearest point found. The range image, surface based and volumetric based methods all provide this information. This has been omitted from the description of the CP methods for simplicity.}\]
4.2. Closest Point Methods

This simple encoding method is summarised in algorithm 4.2.5.

When determining the nearest point a very simple projection technique is used. Given a query point $p_1^1$, a simple test is performed to see whether it lies within the $x,y$ extremities of the encoded range image $P^2$. If it is not within this range, then the projection has failed. The implementation of our method assumes that the range image has a uniform grid, meaning that there is a constant difference in distance between the columns, i.e. the $x$ components, and a constant difference in distance between the rows, i.e. the $y$ components. This grid resolution is denoted by $P_{dx}^1$ and $P_{dy}^1$. It is assumed that $p_1^1$ lies over $P^2$, and so the corresponding range square can be determined by just taking the $x$ and $y$ components into account. Each range square is identified by the top left corner of the range square. This identifying point will also be shared by other neighbouring range squares, since each edge of a range square shares two of its range points with its neighbour. The projection is illustrated in figure 4.11 which also shows how the range squares share their edges.

Once the range square which the projected point $p_1^1'$ occupies has been determined, it is tested to make sure that it is a valid range square. If it is valid, the closest

---

3The actual query point is not projected onto the range square, but its $x,y$ value is used to identify its corresponding range square. Unlike other CP methods, the distance to the query point and not its projected self is used during the search.
Algorithm 4.2.5 Range Image Search: Encoding.

RangeEncode(rangeImage)
{
    Divide rangeImage into rangeSquares and set them to valid.

    For each rangeSquare $s_i$
    {
        If the 4 range points of $s_i$ are not valid
            Set $s_i$ to invalid.
        Else
            Compute surface normals for top left and bottom right triangles.
            If surface normals $\geq$ slopeThreshold
                Set $s_i$ to invalid.
    }

    For each rangeSquare $s_i$
    {
        If $s_i$ is valid and adjacent range squares are invalid
            Set $s_i$ to boundary.
    }

    Return rangeSquares.
}

point to the query point $p^1$ is determined for both constituent triangular patches of the range square. The closest point is taken as the minimum of these two Euclidean distances.

The neighbourhood is then expanded\(^4\). This always happens regardless of whether the original projected range square was valid or not. The concept of the neighbourhood is shown in figure 4.11(d) where the neighbouring range squares of the original projected range square are searched. If the termination criterion is not met then the neighbourhood is expanded again as shown in figure 4.11(e). The criterion for this method comprises of the maximum search distance $d_{\text{max}} = n \times \max(P_{dx}, P_{dy})$ and a

\(^4\)The neighbourhood expands to a maximum size defined by a constant $n_{\text{max}}$. By default, $n_{\text{max}}$ has the value of 2, and so the neighbourhood always expands. However, if it was to have the value of zero, then only the projected range square would be examined, and no other range square.
4.2. Closest Point Methods

Figure 4.11: Range image search: determining nearest point. (a) Projecting a query point onto the range image. (b) Seeing a 2D representation of (a), where depth can no longer be perceived. The circle being the projected point. (c) Determining nearest point in the range square. (d) Expanding the search and looking at the neighbouring range squares to determine the nearest point. (e) Expanding the search again and determining the nearest point. The search is expanded until the termination criterion has been met.

maximum searchable neighbourhood $n_{\text{max}}$. By introducing the extra constraint of $n_{\text{max}}$ the maximum search time remains constant. Algorithm 4.2.6 summarises the determination of the closest points.

Due to the simple nature of this algorithm there are two cases when a closest point cannot be found. The simplest case is when the projection fails, the result is that an associated weight is set to zero to indicate that failure. Another case is when a
Algorithm 4.2.6 Range Image Search: Determining Closest Points.

RangeGetClosestPoints($P^1$)
{
    For each point $p_i^1$
    {
        Project $p_i^1$ onto the range image.

        If projected point is inside the range image
        {
            Determine the range square that $p_i^1$ has been projected to.

            If the range square is valid
                Determine the closest point $p'_n$, and store its distance in $d$.

            // $n = 1$  3x3       $n = 2$  5x5
            Expand out the neighbourhood, and update $n$.

            // Neighbourhood Threshold, $n_{max} = 2$
            While $d > d_{max}$ and $n \leq n_{max}$
            {
                For each valid range square on the perimeter of the neighbourhood
                {
                    Calculate the local closest point.

                    If $d_{local} < d_n$
                        Update $p'_n$ and $d$.
                }

                $d_{max} = n \times \max(P_{dx}^1, P_{dy}^1)$.

                If $d > d_{max}$
                    Expand the neighbourhood and recalculate $n$.
            }
        }
    }
    Else
        Set $p'_n$ to invalid.

    Add $p'_n$ to list of closestPoints.
}

Return closestPoints.
projected point lands in a region where all the range squares within the maximum searchable neighbourhood are invalid. Again its associated weight would be set to zero, meaning that the registration algorithm can still cope with this. However, if either of these cases happens for every query point then the returned point set will be completely invalid. Also due to the way the range image is encoded, the true approximate closest point for some query points will never be returned, as valid range points contained in invalid range squares have been effectively thrown away.

The memory cost for this method is directly proportional to the number of range points in the encoded range image. The algorithm cost is not necessarily constant because of the iterative nature of the algorithm. However, due to the extra constraint of there being a limit to restrict the maximum neighbourhood size to search, the amount of time used to perform a search has a constant limit. Also because of this extra constraint, finding the correct approximate closest point cannot be guaranteed if the correct approximate closest point lies outside the maximum searchable neighbourhood size.

4.3 Summary

In this chapter the meaning of a surface has been discussed, along with various representations. There are various trade-offs for each representation, for example the ease of computation with linear approximations versus polynomial surfaces with smooth curvature. Another factor is the amount of storage required. Generally the less storage used, the greater likelihood that the accuracy will be affected as the surface becomes more of an approximation. Finally the amount of computation required to determine a point on the surface is important.

Three closest point methods that work on triangulated surfaces have been presented with one method which works with a range image. The factors likely to affect accuracy, speed and possible failures have been discussed. The setting of parameters that affect each algorithm may have to be determined empirically for some cases so that better results can be obtained in respect to accuracy.
Chapter 4. Determining The Closest Point

Closest point methods can use any surface representation providing a method exists for determining a point on that surface representation. Determining such a point for complex techniques may require some considerable time. As always, the requirements for the application will determine the surface type and resulting technique used.

When the query set is close to the surface to be searched, the closest point methods will be efficient. However, as the query set moves away from the surface to be searched, so that it is no longer above the surface or close to it, the efficiency is reduced, meaning that the search time will most likely significantly increase. The surface based search and voxel search will keep iterating until a solution is found. These methods can also have a maximum distance threshold set to terminate the search and in the same way they may or may not return a valid estimate of the closest point. In the case of the range image search a maximum neighbourhood search size is defined and hence time will not be severely affected for reasonable close query points, but whether a point will be found is affected. However, if the query point set is quite far from the surface then the projection for each query point is most likely to fail.

Out of the nearest methods presented in this chapter, only three of them are actually used in the surface registration process discussed in the next chapter. These being the surface based search, voxel search and range image search. All of these three methods determine whether the nearest point found lies on the boundary and the surface normal associated with it, as was mentioned briefly earlier. A point is considered to be on the boundary for the surface based search and voxel search, if the point lies on an edge or vertex of a triangle, where that triangle does not have two adjacent triangles. How the surface registration method uses this additional information is discussed in the following chapter.
Chapter 5

2 View Surface Registration

The first step in 3D surface reconstruction of real world objects is to acquire depth information of the real object, usually via a range scanner. However, a range scanner cannot acquire all the information that is required to reconstruct the whole surface in one go. That is, a range scanner will only acquire depth information for what it can see from a particular view. Hence several different views are required to have all of the surface information relevant to the real world object. Depending on the complexity of the object, many views may be required for certain regions of the object due to occlusions. At the end of the acquisition stage, there will be several views of the real world object with corresponding range images. These range images will most likely be processed to yield polygonised models (for example by using the marching cubes algorithm by Lorensen & Cline [74] or the marching triangles algorithm by Hilton et al. [43] or the ball-pivoting algorithm by Bernardini et al. [10]) turning them into surfaces for the next stage of processing, reconstruction of the whole surface model.

When range images are acquired, the transformations between successive range images may not be known. The exception is when using a range scanner that is mounted to an arm which feeds the computer the current sensor position. However, there will always be an incomplete model as long as some parts of the object are invisible in all views. For example, from a stable pose of an object several range images will be acquired for different views. However, to get a full surface description, it may
be necessary to physically move the object as the surface on which it rests makes surface parts of the object invisible.

If for the successive range images that have been obtained, the transformations between them are unknown or inaccurate, then the data itself has to be used in some way to infer the required transformations to best register them. If there is surface overlap between two range images then determining the transformation ought to be possible. However, it is made difficult as corresponding data must be identified. Also, account must be taken of the fact that the measured surfaces will not be identical due to noisy sensing and artifacts etc.

To build a full 3D surface model a pairwise incremental process can be used. First, two views are registered and then fused (integrated) into one surface. Next a subsequent view is taken and registered to that recently fused surface. These are then also fused. For each remaining view, the pairwise process continues registering a view with the growing surface model, until the full 3D surface model is finally obtained. This incremental process of registering two surfaces at a time will be referred to as 2 view surface registration.

In chapter 1, the standard iterative closest point (ICP) algorithm was introduced for registering two surfaces. The ICP algorithm performs reasonably well when there are large overlaps between surfaces. However, in reality the surfaces obtained from the acquisition stage will not have a high degree of overlap. Therefore, any correspondences established by false matches between the non-overlapping regions will cause the registration process to diverge away from the true solution. Also, the standard ICP algorithm has no means of coping with surface noise. Therefore, it is desirable to improve the standard ICP algorithm so that it is more robust to noise and mismatches. In this chapter, the standard ICP algorithm will be covered in more detail and techniques to improve its robustness will be introduced. The ICP algorithm consists of two key components: a point set alignment method and a closest point method. The closest point method being the most time consuming part of the ICP algorithm. Thus the work in this chapter examines how the choice of the closest point method affects the overall performance of the ICP algorithm. As
well as evaluating the usefulness of robustness procedures introduced and how the ICP algorithm is affected by varying degrees of displacement between the surfaces to be registered.

5.1 Overview Of The Iterative Closest Point Algorithm

Point set alignment is a key component of the ICP algorithm that estimates the transformation required to align two corresponding point sets. The point set alignment method uses a least squares fitting technique (see 3.2) where the distances between corresponding points on the two surfaces that we want to register is minimised. Therefore, point correspondences need to be established between the two surfaces. If true correspondences are established, then once the cost function is minimised the surfaces will be registered.

To establish these correspondences (or approximate correspondences) a closest point method is used. If one surface called the moveable surface, is sampled, then a point set can be obtained and the corresponding set of points on the fixed surface can be calculated using a closest point method. A rigid body transformation that causes the corresponding points to coincide can be calculated. This transformation is then applied to the moveable point set to bring it into registration with the fixed surface. However, it is unlikely it will be correctly aligned if the correspondences used are incorrect. However, it can be shown that under certain conditions the moveable surface will have moved closer towards its correctly registered position. Hence, iterative application of this strategy will lead eventually to proper registration. The process of establishing correspondences and estimating the transformation continues until some criterion is met. The composition of transformations ascertained through this iterative procedure is the transformation that registers the original moveable surface to the original fixed surface. This procedure is known as the iterative closest point (ICP) algorithm and was developed by Besl and McKay [11].

The ICP algorithm works by performing an iterative minimisation of distances between corresponding points which have been determined by a closest point method. From a closest point method perspective, the sampled points from the moveable
Chapter 5. 2 View Surface Registration

Figure 5.1: Flowchart overview of 2 view surface registration.
surface are the query points. Each query results in a corresponding closest point located on the fixed surface. Thus, two sets of points with correspondences are established. These will be referred to as a correspondence set. This correspondence set is given to a 2 viewpoint set alignment method which determines the optimal transformation to align these two point sets. This transform is then applied to the moveable point set and the process iterates until a criterion has been met. Further details of the registration termination criterion can be found in section 5.3. The ICP process of registering two surfaces is summarised diagrammatically in figure 5.1.

5.2 Measuring Error And Incorporating Robustness

To be able to determine whether the registration of two surfaces is correct, a quantitative error measure needs to be defined. The error might use the distance between corresponding points associated with the two surfaces. As these are generally unknown the distance between closest points as determined in the ICP algorithm must be used. Many error measures can be proposed. We will use the simple sum of distances as it is easy to compute and in many simple cases it is a good approximation to the ideal measure of error. To measure the error, the distances between the corresponding points in the sets associated with the fixed and moveable surface will be used. During each iteration $k$, the error will be computed, and if it meets the registration termination criterion, then the registration process will stop.

The concept of a weight which signifies our confidence in a measurement was introduced in section 3.2.3. These can be used to increase the robustness of a statistical procedure, i.e. a way to ensure that the procedure is not severely affected by errors (outliers) in the data. If the value of the weights have been set accurately, e.g. all outliers have their corresponding weights set to zero, then a more accurate measure of error can be obtained. If the corresponding point sets of the moveable and fixed surface have $n$ points, then a set of weights $W$ associated with the correspondence sets is defined as follows.

$$W = \{w_i \in \mathbb{R} : i = 1 \ldots n \text{ and } 0 \leq w_i \leq 1\}$$  (5.1)

Initially in the ICP algorithm all weights are set to the value of 1, i.e. each point
is given equal importance. During each iteration a closest point is determined for each moveable point. If one cannot be found the corresponding weight is set to zero. If robustness procedures are in use, then the values of the weights are modified, see section 5.5 for more details. Then the transform is estimated and applied, and the resultant error is computed. This process iterates until registration has been achieved. At the beginning of each iteration the weight values are re-initialised to the value of 1, since a new correspondence set is about to be built.

5.2.1 Mean Square Error

The most commonly used measurement of error in statistics is the mean square error (MSE). The MSE calculates the average squared distance between the point sets. If the distance is small then the MSE error will be small, and this indicates that the registration is fairly good. However, the use of the MSE does not guarantee that the registration is correct when there is a degenerate case, see section 5.4 for further details.

As already mentioned we have two corresponding point sets, the fixed point set with its elements denoted as $f_i$, and the moveable point set with its elements denoted as $m_i$. For each corresponding $f_i$ and $m_i$ there is a corresponding weight $w_i$. If methods are available for determining the confidence in an observation or measurement, then a weighting scheme can be incorporated into the MSE as shown below.

1. Create a list of distances $D$.

$$D = \{d_i \in \mathbb{R} : d_i = \| m_i - f_i \| \text{ where } w_i > 0 \text{ and } i = 1 \ldots n\}$$  \hspace{1cm} (5.2)

2. Calculate $MSE^k$

$$MSE^k = \frac{\sum_{i=1}^{N} d_i^2 * w_i}{\sum_{i=1}^{N} w_i}$$  \hspace{1cm} (5.3)

where $MSE^k$ is the MSE for iteration $k$.

The root mean square (RMS) error can be computed from the MSE by simply taking the square root of it.
5.3. The Registration Termination Criterion

5.2.2 Median Absolute Deviation Estimator

The robustness of a least squares method depends on how reliably the error is computed. The RMS is only a good estimator if the noise present is Gaussian and there are no significant outliers. Therefore if there are outliers and/or the noise is non Gaussian, then something more suitable needs to be used. A good choice is the median absolute deviation (MAD) estimator which is a measure of error that is insensitive to both distribution assumptions and outliers. Further information on MAD and other estimation techniques can be found in Zhang [120] and references therein. The modified method of the MAD estimator using weights is shown below.

1. Create in ascending order a list of distances \( D_s \) where the weights associated with \( M \) (the moveable point set) and \( F \) (the fixed point set) are non zero.

\[
D_s = \{ d_i \in \mathbb{R} : d_i \leq d_{i+1} \text{ where } w_i > 0 \text{ and } i = 1 \ldots n \} \tag{5.4}
\]

2. Calculate the median of \( D_s \), \( d_{\text{median}} \).

3. Calculate the root mean square error.

\[
RMS = 1.4826 \times d_{\text{median}} \tag{5.5}
\]

4. Calculate the MSE.

\[
MSE^k = RMS^2 \tag{5.6}
\]

The factor of 1.4826 is chosen to produce an estimate consistent with least squares for a Gaussian distribution.

5.3 The Registration Termination Criterion

As already mentioned, after each iteration the mean square error is calculated, from which the square root is taken giving the root mean square error, RMS. During the registration process, one would expect the RMS error of the subsequent iteration to be less than the current iteration’s RMS error. Therefore, towards the end of
registration, the distribution of RMS error as a function of iteration number should
flatten out and so have a gradient of almost zero. Thus if the differences in RMS
for the last three iterations are almost zero then registration can be assumed to
be complete. The registration termination criterion uses the ratio of successive
RMS values which should approach one as registration nears completion. Before
the termination criteria can be applied, at least three iterations need to have been
performed since three RMSs are required for the ratios. The current RMS is denoted
as $RMS^k$ and the previous two RMSs are denoted as $RMS^{k-1}$ and $RMS^{k-2}$. Ideally
the RMS threshold used with respect to the ratios would have the value of 1, but due
to numerical rounding errors it is set to 0.999. The rules governing the registration
termination criterion are shown more precisely below.

1. If $k \geq Max\ Iterations$ then send termination signal. (Puts a limit on the
maximum number of iterations allowed.)

2. Calculate $RMS^k$, $RMS^{k-2}$ and $RMS^{k-3}$, where $RMS^k = \sqrt{MSE^k}$,
$RMS^{k-1} = \sqrt{MSE^{k-1}}$, $RMS^{k-2} = \sqrt{MSE^{k-2}}$.

3. If $RMS^k \leq 1 \times 10^{-20}$ then send termination signal as registration is complete.
(Terminates if RMS approaches machine precision limits.)

4. If $k < 3$ then do another ICP iteration. (Must do at least three iterations
before doing test in step 5.)

5. If $\frac{RMS^k}{RMS^{k-1}} > rmsthreshold$ and $\frac{RMS^{k-1}}{RMS^{k-2}} > rmsthreshold$ then send the termi-
nation signal as the registration criterion has been met, else do another ICP
iteration.

5.4 Degeneracy, Locking Features And Local Minimum

When registering two surfaces, the assumption has been made that if the RMS is
zero or very nearly so, then the registration is complete and the transform found is
the correct one. However, the actual registration obtained may be incorrect. For
example, if we had two planar surfaces, the moveable surface being smaller than the
fixed one, then when registered, the error would be zero. However, the moveable surface could be placed anywhere on the fixed surface and would yield an error of zero. This is just one example of a degenerate case.

The possibility of having degeneracy when registering two surfaces should be taken into account when building models. Therefore, it is desirable to have what have been called locking features in the overlap of the two surfaces. This is where the surface has very distinctive localised curvature patches that appear in both range image scans. This then ensures that degeneracy is reduced and the registration algorithm can home in on a unique transformation, i.e. there is a single global minimum MSE configuration to allow the two surfaces to lock together at the correct points.

It should also be noted that the standard ICP algorithm suffers from a local minimum problem. When registering, all that is happening is that the average distance between the two surfaces is being minimised by finding an appropriate transformation. As the registration process proceeds, the RMS error gets less as the transform gets nearer the required transformation needed to have the correct registration. However, along this journey the RMS may suddenly decrease and then start to rise again as illustrated in figure 5.2, meaning that the registration algorithm believes its found the answer. Unfortunately in this case it has found a local minimum and not the global minimum. Further adaption of the ICP algorithm has been done to try to reduce the possibility of the ICP algorithm getting stuck in a local minimum and is now discussed.

Figure 5.2: ICP and local minimum.
5.5 Extra Checks For Improvement Of Registration

In section 5.2 the idea of incorporating weights into statistical procedures, in our case least squares, to improve their robustness was reintroduced. Also the idea of having weights in procedures for calculating the overall error was introduced so that a more accurate value for the error could be obtained by reducing the influence of any outliers. Therefore the ICP algorithm can have its robustness improved by incorporating weights. Hence the ICP algorithm will now solve the registration problem using an iterated re-weighted least squares method.

To take advantage of the robustness weights can offer, reliable methods of setting weight values need to be employed. Three such methods are discussed shortly, but first the role that the closest point method plays in setting weight values is discussed. When building a set of corresponding points their associated weight values under ideal circumstances will all be 1. However, if the closest point method being used cannot determine a closest point, then its corresponding weight will be set to zero. In our implementation of the ICP algorithm all weights are initialised to the value of 1 at the start of each iteration. If the weights were not reinitialised, then in the next iteration any query points that had associated weights with the value of zero would be ignored by the closest point method. Obviously this is an undesirable affect, leading to the possibility of ending up with a query set where all its associated weights were zero. Also, it would not make sense to keep the same weightings since an entirely new correspondence set is being created each iteration.

The flowchart in figure 5.1 shows a fairly simple representation of ICP algorithm where the RMS (the square root of the MSE) only gets calculated once per iteration after the transform has been applied. However, extra checks (omitted from the flowchart) of which one is novel, have now been introduced which alter the value of the weights. Hence, the value of the MSE will become more accurate once recomputed using the new weight values. The downside of this is that the MSE needs to be recomputed more than once per iteration. Only one of the improvement checks uses the current value of the MSE. Therefore it is advantageous to have a more accurate value for the MSE even though there is a slight expense of recomputing the MSE.
before using that particular check. At worst the MSE needs to be calculated twice per iteration. Firstly for the improvement check that requires it, and secondly after the rigid transform acquired has been applied.

5.5.1 Boundary Check

The key to success for surface registration is being able to determine corresponding points accurately. If registration of a surface to itself were to be performed, then for each query point, a true corresponding point is known to exist. However, when building surface models, the rigid bodies used will not be identical and there will generally be only a partial overlap between the two views. This means that for the regions of the surface that do not overlap, no true corresponding points exist. Hence when registering two surfaces with partial overlap, not all corresponding (closest) points returned by the closest point method will be the true corresponding points. Figure 5.3 illustrates such an example where the query points for the non-overlapping region of the moveable surface is associating the closest corresponding points with the boundary of the fixed surface, and as a result all of those particular query points are ending up with the same closest point. This would mean that when solving for the rigid transform an incorrect drag associated with these false correspondences would occur. Existing registration methods in the literature take the approach of ignoring all correspondences that lie on the boundary of a surface, and therefore potentially throw away some good matches. Thus a novel method called the boundary check is introduced which uses distance information associated with the correspondences to filter out bad matches that lie on the boundary of a surface.

Each query point given to the closest point method has returned its closest point, along with additional information. This information includes whether the closest point lies on the boundary of the surface (as defined by the closest point method used), the surface normal (the normal associated with the triangular plane that the closest point lies on) and the distance to the query point. Therefore when building up the corresponding point set of closest points $F$, if a member that is about to be added to $F$ is indicated by the closest point method as lying on the boundary,
then that point has the boundary check applied. The boundary check makes use of the standard deviation (RMS) from the previous iteration. The current standard deviation is not used since the correspondence set is not yet fully built. However, the standard deviation is known from the previous iteration\(^1\) and is therefore the best measure of error currently known.

The value of a weight can be anywhere between zero and one. One indicates full confidence, and zero indicates absolutely no confidence. Using the standard deviation (RMS) \(\sigma\) of the distances between corresponding points, along with a closest point's distance \(d_i\) to its query point and its boundary information, its corresponding weight can be set appropriately. If a point returned from a closest point method lies on the boundary of a surface and is a good correspondence, then its associated distance to the query point will be small. However, if it is a bad correspondence, its associated distance to the query point will be large. To define what is considered large, \(\sigma\) is used. For a Gaussian distribution certain characteristics are known regarding the population under the curve, and how they are spread with respect to the standard deviation. From this it is known that 95% of a population lies under the curve within

\(^1\)From an implementation point of view during the very first iteration, the standard deviation for the previous iteration does not really exist, and so is initialised to infinity. In reality, this is the maximum value possible for the floating point representation specific to the machine's architecture that the ICP code is being executed on.
5.5. *Extra Checks For Improvement Of Registration*

Figure 5.4: Boundary check weight function. The dashed line shows where the slope would continue to $3\sigma$ if it was allowed to proceed past $2\sigma$.

$2\sigma$. Thus non-boundary correspondences should yield distances within $2\sigma$. Therefore any points that lie on the boundary of a surface that have distances greater than $2\sigma$ will be considered as incorrect correspondences and have their weights set to zero. It should be stressed that all the distances between a query point and its corresponding closest point have been made absolute. Points that lie on a boundary where their distance is less than or equal to one standard deviation are considered good correspondences. Those points which lie on a boundary and have distances between $\sigma$ and $2\sigma$ have their weights linearly decreased as they move towards $2\sigma$ using a ramping function, that if allowed would reach zero at $3\sigma$, as illustrated in figure 5.4. The boundary check weight update is more precisely described below.

1. Use the standard deviation $\sigma$ from the previous iteration.

$$\sigma = \sqrt{MSE^{k-1}}$$ (5.7)

2. For the boundary point $p_i$, update its corresponding weight $w_i$.

(a) If $d_i < \sigma$ then $w_i$ remains the same.

(b) Else

i. If $d_i < 2\sigma$ then

$$w_i = w_i \ast \left(1.0 - \frac{d_i - \sigma}{2\sigma}\right)$$ (5.8)
ii. Otherwise \( w_i = 0 \).

When the two surfaces that are to be registered have a large displacement between them, it is likely that the majority of all initial correspondences determined by the closest point method will lie on the boundary of the fixed surface. Thus by using a boundary check weight function that accepts boundary points with distances between \( \sigma \) and \( 2\sigma \) (although there is less confidence associated with them), the registration process is less likely to be hindered.

### 5.5.2 Normal Angle Check

Each point on a surface has a corresponding surface normal. Therefore when registering two surfaces, each pair of corresponding points \((m_i, f_i)\) that truly correspond should have a small angular difference between their surface normals. Large angle differences can therefore be used to identify bad correspondences. Such an example of bad correspondences is illustrated in figure 5.5 where the surface normal difference is shown by the arrows. Using angular differences between correspondences tends to be commonly used in registration methods for determining good correspondences. Some methods use a simple thresholding scheme where correspondences are simply accepted if their angular difference is below the angular threshold, otherwise they are rejected. Other methods will only accept correspondences if the surface normals point in the same direction.

![Figure 5.5: Bad correspondence pairs shown by difference in surface normals orientation.](image-url)
To be able to implement the normal check, two additional sets containing surface normal information need to be introduced. We denote $\mathcal{N}_\mathcal{F}$ as the set of normals corresponding with $\mathcal{F}$, and $\mathcal{N}_\mathcal{M}$ as the set of normals corresponding with $\mathcal{M}$. These are more precisely defined as follows.

$$\mathcal{N}_\mathcal{F} = \{n_{f_i} \in \mathbb{R}^3 : i = 1 \ldots n\} \quad (5.9)$$

$$\mathcal{N}_\mathcal{M} = \{n_{m_i} \in \mathbb{R}^3 : i = 1 \ldots n\} \quad (5.10)$$

The angular differences that are considered acceptable and unacceptable are now defined. The maximum value for acceptable angles is denoted as $\delta \theta$. Weight values remain unchanged, providing the angular difference between surface normals do not exceed $\delta \theta$. Once the angle exceeds this threshold, the weight value will linearly decrease towards zero. The point at which it reaches zero is a second threshold $\delta \phi$. The two values used for the thresholds are: $\delta \theta = 40^\circ$ and $\delta \phi = 70^\circ$. By using these thresholds, it allows surfaces that have initially a large rotational difference between them to be registered. It also allows for the variations of angle between surface normals due to noise. The weight function for the normal angle check is illustrated in figure 5.6. The normal angle check weight update is described below.

For each $w_i > 0$

1. Calculate $\theta$, where

$$\theta = \arccos(n_{m_i} \cdot n_{f_i}) \quad (5.11)$$
2. If \( \theta < \delta \theta \) then \( w_i \) remains the same.

3. Else
   
   (a) If \( \theta < \delta \phi \) then
   
   \[
   w_i = w_i \ast \left( 1.0 - \frac{\theta - \delta \theta}{\delta \phi - \delta \theta} \right)
   \]  
   (5.12)

   (b) Otherwise \( w_i = 0 \).

### 5.5.3 Distance Check

After generating a correspondence set, one way of assessing how good the correspondences are is by using their distance as an indicator. If all correspondences in the set are good, then the distance between each corresponding pair of points should be close to the average distance, the mean square error. In a correspondence set where the correspondences are not all good, bad correspondences will have larger distances than the average. Such an example is shown in figure 5.7 where there are bad correspondence pairs are shown by the dotted (rather than dashed) lines. Zhang [119] introduced this idea based on distance and only accepted correspondences with a distance less than a maximum tolerance distance. However, this maximal tolerance is

---

**Figure 5.7:** Three bad correspondence pairs shown by a large difference in distances compared to the average.
computed using rules that involve a parameter that has to be user defined. This parameter indicates the average distance expected when the registration is considered good, and therefore tends to have a value associated with the resolution of the data. The distance check used within the modified ICP implementation uses a data independent method that uses the distribution of the distances between corresponding points to appropriately modify the weight of each corresponding pair of points. The distance check is not applied during the construction of the corresponding point set, as it would have to use the MSE from the previous iteration which would represent that iterations correspondence set and not the current iteration's correspondence set. Once the correspondence set has been built, the MSE is computed and then the distance check is applied. The weight function of the distance check also uses a linear ramp, where the weight remains at the value of one up to one standard deviation and decreases linearly to zero at $3\sigma$. The distance check weight update is described more precisely below.

For each $w_i > 0$

1. If $d_i^2 > 3 \times MSE^k$ then $w_i = 0$.

2. Else

   (a) If $d_i^2 > MSE^k$ then

   $$w_i = 1 - \frac{d_i^2 - MSE^k}{2 \times MSE^k}$$

   (5.13)

   (b) Otherwise $w_i$ remains the same.

5.6 The Modified Iterative Closest Point Algorithm

The modified iterative closest point (ICP) algorithm solves the registration problem using an iterated re-weighted least squares method. An initial guess in the form of a transformation is supplied. This guess will move the moveable surface so that it is close to the fixed surface. If the required transformation is unknown, then a null transformation can be passed, i.e. one with no translation or rotation. Next, the moveable surface is sampled to obtain a point set $\mathcal{M}$. The fixed surface is CP
(closest point) encoded. At the start of each iteration the weights associated with
the moveable points \( \{m_i\} \) and their corresponding closest points are initialised to
the value of 1. The moveable point set \( \mathcal{M} \) is passed as a query set to a CP method
which obtains another point set \( \mathcal{F} \) containing the corresponding closest points on the
fixed surface to \( \mathcal{M} \). If the CP method failed to find a closest point for a query point
then its weight is set to zero. If the boundary check is in use, then when each closest
point is returned, its weight is appropriately modified. If the normal check is in use,
the angular differences between the corresponding point pairs are checked and their
weights modified accordingly. Next, the MSE is computed only if the distance check
is to be used. After the robustness checks have been applied, the transformation
required to minimise the distances between corresponding points is estimated using a
2 view point set alignment technique. The resultant transformation is applied to the
moveable point set \( \mathcal{M} \) and the MSE is recomputed. If the registration termination
criterion is not met, the process iterates, otherwise the surfaces are assumed to be
registered. Our modified ICP algorithm is described more precisely below.

1. Assume we are given the initial guess \( g^0 \).

2. Sample points from the moveable surface and apply \( g^0 \) to them and store in
the set \( \mathcal{M} \).

\[
\mathcal{M} = \{m_i \in \mathbb{R}^3 : i = 1 \ldots n\} \quad (5.14)
\]

3. Let \( k = 0 \), where \( k \) denotes the iteration.

4. Iterate over \( k \) while the termination criteria has not been met (see section 5.3).

   (a) Initialise weights.

\[
W = \{w_i \in \mathbb{R} : \forall w_i = 1 \text{ and } i = 1 \ldots n\} \quad (5.15)
\]

   (b) Determine the closest points for \( \mathcal{M} \) using a closest point (CP) method
and store in \( \mathcal{F} \).

\[
\mathcal{F} = \{f_i \in \mathbb{R}^3 : f_i = \text{Closest}[g^k \ast m_i] \text{ and } i = 1 \ldots n\} \quad (5.16)
\]

If the CP method could not find a closest point for \( m_i \), then \( f_i \) is invalid
and the corresponding weight \( w_i \) is set to zero.
While determining the closest point for each query point, if the boundary check is in use, it appropriately modifies its corresponding weight.

(c) If the normal check is in use, the weights $W$ are appropriately modified.

(d) If the distance check is in use, calculate the new $MSE^k$ and modify the weights $W$ as necessary.

(e) Calculate the transform $\Delta g$ to minimise equation (5.17), to align the moveable point set $\mathcal{M}$ with their corresponding closest points $\mathcal{F}$ that lie on the fixed surface.

\[
E(\Delta g) = \sum_{i=1}^{n} w_i (f_i - \Delta g \circ g^k \ast m_i)^2
\] (5.17)

Then set transform $g^{k+1}$ to

\[
g^{k+1} = \Delta g \circ g^k
\] (5.18)

(f) Apply transform, $g^{k+1} \ast \mathcal{M}$, and recompute $MSE^k$.

(g) Test the registration termination criterion (see section 5.3). If not met, increment the iteration $k$.

\[
k = k + 1.
\]

5. End of while. Return the rigid transform $g^{k+1}$ that was obtained from the final iteration.

5.7 The Accelerated ICP Algorithm

Besl and McKay noticed when performing experiments with the ICP algorithm that there was fast convergence during the first few iterations but that it then slowed down as it approached the local minimum. They also noticed that the registration convergence happened in an exponential way and so decided to take advantage of it by performing prediction to speed up the process of registration.

When performing registration the translation $T$ ($3 \times 1$ vector) and rotation $R$ ($4 \times 1$ quaternion vector, see section 3.1.5.4) are determined each iteration to minimise the
Figure 5.8: Accelerated ICP: exploiting properties of the convergence curve.

MSE. As the MSE decreases each iteration, $T$ and $R$ get nearer to their optimal values. This iterative optimisation can be seen as a process which involves selecting a direction and a step size a multidimensional space each iteration. Figure 5.8 shows a schematic illustration of how the MSE might vary as a function of this multidimensional space ($R$ and $T$) for various iterations, $k$. Thus, during each iteration of the ICP algorithm a new registration vector $q^{k}$ consisting of $R$ and $T$ is computed reducing the value of the MSE error. By using three consecutive registration vectors ($q^{k-2}$, $q^{k-1}$, $q^{k}$) a transformation that further reduces the MSE can be predicted under the right conditions. This can either be linear, parabolic or higher order prediction. If no prediction update can be used then a maximum allowable update is performed only if the right conditions are met. From the last two registration vectors $q^{k-1}$ and $q^{k}$ the transformation made between them can be determined, i.e. $\Delta q^{k} = q^{k} - q^{k-1}$. It is assumed that the transformation $\Delta q^{k}$ is reliable, and so the next transformation to be used by the maximum allowable update is simply $q^{k+1} = q^{k} + n\Delta q^{k}$. A suitable value for the step size $n$ was found through experimentation by Besl and McKay to be 25.

When calculating the gradient for a straight line, two things need to be known. The
horizontal distance between two points and the vertical distance between them. By using approximate arc length values, a coordinate system for the horizontal axis can be built up where the last registration vector $q^k$ will effectively be treated as being at the origin $v^k$. The previous two registration vectors will have the coordinates $v^{k-1}$ and $v^{k-2}$ which will be on the negative side of the axis allowing the horizontal distance between points to be computed. During each iteration $k$ a MSE value will be associated with the registration vector $q^k$. Thus the parametric function $d(k)$ is introduced which returns the value of the MSE for the iteration specified. The MSE values returned by the parametric function are treated as coordinates on the vertical axis. To be able to do a linear update the two coefficients of the straight line equation need to be determined, and then the zero crossing of the straight line where the MSE will be zero (the optimal update) is ascertained. The resultant rigid transform will be $q^k$ plus some distance along the vector, i.e. a coefficient times the unit vector of $q^k$. The mathematical notation relevant to the original figure 5.8 is re-summarised in figure 5.9.

To do the parabolic update, Besl and McKay fit a polynomial curve to the last three iterations' MSE values, denoted parametrically as $d(k-2)$, $d(k-1)$ and $d(k)$. To simplify the complexity of the polynomial fit, a 2D polynomial curve (straight line or parabola) can be fitted to the MSE values. To be able to do this, the three registration vectors involved need to be reasonably aligned, so that they effectively all lie on the same plane. A method of determining their alignment is by measuring the two angles between the three registration vectors which are denoted $\theta^{k-1}$, $\theta^k$. If both angles are below a threshold $\delta \theta$, then they are considered to be aligned. Besl and McKay decided upon a value of 10 degrees for this threshold. If they are considered aligned, the coefficients for a parabola are solved. The minima of the parabola where the MSE will be at its lowest can then be determined and the appropriate transformation update is applied.

Which transformation update is performed depends on some logic decisions which are covered in more detail in the description of Besl and McKay's [11] accelerated ICP algorithm below. The algorithm has been extended in the last step, so that if a worse MSE is obtained after the prediction update, the prediction update is undone.
1. Do three basic ICP iterations.

2. Compute the registration vector differences.

The current registration vector is denoted as \( q^k \), and the three previous registration vectors will be denoted as \( q^{k-1} \), \( q^{k-2} \) and \( q^{k-3} \).

The last three vector differences \( \Delta q^k \), \( \Delta q^{k-1} \) and \( \Delta q^{k-2} \) are defined as:

\[
\Delta q^k = q^k - q^{k-1} \quad (5.19)
\]

\[
\Delta q^{k-1} = q^{k-1} - q^{k-2} \quad (5.20)
\]

\[
\Delta q^{k-2} = q^{k-2} - q^{k-3} \quad (5.21)
\]

3. Compute the angles between the last three registration vectors.

The angle between \( \Delta q^k \) and \( \Delta q^{k-1} \) is denoted as \( \theta^k \). Similarly \( \theta^{k-1} \) is the
5.7. The Accelerated ICP Algorithm

angle between \( \Delta q^{k-1} \) and \( \Delta q^{k-2} \). They are defined as follows.

\[
\theta^k = \arccos \left( \frac{\Delta q^k \Delta q^{k-1}}{\|\Delta q^k\| \|\Delta q^{k-1}\|} \right) \quad (5.22)
\]

\[
\theta^{k-1} = \arccos \left( \frac{\Delta q^{k-1} \Delta q^{k-2}}{\|\Delta q^{k-1}\| \|\Delta q^{k-2}\|} \right) \quad (5.23)
\]

4. If last three registration vectors \( q^k, q^{k-1} \) and \( q^{k-2} \) are in good alignment, then attempt prediction.

We define an angular alignment threshold \( \delta \theta = 10^\circ \).

If \( \theta^k < \delta \theta \) and \( \theta^{k-1} < \delta \theta \) then there is good directional alignment.

(a) Calculate coefficients for prediction update.

i. Calculate \( v_{\text{max}} \).

\[
v_{\text{max}} = 25 \|\Delta q^k\| \quad (5.24)
\]

ii. Calculate the approximate arc length argument values, \( v^k, v^{k-1} \) and \( v^{k-2} \).

\[
v^k = 0, \quad v^{k-1} = -\|\Delta q^k\|, \quad v^{k-2} = -\|\Delta q^{k-1}\| + v^{k-1} \quad (5.25)
\]

iii. Calculate \( a_1 \) and \( b_1 \), used in straight line prediction where the parameterised straight line is \( v_1 = d(v) = a_1 v + b_1 \).

\[
a_1 = \frac{MSE^{k-1} - MSE^k}{v^{k-1} - v^k} \quad (5.26)
\]

\[
b_1 = MSE^k - (a_1 * v^k) \quad (5.27)
\]

iv. Calculate \( a_2 \) and \( b_2 \), used in parabolic prediction where the parameterised parabola is \( v_2 = d(v) = a_2 v^2 + b_2 v + c_2 \). Therefore the extremal point will be at \( \frac{\delta^2 d}{\delta v^2} = 2a_2 + b_2 \) and hence only the values of \( a_2 \) and \( b_2 \) will be required.

\[
\begin{pmatrix}
    a_2 \\
    b_2 \\
    c_2
\end{pmatrix} =
\begin{pmatrix}
    v^{k-2} & v^{k-1} & 1 \\
    v^{k-2} & v^{k-1} & 1 \\
    v^{k-2} & v^{k-1} & 1
\end{pmatrix}^{-1}
\begin{pmatrix}
    MSE^k \\
    MSE^{k-1} \\
    MSE^{k-2}
\end{pmatrix} \quad (5.28)
\]
v. Calculate \( v_1 \) and \( v_2 \) which are the parametric values which when supplied to \( d() \) will obtain a low MSE associated with the zero crossing \( v_1 \) or the minima of the parabola \( v_2 \).

\[
v_1 = \frac{-b_1}{a_1} \quad (5.29)
\]

\[
v_2 = \frac{-b_2}{2 \times a_2} \quad (5.30)
\]

(b) If logic tests are passed, perform update.

i. Use parabola based updated registration vector

\[
g^{k+1} = q^k + v_2 \times \frac{\Delta q^k}{\| \Delta q^k \|} \quad (5.31)
\]

if \( 0 < v_2 < v_1 < v_{\text{max}} \) or \( 0 < v_2 < v_{\text{max}} < v_1 \)

ii. Use the line based updated registration vector

\[
g^{k+1} = q^k + v_1 \times \frac{\Delta q^k}{\| \Delta q^k \|} \quad (5.32)
\]

if \( 0 < v_1 < v_2 < v_{\text{max}} \) or \( 0 < v_1 < v_{\text{max}} < v_2 \) or \( v_2 < 0 < v_1 < v_{\text{max}} \)

iii. Use the maximum allowable update

\[
g^{k+1} = q^k + v_{\text{max}} \times \frac{\Delta q^k}{\| \Delta q^k \|} \quad (5.33)
\]

if \( v_1 > v_{\text{max}} \) and \( v_2 > v_{\text{max}} \)

iv. Otherwise cannot do a prediction update.

(c) If we can do a predicted update

i. Apply transformation \( g^{k+1} \).

ii. Determine the new set of closest points.

iii. Recompute the MSE.

iv. If the new MSE is worse than MSE before the prediction update then recover, i.e. reapply transformation acquired by last basic ICP.

When doing a linear or parabolic update, the parameter \( v_1 \) or \( v_2 \) that gets used in the transform update, must be greater than zero. Otherwise, the update would be going back on itself. The prediction logic used takes a cautious approach to doing
updates. When a parabolic update occurs $v_2$ is always less than $v_1$ meaning that only a small parabola curve is trusted. When larger updates occur, the linear update is considered more reliable but if either $v_1$ or $v_2$ are greater than $v_{\text{max}}$ then the leap forward in prediction is considered too great, and so the maximum allowable update is done instead.

In the standard ICP algorithm which will be referred to as the basic ICP algorithm, the rigid transform gets updated as a composition, $g^{k+1} = \Delta g \circ g^k$, as was defined in equation (5.18). However, in the accelerated ICP algorithm when prediction occurs, the rigid transform is directly updated with the new prediction effectively wiping out all previous compositions, as is clearly shown in equations (5.31), (5.32) and (5.33).

In our implementation of the accelerated ICP, the registration termination criterion has been modified to take into account the actual MSEs obtained at the end of each accelerated ICP iteration, and not the three basic ICP iterations as would normally be the case.

### 5.8 Results

Three different closest point methods have been discussed in detail in the previous chapter. These being the surface based search, voxel based search and range image search. Any of these methods can be used by the ICP algorithm to perform registration. A comparison of how the ICP performs using these three different closest point methods is now presented. Characteristics of interest will be rate of convergence, accuracy and computational time required to complete the registration. The ICP method using the surface based CP search will be referred to as PolRegis. Likewise the voxel based ICP will be referred to as VoxRegis, and the range image search as RanRegis.
5.8.1 Data Used

In the following experiments, two data sets are used. The foot model shown in figure 5.10(c) is a synthetic model. The surface curvature of this model is fairly smooth with the exceptions of sharper curvature at the heel and toes of the foot. The Beethoven model is also synthetic and is shown in figure 5.11(c). The surface of this model is much more complicated with a lot more sharper curvature. Both the Beethoven and foot model were produced by Viewpoint Animation Engineering [29] which is a company that specialises in producing polygon mesh data sets. These particular models can be downloaded from the FTP site avalon.chinalake.navy.mil.

From the existing synthetic data sets of Beethoven and the foot, additional views of the models are created giving partial views of an object. These additional generated views are available in two formats. One of these formats is a polygon mesh, more specifically a triangulated mesh and the other format is in the form of range data. Given the nature of the surface representations used by the closest point methods, PolRegis and VoxRegis work with the triangulated meshes and RanRegis works with the range images.

By using synthetic data, the ground truth is known, and so the performance of the final outcome can be assessed. As can be seen from figures 5.10 and 5.11 the synthetic data is not simplistic, especially the Beethoven data set. The use of synthetic data allows known amounts of noise of various noise types to be added.

5.8.1.1 Generation Of Additional Synthetic Data

To be able to determine the ground truth for the RanRegis experiments, synthetic range images need to be created. To be able to do this a program called cri (create synthetic range image) was implemented. To produce a synthetic range image, the cri program is given an existing synthetic full 3D model (such as Beethoven) in triangulated mesh form, a viewing vector and a range image grid resolution is specified. The viewing vector is normalised, and simulates the direction from which the virtual range scanner points at the object. As well as the range image being created, a uniform triangulated mesh is also generated. The generated mesh which has
Figure 5.10: Foot data set. (a) and (b) First partial view, triangular mesh and range image respectively. (c) Original foot model. (d) and (e) Second partial view.

its uniformity corresponding to the resolution of the range image may substantially differ to the input mesh, which may be non-uniform and of a lower or higher resolution. When a synthetic range image is created an associated file containing its pose information is also generated. Since the pose information is not contained within the range image it is saved to an associated file. This means that when RanRegis
loads the range image and its associated file, the pose of the range image is known. This allows ground truth to be known when running RanRegis.

The cri program works by determining the rotation required to align the viewing vector onto the z-axis. The cross product is used to determine the angle between the
z-axis and viewing vector. By using unit vectors for the z-axis and viewing vector, the calculation of the angle between them is made easier since the cross product $|| a \times b || = || a || || b || \sin \theta$ simplifies to $|| a \times b || = \sin \theta$. Once the rotation has been applied to the surface, a bounding box (see section 5.8.2) is determined so that the space can be appropriately divided into the number of rows and columns specified for the range image resolution. Each facet in the triangular mesh is processed, and those with a normal vector that has a non-zero positive z component are used to update the range image. The region of the range image that the facet covers is determined and the z (depth) values are determined for those positions within the region by using the plane equation. When updating range image points a z buffering technique is used. If a depth value already exists at that range point, it will only get overwritten if the new depth is closer to the virtual range scanner. During this stage no slope thresholds are applied to the range image, they are done in the encoding stage of the range image search. The newly created range image is also exported as a triangulated mesh with slope thresholds applied of 1.46. The slope threshold simulates the level of surface curvature that will be seen by a range scanner. This value of the threshold was obtained through experimentation after seeing which surfaces looked more realistic.

In figure 5.10(c) the original foot model is shown along with generated partial views of them. The original model contains 1450 vertices and 2897 faces. The first partial view seen at the top right of the figure was generated with the viewing vector (1.0, 1.0, 1.0) at the resolution of 100x100. The corresponding triangular mesh in figure 5.10(a) has 5030 vertices and 9775 faces. A significant increase in the number of vertices, almost $3\frac{1}{2}$ times more vertices. This is to be expected since the surface has been sampled at a higher resolution by the cri program, compared to the original surface detail. When lower resolutions are specified to the cri program, a uniform surface mesh with fewer facets will result. The partial view generated for figure 5.10(e) used the viewing vector (0.5, -0.5, 2.5) with a resolution of 100x100. Its associated triangular mesh in figure 5.10(d) has 8272 vertices and 16219 faces. An increase by a factor 5.7 in the number of vertices. The partial meshes have been rotated to illustrate the partial shape of the foot. Distortion shown in the range
image partial views is an artifact of being enlarged for presentation purposes.

The original Beethoven model is shown in figure 5.11(c) along with generated partial views. The original model contains 2532 vertices and 5029 faces. The first partial view again shown at the top right was generated with the viewing vector \((1.0, 1.0, 1.0)\) at a resolution of 100x100. Its associated mesh shown to the left of it has 6340 vertices and 12152 faces. As to be expected, the number of vertices increased by a factor of \(2\frac{1}{2}\). The second partial view was generated by the viewing vector \((0.1, 0.5, -0.5)\) again at the resolution of 100x100. The corresponding mesh has 6728 vertices and 12948 faces. The partial meshes in figure 5.11(a) and (d) have been rotated to illustrate that there are gaps in the surface, and that it is not one whole continuous surface, although they will be treated as one surface during registration.

Later during the experiments the same partial views will be used but at varying resolutions to determine how the speed and accuracy is affected. The major effects will mainly be attributed to the closest point methods used when performing the registration.

### 5.8.2 General Approach

As already mentioned, ground truth is achieved by sampling patches from full synthetic data and thereby having surfaces that are already aligned. Prior to each experiment the moveable surface is transformed by a known amount. The performance of the algorithm can be assessed using two additional measures: an angular difference and a translational difference. The angular difference \(\delta r\) is defined in equation (5.34) where the expected rotation is denoted as \(R_e\) and the actual rotation obtained after registration is denoted as \(R\). The translational difference \(\delta t\) is defined in equation (5.35) where the expected translation is denoted as \(T_e\) and the actual translation obtained is denoted as \(T\). The bounding box vector described shortly is denoted as \(B\).

\[
\begin{align*}
\delta r &= \text{AngleOf}[R_e R^{-1}] \\
\delta t &= \frac{\|T_e - T\|}{\|B\|}
\end{align*}
\]  

(5.34)  

(5.35)
The registration software has a wide range of parameters available to it. To perform experiments using all the parameters would be very time consuming and would not likely show the significance of certain parameters clearly. Thus some parameters will be kept fixed unless otherwise stated throughout the experiments. When the moveable surface is initially rotated away from the fixed surface prior to starting the experiment, the rotation involved for the moveable surface will always occur around the axis \((1, 2, 3)\). The ICP algorithm has a termination criterion for a RMS threshold that is set to 0.999. There is also a limit to the maximum number of iterations performed, and this has been set to 100 iterations. The error will also usually be calculated using the standard MSE. However, the MAD estimator will at times be used to see if it has any significant benefits. Initially the experiments will not use any of the improvement checks, i.e. boundary, normal or distance checks. The accelerated ICP will also not be initially used.

To perform registration using the PolRegis and VoxRegis method the moveable surface needs to be sampled. However the RanRegis method works directly with a range image, and no sampling is performed. To enable a fair comparison the number of randomly sampled points from the moveable surface for the PolRegis and VoxRegis methods will be made to match the number of points used by the RanRegis method.

As mentioned earlier, by using synthetic data, ground truth can be obtained. In addition to modelling data, we can model noise too. There are various types of noise, but the noise chosen is Gaussian. When an experiment involves noise, only the moveable surface has noise applied.

The amount of Gaussian noise added is defined as a fraction of the length of the bounding box vector. The bounding box vector \(B\) is a length which is characteristic of the size of the surface. For our purposes a 2D bounding box vector is illustrated in figure 5.12. In the 3D case, the bounding box vector refers to the difference between the maximum and the minimum along each of the \(x, y, z\) axes. The noise is set in terms of a percentage \((n_s)\) of the bounding box. The noise level \(\sigma\) defined in equation (5.36) can then be multiplied by the \texttt{RandomGauss} function which returns
a value from a Gaussian distribution that has a RMS of 1.

\[ \sigma = \frac{n_s}{100} \times ||B|| \quad (5.36) \]

For the PolRegis and VoxRegis method the noise is added as shown in equation (5.37),

\[
\begin{pmatrix}
x' \\
y' \\
z'
\end{pmatrix} = \begin{pmatrix}
x + \sigma \times \text{RandomGauss} \\
y + \sigma \times \text{RandomGauss} \\
z + \sigma \times \text{RandomGauss}
\end{pmatrix}
\quad (5.37)
\]

and for the RanRegis method that works on a 2\(\frac{1}{2}\)D range image the noise is added as shown in equation (5.38).

\[
\begin{pmatrix}
x' \\
y' \\
z'
\end{pmatrix} = \begin{pmatrix}
x \\
y \\
z + \sigma \times \text{RandomGauss}
\end{pmatrix}
\quad (5.38)
\]

Before the registration process is started, the moveable surface is translated away from the fixed surface according to the specified values. Due to the fact that the data used is of various sizes, it would be ideal if the translation could be kept consistent throughout the experiments when using different data. Hence relative translation is used where the surface is translated according to a percentage of the maximum axis length \(s\) of the bounding box specific to the surface model being used. The
function MaxXYorZ returns the largest component value of the bounding box vector, i.e. \( x, y \) or \( z \). The percentage is defined by the components of \( T \), where for example if the component \( t_x \) had the value of 0.90 this would represent a 90% translational movement along the \( x \)-axis. The initial translation vector which is applied to the moveable surface is denoted by \( T_{\text{init}} \) as defined in equation (5.40).

\[
\begin{align*}
    s &= \text{MaxXYorZ} \, \| \, B \, \| \\
    T_{\text{init}} &= (s \, t_x, s \, t_y, s \, t_z)^T
\end{align*}
\]  

(5.39)  

(5.40)

If the data patches being registered are really the same surface patch then when applying improvement checks there will be two measures indicating how well the registration is being improved. Firstly the MSE, and secondly how much confidence there is in the data indicated by the weight values. A rough gauge can be given by using the ratio between the absolute full confidence and the actual estimated confidence. If there is full confidence, then each randomly sampled point will have an associated weight at the value of 1. Therefore the number of randomly sampled points \( n_p \) indicates the full confidence. The sum of the weights gives the estimated confidence. The normalised confidence \( W_c \) is defined as follows.

\[
W_c = \frac{\sum_{i=1}^{n_p} w_i}{n_p} \times 100
\]  

(5.41)

Each iteration the registration software outputs not only the RMS, \( \delta r \), \( \delta t \) and \( W_c \) values but also a Boolean statement (yes/no) for each robustness check to indicate whether the boundary, normal or distance check affected any weights. When the accelerated ICP is in use, additional information about the type of prediction that was used (providing the right conditions were met), and whether the prediction update was committed or rolled back\(^2\) is provided. This additional information allows the registration methods to be further analysed.

\(^2\) Roll back is where the error obtained after the prediction update is worse than the error prior to the update, thus the transformation is reverted back to the state prior to the prediction update. If the prediction update is committed, then the error has improved after doing prediction and so the transformation is kept.
5.8.3 The Experiments

The experiments have been grouped into several series, each demonstrating specific behaviour. For all series, any differences in behaviour of the ICP algorithm caused by the closest point methods will be commented upon. In the Series 1 experiments, the same surface is registered to itself to show what the standard ICP algorithm can achieve when there is full overlap. The performance of the accelerated ICP will also be examined. It is expected that all methods will work well considering there is full overlap.

In Series 2 the experiments determine how the standard ICP performs when two different but partially overlapping surfaces are to be registered. Also, how the number of sample points used affect the standard ICP, basic and accelerated, will be examined. In Series 3 the robustness checks are introduced to see how the modified ICP algorithm performs. Simulated noise is introduced in series 4 and a comparison between the MSE and the MAD estimator calculation for the RMS is performed.

5.8.3.1 Series 1

In this series of experiments the same surface is registered to itself. In this case, the foot model is used. For PolRegis and VoxRegis the original model is used, whereas for RanRegis a partial view generated via cri is used. The viewing direction for the partial view is (1,1,1) at a resolution of a 100 × 100. The rigid transform that is applied to the moveable surface consists of a 5° rotation and a (0.01,0,0) relative translation. Since this series concentrates on basic behaviour expected from the standard ICP algorithm the default (for our implementation) number of sample points are used. These being 2000 for PolRegis and VoxRegis, and RanRegis being 5036 determined by the range image search closest point method. For other series the same number of sample points are used for all closest point methods. Any characteristics that have been noticed that are attributable to one or more methods will also be commented upon. No real comparisons of the closest point methods are made at this stage. Only simple comparisons between the basic ICP and accelerated ICP will be made. Since each accelerated ICP iteration consists of three basic
ICP iterations, when comparisons are made between the basic and accelerated ICP version, the number of accelerated ICP iterations should be multiplied by three.

During initial trials with VoxRegis, it was found that its behaviour was seriously affected when direct lookup failed (i.e. the initial voxel the query point resides in contains no closest point) and the iterative search (i.e. looking at the surrounding neighbourhood after the direct lookup has failed) was used. As a consequence of just using direct lookup, more failures in determining a closest point are likely to occur. Therefore graphs of both the direct lookup and the iterative search will be shown, to highlight the difference in behaviour.

In the following tables, the information presented will show how long each method took, and the error minima for the final iteration. They may also include information such as whether it was the basic ICP or accelerated ICP that was performed. The graphs show RMS/δr/δt against iterations and therefore show the convergence of each registration test.

In ideal circumstances it is expected that the RMS, δr and δt errors would decrease monotonically. At the very least the RMS should decrease monotonically. In this particular experiment, the largest error will be associated with the rotation rather than the translation. As the RMS error is a sum of effects from both R and T an increase in one of them does not necessarily lead to an increase in the overall RMS, as it may be smaller than a decrease due to change in the other. An example of this is shown in the first few iterations and can be seen in figure 5.13(e) for δt. PolRegis dips first and rises before VoxRegis direct lookup (Lk) which can be seen at the bottom of the graph, whereas RanRegis does a smaller version of this dip at the top end of graph near 0.005. In figure 5.13(c) and (d) the errors associated with VoxRegis Iterate can be partially seen, since the graphs produced have been cropped to focus on the finer details of the other methods. If true correspondences are used in the registration process, then as δr and δt decrease the RMS will also decrease. However, if not all of the correspondences are true ones, the RMS may fluctuate, and so the RMS will be dependent on the closest point method. The rotational and translational errors are always measured consistently and therefore are a reliable
Figure 5.13: Series 1: Basic ICP vs accelerated ICP for foot.

way of determining how well the methods perform. The RMS dependence on the closest point method is shown in figure 5.14 for iterative and direct lookup methods of VoxRegis. It can be quite clearly seen that the correspondences established by VoxRegis’s iterative search are questionable since the RMS error is erratic.

From table 5.1 it can be seen that for the basic ICP runs that PolRegis and RanRegis
5.8. Results

Figure 5.14: Series 1: VoxRegis direct lookup vs iterative search.

do a high number of iterations compared to the two VoxRegis runs. PolRegis is approximately $5\frac{1}{2}$ times slower than RanRegis which is using $2\frac{1}{2}$ times more sample points. This can be attributed to the closest point method of PolRegis doing more iterative searches to find each closest point, where RanRegis has its search restricted. Hence PolRegis has faster convergence due to more accurate correspondences at the
The rotation errors for the accelerated runs have all been reduced as can be seen in table 5.1 with the exception of the iterative search for VoxRegis. In PolRegis’s case, the rotation and RMS errors decreased significantly down to machine precision level. The basic ICP run reached the limit of 100 iterations. The accelerated run did 50 iterations (150 basic iterations) and hence took $1\frac{1}{2}$ times longer to converge. The accelerated run of RanRegis took $\frac{2}{3}$rd s of the time compared to the basic version and came close to halving the rotational error. The accelerated run of VoxRegis direct lookup reduced its rotational error by half and took approximately the same time as the basic ICP run. The iterative search of VoxRegis had an increase in rotational and RMS error.

For PolRegis’s accelerated ICP run, mainly linear updates were performed during the first 36 iterations except for one parabolic update performed in the fifth iteration. Out of the remaining thirteen iterations, only three parabolic updates were performed. Prediction was not used for these other iterations due to poor angular alignment. For the first four iterations of RanRegis’s accelerated run, one linear update was performed for the third iteration. Prediction was attempted for all of the remaining iterations, the majority being of the parabolic type. A maximum allowable update was attempted in the 7th iteration, but was rolled back (undone), due to a higher RMS resulting. In the final iteration, a maximum allowable update was
5.8. Results

performed which was successful and therefore committed (kept). When VoxRegis used direct lookup, a linear prediction update was done for the first iteration only, and after that no prediction updates occurred. For the iterative search, there were large angular differences between the registration vectors and so no prediction occurred. From all of the accelerated runs, it appears when the registration is in the final stages of convergence, i.e. very little RMS change, the parabolic update gets performed more due to the nature of the prediction logic, since the prediction logic suggests that we have more confidence in where the current registration vectors are heading.

The RMS, rotation and translational errors only show the overall performance of how each closest point method affects the ICP algorithm. Another indicator of how well each closest point method performs is the overall weight confidence. The only way a weight can be set to zero is if the closest point method fails to find a closest point. In PolRegis's case, there is 100% confidence in the corresponding point set that its built. However, in RanRegis's and VoxRegis's case this is not so. In the basic ICP version of RanRegis, there is a 11% failure in determining the closest points for the first iteration. In the second iteration that reduces to 3.5%, and in the third iteration it drops further to 0.5% and continues to drop until the 31st iteration where full confidence is reached. For the basic ICP version of VoxRegis using direct lookup there is initially a 35% failure which reduces to 25% in the second iteration, and by the fifth iteration full confidence in the correspondence set is achieved. When using the direct lookup approach only, if the moveable point set is far from the fixed surface, the likelihood of each moveable point residing in a voxel that is empty is high. The iterative search of VoxRegis had full confidence but it still had high RMS errors.

In chapter 4, the maximum searchable neighbourhood size was discussed for the range image search closest point method. This specified a limit on the area of the range image that was searched after the initial projection, i.e. its neighbourhood, and as a result the maximum search time is limited but at the potential cost of not being able to determine a closest point or not allowing the best closest point to be found. Therefore, it is interesting to see what affect the neighbourhood size has on
Table 5.2: Series 1: Effects of varying neighbourhood size for RanRegis using foot.

<table>
<thead>
<tr>
<th>Method</th>
<th>NS</th>
<th>Its</th>
<th>RMS</th>
<th>$\delta r$</th>
<th>$\delta t$</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic</td>
<td>1</td>
<td>99</td>
<td>5.193x10^{-2}</td>
<td>8.22x10^{-4}</td>
<td>8.73x10^{-4}</td>
<td>1:15</td>
</tr>
<tr>
<td>Basic</td>
<td>2</td>
<td>77</td>
<td>6.193x10^{-2}</td>
<td>9.39x10^{-4}</td>
<td>8.88x10^{-4}</td>
<td>0:59</td>
</tr>
<tr>
<td>Basic</td>
<td>3</td>
<td>75</td>
<td>6.194x10^{-2}</td>
<td>9.42x10^{-4}</td>
<td>8.89x10^{-4}</td>
<td>0:57</td>
</tr>
<tr>
<td>Accel</td>
<td>1</td>
<td>17</td>
<td>5.142x10^{-2}</td>
<td>5.00x10^{-4}</td>
<td>7.46x10^{-4}</td>
<td>0:45</td>
</tr>
<tr>
<td>Accel</td>
<td>2</td>
<td>12</td>
<td>6.134x10^{-2}</td>
<td>5.58x10^{-4}</td>
<td>7.28x10^{-4}</td>
<td>0:33</td>
</tr>
<tr>
<td>Accel</td>
<td>3</td>
<td>12</td>
<td>6.134x10^{-2}</td>
<td>5.69x10^{-4}</td>
<td>7.36x10^{-4}</td>
<td>0:31</td>
</tr>
</tbody>
</table>

the end result. Thus the experiment for RanRegis was re-run using neighbourhood sizes of 1 (the projected range square), 2 (3x3 range squares), 3 (5x5), 5, 7 and 9. In this particular case there was no difference in the end results or time taken after increasing the neighbourhood size (NS) beyond 3, and so have been omitted from figure 5.15 and table 5.2.

From figure 5.15 it can be seen that for the neighbourhood size of 1, the initial RMS is slightly better than those with greater neighbourhood sizes for both basic and accelerated ICP runs. However, it suffers from a slower convergence. This is to be expected since the closest point being determined will not be the best one and so more ICP iterations are required to get better correspondences to align the surfaces. This can be clearly seen in table 5.2 for the basic ICP run.

In this particular experiment for the accelerated ICP runs, it appears that the number of linear updates performed and how early on during the run, had a direct influence on the final RMS obtained. The earlier the linear updates occurred, the better the final RMS and rotational error. Each initial linear update saw a reduction in RMS error of 50%. For the neighbourhood size of 1, two linear updates were performed in the first and seventh iteration. One linear update was performed in the third iteration for the neighbourhood size of 2, and one linear update was done in the fifth iteration for the neighbourhood size of 3. For all neighbourhood sizes the majority of subsequent updates performed were parabolic.

In chapter 4, the voxel size was discussed with respect to the voxel search closest
5.8. Results

Figure 5.15: Series 1: Effects of varying neighbourhood size for RanRegis using point method. In the voxel search closest point method, the space that the fixed surface occupies is uniformly divided into voxels. One of the trade offs was storage requirements versus speed, and another factor was the effect on accuracy. Hence a set of experiments was made to investigate the effect of changing the voxel size. The
Table 5.3: Series 1: Effects of varying voxel size for VoxRegis direct lookup.

<table>
<thead>
<tr>
<th>Method</th>
<th>( n_{\text{div}} )</th>
<th>Its</th>
<th>RMS</th>
<th>( \delta r )</th>
<th>( \delta t )</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic</td>
<td>128</td>
<td>16</td>
<td>4.917x10^{-1}</td>
<td>1.095x10^{-2}</td>
<td>7.15x10^{-4}</td>
<td>0:03</td>
</tr>
<tr>
<td>Basic</td>
<td>200</td>
<td>25</td>
<td>3.154x10^{-1}</td>
<td>4.934x10^{-3}</td>
<td>5.88x10^{-4}</td>
<td>0:06</td>
</tr>
<tr>
<td>Basic</td>
<td>300</td>
<td>29</td>
<td>2.149x10^{-1}</td>
<td>5.651x10^{-3}</td>
<td>6.12x10^{-4}</td>
<td>0:11</td>
</tr>
<tr>
<td>Basic</td>
<td>400</td>
<td>46</td>
<td>1.591x10^{-1}</td>
<td>2.056x10^{-2}</td>
<td>5.44x10^{-4}</td>
<td>0:20</td>
</tr>
<tr>
<td>Basic</td>
<td>500</td>
<td>8</td>
<td>3.424x10^{-1}</td>
<td>6.831x10^{-2}</td>
<td>7.16x10^{-4}</td>
<td>0:27</td>
</tr>
<tr>
<td>Accel</td>
<td>128</td>
<td>12</td>
<td>4.875x10^{-1}</td>
<td>3.272x10^{-3}</td>
<td>3.43x10^{-4}</td>
<td>0:04</td>
</tr>
<tr>
<td>Accel</td>
<td>200</td>
<td>16</td>
<td>3.119x10^{-1}</td>
<td>7.210x10^{-4}</td>
<td>7.70x10^{-4}</td>
<td>0:07</td>
</tr>
<tr>
<td>Accel</td>
<td>300</td>
<td>14</td>
<td>2.122x10^{-1}</td>
<td>1.587x10^{-3}</td>
<td>4.61x10^{-4}</td>
<td>0:12</td>
</tr>
<tr>
<td>Accel</td>
<td>400</td>
<td>23</td>
<td>1.567x10^{-1}</td>
<td>1.134x10^{-3}</td>
<td>7.22x10^{-4}</td>
<td>0:22</td>
</tr>
<tr>
<td>Accel</td>
<td>500</td>
<td>20</td>
<td>1.249x10^{-1}</td>
<td>4.390x10^{-4}</td>
<td>7.03x10^{-4}</td>
<td>0:31</td>
</tr>
</tbody>
</table>

Voxel size \( \Delta b \) is set to \( \| \max(x, y, z) - \min(x, y, z) \| / n_{\text{div}} \), the voxel volume being \( \Delta b^3 \). The default division size being 128. The value of \( n_{\text{div}} \) has been varied from 128 to 500. The results are illustrated in table 5.3 for the direct lookup version of VoxRegis.

As \( n_{\text{div}} \) is increased in the basic ICP VoxRegis direct lookup experiments the RMS value decreases as expected, except for the value of \( n_{\text{div}} = 500 \) due to a local minima as shown in figure 5.16(a). A higher \( n_{\text{div}} \) value results initially in having higher \( \delta r \) and \( \delta t \) errors. However, the more iterations performed, the lower the \( \delta r \) and \( \delta t \) errors finally obtained. This can be seen for \( n_{\text{div}} = 400 \) in figure 5.16(c) which has initially a higher error than the lower resolutions but ends with the smallest error. In the accelerated case, the local minima is overcome \( n_{\text{div}} = 500 \). Although prediction does play a part in obtaining a better answer, it is not the reason for overcoming the local minima. When the accelerated ICP is used, the termination criterion uses the RMS obtained at the end of each accelerated ICP run, i.e. the third basic ICP iteration done within an accelerated ICP iteration. So when in the basic ICP mode, the last three real iterations are compared and each RMS increases, it results in termination. However, with the accelerated ICP, the comparison occurs over 9 basic ICP iterations and overcomes the local minima in this case.
5.8. Results

Figure 5.16: Series 1: Effects of varying voxel size for VoxRegis direct lookup using foot.

As the value of $n_{div}$ is increased resulting in higher voxel resolution, the failure rate for determining closest points also increases. This is to be expected with the VoxRegis direct lookup approach, since likelihood of a query point residing in an empty voxel increases with voxel resolution, thus resulting in its weight being set to
zero. For the first iteration of the basic ICP the failure rates for the $n_{div}$ values of 128, 200, 300, 400 and 500 are 35%, 50%, 61%, 70% and 77% respectively.

When using the VoxRegis iterative search the failure rate is very small. Failures
only occur $n_{div} = 400$ and $n_{div} = 500$, where the maximum failure is 3.5% for $n_{div} = 500$. However, the quality of the closest points returned are in question, since there are very high RMS and rotational errors. For the basic ICP, the errors increase linearly as the voxel resolutions are increased, as can be seen in figure 5.17(a), (c) and (e). When using the accelerated ICP, the errors oscillate and the failure rate for determining closest points increases as the resolution increases. For the $n_{div}$ values of 200, 300, 400 and 500 the highest failure rates are 7%, 11%, 12% and 21% respectively. From figure 5.17 it can be seen that the reliability of the ICP algorithm is heavily dependent on the closest point method used.

A summary of conclusions for this series is listed below.

- The reliability of the ICP algorithm is heavily dependent on the closest point method used as there is a direct relationship between RMS error and the closest point method used.

- When the accelerated ICP was being used, the more linear updates performed in the initial stages, the more likely a better RMS would be obtained. Also parabolic updates tended to occur more towards the end of a run when the RMS was converging, although not always.

- The rotational error obtained by the basic ICP tended to be reduced by half or more (down to machine precision level) when using the accelerated ICP.

- PolRegis had no failures when determining a closest point. Failures by RanRegis were due to the restricted neighbourhood search of the method. For VoxRegis direct lookup the failure rate increased as the voxel resolution increased.

- The VoxRegis iterative search is currently not suitable for registering surfaces, and its behaviour needs to be further investigated.

5.8.3.2 Series 2

In this series, two different but partially overlapping surfaces are used to see how the methods perform in near real world cases. How the performance of the standard
ICP, basic and accelerated, is examined as the number of moveable points used in the registration is increased along with the fixed surfaces' facets or points in the case of range images. It is expected that as the number of moveable points and the fixed surface resolution increases, that the time taken to determine a closest point will increase and so will the accuracy of the closest point determination.

A compromise between accuracy and failure rates for the direct lookup method of VoxRegis has been made, and so the voxel division size that will be used within this series and others will remain at 300, unless stated otherwise. Likewise for RanRegis, the neighbourhood search size will be fixed at 3. The same number of points are used in this series and subsequent ones to allow fairer comparisons between the methods. For PolRegis and VoxRegis direct lookup, although the number of points will be the same, the moveable surface is randomly sampled every time and therefore the actual points used will be different each time. In contrast the RanRegis method will use valid range points within the moveable range image and will always use the same points. Cri was used to generate the data using the resolutions: 100x100, 50x50, 38x38 and 25x25. The number of points to be randomly sampled by PolRegis and VoxRegis direct lookup is determined by the number of valid range points within the moveable range image.

The first two overlapping views were generated from the Beethoven model. The fixed surface was generated from the view (1, 1, 1) and the moveable surface from the view (0.1, 0.5, −0.5) for all resolutions. The registration test consisted of a small rotation and a relatively large translation compared to the small translation and large rotation used in series 1. The moveable surface was rotated by 2° and translated by (−0.4, 0.6, 0.2). Figure 5.18 shows the results for these different resolutions when used with PolRegis. From looking at the $\delta r$ and $\delta t$ graphs a minima is reached around 20-30 iterations for the basic ICP and then rises except for the resolution of 100x100. Similar behaviour occurs in the accelerated case. Although the optimum rotation and translation are not being achieved, this behaviour is expected as incorrect correspondences are being used from the non-overlapping portions of the surfaces. Thus the standard ICP fails in this case.
5.8. Results

Figure 5.18: Series 2: Effects of varying the number of points for PolRegis using Beethoven.

The RMS graphs for VoxRegis direct lookup and RanRegis are shown in figure 5.19. As the resolution is increased the RMS error of RanRegis improves for both the basic and accelerated ICP. However, there is a very high degree of failure for determining the closest points, where the failure rate goes from 93% to 99% as
the resolution is increased. This is solely down to the large translational difference between the two range images causing projection failure. This high failure rate did not cause the RMS error to contradict the rotational and translation errors. For the basic ICP of VoxRegis direct lookup, the RMS starts off high, but then settles down to around the same RMS error level for all resolutions. In the accelerated case, the lower the resolution the better the RMS error. VoxRegis direct lookup had a very high failure rate of 97% for determining correspondences at all resolutions which is due to the large translational difference between the partial surfaces. Thus in this case, VoxRegis direct lookup’s RMS error for both basic and accelerated cases is an unreliable indicator of how good the registration is since \( \delta r \) and \( \delta t \) were approximately 2.6 and 1.0 respectively for all runs.

From looking at table 5.4 it can be seen that as the resolution is increased, there is
5.8. Results

Table 5.4: Series 2: Effects of varying the number of points for Beethoven.

<table>
<thead>
<tr>
<th>Method</th>
<th>Resolutions: RMS/Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>25x25: 424pts</td>
</tr>
<tr>
<td>PolRegis</td>
<td>0.214 / 0:24</td>
</tr>
<tr>
<td>RanRegis</td>
<td>0.663 / 0:01</td>
</tr>
<tr>
<td>VoxRegis Lk</td>
<td>0.059 / 0:08</td>
</tr>
<tr>
<td>PolRegis Acl</td>
<td>0.213 / 0:23</td>
</tr>
<tr>
<td>RanRegis Acl</td>
<td>0.583 / 0:02</td>
</tr>
<tr>
<td>VoxRegis Lk Acl</td>
<td>0.059 / 0:08</td>
</tr>
</tbody>
</table>

no significant improvement in the final RMS obtained for PolRegis using the basic or accelerated ICP. A significant amount of time is taken for the resolution of 100x100 and PolRegis is the slowest of all methods. RanRegis is the fastest taking only 6 seconds in the basic ICP case at the resolution of 100x100. However, in RanRegis’s case the number of points used does affect the RMS, rotational and translation errors equally. For the accelerated ICP no prediction was used at any resolution for RanRegis. Although VoxRegis direct lookup is fast, the registration suffered due to the high failure rate. No prediction took place in the accelerated case.

Next, the foot model was used. The fixed surface was generated from the view (1,1,1) and the moveable surface from the view (0.5, −0.5, 2.5) using the same four resolutions. The registration test consisted of a large rotation and a small relative translation. The opposite to the previous test. The moveable surface was rotated by 6° and translated by (0.03, 0.02, 0.01).

In series 1, it was noted that the iterative search method of VoxRegis tended to oscillate. This behaviour originally associated with the iterative search also occurs in some experiments when direct lookup is used, as can be seen in figure 5.20 for the resolutions of 38x38 and 50x50 when using the basic ICP. From the graphs it would seem that VoxRegis direct lookup has registered the surfaces best. However, from table 5.5 it can be seen that RanRegis has the lowest rotational and translational error.
Chapter 5. 2 View Surface Registration

Figure 5.20: Series 2: Stability at resolution 38x38 and 50x50 for Foot.

Table 5.5: Series 2: Stability at resolution 50x50.

<table>
<thead>
<tr>
<th>Method</th>
<th>Its</th>
<th>RMS</th>
<th>$\delta r$</th>
<th>$\delta t$</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>PolRegis</td>
<td>38</td>
<td>1.5285</td>
<td>12.12x10^{-02}</td>
<td>21.19x10^{-03}</td>
<td>1:49</td>
</tr>
<tr>
<td>RanRegis</td>
<td>18</td>
<td>1.2926</td>
<td>6.917x10^{-02}</td>
<td>7.653x10^{-03}</td>
<td>0:06</td>
</tr>
<tr>
<td>VoxRegis Lk</td>
<td>18</td>
<td>0.4481</td>
<td>13.24x10^{-02}</td>
<td>18.43x10^{-03}</td>
<td>0:05</td>
</tr>
</tbody>
</table>

Figure 5.21 shows how the methods behave for the resolution of 100x100. In the basic ICP case, RanRegis and VoxRegis direct lookup only perform three iterations due to a rising RMS error. PolRegis achieves the best rotational error in the 4th iteration but seeks another solution due to incorrect correspondences being used. In the accelerated ICP case, VoxRegis direct lookup’s oscillating behaviour is clearly shown while the $\delta r$ and $\delta t$ graphs decrease slowly. VoxRegis direct lookup has a high initial failure rate for all resolutions of around 80% to 90%. Thus the oscillation can be attributed to these failures and also the precomputed closest point as RanRegis does not oscillate when it has high failure rates. RanRegis’s initial failure rates are not so high, and they also dramatically decrease after each iteration. The initial failure rates for the resolutions of 25x25, 38x38, 50x50 and 100x100 were 10%, 14%, 21% and 58% respectively. Increasing RanRegis’s maximum neighbourhood search size does not dramatically reduce the initial failure rate, meaning that the initial projections land outside of the range image in this particular case. PolRegis experienced no problems in determining closest points.
Table 5.6 summarises the RMSs obtained and their corresponding times taken for each method at the various resolutions. The time taken by PolRegis is seriously affected by the number of points used, but is reliable and behaves consistently. Above the resolution 38x38, not much significant gain in the final RMS obtained is achieved for PolRegis. RanRegis and VoxRegis tend to be consistent on the amount
Table 5.6: Series 2: Effects of varying the number of points for Foot.

<table>
<thead>
<tr>
<th>Method</th>
<th>Resolutions: RMS/Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>25x25: 523pts</td>
</tr>
<tr>
<td>PolRegis</td>
<td>2.010 / 0:21</td>
</tr>
<tr>
<td>RanRegis</td>
<td>2.234 / 0:02</td>
</tr>
<tr>
<td>VoxRegis Lk</td>
<td>0.487 / 0:04</td>
</tr>
<tr>
<td>PolRegis Acl</td>
<td>2.000 / 0:15</td>
</tr>
<tr>
<td>RanRegis Acl</td>
<td>2.191 / 0:02</td>
</tr>
<tr>
<td>VoxRegis Lk Acl</td>
<td>0.481 / 0:04</td>
</tr>
</tbody>
</table>

of time taken to register, which is not seriously affected by using a high number of points. The RMS error generally improves for RanRegis as the resolution is increased. VoxRegis direct lookup's RMS error is approximately the same except for the resolution of 100x100.

A summary of conclusions for this series is listed below.

- As expected the standard ICP algorithm fails when surfaces that contain partial overlaps are to be registered, since the whole of each surface is being registered even though they are not identical.

- Generally as the number of moveable points are increased the RMS improves. However, the amount it improves is dependent on the closest point method being used. In PolRegis's case, it does not improve much beyond the resolution of 38x38, but in RanRegis's case it improves each time the resolution is increased.

- The execution time of PolRegis is heavily dependent on the number of moveable points used, and is the most time intensive method. RanRegis and VoxRegis direct lookup are less affected by the number of points used.

- VoxRegis direct lookup and RanRegis had very high initial failure rates for determining the closest point in these experiments when the moveable point set was translationally far from the fixed surface. This was simply due to
projection failure. They were less affected when there was a high rotational
difference between the moveable point set and fixed surface.

- VoxRegis direct lookup also appears to have a stability problem which is
thought to be associated with the precomputed closest point and its high fail­
ure rate. The RMS error tends to contradict the rotational and translational
errors.

5.8.3.3 Series 3

In this series the robustness checks will be introduced to see how the modified
ICP algorithm performs. Only the basic ICP is used, so that the affect on each
iterations' correspondence set can be examined. The core series 1 experiments and
the series 2 experiments using the resolution of 38x38 will be re-run to see effects of
the robustness checks when registering surfaces with full and partial overlap.

Figure 5.22 shows the re-run of the core series 1 experiments with and without the
robustness checks. It can be clearly seen that RanRegis's run was extended with
the use of the normal check compared to its original run. For RanRegis the normal
check affected weights each iteration. The initial confidence was 48% and ended with
52%. The final confidence is solely down to the normal check. For VoxRegis direct
lookup's RMS error the normal check only affects weights for the first two iterations,
and its affect is very negligible. The weight confidence for the direct lookup case of
VoxRegis for the the first two iterations was 40% and 43%. On the third iteration
when the normal check has no influence, the confidence in the correspondence set
was 45% due to closest point determination failure. For PolRegis, the normal check
only affected weights for the first two iterations (the confidence being 99.87% and
99.99%) since that is when there is the most angular difference between the surfaces
being registered.

When the boundary check is introduced, it does nothing in PolRegis's and VoxRegis
direct lookup's case. This might be expected since it is the same whole surface being
registered to itself. However, even when registering a partial surface to itself, there
will be points that lie on the boundary of the surface. Thus with RanRegis, the
boundary check does affect weights, although it does not for the first iteration as is to be expected, since the boundary check uses the RMS from the previous iteration which does not exist for the very first iteration. The initial confidence is 95%. During the registration process the angular and translational errors drop resulting in a final confidence of 85%. The boundary check is to be expected to affect more weights with RanRegis due to its encoding scheme where the range squares that were not considered valid were on the true boundaries of the surface. Therefore, when registering the points associated with the true boundaries there will not be true correspondences and their distances will be greater than the RMS as the transform gets closer to the optimum one.

When the distance check is introduced, weights are affected each iteration for all methods and the rate of convergence is also slowed down. PolRegis’s confidence
starts at 80% and increases gradually to 87%. For RanRegis the confidence starts at 83% and ends at 97% and for VoxRegis direct lookup it starts at 34% and ends at 46% due to a short run caused by a rise in RMS error. Since the same surface is being registered to itself, intuitively it would not be expected for the distance check to have such a negative impact on the registration.

In section 5.5.3, the distance check was described where the linear ramp started at $\sigma$ and decreased to zero at $3\sigma$. The associated equation was

$$w_i = 1 - \frac{d_i^2 - MSE^k}{2 * MSE^k}.$$

In light of the effect of the distance check in this case, it was decided that varying the ramp starting point would be interesting. Therefore using the variable $d_{c\sigma}$ to indicate where the distance check ramp start occurs, the equation is re-written to

$$w_i = 1 - \frac{d_i^2 - d_{c\sigma} * MSE^k}{3 * MSE^k - d_{c\sigma} * MSE^k}.$$

d_{c\sigma}$ was varied from 1 to 3, and when being at 3, the equation is ignored and a binary decision is used, i.e. for each point whose distance is equal to or greater than $3 * MSE^k$ its weight is set to zero. The $d_{c\sigma}$ values used were 1, 1.5, 2, 2.5, 2.85 and 3. The graphs in figure 5.23 are plotted on a logarithmic scale for the y-axis so that more distinctions between the different $d_{c\sigma}$ values can be seen. By changing the value of the distance check ramp start, the rate of convergence is increased but not dramatically as was hoped. Therefore in this case, a significant number of points had distances that were greater than $3 * MSE^k$ which slowed down the rate of convergence.

Next the Beethoven partial overlaps used in series 2 were re-run. In the boundary and normal check case shown in figure 5.24 PolRegis achieved a better RMS than obtained without the robustness checks. RanRegis's RMS increased for the first three iterations and terminated early, which happened in the original case. VoxRegis direct lookup failed during the second iteration due to all of the correspondence sets weights being set to zero reported by the routine VoxelGetClosestPoints. When using the boundary check, it is assumed that the RMS error always decreases monotonically. Hence it is therefore safe to use the RMS from the previous iteration when applying
the boundary check during the \texttt{VoxelGetClosestPoints} function. However, when \texttt{VoxRegis} direct lookup oscillates, the actual RMS in the current iteration when closest points are actually being determined can be significantly higher than the RMS in the former iteration. This means that the boundary check will most likely set each point’s weight value to zero if it lies on a boundary. The distance check will not suffer from this problem as it uses the MSE for the current iteration, i.e. it is applied after the closest points have been found and MSE has been recomputed.

When using the distance check (as originally specified) along with the normal and boundary check, \texttt{PolRegis} converged quickly but had a higher RMS value. \texttt{RanRegis} again finished after the third iteration due to a rising RMS error. \texttt{VoxRegis} direct lookup does not suffer from the sum of weights being zero problem in this case. Although the RMS is extremely low, it is a very unreliable measure in this case as there is only a 1% confidence in the correspondences. This poor confidence is not
5.8. Results

Figure 5.24: Series 3: Beethoven from series 2 re-run with robustness checks.

solely due to the robustness checks, since the original series 2 experiments has a 97% failure rate in determining correspondences.

Next the partial foot view experiments from series 2 were re-run with the robustness checks. Figure 5.25(a) shows those same experiments without the robustness checks, where the graph has been slightly cropped in the \( y \)-axis for comparison between the original and robustness check runs. For both robustness check cases, VoxRegis only did 3 iterations due to a rising RMS error. RanRegis achieves better RMS values in both cases compared to PolRegis, but PolRegis obtains slightly better rotation and translation errors. However, the maximum limit of 100 iterations was hit by both methods, meaning that RanRegis may achieve a better result if it was allowed to do more iterations. Again it can be seen that the distance check slows down the convergence rate.
When using the foot views the normal check only affected the first couple of iterations. However, in Beethoven’s case, the normal check was more consistently active due to the high surface curvature associated with the model.

A summary of conclusions for this series is listed below.

- The registration of partially overlapping surfaces is successful when using the boundary and normal robustness checks.

- The boundary check was the most effective robustness check when using partial surfaces, compared to whole surfaces that have no boundary.

- The normal check only had influence for the first few iterations while the rotational difference between the two surfaces was large. The normal check...
5.8. Results

has more influence in cases where partial surfaces are used and there is not complete overlap, and non-corresponding parts of the surface have a high curvature.

- The distance check tends to hinder the registration. Sometimes the initial rate of convergence is fast, other times it is slow. However, the RMS obtained is always higher than the RMS obtained when not using the distance check. Altering the weight function for the distance check had no significant affect.

- The robustness checks made closest point determination problems associated with a high translational difference between two surfaces very apparent for VoxRegis direct lookup resulting in failure.

5.8.3.4 Series 4

In this series simulated noise is introduced. The RMS in the previous series has been calculated using the standard MSE method as described in section 5.2.1. A comparison between the MSE method and the MAD estimator will be done. The series 1 and 2 core experiments will again be used with the exception of Beethoven due to the high closest point failures. The noise levels used are 0%, 0.2%, 0.5%, 1%, 2% and 5%. The distance check is also used to see how it performs when noise is present.

Figure 5.26 shows the core series 1 experiments re-run with noise of various levels. All of the robustness checks have been used simultaneously. As the noise levels are increased, both PolRegis and RanRegis behave as is expected for both MSE and MAD cases where the RMS increases as noise increases. However, in VoxRegis direct lookup's case there is not a direct relationship between RMS and noise level. For the MSE and MAD case, the 5% noise curve is actually lower than the 0.5%, 1% and 2% curves. This behaviour is related to the poor confidence given to the closest points found. In the MSE case, the initial confidence values were 34%, 30%, 27%, 22%, 18% and 7% for their respective noise levels 0%, 0.2%, 0.5%, 1% and 5%.

As the noise is increased, RanRegis's error rates are less affected than that of PolRegis. Interestingly the difference between the 2% and 5% curve is more noticeable
Chapter 5. 2 View Surface Registration

Figure 5.26: Series 4: Effects of noise and MSE vs MAD using boundary, normal and distance check. Re-run of series 1 core.

in the MAD case for both PolRegis and RanRegis. The RMS is lower when using the MSE compared to the MAD estimator.

Next the foot experiment from the core series 2 experiments is re-run. Figure 5.27 shows it re-run using all robustness checks and figure 5.28 without the distance
5.8. Results

Figure 5.27: Series 4: Effects of noise and MSE vs MAD using boundary, normal and distance check. Re-run of series 2 core, Foot.

check. Both RanRegis and PolRegis perform well. Again VoxRegis direct lookup fails due to a low confidence in the closest points determined. Thus the stability of VoxRegis direct lookup is also in question considering the crossing of the curves for both figures 5.27 and 5.28.
Figure 5.28: Series 4: Effects of noise and MSE vs MAD using boundary and normal check only. Re-run of series 2 core, Foot.

When comparing the RMSs of the graphs obtained using the distance check, and those that do not use the distance check, there appears to be better RMS values obtained with the distance check. However, the final confidence $W_c$ in the closest points determined is playing a role as illustrated for the 2% noise case using mean
square error in table 5.7. Also it is noted that in the 2% case better rotational and translational errors are achieved when using the distance check due to the runs reaching the 100 iterations limit. Generally, the rotational and translational errors are always better when the distance check is not used, whether using the MSE or MAD estimator.

Table 5.7: Series 4: Foot with 2% noise using MSE.

<table>
<thead>
<tr>
<th>Method</th>
<th>Distance Check</th>
<th>No Distance Check</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$W_c$</td>
<td>RMS</td>
<td>$\delta r$</td>
<td>$\delta t$</td>
<td>$W_c$</td>
<td>RMS</td>
<td>$\delta r$</td>
<td>$\delta t$</td>
</tr>
<tr>
<td>PolRegis</td>
<td>69.24%</td>
<td>2.104</td>
<td>0.0207</td>
<td>0.0155</td>
<td>89.84%</td>
<td>3.011</td>
<td>0.0453</td>
<td>0.0082</td>
</tr>
<tr>
<td>RanRegis</td>
<td>54.89%</td>
<td>1.705</td>
<td>0.0192</td>
<td>0.0094</td>
<td>71.99%</td>
<td>2.646</td>
<td>0.0284</td>
<td>0.0074</td>
</tr>
<tr>
<td>VoxRegis Lk</td>
<td>9.39%</td>
<td>0.466</td>
<td>0.1051</td>
<td>0.0347</td>
<td>0.12%</td>
<td>0.580</td>
<td>0.1038</td>
<td>0.0351</td>
</tr>
</tbody>
</table>

Table 5.8 shows the 0.5% noise case. Better rotational and translational error is achieved when no distance check is used (except for VoxRegis direct lookup which is unreliable). Higher RMS values exist for the runs that do not use the distance check. However, the RMS is affected by weight values which the distance check alters. The MAD estimator reports a higher RMS compared to the standard MSE method of determining the RMS. When using the MSE method, the rotational and translational errors are generally lower than the MAD estimator in the case of PolRegis. However, with the RanRegis experiments it is not clear cut, and this is most likely due to the varying confidence levels.

Table 5.8: Series 4: Foot with 0.5% noise, MAD and MSE.

<table>
<thead>
<tr>
<th>Method</th>
<th>MSE</th>
<th>MAD</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RMS</td>
<td>$\delta r$</td>
<td>$\delta t$</td>
<td>RMS</td>
<td>$\delta r$</td>
<td>$\delta t$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PolRegis DC=1</td>
<td>0.553</td>
<td>0.0190</td>
<td>0.0133</td>
<td>0.805</td>
<td>0.0531</td>
<td>0.0163</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RanRegis DC=1</td>
<td>0.485</td>
<td>0.0531</td>
<td>0.0032</td>
<td>0.654</td>
<td>0.0479</td>
<td>0.0027</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VoxRegis Lk DC=1</td>
<td>0.467</td>
<td>0.2211</td>
<td>0.0238</td>
<td>0.719</td>
<td>0.2468</td>
<td>0.0209</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PolRegis DC=0</td>
<td>0.765</td>
<td>0.0135</td>
<td>0.0099</td>
<td>0.781</td>
<td>0.0148</td>
<td>0.0108</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RanRegis DC=0</td>
<td>0.696</td>
<td>0.0287</td>
<td>0.0024</td>
<td>0.708</td>
<td>0.0285</td>
<td>0.0024</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VoxRegis Lk DC=0</td>
<td>0.550</td>
<td>0.2323</td>
<td>0.0221</td>
<td>0.715</td>
<td>0.2481</td>
<td>0.0208</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
In a low noise case the normal check only affects the first couple of iterations. However, when higher levels of noise are present it has a higher activity.

A summary of conclusions for this series is listed below.

- The distance check generally hindered the registration process and gave worse translational and rotational errors even when noise is present.

- The RMS error calculated using the MSE is lower than when the MAD estimator is used.

- The rotational errors and translational errors obtained are still dependent on the closest point method used. For PolRegis, better registration was achieved when using MSE, but for RanRegis the MSE was better for 50% of the experiments. The MAD estimator may perform better with real data, rather than synthetic data that involves a single Gaussian noise distribution.

- The normal check is more active in cases where noise is present.

### 5.8.3.5 Conclusions

The speed, accuracy and stability of the ICP algorithm is affected by the choice of the closest point method used, and any fine tuning parameters passed to these methods. The surface based closest point method is the most reliable, achieving low errors but at the expense of high execution time. Significantly increasing the size of the number of moveable points does not increase accuracy for this method but it does significantly increase the execution time. The voxel search tends to be unreliable if the moveable surface is not close to the fixed surface due to the precomputed closest point strategy it uses, and the voxel size when only using direct lookup. Increasing the number of moveable points has no significant impact on execution time. In contrast, the range image search’s accuracy is improved as the resolution of the range images increase. Although execution time does increase, it does not increase significantly. The range image search is effected by the translational difference between the two range images to be registered due to the simple projection technique the method uses.
When the accelerated ICP is used, the more times a linear update occurs in the initial iterations the better the final RMS obtained. The parabolic updates tended to occur towards the end of convergence but not always.

Using robustness checks allows partially overlapping surfaces to be registered, whereas the standard ICP algorithm fails. The most effective robustness check is the boundary check. It is most useful when there is a low degree of overlap between the surfaces being registered. The normal check tends to be useful when there is high curvature in the views which do not form part of the overlap and is more active when noise is present. The distance check affects the rate of convergence, either by making it faster or slower. However, the final RMS obtained always tends to be higher than the RMS obtained without using the distance check in the noiseless case. When using the distance check, the value of the RMS tends to be a less reliable measure of how good the registration is, especially when noise is present. The boundary check and distance check use a similar weighting function, where the vital difference is knowing whether a corresponding point with a large distance actually lies on the boundary of the fixed surface or not. Having this knowledge is what makes the boundary check so effective.

When noise is modelled and the level of that noise is increased, the final RMS obtained also increases, which is to be expected. However, the rotational and translational errors still remain low. When using the MSE or MAD estimator method to obtain the RMS the rotational and translational errors obtained were still dependent on the closest point method used. When using the surface based closest point method, the MSE method gave better translational and rotational errors. However, the MAD estimator may perform better with real data, rather than synthetic data that involves a single Gaussian noise distribution.

5.9 Summary

In this chapter an experimental comparison has been presented. The standard ICP algorithm has been described and it has been shown that it is dependent on the closest point method used and therefore can fail even when using fully overlapping
surfaces. The accelerated ICP was also examined. It has also been shown that the standard ICP algorithm fails when there is only partial overlap between the surfaces to be registered. The ICP algorithm was extended by introducing robustness checks and it has been shown that the registration of partially overlapping surfaces succeeds. It has also been shown that low rotational and translational errors are achieved in the presence of noise.
Chapter 6

N-View Point Set Alignment: A Comparison

Several authors have considered the problem of building complete surface models of complex objects using range images taken from several views, see Bergevin et al. [8], Blais and Levine [14], Chen and Medioni [17] and Dorai et al. [23]. Since the viewpoints (or object poses) are usually not known it is necessary to register the surfaces taken from various views prior to fusion, see Hilton et al. [45].

In the case of 2 views, the iterated closest point (ICP) algorithm by Besl and McKay [11] may be used to register surfaces. To be able to do this, the ICP algorithm needs two key things. Firstly, a method of determining corresponding points between the two surfaces to be registered. Secondly, the ICP algorithm requires a method which will solve the geometric transformation (the rotation and translation) that will map the two corresponding point sets onto each other. For this second requirement, several analytic solutions for solving the 3D point set registration problem are available, see Kanatani [68] and references contained therein.

When more than 2 views must be registered a strategy similar to the ICP algorithm may still be used, provided that a solution for the N-view point set alignment problem is available. The N-view point set alignment problem may be reduced to a chain of pairwise problems and solved with a 2 view algorithm. However, this is not an op-
timal solution. For example, if there are three overlapping surfaces and the first two surfaces are registered and fused, then any error in the determined transformation is now fixed in the new fused surface. If the final view is registered to the newly fused surface by using a pairwise step technique, errors will propagate through each step. However, if information in all views is used simultaneously rather than sequentially, then a global optimal solution can be sought with no error propagation occurring.

Recently 3 algorithms for alignment of multiple partially overlapping point sets have been published by Pennec [81], Stoddart and Hilton [103] and Benjemaa and Schmitt [6]. The relative merits have not yet been studied. In the 2 view case a thorough evaluation of the various techniques has been performed by Eggert et al. [28]. The purpose of the work in this chapter is similar to that of the Eggert work but for the N-View alignment methods.

6.1 N-View Alignment

Several analytical solutions exist for the 2 view point set alignment problem. These methods decouple the rotation and translation, and solve for the rotation by computing the SVD of a \((3 \times 3)\) matrix, see Kanatani [68] or the eigenvectors of a \((4 \times 4)\) matrix, see Horn [53]. The translation is then usually solved by calculating the displacement between rotated centroids.

In this chapter the methods of Stoddart and Hilton [103], Pennec [81], and Benjemaa and Schmitt [6] will be studied. All the current methods for N-view registration are iterative. However, Benjemaa and Schmitt made a significant advance insofar as they have been able to analytically decouple the rotation and translation.

6.1.1 Problem definition

Each of the three main papers relating to N point alignment are described using the notation of Benjemaa and Schmitt. Benjemaa and Schmitt assume that there are \(M\) point sets each taken from a different viewpoint, \(S^1 \ldots S^M\), where \(S^\alpha = \{p_1^\alpha \ldots p_N^\alpha\}\). The objective is to find the best rigid body transforms, \(f^1 \ldots f^M\), which when applied
6.1. N-View Alignment

to each point set results in alignment of all corresponding points. The rigid body transform is denoted as \( f^\alpha \) and comprises of a rotation \( R^\alpha \) and translation \( T^\alpha \). Hence, \( f^\alpha \ast p = R^\alpha p + T^\alpha \).

The overlap of \( S^\alpha \) with \( S^\beta \) is denoted as \( O^{\alpha\beta} \subset S^\alpha \) where \( O^{\alpha\beta} = \{ p_1^{\alpha\beta} \ldots p_{N^{\alpha\beta}}^{\alpha\beta} \} \). \( O^{\alpha\beta} \) has \( N^{\alpha\beta} \) points where each point \( p_i^{\alpha\beta} \) is matched with \( p_i^{\beta\alpha} \in O^{\beta\alpha} \subset S^\beta \). Therefore \( N^{\alpha\beta} = N^{\beta\alpha} \). Benjemaa also states that \( O^{\alpha\alpha} = \emptyset \) and \( N^{\alpha\alpha} = 0 \) for convenience in subsequent formulae. The idea of overlap is illustrated in figure 6.1. Each view only has points in its set of the object that it can see. Thus the points which are visible in both views are considered as the overlap, since they will be identical and thus are classified as the corresponding points between both views.

\[ \text{Figure 6.1: Illustration of overlap between point set views.} \]

The problem may be specified as minimising over the \( N \) transforms \( f^\alpha \) a cost \( E \) where

\[
E[f^1 \ldots f^M] = \sum_{\alpha=1}^{M} \sum_{\beta=1}^{M} \sum_{i=1}^{N^{\alpha\beta}} w_i^{\alpha\beta} \| f^\alpha \ast p_i^{\alpha\beta} - f^\beta \ast p_i^{\beta\alpha} \|^2
\]

where \( w_i^{\alpha\beta} \) are the weights. It is noted that the problem is undetermined up to a global transformation applied to all point sets, i.e. when all transforms are applied
simultaneously. With no loss of generality this can be removed by requiring that $f^1$ is the identity transform.

### 6.1.2 Pennec

Pennec’s [81] method is by far the easiest to implement, provided that a 2 view point set alignment algorithm is already available! It is iterative and based on the concept of ‘mean shape’. Each view has corresponding points on the mean shape to which it is aligned using a standard point set alignment method, such as Horn et al. [53]. At the beginning of each iteration, a new mean shape is calculated, and again the views are aligned to it. This continues until convergence.

The mean shape $M = \{ m_1 \ldots m_k \}$ is constructed by using averaging of corresponding points. Thus each element of the mean shape is constructed using corresponding points $p_i^{\alpha \beta}$ and $p_i^{\beta \alpha}$. Pennec’s method has been modified so that these corresponding points have an associated weight $w_i^{\alpha \beta}$ indicating the confidence that these two points are actually true correspondences. The idea of mean shape is illustrated in figure 6.2. During construction, it is noted which elements of the mean shape correspond to a specified view by using a mapping function $r(i, \alpha, \beta)$ and thus the weight $w_i^{\alpha \beta}$ becomes associated with that element of the mean shape. Thus when a view is to be aligned to the mean shape, the correspondences on the mean shape are known.

![Figure 6.2: Pennec's mean shape.](image)

Each view’s point set is aligned one at a time to the mean shape. Thus the determination of each transform is done independently from each of the other transforms $f^1 \ldots f^M$. The optimal transform for a view is found by determining the minimal distances between the points of that view and their corresponding points on the
mean shape.

\[ E[f^\alpha] = \sum_{r=1}^{k} \sum_{\beta=1}^{M} \sum_{i=1}^{N^{\alpha\beta}} w_{i}^{\alpha\beta} r(i, \alpha, \beta) \| m_r - f^\alpha \ast p_{i}^{\alpha\beta} \|^2 \]  

(6.2)

The mapping function \( r(i, \alpha, \beta) \) makes sure that only points belonging to a particular view are aligned to their corresponding points on the mean shape. Thus if a point does not belong to a view, its associated contribution to the cost will be zero. The transform \( f^\alpha \) for two corresponding point sets is solved by using a closed-form solution such as Horn et al. [53].

The process of building a mean shape and aligning the views to the mean shape continues until convergence is met. Convergence is determined by using ratios of the overall cost for the present and the past two iterations. If the ratios approach one, the convergence curve is assumed to be levelling out and alignment is considered complete. As in the modified ICP termination registration criterion at least three iterations have to be performed. In addition there is a maximum limit set on the total number of iterations. This is set to the value of 1000. Finally there is a minimum number of iterations that must be performed which is set to 100. The implementations of the Stoddart and Hilton and the Benjemaa and Schmitt methods use the same minimum and maximum limits.

To summarise, the mean shape is first computed. Then the optimal transforms for each view are solved. The mean shape is recomputed taking into account the new transforms, and the transforms are solved again. This process iterates until convergence.

6.1.3 Stoddart and Hilton

Stoddart and Hilton [103] use an iterative numerical method based on gradient descent. Their criterion is to determine the minimum cost to obtain optimal transformations to minimise distances between corresponding point pairs. The problem is solved by analogy with a physical system of rigid bodies connected by springs, as illustrated in figure 6.3.
The method works by first defining a centre of mass for each view. Once this is done, the points for each view can be defined to be relative to its centre of mass. For each view, a local centroid\(^1\) relative to the view's centre of mass is defined for each subset of points that correspond to another view's points, as illustrated in figure 6.4. If the views are not aligned, then the springs between the connected local centroids will be stretched causing a force to pull the connected local centroids together. Thus the translation to align two connected local centroids can be determined by calculating the force of the spring between the two local centroids. Similarly by computing the torque, the rotation around the centre of mass is obtained. Once the force and torque have been computed for all correspondences, a view's overall force \(F_{\alpha}^{\text{tot}}\) becomes the vector sum of these forces calculated for corresponding views, i.e. \(F_{\alpha}^{\text{tot}} = \sum_{\beta=1}^{M} F^{\alpha\beta}\). Similarly the overall torque \(\tau_{\alpha}^{\text{tot}}\) is computed for each view.

Once all the spring forces have been calculated for a view, the individual spring forces may be combined into an overall force \(F_{\alpha}^{\text{tot}}\) acting on the centre of mass and a torque \(\tau_{\alpha}^{\text{tot}}\) around the centre of mass for each view. In Stoddart and Hilton [103] it is shown how to compute these in a very efficient way.

Each view is associated with a rigid body having an arbitrary centre of mass and moment of inertia. Stoddart and Hilton state that the choice of value for these parameters do have considerable effect on the rate of convergence, and that a sensible

---

\(^1\)The local centroid is defined as the average of the weighted subset of points. Whereas, the centre of mass is defined as the average of all points within the view.
6.1. N-View Alignment

The view's centre of mass, and the moment of inertia should be chosen as if each object were a sphere of radius equal to half the diagonal of a bounding box containing the data.

Once the force and torque have been obtained they can be inserted into a dynamical system which moves towards a potential minimum. The following friction dominated equations of motion are chosen.

\[ \gamma \frac{dy_{cm}}{dt} = F_{\text{tot}}^\alpha \quad (6.3) \]
\[ \Gamma \omega^\alpha = \tau_{\text{tot}}^\alpha \quad (6.4) \]

where \( \gamma \) resembles the mass and \( \Gamma \) the moment of inertia, but here they represent the drag and rotational drag coefficients. \( \omega \) is the angular velocity (rate of change of orientation with respect to time). Only first order equations of Newton's laws are used since at the minima the acceleration will be zero.

Stoddart and Hilton use friction dominated equations of motion to dictate a solution that evolves over time to a local minimum in potential energy. This is very similar to the method of gradient descent in optimisation where the potential energy plays the role of the cost function, and the physical forces \( F^\alpha \) play the role of the gradients. Secondly, the torque \( \tau^\alpha \) around the centre of mass \( cm \) of each view is also considered in this dynamical system. The alignment problem is solved by integrating the equations of motion over time.
To start off the process, an initial estimate must be supplied. During the computation, all transforms associated with each view vary simultaneously. The system of equations are then integrated by a simple quality controlled Euler method which can solve the dynamical system with adaptive step size. It is guaranteed to converge to a local minimum. Once the computation is complete, all views are transformed so that the first view is transformed by the identity transform.

6.1.4 Benjemaa And Schmitt

Benjemaa and Schmitt [6] extend the unit quaternion approach used by Horn [52] to the multi-view case. Like Horn, Benjemaa and Schmitt take the cost function and decouple the rotations from the translations, re-expressing the same cost function in terms of rotation only. Unlike Horn, Benjemaa and Schmitt’s method is not closed form, and thus the solution is estimated using an iterative process. During the iterative process, only the rotation is estimated since the translational component of the cost function has been re-expressed in terms of rotation. However, from an implementational point of view, the translation components are calculated due to the way the transformation update is applied to be able to determine the current error between views, and therefore whether convergence has been achieved.

The general approach of their method is as follows. One view (point set) is used as a reference frame by having an identity translation and rotation associated with it which remains constant and thus remains fixed. During each iteration, each view’s rotation is estimated independently. This is achieved by allowing the point set whose rotation is to be estimated to move while the other point sets are kept fixed. Once that rotation has been estimated, the rotation for the next point set is determined. This continues until all of the optimal rotations have been obtained. The optimal translations are then calculated by using a linear combination of differences between the rotated centroids. This process is continued until convergence.
6.1. N-View Alignment

6.1.4.1 Implementation Overview

Derivation of the method of Benjemaa and Schmitt is mathematically sophisticated. However, the implementation is based around a few formulae, and thus the basic recipe for implementation which includes these formulae follows. Additional background material for the equations used can be found in Benjemaa and Schmitt [6, 5] as well as Horn [52].

Prior to the main iterative loop, various matrices shown below are constructed that will be needed subsequently and remain unchanged throughout the iterative process.

1. Initialise the matrix $\tilde{A}$.

$\tilde{A}$ is a $(M - 1 \times M - 1)$ matrix, where $M$ is the number of point sets. $N^{\alpha\beta}$ refers to the number of points between set $\alpha$ and $\beta$. Hence $N^{\alpha\beta} = N^{\beta\alpha}$. The diagonal elements of $\tilde{A}$ are set to $N^{\alpha\alpha}$, due to suppression of the first column and row of $A$ which gives $\tilde{A}$. This is done so that there is a reference frame. $N^{\alpha} = \sum_{\beta=1}^{M} N^{\alpha\beta}$. The remaining elements, $\tilde{A}_{ij}$ are set to $-N^{\alpha\beta}$, where the subscripts are $i = \alpha - 1$ and $j = \beta - 1$, again due to the suppression.

2. Determine $\tilde{A}^{-1}$ which is used in the calculation of translations stage.

A padded version of $\tilde{A}^{-1}$ is required and therefore an extra first row and column is added containing the values of zero.

3. Calculate $Q^{\alpha\beta}$ which are used in the calculation of rotations stage.

$Q^{\alpha\beta} = Q^{\alpha\beta}_R + Q^{\alpha\beta}_t$, where $Q^{\alpha\beta}_R$ and $Q^{\alpha\beta}_t$ are $4 \times 4$ symmetrical matrices.

$$Q^{\alpha\beta}_R = \begin{bmatrix}
S_{xx}^{\alpha\beta} + S_{yy}^{\alpha\beta} + S_{zz}^{\alpha\beta} & S_{xy}^{\alpha\beta} - S_{zy}^{\alpha\beta} & S_{xz}^{\alpha\beta} - S_{zx}^{\alpha\beta} & S_{yx}^{\alpha\beta} - S_{xy}^{\alpha\beta} \\
S_{xy}^{\alpha\beta} - S_{yx}^{\alpha\beta} & S_{xx}^{\alpha\beta} - S_{yy}^{\alpha\beta} - S_{zz}^{\alpha\beta} & S_{xz}^{\alpha\beta} + S_{zx}^{\alpha\beta} & S_{xy}^{\alpha\beta} + S_{yx}^{\alpha\beta} \\
S_{xz}^{\alpha\beta} - S_{zx}^{\alpha\beta} & S_{xy}^{\alpha\beta} + S_{yx}^{\alpha\beta} & S_{xx}^{\alpha\beta} + S_{yy}^{\alpha\beta} & -S_{xz}^{\alpha\beta} + S_{zx}^{\alpha\beta} \\
S_{xy}^{\alpha\beta} - S_{yx}^{\alpha\beta} & S_{xz}^{\alpha\beta} + S_{zx}^{\alpha\beta} & S_{xy}^{\alpha\beta} + S_{yx}^{\alpha\beta} & -S_{xx}^{\alpha\beta} - S_{yy}^{\alpha\beta} + S_{zz}^{\alpha\beta}
\end{bmatrix},$$

where

$$S_{xx}^{\alpha\beta} = \sum_{i=1}^{N^{\alpha\beta}} x_i^{\alpha\beta} x_i^{\beta\alpha}, \quad S_{xy}^{\alpha\beta} = \sum_{i=1}^{N^{\alpha\beta}} x_i^{\alpha\beta} y_i^{\beta\alpha}, \quad S_{xz}^{\alpha\beta} = \sum_{i=1}^{N^{\alpha\beta}} x_i^{\alpha\beta} z_i^{\beta\alpha}, \ldots,$$
with $(x_i^{\alpha}, y_i^{\alpha}, z_i^{\alpha})^t = P_i^{\alpha}$ and $(x_i^{\beta}, y_i^{\beta}, z_i^{\beta})^t = P_i^{\beta}$.

$$
\Theta_i^{\alpha\beta} = \begin{bmatrix}
\tilde{S}_{xx}^{\alpha\beta} + S_{yy}^{\alpha\beta} + S_{zz}^{\alpha\beta} & \tilde{S}_{xy}^{\alpha\beta} - S_{zx}^{\alpha\beta} & \tilde{S}_{xz}^{\alpha\beta} - S_{yz}^{\alpha\beta} & \tilde{S}_{zy}^{\alpha\beta} - S_{yx}^{\alpha\beta} \\
S_{yx}^{\alpha\beta} - \tilde{S}_{xy}^{\alpha\beta} & \tilde{S}_{xx}^{\alpha\beta} + S_{yy}^{\alpha\beta} - S_{zz}^{\alpha\beta} & S_{zx}^{\alpha\beta} + S_{yz}^{\alpha\beta} - S_{yz}^{\alpha\beta} & S_{zy}^{\alpha\beta} + S_{yx}^{\alpha\beta} \\
S_{zx}^{\alpha\beta} - \tilde{S}_{xz}^{\alpha\beta} & \tilde{S}_{yz}^{\alpha\beta} - S_{zy}^{\alpha\beta} & \tilde{S}_{xx}^{\alpha\beta} + S_{yy}^{\alpha\beta} - S_{zz}^{\alpha\beta} & S_{zx}^{\alpha\beta} + S_{yz}^{\alpha\beta} - S_{yz}^{\alpha\beta} \\
\tilde{S}_{zy}^{\alpha\beta} - S_{yx}^{\alpha\beta} & S_{zy}^{\alpha\beta} + \tilde{S}_{yz}^{\alpha\beta} & S_{zx}^{\alpha\beta} + S_{yz}^{\alpha\beta} - S_{yz}^{\alpha\beta} & \tilde{S}_{xx}^{\alpha\beta} + S_{yy}^{\alpha\beta} - S_{zz}^{\alpha\beta}
\end{bmatrix},
$$

where

$$
\tilde{S}_{xx}^{\alpha\beta} = \sum_{k=1}^{M} \sum_{l=1}^{M} \mu_{kl}^{\alpha\beta} x_{ak} x_{bl}, \quad \tilde{S}_{xy}^{\alpha\beta} = \sum_{k=1}^{M} \sum_{l=1}^{M} \mu_{kl}^{\alpha\beta} x_{ak} y_{bl}, \ldots,
$$

with $(x^{ak}, y^{ak}, z^{ak})^t = N^{ak} \tilde{P}^{ak}$ and $(x^{bl}, y^{bl}, z^{bl})^t = N^{bl} \tilde{P}^{bl}$.

$$
\mu_{kl}^{\alpha\beta} = a_{kl} - a_{k\beta} - a_{\alpha l} + a_{\alpha\beta}, \text{ where } a_{ij} \text{ is the element } (i, j) \text{ of the matrix } \tilde{A}^{-1}\n$$

where $i$ and $j \in [2..M]$. In order to homogenise the indices of the sums, null terms with an index 1 are introduced for $a_{ij} = a_{i1} = 0$.

To obtain the optimal rotation, Benjemaa and Schmitt state that $H$ needs to be maximised.

$$
H = \sum_{\alpha=1}^{M} \sum_{\beta=1}^{M} (\hat{q}^{\alpha\beta} \hat{q}^{\alpha})^t \Theta_i^{\alpha\beta} (\hat{q}^{\alpha\beta} \hat{q}^{\alpha})
$$

where $\hat{q}$ is a unit quaternion, which is a four-dimensional vector $(q_0, q_x, q_y, q_z)^t$, and $\hat{q}^*$ is the conjugate of $\hat{q}$ ($\hat{q}^* = q_0 - iq_x - jq_y - kq_z$). When keeping all quaternions $q = (q_2, q_3, \ldots, q^M)^t$ fixed except for one, the maximisation of $H$ becomes a much simpler problem. The quaternion $q^1$ is set to the identity quaternion, meaning that the first view is kept as the reference frame. By estimating one rotation at a time by keeping all other views fixed the vector $q$ containing the quaternions is built up incrementally. Before the iterative process commences $q_0$ will be initially defined to contain identity quaternions for each view. For each subsequent iteration, the transition from $q_m$ to $q_{m+1}$ is done in $(M - 1)$ steps. The first step determines $q^2_{m+1}$, the second $q^3_{m+1}$ and so on.

Benjemaa and Schmitt define this simpler maximisation equation for each individual quaternion $\hat{q}^i$ as

$$
H(\hat{q}^i) = 2\hat{q}^i N_i \hat{q}^i,
$$
where \( N^j = \sum_{\beta=1,\beta\neq j}^M Q^{\beta\beta} Q^{j\beta} Q^{*\beta} \), and where

\[
Q^{*\beta} = \begin{bmatrix}
q_0 & -q_x & -q_y & -q_z \\
q_x & q_0 & -q_z & q_y \\
q_y & q_z & q_0 & -q_x \\
q_z & -q_y & q_x & q_0 
\end{bmatrix}
\]

However, Benjemaa and Schmitt point out that the optimal unit quaternion which maximises this function is the eigenvector corresponding to the highest eigenvalue of the matrix \( N^j \) since \( H(q^j) \) is a quadratic form. Therefore once this eigenvector is obtained the associated quaternion in the vector \( q \) is updated. Afterwards, all of the other views' rotations are subsequently estimated individually.

Once the rotations have been estimated, the translations can be calculated. To determine the translations, the vector differences between rotated centroids are determined, and by using the matrix \( \tilde{A}^{-1} \) created in the initialisation step, a list of translations corresponding to each view are calculated.

1. Calculate matrix \( \tilde{B} \).

\( \tilde{B} \) is a \((1 \times M)\) matrix, containing the differences between the rotated centroids of the overlaps. \( \tilde{B} \) is \( B \) without its first element, again for the reason of having a reference frame, i.e. the rotation and translation of the first set having the identity values (a null transformation). An element of \( \tilde{B} \) is therefore

\[
\tilde{B}_i = \sum_{\beta=1}^M N^{e\beta} [R^e(\tilde{p}^{e\beta}) - R^\beta(\tilde{p}^{e\beta})], \quad \text{where } e = i + 1 \text{ and where } \tilde{p}^{\alpha\beta} = \frac{1}{N^{\alpha\beta}} \sum_{i=1}^{N^{\alpha\beta}} p_i^{\alpha\beta}.
\]

2. Calculate \( \bar{x}_{\text{min}} \) which contains the optimal translations.

Matrix \( \bar{x}_{\text{min}} \) is then computed by calculating \( \bar{x}_{\text{min}} = -\tilde{A}^{-1} \tilde{B} \).

To summarise the matrices \( \tilde{A}^{-1} \) and \( Q^{\alpha\beta} \) \( \forall \alpha\beta \) are created prior to the iterative loop. Initially each view’s quaternion is set to the identity quaternion within the vector list \( q \). However, the first view is kept fixed and so its quaternion \( q^1 \) is never updated. During the iterative loop, each of the remaining quaternions are calculated individuially by determining the eigenvector that has the highest eigenvalue of the
matrix $N^j$ associated with $H(q^j)$. Once all of these quaternions within $q$ have been updated by composition, the loop could recommence. However, a measure needs to be calculated to determine whether the point sets have become aligned. This measure is provided by the mean square error (MSE) and the past three iterations’ MSEs are used to determine whether convergence has occurred. To be able to calculate the MSE, the translations need to be known, and thus they are computed each iteration using the difference between rotated centroids $\tilde{B}$ to ascertain the list of translations $\tilde{x}_{\min} = -\tilde{A}^{-1}\tilde{B}$ for all the views.

### 6.2 Results

To characterise the three methods a series of numerical experiments were performed to determine the rate of convergence, accuracy, stability, and computational time required by the methods.

#### 6.2.1 Implementation

The Pennec algorithm is the easiest to implement providing a 2 view point set alignment method is available. Building of the mean shape is relatively easy, and all that is then required is to be able to align each view to the mean shape. Recomputation of the mean shape using the estimated transformations is straightforward. For the other two algorithms, the implementations require more effort as the calculations are more complex.

The Stoddart and Hilton algorithm has several free parameters which are chosen heuristically. The present implementation is based on a quality controlled Euler routine which requires some tuning. In contrast, the Pennec algorithm and the Benjemaa and Schmitt algorithm are parameter free other than the termination criterion and threshold.
6.2.2 Creation of Synthetic Data Sets

Synthetic data sets were generated from 3D surface models by a process intended to emulate a multiple view range data acquisition. First of all, a 3D model is selected and subsequently points were randomly sampled from the surface. Each of the $\alpha = 1..M$ views had an associated view direction and thereby only a subset of the randomly sampled points from the surface were visible to that view resulting in a subset of points $S^\alpha$, i.e. the view's point set. For each pair of views a subset of points are likely to correspond, and thus those points create a correspondence set. A correspondence set is a set of pairs of points such that $p^\alpha \in \text{view } \alpha$ and $p^\beta \in \text{view } \beta$ where $p^\alpha = p^\beta$ is believed to be the same point. In the registration of surfaces, matches are established and the correspondences used are approximate ones. In these $n$ view point set alignment experiments, the actual correspondences used are true correspondences, and hence the use of synthetic data.

A correspondence set is constructed between two views when the randomly sampled points lying on the 3D model are simultaneously visible from both views. Each random point on the surface is evaluated to see if it is visible from both views. If it is, then each view within the correspondence set (overlap set) will have an identical copy of that point, i.e. $O^{\alpha\beta} = O^{\beta\alpha}$. Hence the views within each correspondence set are perfectly aligned. However, after adding noise it will no longer be true that $O^{\alpha\beta} = O^{\beta\alpha}$, i.e. $O^{\alpha\beta} \neq O^{\beta\alpha}$. This generation of correspondence sets is performed for all combinations of views used. The number of correspondence sets generated will depend on the number of views specified and the characteristics of the 3D model used.

The views used in the experiments were chosen to get the maximum coverage of the 3D model. For each experiment the same views were used. The number of views chosen were 2, 3, 6 and 18. This allows a sequence of tests of increasing difficulty. The views $(0,0,1)$ and $(0,1,0)$ were used in the two view case. The three view case used the additional view $(0,0,-1)$, and for the six view case, the additional views $(-1,0,0)$, $(1,0,0)$ and $(0,-1,0)$ were used. These are illustrated in figure 6.5. The eighteen view case was generated by rotating the six views around the x, y and z axes individually by $45^\circ$. 
6.2.2.1 Chosen Transforms

In order to test the algorithms each view's point set has a transformation applied to move it away from its pre-aligned state. This allows ground truth to be known. The transform for the first view (view 0) is always null, and thus can be used as a reference frame. For each view the rotation and translation is incremented. The rotation for the second view (view 1) is $1^\circ$ and for each subsequent view the angle is increased by $1^\circ$. The rotation axis is always $(1, 1, 1)$. The translation for view 1 is $(0.2, 0.2, 0.2)$, and for subsequent views the $x$, $y$ and $z$ components are incremented.
by 0.2. Hence each view has a unique rotation and translation associated with it. The last view having the largest transform applied. By not using randomly generated transforms, any differences between successive experiments using the same 3D data and number of views can be attributed solely to differences other than the initial transforms, such as varying levels of noise.

### 6.2.2.2 Adding Noise

To be able to test how well the methods perform when noise is present, each coordinate of the synthetic measurements had zero mean Gaussian noise with RMS $\sigma$ added. This corresponds to isotropic noise with RMS $\sqrt{3}\sigma$ when considering the RMS error on vectors. The noise is set in terms of a percentage $p$ of the diagonal of the bounding box of the noise free data, $B$ as follows

$$\sigma = pB/100 \quad (6.5)$$

### 6.2.3 Quantitative Measures Used

There are two quantitative measures that we can use to evaluate the result of registration. Firstly, since the transformations required to register the views are known, the error between the ground truth and the estimated rotations and translations for each view determined by the three methods can be used. For convenience only the error associated with the last view is reported. The rotational error in units of degrees is denoted as $\delta \theta$ and the translational error as $\delta T$.

The second measure is the residuals between corresponding points after registration. This should be a weighted average over all the point pairs and is given by

$$e = \frac{\sum_{\alpha} \sum_{\beta} \sum_{i} \sum_{\beta} \sum_{i} \sum_{\alpha} \sum_{\beta} w_{i}^{\alpha\beta} \| f^{\alpha} \ast p_{i}^{\alpha\beta} - f^{\beta} \ast p_{i}^{\beta\alpha} \|^2}{\sum_{\alpha} \sum_{\beta} \sum_{i} \sum_{\beta} \sum_{i} \sum_{\alpha} \sum_{\beta} w_{i}^{\alpha\beta}} \quad (6.6)$$

Since a known amount of noise $\sigma$ has been added to each component of the point pair it is expected that

$$e = \sqrt{3}\sqrt{2}\sigma \quad (6.7)$$
Chapter 6. N-View Point Set Alignment: A Comparison

The $\sqrt{3}$ takes account of the 3 components $(x, y, z)$, and the $\sqrt{2}$ accounts for the fact that noise has been added to both points.

6.2.4 Convergence

The experiments start by considering data sets with no noise added. Initially 2 views are used which is an artificial problem for n point set registration as it is possible to solve the problem by registering views in a pairwise manner. However, this case is very useful for determining the rate of convergence of the algorithm.

The dataset is derived from a surface model of an icosahedron with unit radius. The icosahedron had 50 random points chosen from which the 2 views are built from. Figure 6.6 shows the convergence of $e$ as a function of iteration number.

![Figure 6.6: 2 views: $e$, 50 points, no noise.](image)

It can be seen that Benjemaa converges in 1 step, Pennec converges in 1 step and Stoddart converges in 45 steps. The one step convergence of Benjemaa is to be expected since in the 2 view case it is equivalent to existing analytic methods. As a purely numerical method the convergence of Stoddart is as expected. The method of Pennec is somewhat faster than might be expected in this case, but it too contains a 2 view analytic method within. For this problem it is expected that $e$ will converge to zero, it is observed that all methods converge to a number in the region of $10^{-15}$. In other words the algorithms all converge to a number close to full machine precision.

A more meaningful test of the algorithm is a case where there are more than 2 views.

The next case that is considered has 200 points sampled from the icosahedron and
6.2. Results

pg 181

Figure 6.7: 6 views e, 200 points, no noise.

uses 6 views. There were 12 overlap sets. No noise was added. The convergence is illustrated by the graphs in figure 6.7. As can be seen all methods show geometric convergence but Stoddart and Pennec converge faster than Benjemaa. The results are summarised in table 6.1. It can be seen that all methods converge to full machine precision. The fastest method is Stoddart.

Table 6.1: 6 views: 200 points, no noise.

<table>
<thead>
<tr>
<th>method</th>
<th>iterations</th>
<th>cpu</th>
<th>$e\ 10^{-16}$</th>
<th>$\delta \theta\ 10^{-14}$</th>
<th>$\delta T\ 10^{-16}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pennec</td>
<td>51</td>
<td>4.99</td>
<td>7.81</td>
<td>2.99</td>
<td>9.93</td>
</tr>
<tr>
<td>Benjemaa</td>
<td>156</td>
<td>0.80</td>
<td>5.60</td>
<td>2.62</td>
<td>5.44</td>
</tr>
<tr>
<td>Stoddart</td>
<td>48</td>
<td>0.24</td>
<td>81.99</td>
<td>32.28</td>
<td>25.51</td>
</tr>
</tbody>
</table>

In the next case noise equivalent to 0.5% of the diagonal of the bounding box is added. The results are summarised in figure 6.8 and table 6.2. The predicted value for $e$ is 0.0353 which is consistent with the result in the table.

Table 6.2: 6 views: 200 points, 0.5 noise.

<table>
<thead>
<tr>
<th>method</th>
<th>iterations</th>
<th>cpu</th>
<th>$e$</th>
<th>$\delta \theta$</th>
<th>$\delta T$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pennec</td>
<td>10</td>
<td>1.00</td>
<td>0.0356877</td>
<td>0.303154</td>
<td>0.00305913</td>
</tr>
<tr>
<td>Benjemaa</td>
<td>12</td>
<td>0.06</td>
<td>0.0356877</td>
<td>0.303154</td>
<td>0.00305913</td>
</tr>
<tr>
<td>Stoddart</td>
<td>15</td>
<td>0.07</td>
<td>0.0356877</td>
<td>0.303154</td>
<td>0.00305913</td>
</tr>
</tbody>
</table>

$^2$Geometric convergence is where each subsequent value of an iteration differs by a constant multiplier. On a logarithmic plot this can be seen as a straight line.
An unexpected result is the overshoot of Benjemaa in the angle graph which is not visible in the graph of e. It does seem that Benjemaa is more affected by increasing the number of views, as can be seen in a figure of δθ convergence for the 200 point 18 view case shown in figure 6.9.

6.2.5 Highly Nonspherical Models

The results in the previous section are representative of the overall behaviour of the various methods as applied to a dataset that comes from a regular approximately spherical shape. It is believed that there are several situations where the behaviour of the algorithm may be much worse. Thus one such case in which the data comes from a highly non spherical object is tested. The object is generated from the previously used icosahedron by scaling two axes by a factor of 1000. The result is a long thin cigar shaped object. Hence when points are collinear in the two view case,
6.2. Results

The 6 view case with 200 points and no noise is first considered. The results are shown in figure 6.10 and table 6.3. It is clear that the Stoddart method now fails completely and Benjemaa produces a significantly worse answer than Pennec. As expected the angular error has become much worse \((10^{-10})\) due to the fact that this particular experiment has begun to approach a degenerate case.

Table 6.3: Degenerate, 6 views: 200 points, no noise.

<table>
<thead>
<tr>
<th>method</th>
<th>iterations</th>
<th>cpu</th>
<th>(e)</th>
<th>(\delta \theta)</th>
<th>(\delta T)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pennec</td>
<td>50</td>
<td>5.44</td>
<td>1.889e-15</td>
<td>1.186e-10</td>
<td>2.927e-12</td>
</tr>
<tr>
<td>Benjemaa</td>
<td>129</td>
<td>0.70</td>
<td>1.729e-13</td>
<td>9.183e-09</td>
<td>2.268e-10</td>
</tr>
<tr>
<td>Stoddart</td>
<td>22</td>
<td>0.13</td>
<td>1.978e-05</td>
<td>2.848</td>
<td>0.070</td>
</tr>
</tbody>
</table>

If noise of 0.001% of the bounding diagonal is added the results shown in figure 6.11 and table 6.4 are obtained.

Table 6.4: Degenerate, 6 views: 200 points, 0.001 noise.

<table>
<thead>
<tr>
<th>method</th>
<th>iterations</th>
<th>cpu</th>
<th>(e)</th>
<th>(\delta \theta)</th>
<th>(\delta T)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pennec</td>
<td>18</td>
<td>1.94</td>
<td>3.847e-05</td>
<td>0.0266727</td>
<td>0.0006627</td>
</tr>
<tr>
<td>Benjemaa</td>
<td>41</td>
<td>0.21</td>
<td>3.847e-05</td>
<td>0.0266725</td>
<td>0.0006627</td>
</tr>
<tr>
<td>Stoddart</td>
<td>24</td>
<td>0.14</td>
<td>4.308e-05</td>
<td>2.84937</td>
<td>0.0703454</td>
</tr>
</tbody>
</table>

In figure 6.12 the behaviour of \(\delta \theta\) under noise for 3, 6 and 18 views is shown. Some
unusual convergence behaviour is visible for the Benjemaa method but it does make steady progress to the solution as measured by $e$.

### 6.2.6 Conclusion

It is clear that Pennec’s method is by far the easiest to implement. There are no parameters to choose. Its rate of convergence is geometrical. It is the only method
that consistently gives high accuracy solutions.

It is noted that Pennec's method is by far the slowest and in applications where accuracy is important, the additional CPU time would not be a major disadvantage. That it is the slowest is an inevitable consequence of the fact that the other algorithms use CPU time proportional to the sum of the number of points and the number of iterations, whereas Pennec uses time proportional to the product of the number of points and the number of iterations.

The method of Benjemaa and Schmitt is harder to implement and suffers from a slight loss of accuracy for the near degenerate case. If speed is the most important criteria it is the best algorithm.

The Stoddart and Hilton method has the disadvantage of requiring additional parameters to be chosen. It fails in the near-degenerate case.

6.3 Summary

In this chapter, three n view point set alignment methods have been presented and discussed. A comparison of their behaviour has been performed where it is noted that all perform well in a non-degenerate case. Benjemaa and Schmitt's method is fast and performs well in both degenerate and non-degenerate cases. Stoddart and Hilton's fails in the degenerate case and Pennec's method is slow but is the most accurate.
Chapter 7

N View Surface Registration

In the process of model building it is known that several views of an object will be required to reconstruct it. Traditionally these models have been constructed by a pairwise technique of registering two views, fusing them, and registering subsequent views and fusing. The iterative closest point (ICP) algorithm for registering two surfaces was discussed in chapter 5.

The problem with registering pairs of views and fusing them is that any error present in registration between two views will be fixed when they become fused. Thus error propagation occurs during this pairwise approach. Also since the surfaces are not registered simultaneously, the information in all overlaps is not exploited.

To register two surfaces that have been acquired from two views, a 2 view surface registration method is required. One such method is the ICP algorithm which needs two components, a method for determining a closest point on a surface, and a way of aligning two point sets. Therefore to extend the 2 view surface registration method to register n surfaces, it should just be a case of substituting a n view point set alignment method for the 2 view point set alignment method. Three n view point set alignment methods have been implemented and tested as shown in the previous chapter. The work in this chapter shows how the modified ICP implementation can be extended to the multi-view problem.
7.1 Overview

Extending the 2 view ICP implementation to n view is not as trivial as just plugging in an implemented n view point set alignment method. This is due to the fact that instead of determining one set of correspondences, multiple sets of correspondences need to be considered.

In the 2 view case, the moveable surface was randomly sampled, obtaining a point set. This point set was then passed to a closest point method, obtaining another set of corresponding closest points. In the implementation of the n point set alignment methods, these two point sets were referred to as a correspondence set. Thus in the n view case, several correspondence sets exist. It is assumed for now that it is known how the surfaces overlap with each other. Then, for each of the overlaps a correspondence set is determined.

Once the correspondence sets have been built, they are then passed to a n view point set alignment method. The optimal rotations and translations for each view are then determined. These transformations are applied to all sets of sampled points. The transforms are also applied to the surfaces since the projection criteria for closest point methods use the position of the encoded surface. This means that each surface has to be re-encoded by the closest point method every iteration. Thus the method consists of building correspondence sets, determining transforms, applying these transforms to the surfaces and their associated sets of sampled points, re-encoding the surfaces and then iterating until convergence. The method for determining convergence will be similar to that presented in the previous chapters with the addition that there is a minimum number of 30 iterations to be performed.

To build correspondence sets, those surfaces that overlap with each other need to be found. If the surfaces are reasonably close, then one way of determining this is by using an average surface normal. Thus the normals associated with the sampled points (which are assumed to be evenly sampled from the surface) would be averaged and thereby provide an estimate of the direction in which the surface is pointing. If the angular difference between two average surface normals is small, the surfaces are assumed to overlap. A problem with the average surface normal is that if the
7.2. Implementation

A surface has a constant curvature, i.e. is almost spherical, then globally the average surface normal would not be meaningful. Another way of determining those surfaces that may correspond is by using proximities, i.e. if two surfaces are reasonably close, then they are assumed to overlap. One way of doing this is to use the difference between centroids of the two surfaces.

Now that two methods have been defined for determining which surfaces overlap, correspondence sets can be built. However, there is the problem that in cases where there is little overlap, only a subset of sampled points have true correspondences. Thus the boundary check and normal check introduced in section 5.5 will have to be used to reduce any incorrect drag caused by incorrect correspondences. The overall scheme for the n view surface registration is illustrated by a flow diagram in figure 7.1.

7.2 Implementation

In this section implementation issues are discussed. The most vital data structure is the correspondence set. A basic correspondence set consists of the two view identities, their two point sets, and an associated set of weights. In addition the correspondence set may include sets of surface normals and a list of distances between points.

A correspondence set is constructed in two stages. Firstly a view’s sampled points and its view ID are copied to the correspondence set, along with the second view ID that it overlaps with. The sampled points along with the second view’s ID is passed to a closest point method, which then determines the correspondences. This identifies the values of the second point set. If the closest point method fails to determine a closest point at any stage, the associated weight list gets updated accordingly. When each correspondence is determined the distance between the two corresponding points is computed and the distance list is built. This is useful as these distances are required by the robustness checks and they do not have to be re-computed each time.
Figure 7.1: Flowchart overview of the n view surface registration scheme.
7.2. Implementation

When the surfaces are initially considered, they are stored in an encapsulated structure, which stores the original surface and its closest point encoded form. The encoded form is obtained by applying the view's current overall transform (the compositions of all previously computed transforms for that view obtained in previous iterations) to an on-the-fly copy (a copy made during execution of the software) of the original surface and thus is then encoded. This is done to reduce rounding errors, i.e. if the surface is rotated each iteration, rounding errors occur and they start to accumulate. It was noted that the use of the on-the-fly copy and surface re-encoding added little additional time to the overall computing time required to perform 2 view surface registration. The closest point method used was the surface based method. Additional computing time may be required when using different closest point methods. When transformations were applied to both surfaces, the computing time did increase but only marginally.

Similarly the data structure that stores sampled points of the surfaces and their normals, have originals and their working copies since these are transformed each iteration prior to the construction of the correspondence sets. This encapsulated data structure also includes the average surface normal which is updated each iteration and initially determined from the sampled point set representing the surface. When determining the average surface normal it is assumed that the randomly sampled points will be evenly distributed across the surface and so their associated normals will give a good approximation of the average surface normal. However, if there was a cluster of the points lying on a high curvature part of the surface, their normal vectors would likely skew the result of the average surface normal vector. Therefore an additional method may need to be used to assist in determining whether surfaces may overlap. One approach is to use the proximity of the two surfaces being considered. Thus this structure also stores the centroid of the sampled points which is updated each iteration. The average bounding box vector for all the surfaces can be computed and used as a relative measure to determine the closeness between surfaces. Thus surfaces are considered to overlap, if the distance between the centroids of two sampled point sets is less than a quarter of the average bounding box length. When using the average surface normal (angle) approach, the surfaces are
considered to overlap if the angular difference is less than 90°. Using both the angle and centroid approaches a better determination of whether surfaces overlap can be obtained.

The implementation has three main arrays. These are the correspondence sets, the closest point encoded structures and the sampled points structures. The updates for the latter two structures are relatively straightforward. The correspondence set array gets recreated each iteration when the correspondence sets are redetermined, and the size of this array is initially likely to fluctuate.

As mentioned earlier, distance lists are used to reduce unnecessary recomputation of distances between corresponding points since three processes use them. These are the boundary check, the distance check and the MSE computation itself. At the end of each iteration the overall (composite) transforms for each view are applied to on-the-fly copies of the original sampled point sets for use in the next iteration. Also, all correspondence sets need to have the transforms obtained in this iteration applied to them. Once this is done, the distance list is recomputed to reflect the new state, and then the MSE can be recalculated.

### 7.3 Results

In these experiments only the surface based closest point method is evaluated. This method was chosen since it was shown to be the most reliable closest point method in the 2 view case. Also the main focus of these experiments is on how the n view point set methods perform in the n view surface registration scheme. The two data sets Foot and Beethoven were used in these experiments. Again the create synthetic range image (cri) program was used to generate partial views. Both triangulated meshes and range images are produced by this program, but only the meshes are used in these experiments. The resolution used by the program cri to generate the data was 50 × 50.

1The actual transformations obtained from the n view point set alignment methods are applied to the correspondence sets, since the composite transforms from the last iteration have been applied to the on-the-fly copies of the sampled points during the construction of the current correspondence sets.
An additional data set is also used which is real data. Ten views of the bunny were captured via a rotational turntable as shown in figure 7.2, of which one view has been omitted from the figure. The surface curvature is complex, and obviously the results of registering all of these partial views will be of interest. These polygon meshes were derived from range data scans of a clay rabbit model that was captured by the Stanford Computer Graphics Laboratory. A Cyberware [20] 3030MS optical triangulation scanner was used. The bunny data set can be downloaded from http://www-graphics.stanford.edu/data/3Dscanrep.
The number of random sample points for experiments were kept the same, the number being 200, unless otherwise stated. The rotation axis remained the same throughout, which was set to the normalised version of the vector (1, 2, 3). The first view always remained fixed. The boundary check and normal check were used unless otherwise stated. Three quantitative measures are used: the RMS, the angular error and the translational error. For the angular error $\delta R$ and translational error $\delta T$, the results are reported for the last view only.

### 7.3.1 2 View Case

A simple 2 view case is used to validate the implementation and compare convergence curves between the methods. In this experiment, the implemented 2 view point set alignment code based on Arun et al. (used in the 2 body surface registration) is also used. The data model used is Beethoven, and the partial views used are (1, 1, 1) and (0.1, 0.5, −0.5) the same that were used in the 2 view surface registration experiments. To move the two views from their pre-aligned positions the rotation used was $6^\circ$ and the translation was (0.03, 0.02, 0.01).

![Figure 7.3: Beethoven, 2 view.](image)

From figure 7.3 it can be seen that all methods converge except for Benjemaa which terminates early after the minimum 30 iterations is exceeded. This is related to the fact that the implementation of Benjemaa does not fully use weights\(^2\). The actual

---

\(^2\)The Benjemaa n-view point set alignment implementation only uses weights internally when calculating the MSE.
RMS plotted on graphs and reported, always uses weights in its calculation. The behaviour of Benjemaa in this case just highlights the importance of weights. In table 7.1 it can be seen that all methods apart from Benjemaa converge to the same level of accuracy. The time taken is approximately the same for Arun et al. and Stoddart which are the faster methods. Pennec as expected takes longer by a factor of two.

<table>
<thead>
<tr>
<th>Method</th>
<th>Its</th>
<th>RMS</th>
<th>$\delta r$</th>
<th>$\delta t$</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arun</td>
<td>100</td>
<td>6.415x10^{-02}</td>
<td>4.744x10^{-03}</td>
<td>1.370x10^{-03}</td>
<td>1:50</td>
</tr>
<tr>
<td>Pennec</td>
<td>100</td>
<td>6.415x10^{-02}</td>
<td>4.744x10^{-03}</td>
<td>1.370x10^{-03}</td>
<td>3:26</td>
</tr>
<tr>
<td>Benjemaa</td>
<td>30</td>
<td>6.194x10^{-02}</td>
<td>1.225x10^{-01}</td>
<td>1.905x10^{-02}</td>
<td>0:35</td>
</tr>
<tr>
<td>Stoddart</td>
<td>100</td>
<td>2.040x10^{-12}</td>
<td>4.744x10^{-03}</td>
<td>1.370x10^{-03}</td>
<td>1:51</td>
</tr>
</tbody>
</table>

### 7.3.2 3 Views

In these experiments 3 views were used. The translation used for the second view was (0.03,0.02,0.01) and that for the third view was a multiple of this, i.e. (0.06,0.04,0.02). Similarly view 2 had a rotation of 1 degree and view 3 had a rotation of 2 degrees. The data set used was Beethoven, but in this case the views used were (0,0,1), (0,1,0) and (0,0,−1). This means that the views are all orthogonal to each other and that there should be some overlap between them.

When determining overlaps two methods were available, proximity and average surface normals. Normally the two methods are combined with a logical or operator. Hence, if at least one of the methods considers there to be an overlap then a correspondence set will be built. In this experiment the proximity method alone found 3 overlaps showing that this method does have some value.

The results of this experiment are shown in figure 7.4. Again in this experiment Benjemaa fails. It starts to converge, but then the incorrect correspondences are then highlighted as it starts to move away from the true optimal transformation.
What is interesting to note is that both Pennec and Stoddart and Hilton converge identically. The rotation acquired by both for view 2 was 1.69° and for view 3 it was 2.08°. The only difference here is that Stoddart takes approximately half the time to get to the same result. Table 7.2 shows the differences between the methods for the last view.

In the next experiment the foot dataset is used. The viewing directions used for

<table>
<thead>
<tr>
<th>Method</th>
<th>Its</th>
<th>RMS</th>
<th>$\delta r$</th>
<th>$\delta t$</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pennec</td>
<td>49</td>
<td>3.826x10^{-02}</td>
<td>1.695x10^{-03}</td>
<td>1.904x10^{-03}</td>
<td>6:39</td>
</tr>
<tr>
<td>Benjemaa</td>
<td>33</td>
<td>5.650x10^{-01}</td>
<td>1.005x10^{-01}</td>
<td>2.093x10^{-02}</td>
<td>2:06</td>
</tr>
<tr>
<td>Stoddart</td>
<td>100</td>
<td>3.826x10^{-02}</td>
<td>1.695x10^{-03}</td>
<td>1.904x10^{-03}</td>
<td>3:28</td>
</tr>
</tbody>
</table>
the 3 views were \((1, -1, 0), (0, -1, 0)\) and \((-1, -1, 0)\). This means that there is a 45 degree difference between each of the views. The same rotations and translations were used to move the three pre-aligned partial surfaces away from each other.

The program was first run using just the proximity (centroid difference) method to determine overlaps. When this was done the method considered that there was only one overlap, that being between view 1 and view 3. In this case both the methods of Stoddart and Benjemaa failed. The Stoddart method considered that the overall confidence as measured by the weights was too small, and the program terminated. With Benjemaa’s method one of the matrices involved was singular. Pennec’s method converged but not to the optimal solution. This is not surprising since only one pair of overlaps is considered. Figure 7.5(a), (b) and (c) shows the convergence for Pennec’s method in this case.

The program was re-run including the average surface normal (angle) method. In
this case an additional pair of views are considered overlapping, views 1 and 2. In this case all methods perform without failing. Benjemaa does not find the optimal solution due to weights not being used. The convergence behaviour of these graphs are shown in figure 7.6(a), (b) and (c). Pennec and Stoddart behave identically, except Pennec takes 4 minutes and 38 seconds to run, whereas Stoddart takes 2 minutes and 25 seconds to run. Again Stoddart takes approximately half the time. The rotation obtained by Stoddart and Pennec for view 2 was 1.24° and for view 3, 9.08°. The true values being 1° for view 2 and 2° for view 3.

In a further test the overlap determination methods were ignored and all unique correspondence sets were built, that being 3 in this case. This introduced the additional corresponding view pairs 2 and 3. The behaviour of the methods for this case is shown in figure 7.7(a), (b) and (c). Pennec and Stoddart behave identically. The RMS in the previous experiment was 0.24 and in this experiment it is 0.08. This
emphasises the importance of determining correctly which views overlap, even if the overlap is extremely small as appears to be the case for view 1 and 3. The final rotation obtained by Stoddart and Pennec for view 2 in this case was 1.12° and for view 3 it was 3.34°.
When registering only two surfaces the normal check was only found to be generally useful in the first couple of iterations, or when there was very little overlap between surfaces. To evaluate the usefulness of the normal check in the n view case the program was re-run with all correspondence sets built, but with the normal check switched off. In this case all methods converge but not to the ideal, see figure 7.8. This demonstrates that the normal check plays a vital role in helping to obtain the optimal transformations.

![Graphs](image)

Figure 7.9: Foot 3 view, angle and centroid, 2 csets. Normal, boundary and distance check.

Next the distance check was used to see what affects it has in the n view case. All correspondence sets were used, and all the methods failed because the last correspondence sets weights summed to zero. The correspondence set that was rejected by the distance check was between views 2 and 3. The distance check ramp start was varied from the default 1 standard deviation, to two and three times, with no
7.3. Results

success. The angle and centroid version of selecting views that are considered to overlap was also run. For the cases of 1 and 2 standard deviation for the distance check ramp start (dcrs), all methods failed. When the dcrs was set to 3, Pennec failed in the first iteration due to a SVD failure. Benjemaa failed in the eighth iteration due to the sum of weights being zero. Stoddart ran to full completion of the 100 iterations, although the RMS fluctuated (probably due to the distance check). Although this behaviour of oscillation has been observed before in the n view point set alignment comparative study in section 6.2.4, figure 6.7. The graphs for this angle and centroid run with the dcrs of 3 is shown in figure 7.9. As can be seen from this figure, most notably from the Stoddart run, is that while the RMS generally is decreasing the angular and translation errors are increasing. Therefore in this case the distance check causes divergence of the solution away from the optimal solution.

7.3.3 Increasing Sample Points

So far, only 200 sample points have been used per surface, meaning that each correspondence set contains 200 points. Increasing the number of sample points might improve the answers obtained. Two different sample point set sizes are used: 400 and 800 points. The foot experiment was re-run using the angle and centroid method of determining correspondence sets as well as the exhaustive correspondence set approach (using all unique correspondence sets).

The graphs for 400 points are shown in figure 7.10. It can be seen that faster RMS convergence happens when using the exhaustive correspondence set approach. Also in the angle and centroid case, the rotation for the last view moves away from the optimal value after 12 iterations. In the exhaustive correspondence set case, it is likely that better values would have been obtained if the process was not prematurely terminated after 100 iterations.

In the angle and centroid case, Benjemaa's method initially achieves a better rotational value but then diverges away from it due to not using weight values. However, Benjemaa's method for the last view obtains a better rotational value than the other
methods. For view 2, the rotation obtained is extremely high being 10.77° compared to the rotation of 1.30° achieved by both Pennec and Stoddart. However, when the exhaustive correspondence set approach is used, Benjemaa obtains even worse errors: 16° for view 2 and 9.82° for view 3. Again Pennec’s and Stoddart’s methods perform identically from a convergence point of view. In the all correspondence sets case, Pennec’s method takes roughly one and a half times longer to complete than
7.3. Results

In the 800 point case, it can be seen in figure 7.11 that there is a slightly slower convergence compared to the 400 point case. Due to the 100 iteration limit, the methods achieve a slightly worse result. If both 400 and 800 point cases are allowed to run with no upper limit on maximum iterations, a better result might be achieved in the 800 point case compared to the 400 point case.
Chapter 7. N View Surface Registration

Table 7.3 shows the differences between the 200, 400 and 800 point set case, for both angle and centroid runs and the exhaustive correspondence set runs. The most noticeable difference is the time taken to perform the registration. As expected Pennec takes the longest time to execute. Benjemaa completes the quickest in all cases due to not using weights. Stoddart’s execution time in these cases is approximately a half to two thirds of Pennec’s execution time in each case. The difference in execution time between two correspondence sets and three correspondence sets is fairly large, and this difference increases as the point set size increases. From the times shown for the methods in table 7.3, it can be determined that for Benjemaa the relationship between points and time is linear. For Pennec and Stoddart, the growth is slightly nonlinear for these experiments which only use 2-3 correspondence sets.

Table 7.3: Foot: affects of varying point set size.

<table>
<thead>
<tr>
<th>Method</th>
<th>Angle and Centroid (2 Csets)</th>
<th>Exhaustive CSet (3 Csets)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time</td>
<td>RMS</td>
</tr>
<tr>
<td>Pennc 200</td>
<td>0:04:38</td>
<td>0.238</td>
</tr>
<tr>
<td>Benjemaa 200</td>
<td>0:01:11</td>
<td>2.900</td>
</tr>
<tr>
<td>Stoddart 200</td>
<td>0:02:25</td>
<td>0.238</td>
</tr>
<tr>
<td>Pennc 400</td>
<td>0:05:45</td>
<td>0.268</td>
</tr>
<tr>
<td>Benjemaa 400</td>
<td>0:01:47</td>
<td>2.950</td>
</tr>
<tr>
<td>Stoddart 400</td>
<td>0:02:40</td>
<td>0.268</td>
</tr>
<tr>
<td>Pennc 800</td>
<td>0:11:39</td>
<td>0.183</td>
</tr>
<tr>
<td>Benjemaa 800</td>
<td>0:03:20</td>
<td>3.127</td>
</tr>
<tr>
<td>Stoddart 800</td>
<td>0:05:07</td>
<td>0.183</td>
</tr>
</tbody>
</table>

In the 400 point case for the exhaustive correspondence set run, the rotational error is four and half times less than that obtained when using 200 points, and the RMS is decreased by about half. Therefore, the number of points affects not only the accuracy but also the time taken to obtain a final answer.
7.3. Results

7.3.4 Bunny

In this experiment all 10 views of the bunny data set was used with the various point set sample sizes of 100, 200 and 400. View 2 is rotated by 1° and all subsequent views rotations are incremented by this amount. Similarly the same occurs for the translation, where view 2 is relatively translated by (0.03, 0.02, 0.01).

Figure 7.12: Bunny 10 view: 100 points.
Figure 7.12 shows the 100 point case with comparisons between using the exhaustive correspondence set approach and the centroid and average surface normal approach for deciding which correspondence sets get created and used. As expected, Benjemaa's method does the minimum of thirty iterations as required and then finishes. In the exhaustive correspondence set case, the angular error of Benjemaa actually increases. For Pennec and Stoddart, the methods terminate early due to one cor-
7.3. Results

Figure 7.14: Bunny 10 view: 400 points.

respondence set whose sum of weights are zero. This particular check is done after corresponding points have been determined for a particular set of points, and after robustness checks have been applied. In both the angle and centroid and the exhaustive correspondence set cases, the sum of weights being zero happened for the last correspondence set being determined. In the angle and centroid case, it happened during the 15th iteration for the 33rd correspondence set which was associated with
views 10 and 9. In the exhaustive correspondence set case, the same views were involved for the 45th correspondence set which happened during the 18th iteration.

Figure 7.13 shows the 200 point case and figure 7.14 shows the 400 point case. From looking at both sets of graphs it can be seen that as the point size is increased, the rotational error of Benjemaa for the final view also increases. Both Pennec and Stoddart’s method terminate early due to the sum of weights being zero. Again views 10 and 9 are involved for the last correspondence set. In the exhaustive correspondence set case this occurs in the 19th iteration for the 45th correspondence set for both 200 and 400 point sizes. For the angle and centroid case it happens in the 16th iteration. Here though, the number of correspondence sets used varied. This is likely to be caused by the random point sampling that has some influence on the average surface normal. In the 100 point case, there were 33 correspondence sets, in the 200 point case there were 32 and in the 400 point case 34.

Table 7.4: Bunny: affects of varying point set size.

<table>
<thead>
<tr>
<th>Method</th>
<th>Angle and Centroid (32-34 Csets)</th>
<th>Exhaustive CSet (45 Csets)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time</td>
<td>RMS</td>
</tr>
<tr>
<td>Pennec 100</td>
<td>0:22:53</td>
<td>0.00159</td>
</tr>
<tr>
<td>Benjemaa 100</td>
<td>0:13:36</td>
<td>0.00961</td>
</tr>
<tr>
<td>Stoddart 100</td>
<td>0:08:12</td>
<td>0.00159</td>
</tr>
<tr>
<td>Pennc 200</td>
<td>0:46:47</td>
<td>0.00149</td>
</tr>
<tr>
<td>Benjemaa 200</td>
<td>0:23:45</td>
<td>0.00959</td>
</tr>
<tr>
<td>Stoddart 200</td>
<td>0:15:41</td>
<td>0.00149</td>
</tr>
<tr>
<td>Pennec 400</td>
<td>1:42:31</td>
<td>0.00064</td>
</tr>
<tr>
<td>Benjemaa 400</td>
<td>0:47:34</td>
<td>0.01020</td>
</tr>
<tr>
<td>Stoddart 400</td>
<td>0:33:42</td>
<td>0.00064</td>
</tr>
</tbody>
</table>

Table 7.4 shows for the various dataset sizes the differences in time, RMS, rotational and translation errors for both angle and centroid and the exhaustive correspondence set cases. As expected, the time required to perform the registration increases for all n view alignment methods as the point set size increases and the correspondence
set size increases. Pennec's method takes approximately two to three times longer to execute than Stoddart's method. There is linear growth for the time taken for all of these methods. For all of the exhaustive correspondence set cases, it can be seen that Benjemaa took approximately the same time as Stoddart. However, Benjemaa did approximately twice as many iterations since it was unaffected by the sum of weights being zero. The rotational error can be seen to improve for both Pennec and Stoddart in the angle and centroid case as the point set size increases. In the exhaustive correspondence set case, the errors do not improve, they appear to get slightly worse. However, it has to be remembered that their runs were cut short and that since more correspondence sets are in use, it is likely that the convergence rate might be slightly slower.

Considering that full convergence was never achieved for either Pennec or Stoddart, rotational values for all views were never obtained. To try to get a fuller run, the 100 point case was re-run without building the correspondence set for views 9 and 10. Unfortunately both the angle and centroid and the exhaustive correspondence set cases hit the sum of weights being zero problem. This was for the last correspondence set. This time between views 8 and 9. For the angle and centroid case only 10 iterations were successfully performed, and the exhaustive correspondence set case 13 iterations. It is interesting to note that the sum of weights problem has happened for the last correspondence set each time, which is associated with the last two views being used. The robustness checks that use the RMS always assume that at each subsequent iteration the RMS will be lower during the registration process. However, if between iterations a view's overlap moves further away from its corresponding view's overlap while all other overlaps between views get closer, then the average distances between these correspondences will be greater than those between the other views. Therefore, if these average distances are considerably higher than the previous iterations' RMS, a robustness check may then consider all these points to be outliers and thus set all of their weights to zero. There is a possibility that the last view may be oscillating, moving around quite a lot, and for that particular iteration it is further away from the other views, thereby causing this early termination.
7.3.5 Conclusions

A comparison of the three n view point set alignment methods used by the n view surface registration method was presented. From this it is clear that determining which views actually have overlaps is vital, if the optimal transformations are to be obtained. Also the normal check appears to play a more significant role in n view surface registration than in the 2 view case. The distance check is a significant hindrance in the n view case, and often causes either failures or divergence away from the optimal solution.

Again the importance of the use of weightings is highlighted most strongly by the implementation of Schmitt and Benjemaa which at present does not fully use weights. Stoddart and Hilton's method tended to be two to three times faster than Pennec's method. These two methods converged identically in all experiments bar one, where the Stoddart and Hilton's method oscillated. However, Pennec's method tended to be more robust. Differences between these two methods may only appear as the complexity of experiments increase. In the bunny experiment involving 10 views it was seen that Schmitt and Benjemaa's method took approximately the same time as Stoddart and Hilton's method, but did twice as many iterations. Therefore an interesting prospect would be to see if a modified version of Schmitt and Benjemaa's method that supported weights is faster than Stoddart and Hilton's method while obtaining similar levels of accuracy.

Increasing the point set sizes used linearly increases the time required to perform the registration, but it also starts to slow down the convergence slightly. It appears that no significant gains in the solutions obtained will be achieved after increasing the point set sizes past a certain point.

When more views are used, it appears that the last view may be moving around a lot compared to the other views. So when using robustness checks, it is more likely that the last view's weights will all be set to zero. In the 2 view surface registration case, it made sense that if the correspondence set's sum of weights was zero to no longer proceed. However, in the n view case perhaps it should only terminate when all correspondence sets are in this state.
7.4 Summary

The novel scheme for n view surface registration was presented in this chapter. The strategy used in the implementation was also discussed. The implementation currently only uses the surface based closest point method that was shown to be the most reliable closest point method in the 2 view case. A comparative study of the three n view point set alignment methods used by the n view surface registration scheme was presented. From this, it was found that the normal check played a vital role in the n view case, and that correctly determining which views overlapped is vital to obtain an optimal solution.
Chapter 8

Self-Calibrating Surface Reconstruction For ModelMaker Data

Traditional range scanners are mounted on XY platforms or scan objects mounted on rotating platforms. These scanners have the limitation that they cannot capture complex geometry in a single scan.

Recently a company called 3D Scanners [92] has produced a sensor called the ModelMaker Reality Capture System (see figure 8.1) that facilitates the capture of more complex geometries. The sensor is based on a small hand-held laser striper mounted on an articulated arm that measures in real time the position and orientation of the striper. Fisher et al. [34] presented a prototype system based on a similar concept in 1996. Using this approach a cloud of point measurements can be captured. The data is subsequently processed into a single surface using a surface fusion algorithm developed at Surrey [44, 45] and incorporated into the ModelMaker product.

Using the striper, range measurements are taken from a series of consecutive stripes which do not form a regular grid pattern, and the measurements obtained are subject to error. The use of an arm then assists in determining the pose of the striper. Once the measurements have been obtained, a fusion algorithm is used which is based on a
volumetric scheme in which an implicit surface representation is created. Due to the fact that the volumetric fusion scheme uses voxels of finite size it can lose accuracy relative to the point data (although this is implementation dependent). On the other hand it is recognised that such a volumetric scheme is very good at extracting the correct topology from complex surfaces, which is a weakness of surface based schemes such as Turk and Levoy's [109] method. This is then subsequently triangulated using a marching cubes algorithm which tends to produce a large number of facets for the mesh.

In this chapter we consider a novel postprocessing method to improve the output of the ModelMaker data and thereby reduce the number of faces produced from the marching cubes algorithm. The postprocessing scheme is iterative and involves surface refitting, surface decimation and recalibration of the data. During the iterative process, the surface refitting scheme takes the output surface and the point data associated with that surface and performs an optimisation with respect to a cost function. This initially reduces the errors introduced by the volumetric scheme. Surface decimation is then used to reduce the number of triangles produced by the marching cubes algorithm. The stripes are then recalibrated using an iterative closest point algorithm based on Besl and McKay [11]. The algorithms for surface refitting and the decimation scheme based on edge collapse are derived from the work of Hoppe et al. [46]. Quantitative results for the postprocessing scheme are presented in this chapter for typical data and poorly calibrated sensor data produced by the ModelMaker where it is shown that these techniques significantly reduce the
8.1 Topology

In chapter 3, geometry was discussed along with the application of rigid body transformations. Position of points in space, relationships of lines (parallelism), and measurements of these geometric objects (distance and angle) are generally thought of as geometry. By applying a rigid body transformation to a geometric object, the position in space of that geometric object would change, but the geometric properties would remain the same. Another branch of mathematics is topology which describes the connectiveness of a set of points. With topology, distance is not relevant. The connectivity of a set of points is described by using edges which give the adjacency to each point. The location of each point is ignored. Thus different geometric objects can share a common topology. When a geometric object is rotated in space, the connectiveness of the points and their associated lines (edges) which make up the facets of the object do not change. So, when a surface is under continuous non rigid deformation such as stretching or twisting, the topology stays the same. Thus the connectivity between vertices remain the same. However, if a non-continuous change is made, such as a tear or hole, the topology of the surface will change.

8.2 Surface Refitting And Decimation

When an object’s surface has been obtained through a reconstruction process, the mesh obtained may not be truly representative of the data points acquired during capture. For example, this may be due to the fusion and/or surface polygonisation methods working with a resolution lower than that of the point data, and so finer surface details become averaged. Therefore, by using this point data along with the resultant mesh obtained, a refitting process can be performed so that the mesh is more representative of the point data. A simple example of surface refitting is illustrated in figure 8.2.
When acquiring 3D data of an object, a surface can be produced which may contain many vertices, e.g. a hundred thousand or more. These vertices are then polygonised producing a triangulated surface. It is likely that this process may not yield the most efficient representation. For example, a single plane may be represented by many triangular patches. The process of reducing the number of triangles which represent a surface is called decimation. Ideally decimation would be done without losing accuracy of the representation of a surface. However, in practical circumstances, a compromise on how representative the decimated surface is compared to the original surface is needed. Such compromises are made to increase processing speed of rendering an object or doing complex manipulation of the surface quickly enough for real time applications, or to reduce the memory requirements to store the surface. Hence, decimation is often used where the complexity of a triangulated mesh is to be reduced but where the overall shape of the surface (mesh) is to be retained. Figure 8.3 shows the original mesh and its decimated version, where it can be seen that the number of triangles has been greatly reduced and the overall shape of the mesh has been kept. The theory of surface refitting and decimation is now considered in more detail.

To be able to do surface refitting, topological information about the surface needs to be known and encapsulated in some representation. The surface representation $S$ used is a set of triangles with a list of vertices denoted $V$ and with the topological connectivity of these vertices being denoted by $K$. Thus $S = \{V, K\}$ in the notation of Hoppe et al. [46]. The set of point measurements associated with the surface is denoted by $X = \{x_1...x_D\}$. The surface fitting process may be formulated as
8.2. Surface Refitting And Decimation

A minimisation problem over the surface vertex positions \( V = \{ v_1, v_n \} \) and mesh topology \( K \) with an objective function given by

\[
E_{\text{tot}}(K, V) = E_{\text{dist}}(K, V) + E_{\text{rep}}(K) + E_{\text{spring}}(K, V)
\]  

(8.1)

The objective function \( E_{\text{dist}}(K, V) \) is a sum of a data fidelity term measuring the sum of the squared distances \( d^2 \) from data to the nearest point \( x_i \) on the surface \( S \).

\[
E_{\text{dist}}(K, V) = \sum_{i=1}^{D} d^2(x_i, S)
\]  

(8.2)

During surface refitting, vertex positions are moved and additional vertices may also be added to obtain a better fit. This results in reducing the cost associated with \( E_{\text{dist}}(K, V) \). However, by adding extra vertices overfitting can occur, thereby reducing the cost of \( E_{\text{dist}}(K, V) \) to zero. Hence, a penalty term \( E_{\text{rep}}(K) \) based on the number of vertices, is introduced so that vertices are not added indefinitely. Similarly, it is desirable to remove vertices from a dense surface even if \( E_{\text{dist}}(K, V) \) increases slightly. In this case \( E_{\text{rep}}(K) \) acts as to encourage vertex removal, thereby decimating the surface. Therefore, \( E_{\text{rep}}(K) \) is set to be directly proportional to the number of vertices \( N_K \) of \( K \). The representation constant \( \lambda_{\text{rep}} \) may be chosen by the

---

Figure 8.3: An example of decimation. (a) Original triangulated mesh, (b) Mesh after being decimated.
user to make a compromise between detail and size of the representation.

\[ E_{\text{rep}}(K) = \lambda_{\text{rep}} N_K \quad (8.3) \]

By just minimising \( E_{\text{dist}} + E_{\text{rep}} \) non desirable results tend to be produced. Hoppe et al. showed that by just minimising \( E_{\text{dist}} \) by itself several surface spikes occurred due to data being not present in regions where the spikes occurred. Hoppe et al. stated that a minimum for \( E_{\text{dist}} + E_{\text{rep}} \) may not exist and thus added a spring energy term \( E_{\text{spring}}(K, V) \)

\[ E_{\text{spring}}(K, V) = \lambda_{\text{spring}} \sum_{\{j, k\} \in K} |v_j - v_k|^2 \quad (8.4) \]

where \( v_j, v_k \) are vertex positions of an edge. This term places on each edge of the mesh a spring of rest length zero and a spring constant \( \lambda_{\text{spring}} \). The spring term acts as a regularising term to help guide the optimisation to a desirable local minimum. As the optimisation converges to the solution, the spring constant \( \lambda_{\text{spring}} \) can be gradually reduced. Also, \( \lambda_{\text{spring}} \) need only play a role when there are triangles with no associated data points. After sufficient decimation the spring (regulariser) may be ignored.

### 8.3 A Brief Review Of The ModelMaker

In this section the theoretical model of the ModelMaker measurement process is developed. We suppose that there are \( j = 1..N \) "stripes" of data produced, each with \( i = 1..M_j \) data points denoted \( \tilde{x}_{ji} \). These are 3D coordinates in the striper reference frame. The stripes and data points acquired are illustrated in figure 8.4.

The covariance of these data points is assumed to be given by the 3x3 matrix \( W_S = \begin{bmatrix} \sigma_S^2 & 0 & 0 \\ 0 & \sigma_S^2 & 0 \\ 0 & 0 & \sigma_S^2 \end{bmatrix} \). A crude isotropic noise assumption will be made, i.e. that the covariance matrix is diagonal and that the variance is \( \sigma_S^2 \) for each of the 3 components.

At the instant stripe \( j \) is measured, the coordinate measuring arm reports a position \( \tilde{t}_j \) and a rotation matrix \( R_j \). The translation and rotation are concatenated into a pose \( f_j \) using the the notation of Pennec [80]. This notation was introduced in
8.4 Motivation And Development Of The PostProcessing Scheme

When using ModelMaker to capture depth information of a real world object, each part of the objects' surface is "painted" in patches, and this continues until it is
fully captured (the parts of the surface that are fully accessible). When painting each patch, a foot pedal is used to start and stop the acquisition of what are termed “micropatches”\(^1\). To form the micropatch a series of points on adjacent stripes are triangulated. A threshold ensures that holes or step edges are not closed. The micropatches are then fused into a resulting surface using a volumetric fusion technique by Hilton et al. [44].

During this capture process there are potentially many sources of error but only the 3 most significant are considered. Firstly, the measured striper points are noisy with some RMS \(\sigma_s\). Secondly, the striper pose will be in error by \(e_j\), which will cause the point RMS in the world coordinate system to be a larger value \(\sigma_w\). Finally the volumetric fusion technique may cause additional errors, depending on the exact method chosen. These errors may arise if each voxel stores a condensed version of the voxel contents or if the polygonisation algorithm makes simplifying assumptions. More sophisticated approaches by Hilton et al. [45, 43] can circumvent this, but there is a time vs accuracy trade-off.

The postprocessing chain begins with the output from the ModelMaker. As well as using the fused surface (the output surface from ModelMaker), the original micropatch data is also used. The fused surface becomes an initial guess \(S_0 = \{K_0, V_0\}\) and the micropatch data is converted to a point set \(X = \{x_{ij}, i = 1..N, j = 1..M_j\}\).

Firstly, surface refitting is done by minimising \(E(K, V)\) with respect to \(V\) using \(S_0\) as a starting point. This step has the effect of correcting some of the errors caused by the volumetric fusion, but benefits from having a good starting point. The next step is to choose some value for \(\lambda_{rep}\) and optimise over both \(K, V\) so as to decimate the surface.

A useful measure to assess the effect of these operations is the RMS distance of data points from the closest point on the surface, which we denote \(d_{rms}\). The effect of refitting is always to reduce \(d_{rms}\), provided that the regularising constant is small. The more the surface is decimated the more it will inevitably raise \(d_{rms}\) once again.

\(^1\)The version of software used when obtaining these micropatches from the ModelMaker was version 1.0.1.4.
When the ModelMaker captures data, the points in each stripe are converted to the world coordinate system by a rigid body transform $f_j^e \circ f^A \circ e_j$ where the error modelled is $e_j$. Many of the stripes overlap so there is redundant information present and it is argued that this information may be used to recalibrate the data. A simple way of doing this, is by applying a (small) transform $g_j$ to each stripe, thus building a set of these transforms $G = \{g_1..g_N\}$. The distance cost can be reformulated

$$E_{\text{dist}}(K, V, G) = \sum_{i=1}^{D} d^2(g_j * x_{ji}, S)$$

(8.7)

To minimise $E_{\text{tot}}$ simultaneously over the surface vertex positions $V = \{v_1..v_n\}$, the topological connectivity of these vertices $K$, and $G$ presents formidable implementational obstacles. Therefore some simplifications are introduced. The first simplification is to iterate over alternate minimisations with respect to $V$ and $G$. Therefore the problem is broken down into two familiar tasks, namely surface refitting (optimisation over $V$) and registration of several point sets to a surface (optimisation over $G$), i.e. the "recalibration" step is nothing more than the registration typically solved using the Iterated Closest Point algorithm [11, 104].

The second implementational simplification is to group sets of stripes together into micropatches and let $j$ denote not stripes but micropatches. The error associated with a micropatch is assumed to be small, and thus individual stripes will not be recalibrated. However, from the results it is clear that much of the benefits of the recalibration is retained. This is probably due to the fact that some significant sources of error vary only slowly over the arm workspace, and the short term random noise on our arm is not too large in comparison.

The mesh optimisation, i.e. surface refitting and surface decimation, was implemented by Dr. Andrew Stoddart using standard techniques published in the literature, mainly Hoppe et al. [46]. For the recalibration step the modified 2 view ICP algorithm was used.
8.5 Results

All of the data presented in this section was collected on a "ModelMaker" at the University of Surrey. The striper is mounted on a Faro Bronze B06 arm. The arm has a quoted accuracy of 0.15mm single point repeatability (one standard deviation) for point measurements. The orientation accuracy is not stated. The more expensive Faro Silver arm has a spec of 0.04mm. One source of error is the temporal update rate which is 16.7ms for the Bronze, whereas the Silver can be synchronised to within 0.1ms.

What is the accuracy of the points measured by ModelMaker? 3D Scanners provide spherical and cubic calibration objects and software routines to compute the errors. A portion of the calibration sphere (radius = 37.925mm) was scanned with a single micropatch and with several overlapping micropatches. A sphere was fitted to these measurements and the RMS error $e_{rms}$ from the sphere was computed. For one micropatch the RMS error was 0.1278mm and for several micropatches the RMS error was 0.2387mm.

From these results it is concluded that striper point noise in the striper coordinate system, $\sigma_S$, is about 0.1278mm or less, but that the error in the world coordinate system can rise to 0.2387mm and above due to arm errors. These figures are consistent with the notion that there is no benefit in either the scanner sensor or the CMM (coordinate measuring machine) arm being very much more accurate than the other!

In the following discussion two measures of surface quality will be used. The more reliable is the sphere RMS error $e_{rms}$ which is about as good a measure of the instrument accuracy as can be easily got. The second is the RMS distance to surface $d_{rms}$ which is a questionable measure of accuracy since in the limit of enough triangles this can always be reduced to zero. However, when the number of measurements per face rises above 5, i.e. about 10 points per vertex it is apparent that we are not overfitting. [ Typically there are approximately twice as many faces as vertices. ] In this case it is argued that $d_{rms}$ is a measure of "local accuracy", i.e. the accuracy of points relative to other nearby points. This accuracy measure will not be sensitive
to global distortions but will reliably quantify measurement of local shape variation. In the absence of a large set of calibration objects $d_{rms}$ is useful.

### 8.5.1 Sphere

Results for data taken from the calibration sphere are now presented. All surfaces are shown with flat-shaded rendering for easier interpretation.

Figure 8.5(a) shows the original mesh $S_0$ (4,861 faces) constructed from the 12,498 points obtained via ModelMaker. The accuracy of the original point data is $e_{rms} = 0.2277$. Six micropatches were used in the surface reconstruction.

<table>
<thead>
<tr>
<th>Stage</th>
<th>No. of Triangles</th>
<th>$d_{rms}$ mm</th>
<th>$e_{rms}$ mm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial 1</td>
<td>4,861</td>
<td>1.0692</td>
<td>0.2277</td>
</tr>
<tr>
<td>Refit 1</td>
<td>4,861</td>
<td>0.1715</td>
<td></td>
</tr>
<tr>
<td>Decimation 1</td>
<td>876</td>
<td>0.1844</td>
<td></td>
</tr>
<tr>
<td>Initial 2</td>
<td>876</td>
<td>0.1572</td>
<td>0.1717</td>
</tr>
<tr>
<td>Refit 2</td>
<td>876</td>
<td>0.1389</td>
<td></td>
</tr>
<tr>
<td>Decimation 2</td>
<td>735</td>
<td>0.1419</td>
<td></td>
</tr>
<tr>
<td>Initial 3</td>
<td>735</td>
<td>0.1390</td>
<td>0.1655</td>
</tr>
<tr>
<td>Refit 3</td>
<td>735</td>
<td>0.1352</td>
<td></td>
</tr>
<tr>
<td>Decimation 3</td>
<td>720</td>
<td>0.1355</td>
<td></td>
</tr>
<tr>
<td>Initial 4</td>
<td>720</td>
<td>0.1351</td>
<td>0.1640</td>
</tr>
<tr>
<td>Refit 4</td>
<td>720</td>
<td>0.1331</td>
<td></td>
</tr>
<tr>
<td>Decimation 4</td>
<td>715</td>
<td>0.1332</td>
<td></td>
</tr>
</tbody>
</table>

After the initial refit stage, registration of the micropatches is then performed. These newly registered points are then used in the next refit stage. This process is then repeated. The results are summarised in table 8.1.

The $d_{rms}$ for the original surface, as shown in table 8.1, was 1.0692. This is worse than $e_{rms}$ reflecting approximations made in the surface fusion and possibly some boundary effects. The fusion voxel size was set at 2mm. After doing a surface refit,
$d_{\text{rms}}$ dropped from 1.0692 to 0.1715. The original number of triangles were 4,861 and after decimation they were reduced to 876. $d_{\text{rms}}$ rose slightly to 0.1844 as expected.

Once registration of the micropatches had been performed, the newly registered point set was used in the 2nd surface refit, where $d_{\text{rms}}$ then dropped from 0.1572 (improvement due to registration) to 0.1389. The rest of the results can be seen from Table 8.1.

At 17 data points per face we are not overfitting. The initial refitting causes a big drop in $d_{\text{rms}}$ drop from 1.0692 to 0.1715. The recalibrating stage then causes a further improvement by a factor 0.68 to 0.1332. This suggests that it is of real value.

More compelling evidence for real improvement is provided by comparing the $e_{\text{rms}}$ values on the point data. The drop is from 0.2277 to 0.1640, i.e. improvement by a factor 0.72. We conclude that we have reduced the errors that can be ascribed to the arm significantly, possibly by as much as half.

### 8.5.2 Corner

Figure 8.5(b) shows the original mesh constructed from the 27,957 points obtained via ModelMaker. Eight micropatches were used in the surface reconstruction, one of which is shown in figure 8.5(c). Figure 8.5(d) shows the final result obtained and figure 8.5(e) shows the same result including the triangulation.

<table>
<thead>
<tr>
<th>Stage</th>
<th>No. of Triangles</th>
<th>$d_{\text{rms}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial 1</td>
<td>9,332</td>
<td>0.3864</td>
</tr>
<tr>
<td>Refit 1</td>
<td>9,332</td>
<td>0.1120</td>
</tr>
<tr>
<td>Decimation 1</td>
<td>53</td>
<td>0.1527</td>
</tr>
<tr>
<td>Initial 2</td>
<td>53</td>
<td>0.1382</td>
</tr>
<tr>
<td>Refit 2</td>
<td>53</td>
<td>0.1148</td>
</tr>
</tbody>
</table>

$d_{\text{rms}}$ initially starts at 0.3864 and then drops to 0.1120 after refitting. After decimation, which has reduced the number of triangles from 9,332 down to 53, $d_{\text{rms}}$
8.5. Results

Figure 8.5: (a) Original Sphere. (b) Corner. (c) A micropatch of corners. (d) Final result. (e) Final result showing triangulation.

increases to 0.1528, which can be seen in table 8.2. Once registration of the micropatches has been done, $d_{rms}$ drops to 0.1382, and after the second refit the $d_{rms}$ drops to 0.1148. The refit improves by a factor 0.36 and the recalibration step by a further factor 0.75.

8.5.3 Gravy Dish

Finally results for a real object are shown. The object is a porcelain dish with a patterned relief. On average each part of the surface is scanned twice. In figure 8.6 (a) we show the fusion output with $d_{rms} = 0.3936$ and 12,236 faces. There are 10 micropatches and a total of 80,095 points. In figure 8.6 (b) we show a slightly decimated and refitted result ($d_{rms} = 0.1503$) which can be seen in Table 8.3. There are 23,6615 points per face. In figure 8.6 (c) we show the results after recalibration ($d_{rms} = 0.0964$).
Chapter 8. Self-Calibrating Surface Reconstruction For ModelMaker Data

Figure 8.6: Gravy dish: (a) The fused result from the micropatches. (b) Result after first refit and decimation (no registration). (c) Result from third refit.

Table 8.3: Results from dish refinement.

<table>
<thead>
<tr>
<th>Stage</th>
<th>No. of Triangles</th>
<th>$d_{rms}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial 1</td>
<td>12,214</td>
<td>0.3936</td>
</tr>
<tr>
<td>Refit 1</td>
<td>12,214</td>
<td>0.1446</td>
</tr>
<tr>
<td>Decimation 1</td>
<td>3,817</td>
<td>0.1503</td>
</tr>
<tr>
<td>Initial 2</td>
<td>3,817</td>
<td>0.1142</td>
</tr>
<tr>
<td>Refit 2</td>
<td>3,817</td>
<td>0.0980</td>
</tr>
<tr>
<td>Decimation 2</td>
<td>3,359</td>
<td>0.1006</td>
</tr>
<tr>
<td>Initial 3</td>
<td>3,359</td>
<td>0.1022</td>
</tr>
<tr>
<td>Refit 3</td>
<td>3,359</td>
<td>0.0958</td>
</tr>
<tr>
<td>Decimation 3</td>
<td>3,290</td>
<td>0.0964</td>
</tr>
</tbody>
</table>

8.5.4 Toy Man

The next result is that of a Toy Man, (in fact a footballer well known for his lachrymose tendencies). The model head is only 40mm high and the fusion software was run at a voxel size of 2mm. The ModelMaker has a workspace of about half a meter, so the very small model is a stringent test of its operational limits. If the head were part of a much larger scene the voxel size would be realistic.

In summary, an object has been chosen that is expected to visibly highlight the
improvements aimed for. The initial mesh has 3,699 triangles and 39,471 points. It has $d_{rms} = 0.9029$. After refitting we reduce $d_{rms}$ to 0.3440 (3,699 tris) and recalibration and refitting reduces it still further to $d_{rms} = 0.1349$ (1,535 tris, 22 pts per face). The results are shown in figure 8.7.

### 8.5.5 Bottle

To test how well the method performs with bad data, a fabric softener bottle was captured when it was known that ModelMaker’s striper was poorly calibrated. In

---

Figure 8.7: Toy man: (a) Original constructed mesh (front view). (b) First refit (front view). (c) Final refit with registration (front view). (d) Original constructed mesh (side view). (e) First refit (side view). (f) Final refit with registration (side view).
this particular case, it was thought that the range scanner (striper) itself was poorly calibrated. If one imagines scanning a planar patch of a surface, then the 2D profile seen by the scanner would be a straight line when the scanner is calibrated. However, it is thought that the profile may have been convex (curved). Once the data was acquired, it was fused at 2mm voxel resolution which normally eliminates ramps. However, due to the overlapping curved stripes, ramps occurred during the fusion process. Thus as expected, the data captured was non-typical for the Model Maker.

The strategy used in this particular case was to initially keep the refit springs parameter high, so that the noise from badly calibrated data would not be fitted to. Otherwise, the refit springs parameter, a physical analogy would snap the mesh to the noise. Thus the registration step is given a chance of doing a reasonable job. Decimation was not used for the first few iterations, so that the refit and registration steps could bring out the fine detail, such as the label’s edge on the front of the bottle.

The original constructed model is shown in figure 8.8(a) and (b), front and side view respectively. Making up that surface is 130,019 triangles and 54,566 vertices. From looking at the surface, it looks like an extremely badly painted bottle. The ramping has been quite noticeable around the high degree curvature of the thin left hand side of the bottle to the front, where the label of the product is located to.

After the initial refit, shown in figure 8.8(c) and (d) the definition of the label starts to appear. Also it can be noticed that the seam of the bottle within the handle section appears to have been brought out. However, two seams appear instead of one centre seam. This could be just due to purely bad data, with the arm over time, or a factor of the refit with the springs chosen, or a combination of both. Also from the side view shown in (d), the noise now becomes very distinct, as it manifests itself in a bubble like appearance.

In the third refit, the visible noise is greatly reduced. However, there are still two seams. In the forth iteration, the spring is reduced in the decimation stage, and thereby allowing the visible noise to be present again, as seen in figure 8.9(d). In the fifth iteration, the spring is tightened up in the decimation stage, and again
Figure 8.8: Fabric softener bottle acquired from poorly calibrated sensor.

Although a perfect bottle has not been achieved, as there are still two traces of a seam until the final iteration, where perhaps decimation has removed some of the relaxed in the sixth iteration.
Figure 8.9: Fabric softener bottle acquired from poorly calibrated sensor (cont).

detail of the two seams. However, what is to be noticed is that the RMS initially was 0.7094 where the bottle had 130,019 triangles, and in the final iteration the number
of triangles was reduced to 32,788 with a final RMS of 0.1288. More details of the RMS errors etc during the various iterations is shown in table 8.4. This is quite a significant result. The one difficulty still remaining, is that spring and decimation
Table 8.4: Results from bottle refinement.

<table>
<thead>
<tr>
<th>Stage</th>
<th>No. of Triangles</th>
<th>$d_{rms}$</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial 1</td>
<td>130,019</td>
<td>0.7094</td>
<td></td>
</tr>
<tr>
<td>Refit 1</td>
<td>130,019</td>
<td>0.4439</td>
<td>High spring value</td>
</tr>
<tr>
<td>Initial 2</td>
<td>130,019</td>
<td>0.2528</td>
<td>Spring value halved</td>
</tr>
<tr>
<td>Refit 2</td>
<td>130,019</td>
<td>0.1859</td>
<td>Spring value halved</td>
</tr>
<tr>
<td>Initial 3</td>
<td>130,019</td>
<td>0.3216</td>
<td>Spring reduced to a third</td>
</tr>
<tr>
<td>Refit 3</td>
<td>130,019</td>
<td>0.1292</td>
<td>Spring reduced to a third</td>
</tr>
<tr>
<td>Initial 4</td>
<td>130,019</td>
<td>0.2875</td>
<td>Spring unchanged</td>
</tr>
<tr>
<td>Refit 4</td>
<td>130,019</td>
<td>0.1250</td>
<td>Spring unchanged</td>
</tr>
<tr>
<td>Decimation 4</td>
<td>110,075</td>
<td>0.1267</td>
<td>Spring set to half of refit spring</td>
</tr>
<tr>
<td>Initial 5</td>
<td>110,053</td>
<td>0.2797</td>
<td>Spring unchanged</td>
</tr>
<tr>
<td>Refit 5</td>
<td>110,053</td>
<td>0.1266</td>
<td>Spring unchanged</td>
</tr>
<tr>
<td>Decimation 5</td>
<td>34,922</td>
<td>0.1358</td>
<td>Spring doubled</td>
</tr>
<tr>
<td>Initial 6</td>
<td>34,922</td>
<td>0.2872</td>
<td>Spring unchanged</td>
</tr>
<tr>
<td>Refit 6</td>
<td>34,922</td>
<td>0.1286</td>
<td>Spring unchanged</td>
</tr>
<tr>
<td>Decimation 6</td>
<td>32,788</td>
<td>0.1288</td>
<td>Spring halved</td>
</tr>
</tbody>
</table>

variables (how hard to try, and how many attempts at decimating) still have to be chosen by hand. Depending on how good the initial model is, affects the level of the spring chosen by the user of the technique. However, it is still trial and error.

8.6 Conclusion

A novel postprocessing chain was presented for ModelMaker data and has been shown qualitatively and quantitatively to achieve significant improvement. The resulting surface reduces the errors caused in fusion and in the arm errors so that we may approach the limiting accuracy of the striper.

It is a commercial reality that the higher the cost of the arm the better the accuracy
that may be obtained. Arm prices vary from £1,000 to £60,000 and alternative non-mechanical technologies are available. An intriguing prospect of this work is the possibility of using cheaper arms to obtain results as good as those currently obtained by expensive arms.
In this thesis the process of geometric surface registration for 3D model building has been considered. Problems associated with the standard ICP technique of surface registration have been highlighted and methods of reducing the influence of these have been presented. It was shown how the 2 view surface registration method could be extended to n view, and a technique for re-calibrating ModelMaker data was also presented. A summary of the main conclusions is now given.

9.1 Conclusions

When registering two surfaces which were either triangulated meshes or range images using the standard ICP algorithm, it was shown that the speed, accuracy and stability of the algorithm is affected by the closest point method used. The surface based closest point method was the most reliable, but the execution time was significantly more than the other methods. The range image search was most reliable when the surfaces being registered were close to each other. The voxel search method tended to be unreliable and unstable, and is not suitable for registration of surfaces. Having a reliable and stable closest point algorithm does not guarantee good registration. The accuracy of 2 view surface registration was further improved by adding robustness checks, most notably the novel boundary check. The distance check, surprisingly hindered registration, and the normal check did play a role, but
not a vital one when registering two surfaces. The accelerated ICP algorithm was also evaluated, and it was found that the more linear updates that occurred in the initial iterations, the better the resultant registration. When noise was modelled the error increased as was expected, but the rotation and transformation errors still remained tolerably low.

A comparison of three n-view point set alignment methods was performed and it was found that Pennec's method was the most reliable. Pennec's method was the easiest to implement, providing a 2 view point set alignment method is available. Pennec's method is very slow compared to the other two methods due to the fact that the mean shape is computed each iteration and this involves using all points. However, it is the most accurate method. Benjemaa and Schmitt's method is harder to implement and suffers from a slight loss of accuracy for the near degenerate case. This is the best algorithm to use when speed is the most important criteria. In a near-degenerate case these two methods converge, with Pennec's method achieving better convergence. Stoddart and Hilton's method fails in the near-degenerate case.

A scheme for extending the 2 view surface registration to n view registration was presented. A comparison of this n view surface registration method using the three different n view point set alignment methods was done. It was found that being able to determine which views actually overlap was vital. The role of the robustness checks in n view registration is even more important than in the 2 view registration case. The boundary check again played its role, but it was found that in the experiments performed that the normal check plays a more significant role in helping to determine incorrect correspondences. The distance check is a significant hindrance in the n view registration case, and often causes failures or the solution to diverge away from the optimal solution sought. The importance of an implementation supporting the use of weights was also highlighted by the implementation of Benjemaa and Schmitt's method, which currently does not fully use weights. As a result, the transformations obtained using this method in n view registration were not ideal. Interestingly, it was found that Pennec's method and Stoddart and Hilton's method converged identically when used in n view registration (except for one case, where Stoddart and Hilton's method oscillated). Pennec's method was slow but it was
more robust. Increasing the point set sizes used generally will improve the solutions obtained, but at the expense of a linear increase in time required to perform the registration and a slight slow down in convergence. It also appears that no significant gains in the solutions obtained will be achieved after increasing the point set sizes past a certain point. When more views are used, it appears that the last view may be moving around a lot compared to other views, which can cause problems when using robustness checks as well.

Finally a post-processing chain was shown that significantly reduced errors in the surface produced by the ModelMaker Virtual Reality System. The method consists of three steps, surface refitting, decimation and registration. The errors present in typical models produced by ModelMaker were reduced, approaching the limit of the accuracy of the striper. This post-processing method was also tested with poorly calibrated data which was highly noisy, and it was found that the method copes well. However, whether typical or poorly calibrated data is used, some parameters still have to be chosen by hand for surface refitting and decimation.

The thesis has established how accuracy, speed and stability of the ICP algorithm is related to the closest point method used. Robustness procedures have been found to be very useful in improving the registration of two surfaces, and even more so, when registering multiple surfaces. The self-calibrating post processing technique reduced errors of data acquired by an articulated laser striping capture system, approaching the accuracy of the striper itself. The method is also robust, and copes well when given bad data. The knowledge and methods presented in this thesis can be used in industry in applications such as reverse engineering.

9.2 Future Work

When using the ICP algorithm to register two surfaces it was found that stability and speed of the registration was affected by the choice of the closest point method. Different closest point methods have different strategies. Some assume that the surfaces are close and therefore tend to be fast in determining a closest point, but they tend to fail when the surfaces are far apart. Other methods, tend to cope
well in all situations but they tend to be slow. Therefore, surface registration could be improved further in respect to stability and speed by using dynamic switching of closest point methods. The switching can be determined by using the error associated with the cost function as a guide. When the error is high, the more stable method can be used, and once the error falls below a certain threshold, the other faster method can be used. This would work by using the method of re-encoding surfaces each iteration as introduced in the n view surface registration implementation. Additionally if a method failed in one iteration, other methods could be tried to see if this could be overcome, thereby improving the stability even further.

Currently the n view surface registration method uses a simple approach for determining which surfaces overlap, and as a result is subject to incorrect determinations being made. A better way needs to be found to reliably determine which surfaces overlap. It is also thought that better registration will be achieved if a subset of the sampled points associated with the specific region of overlap are used, instead of all of the sampled points. This might be done, by simply dividing the sampled points into volumes, and just trying each volume to see which volume of points obtains a better transform. Initially, a brute force technique might be used in determining which volumes are best, and a weighting scheme could then be subsequently used. As surfaces get closer, neighbouring volumes are re-evaluated and the confidence in these volumes may be increased.

The n view surface registration implementation has available only the surface based closest point method at present. This means that triangulated meshes can be registered, but not range images. To resolve this the range image search method could be used. The range image search simple projection mechanism should be improved so that if point lands outside the \( x,y \) area of the range image that it snaps to the appropriate border, instead of failing to determine a closest point. This will greatly improve registration of range images when large translational differences exist between them.

During registration of n surfaces, situations can exist where a correspondence set's
weights can be set all to zero. In the 2 view surface registration case, it made sense that if the correspondence set’s sum of weights was zero the process was terminated. This is not true when registering multiple surfaces. Instead, if a correspondence set’s weights are all zero, it should be ignored for that iteration. It should only stop, when all weights for all correspondence sets are zero. This implementational change should allow the bunny experiment to continue until completion allowing proper evaluation of the 10 view case.

A multi-resolution approach should also be used where point set sizes start off small and increase in size as the solution gets closer. This would be tied in with the convergence criterion. So instead of finishing, the point set size is increased, and registration continues. This would allow the best of both worlds, i.e. faster convergence and less time required to get the result.

Benjemaa and Schmitt’s method should also be adapted so that it fully supports weights. The importance of a method being able to use weights has already been highlighted. This would then allow a more meaningful comparison of the n view point set alignment methods used in registering multiple surfaces. It was noted in the original study that Benjemaa and Schmitt’s method has the potential of becoming the fastest method with similar levels of accuracy as the other two methods. Also, the comparison should be extended to examine what happens when noise is introduced.

Prediction was used in the 2 view surface registration to accelerate the search, but was not used in the extended n view surface registration scheme. In the n view case, multiple surfaces are moving, and thus prediction may or may not be possible. If they are assumed to be all moving in a consistent way, without jumping around too much, then prediction might be possible. Thus accelerating the registration process, and with the possibility of improving the result obtained.

The boundary and normal checks currently use weighting functions that initially suit large displacements between the surfaces to be registered. However, the weighting functions could be made dynamic so that as the surfaces become more aligned, the weighting functions can be tightened further improving correspondences.

As mentioned earlier the post-processing method still requires parameters to be
chosen by hand, and this is a process of trial and error. The importance of the initial spring values used in surface fitting is vital. If the surface being post-processed is very noisy and the surface refit spring constraint is set low, the surface will snap to the noise, making the re-calibration step redundant. If the optimal parameters can be determined, the resultant surface obtained is likely to improve. To be able to determine these optimal parameters, the distance cost which determines the average distance between the points associated with the stripes and the surface can be used. The higher the distance cost, the higher the spring initially used, and as this cost reduces over the iterations, the springs can therefore be reduced as the stripes are re-calibrated.
Appendix A

Mathematical Notation

A consistent notation has been used as much as possible throughout this thesis. The main mathematical notation for each chapter is summarised under their main sections. Where possible conflicts of notation exist, or a greater degree of clarity is required, they will appear under relevant subsections.

Sets such as points and weights have generally been represented using capitals, and their elements are in lower case subscripted form. However, matrices and cost functions are also expressed using capitals, although consistent use of lettering has been used to avoid confusion.

A.1 2-View Point Set Alignment

\[ p \]  \quad \text{Generic point.}
\[ p' \]  \quad \text{Modified generic point.}
\[ P^1 \]  \quad \text{Point set (1st set).}
\[ p^1_i \]  \quad \text{Point belonging to } P^1.
\[ P^2 \]  \quad \text{Point set (2nd set).}
\[ p^2_i \]  \quad \text{Point belonging to } P^2.
\[ \sigma_i \]  \quad \text{Noise associated with a specific point.}
\[ R \]  \quad \text{Rotation.}
Appendix A. Mathematical Notation

\[ T \] Translation.
\[ g \] Rigid body transform.
\[ g \ast p \] Rigid body transform applied to \( p \).
\[ g^2 \circ g^1 \] Composition of two rigid body transforms.
\[ m \text{ and } c \] Straight line fit parameters for least squares.
\[ d_i \] Distance between estimated straight line and an actual data point.
\[ \sigma \] RMS error.
\[ E \] Cost function.
\[ k() \] Kernel function.
\[ p^1_c \] Centroid of 1st point set.
\[ p^2_c \] Centroid of 2nd point set.
\[ Q^1 \] Vector difference set (1st).
\[ Q^2 \] Vector difference set (2nd).
\[ q^1_i \] Vector difference (1st).
\[ q^2_i \] Vector difference (2nd).
\[ N \] No. of elements in the sets \( P^1, P^2, Q^1, Q^2 \).
\[ \hat{R} \] Estimated \( R \).
\[ \hat{T} \] Estimated \( T \).
\[ H \] \( 3 \times 3 \) Matrix (SVD).
\[ U \] Upper triangular matrix.
\[ A \] Diagonal matrix.
\[ V \] Lower triangular matrix.

A.2 Determining The Closest Point

\[ S^1 \] First surface (query).
\[ S^2 \] Second surface (search).
\[ P^1 \] Query point set obtained from \( S^1 \).
\[ p^1_i \] A query point.
\[ P^2 \] Search point set.
A.3. 2 View Surface Registration

\[ p'_i \] Projected point (the projection of \( p^1_i \)).
\[ p'_n \] The closest point.
\( d \) Distance between the query point and the closest point.
\( n \) Current search depth (size of the current search neighbourhood).
\( d_{\text{max}} \) Current maximum search distance.
\( n_v \) Number of vertices or points for a range image.
\( n_s \) Number of searches (queries).
\( t \) Time to find one closest point.
\( \Delta b \) A measurement used for the division of space, either planar or volumetric.

**A.2.1 Surface Based Search**

\( t_i \) A specific triangle of the search surface.
\( \Delta b^2 \) Size of the planar squares with associated buckets.

**A.2.2 Voxel Search**

\( \Delta b^3 \) Voxel size.
\( d_v \) Half of the normals volume's height.

**A.2.3 Range Image Search**

\( P^{1} \_x, P^{1} \_y \) Range image grid resolution for \( x \) and \( y \).
slopeThreshold Slope threshold.
\( x, y \) Coordinate within a range square.
\( n_{\text{max}} \) Maximum neighbourhood search size.
\( d \) Distance associated with best closest point currently found.

**A.3 2 View Surface Registration**

\( \mathcal{M} \) Moveable surface (point set).
\( m_i \)  
Moveable point.

\( \mathcal{F} \)  
Fixed surface (point set).

\( f_i \)  
Fixed point.

\( W \)  
Set of weights.

\( w_i \)  
Weight.

\( D \)  
Set of distances between corresponding points.

\( d_i \)  
Distance between two corresponding points.

\( n \)  
Size of sets \( \mathcal{M}, \mathcal{F}, D \) and \( W \).

\( E \)  
Cost function.

\( g^0 \)  
Initial guess.

\( k \)  
Iteration.

\( \text{Closest[]} \)  
Closest point function.

\( MSE^k \)  
The MSE value associated with iteration \( k \).

\( RMS^k \)  
The RMS values of \( MSE^k \).

\( g^k \)  
Rigid body transform associated with iteration \( k \).

\( \Delta g \)  
Transform difference from \( g^k \) to new transform.

\( d_{\text{median}} \)  
The median of \( D \).

\( RMS \)  
Root mean square (standard deviation).

\( rmsthreshold \)  
RMS threshold.

\( \sigma \)  
Standard deviation.

\( \mathcal{N}_\mathcal{M} \)  
Set of normals associated with \( \mathcal{M} \).

\( \mathcal{N}_\mathcal{F} \)  
Set of normals associated with \( \mathcal{F} \).

\( \mathcal{N}_{m_i} \)  
Normal associated with point \( m_i \).

\( \mathcal{N}_{f_i} \)  
Normal associated with point \( f_i \).

\( \delta \theta, \delta \phi \)  
Normal angle check thresholds.

### A.3.1 Accelerated ICP

\( q^k, q^{k-1} \)  
Registration vectors for the current and previous iteration.

\( \Delta q^k \)  
Difference between the registration vectors \( q^k \) and \( q^{k-1} \).

\( \theta^k \)  
Angular difference between \( \Delta q^k \) and \( \Delta q^{k-1} \).

\( \delta \theta \)  
Angular alignment threshold.
A.4. N-View Point Set Alignment

$v^k$ Arc length argument value.
$v_{\text{max}}$ Maximum allowable update coefficient.
$a_1, b_1$ Coefficients for straight line prediction.
$a_2, b_2$ Coefficients for parabolic prediction.
$v_1$ Coefficient for straight line update.
$v_2$ Coefficient for parabolic update.
$q^k'$ Updated registration vector from prediction.

A.3.2 Results

$R_e$ Expected rotation.
$R$ Actual rotation obtained during ICP iteration.
$T_e$ Expected translation.
$T$ Actual translation obtained during ICP iteration.
$\delta r$ Angular difference between $R_e$ and $R$.
$\delta t$ Translational difference between $T_e$ and $T$.
$W_c$ Weight confidence.
$|| B ||$ Length of bounding box vector.
$n_s$ Percentage of $|| B ||$.
$\sigma$ Noise level.
RandomGauss Returns a value from a Gaussian distribution that has an RMS of 1.
MaxXYorZ Returns the largest component value of a vector.
$n_{\text{div}}$ Voxel division size.
$max (x,y,z)$ Returns maximum values for the $x$, $y$, and $z$ components.
$min (x,y,z)$ Returns minimum values for the $x$, $y$, and $z$ components.

A.4 N-View Point Set Alignment

$S^\alpha$ Set of points belonging to view $\alpha$.
$p_i^\alpha$ A point from $S^\alpha$.
$M$ Number of point sets.
Appendix A. Mathematical Notation

\( f^\alpha \) Rigid body transformation associated with view \( \alpha \).

\( R^\alpha \) Rotation associated with view \( \alpha \).

\( T^\alpha \) Translation associated with view \( \alpha \).

\( O^{\alpha\beta} \) Overlap between \( S^\alpha \) and \( S^\beta \).

\( N^{\alpha\beta} \) Number of points in \( O^{\alpha\beta} \).

\( p_i^{\alpha\beta} \) A point from \( O^{\alpha\beta} \).

\( w_i^{\alpha\beta} \) Weight associated with \( p_i^{\alpha\beta} \).

\( E \) Cost function.

A.4.1 Pennec

\( M \) Mean shape.

\( m_r \) Element of mean shape.

\( r(i, \alpha, \beta) \) Mapping function.

A.4.2 Stoddart And Hilton

\( F_{tot}^{\alpha} \) Overall force of springs acting on centre of mass for view \( \alpha \).

\( \tau_{tot}^{\alpha} \) Overall torque of springs acting around centre of mass for view \( \alpha \).

\( \gamma \) Mass.

\( \Gamma \) Moment of inertia.

\( \omega \) Angular velocity.

A.4.3 Results

\( e \) Overall weighted residual error.

\( \delta r \) Angular difference for last view only.

\( \delta t \) Translational difference for last view only.

\( \sigma \) Noise added as a percentage \( p \) of the diagonal of the bounding box \( B \).
A.5 N View Surface Registration

RMS Overall weighted residual error.

δr Angular difference for last view only.

δt Translational difference for last view only.

A.6 Self-Calibrating Surface Reconstruction For ModelMaker Data

S Set of triangles belonging to surface.

V List of vertices belonging to surface.

K Topological connectivity of vertices in V.

X Set of point measurements \( \{x_1, \ldots, x_D\} \) associated with surface.

\( E_{tot} \) Total cost function.

\( E_{dist} \) Cost function measuring squared distances between X and their closest points on the surface.

\( E_{rep} \) A representational cost function.

\( E_{spring} \) A cost function measuring the spring energy.

\( f^E_j \) Rigid body transformation associated with end of the arm.

\( f^A \) Rigid body transformation associated with the striper.

\( e_j \) Error present in arm’s pose.

\( f_j \) Rigid body transformation reported by the CMM arm, i.e. \( f^E_j \circ f^A \circ e_j \).

\( \sigma_s \) Noise associated with striper points.

\( \sigma_w \) Noise in world coordinate system.

\( g_j \) Rigid body transformation determined by the ICP.

\( d_{rms} \) Distance error associated with recalibrated data.

\( e_{rms} \) Distance error determined by fitting a sphere.
Appendix B

Published Papers

This appendix contains the following published papers:


MATERIAL REDACTED AT REQUEST OF UNIVERSITY
Bibliography


[29] Viewpoint Animation Engineering. 870 West Center, Orem, Utah 84057. USA.


Bibliography


