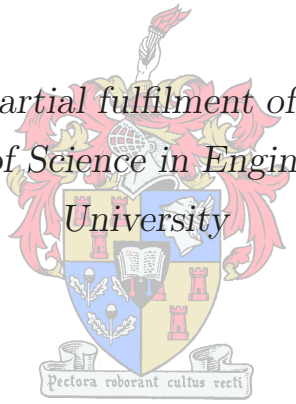
A georeferencing method for an open-pit mine surveying radar

by

Emile Francois Rossouw

Thesis presented in partial fulfilment of the requirements for the degree of Master of Science in Engineering at Stellenbosch University



Department of Electrical and Electronic Engineering
University of Stellenbosch
Private Bag X1, 7602 Matieland, South Africa

Supervisor: Dr. G-J van Rooyen
Co-supervisor: Mr A. Joubert

April 2011

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, and that I am the owner of the copyright thereof (unless to the extend explicitly otherwise stated) and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Signature: .....
E.F. Rossouw

Date: .....

Abstract

A georeferencing method for an open-pit mine surveying radar

E.F. Rossouw

Department of Electrical and Electronic Engineering
University of Stellenbosch
Private Bag X1, 7602 Matieland, South Africa

Thesis: M.ScEng

April 2011

Ground-based mobile mine survey radars are much more common now than they were a few years ago. Their ever growing popularity instigated the need for streamlining their operating procedures. One such a procedure is that of georeferencing the radar within the mine coordinate frame. Mine surveying radars have traditionally been georeferenced using a triangulation technique called survey resectioning, a time consuming process where both models are placed within the common coordinate frame by tying the models together with known targets or beacons. Survey resectioning requires surveying knowledge as well as access to a theodolite, an expensive high precision optical instrument used for measuring horizontal and vertical angles to the known targets. It is also sometimes necessary for this procedure to be performed in extreme weather conditions. Due to the limitations mentioned, this alignment method is not always practical or accurate if not performed correctly by the operator. In this thesis we investigate a new georeferencing scheme for ground-based mobile mine surveying radar, using a software-implementable three-dimensional model alignment. The scheme considers alignment complexity of four degrees of freedom and requires only an estimated radar position for complete convergence. The new scheme is tested on data previously georeferenced using the existing method.

Samevatting

A georeferencing method for an open-pit mine surveying radar

E.F. Rossouw

Department of Electrical and Electronic Engineering
University of Stellenbosch
Private Bag X1, 7602 Matieland, South Africa

Tesis: M.ScEng

April 2011

Die gewildheid van grondgebaseerde mobiele myn-radars is vinnig wêreldwyd aan die toeneem. Hulle word veral in oopgroef-myne aangetref. Hierdie snelgroeiende gewildheid het dit genoodsaak om hul roetiene-gebruiksprosedures te vereenvoudig. 'n Voorbeeld van 'n vereenvoudigde prosedure is om die radar makliker binne die myn-koördinaatstelsel te posisioneer. Dit is van uiterste belang dat hierdie prosedure so vinnig en akkuraat as moontlik moet geskied. Voorheen is hierdie radar-eenhede deur middel van 'n omslagtige driehoeksmetingstegniek geposisioneer (dikwels in ongunstige weersomstandighede). Hierdie tegniek (genoem landmetingskorrelasie), is 'n proses waar die radar-eenheid in die myn se koördinaatstelsel geposisioneer word deur middel van bekende punte of bakens. Landmetingskorrelasie vereis boonop van die gebruiker om landmetingskennis te hê asook om duur en ingewikkelde toerusting soos 'n teodoliet te gebruik. 'n Teodoliet word algemeen gebruik vir die opmeting van horisontale- en vertikale hoeke na bekende punte. Bogenoemde beperkinge van landmetingskorrelasie het tot gevolg dat die moontlikheid van verkeerde posisionering bestaan indien die oprerateur onervare is. Die proses kan ook lank neem. In hierdie studie word 'n nuwe radarposisioneringsproses ondersoek wat deur middel van drie-dimensionele modelbelyning geimplementeer word as 'n rekenaarprogram. Die proses neem vier grade van vryheid in ag, en benodig slegs 'n benaderde posisie van die radar-eenheid vir konvergensie. Die nuwe proses is getoets op bestaande data wat deur middel van die landmetingskorrelasie-metode belyn was.

Acknowledgements

I would like to thank those who helped make this thesis possible:

- My university supervisor, Dr. Gert-Jan van Rooyen, for your enthusiasm and believe towards my research. Your keen error-spotting eye and invaluable guidance made all the difference. You are a great supervisor!
- Anton Joubert, whom without this thesis would not have been possible. You selflessly gave your time when it was not expected of you. Your insight and uncompromising dedication as an engineer will inspire me always. You are a true mentor.
- Everybody at Reutech, including Prof. van der Walt, Johan Bras and Jaco Loots who helped to make this study possible.
- My parents, whom never even once asked "How long still...?" Your affectionate support and understanding will forever be appreciated.
- To my loving girlfriend Nadine, who always cheered me up when nothing seemed to make sense anymore. Your support means the world to me.

Contents

List of Figures

List of Tables

List of Abbreviations

3D	Three dimensional
ADS	Abstract data structure
BTS	Binary tree structure
CPC	Closest point correspondence
CSV	Comma seperated variable
DEM	Digital elevation model
DLL	Dynamically linked library
DTM	Digital terrain model
DXF	Drawing exchange format
FMCW	Frequency modulated continuous wave
GIS	Geographic information system
GPS	Global positioning system
GPU	Graphics processing unit
ICA	Iterative correspondence alignment
ICP	Iterative closest point
LAPACK	Linear algebra package
MinGW	Microsoft GNU for Windows
MS	Mean squared
PDF	Probability density function

Radar	Radio detection and ranging
RF	Radio frequency
RMS	Root mean square
SM	Synthetic map
STL	Standard template library
SVD	Singular value decomposition
TIN	Triangulated irregular network

List of Symbols

$\mathbb{N}$	The set of natural numbers
$\mathbb{N}_k$	The set of natural numbers $\leq k$
$\mathbb{Z}$	The set of integers
$\mathbb{Z}_k$	The set of integers $\leq k$
$\mathbb{R}$	The set of real numbers
$\boldsymbol{p}$	Vector
$\boldsymbol{p} = (x, y, z)$	$\boldsymbol{p}$ as a three-dimensional vector
$\boldsymbol{P} = \{\boldsymbol{p}_i\}$	Point set
N_P	The number of points in set $\boldsymbol{P}$
x	Scalar
$\boldsymbol{I}_n$	Identity matrix of dimension n × n
$\boldsymbol{A}^T$	The transposition of matrix $\boldsymbol{A}$
$\lvert\boldsymbol{A}\rvert$	The determinant of matrix $\boldsymbol{A}$
$\boldsymbol{R}$	A rotation matrix
$\boldsymbol{R}_x$	A 3 × 3 rotation matrix
$\boldsymbol{A} \subseteq \boldsymbol{B}$	$\boldsymbol{A}$ is a subset of $\boldsymbol{B}$
$\boldsymbol{A} \subset \boldsymbol{B}$	$\boldsymbol{A}$ is a proper subset of $\boldsymbol{B}$
$\boldsymbol{A} \times \boldsymbol{B}$	The Cartesian product of $\boldsymbol{A}$ and $\boldsymbol{B}$
$a \vee b$	The logical disjunction operator
$a \wedge b$	The logical conjunction operator
$\lVert\boldsymbol{p}\rVert$	The Euclidean norm of vector $\boldsymbol{p}$
$\operatorname{atan2}(y, x)$	A variant arctangent function
$\operatorname{random}(\boldsymbol{P})$	Elements of $\boldsymbol{P}$ are randomly permutated using a uniform distribution
$\min(\boldsymbol{P})$	Function which selects the element with the lowest value
$\max(\boldsymbol{P})$	Function which selects the element with the highest value

modulo (a, b)	The modulus of a using divisor b
cart2sph (x, y, z)	Function which converts three-dimensional Cartesian coordinates into spherical coordinates
delaunay$_{2D}$ $(\boldsymbol{X}, \boldsymbol{Y})$	Function which applies two-dimensional delaunay triangulation
unwrap $(\boldsymbol{A})$	Function which removes 2π jumps from the elements in $\boldsymbol{A}$
$\boldsymbol{F}[\boldsymbol{l}]$	The Fourier transform of $\boldsymbol{l}$
$c_{\text{tolerance}}$	The allowable tolerance between the calculated centre and the estimated centre
$r_{\text{tolerance}}$	The allowable tolerance between the estimated centre and template set corner points
$z_{\text{tolerance}}$	The allowable height difference tolerance of the height components of the template set cornet points

Chapter 1

Introduction

1.1 Motivation and topicality of this work

Ground-based mobile mine survey radars are much more common now than they were a few years ago [1]. They are considered to be extremely useful survey instruments, having the ability to construct a 3D model of an open pit mine within minutes [1].

Radars operate on the following principle: a radio-frequency (RF) signal is transmitted from the radar's antenna and is reflected by any object in the radar's beam (e.g. an aircraft or a mountain) (e.g. [2], [3]). A small amount of the reflected RF energy returns to the receiving antenna. The round-trip time is proportional to the range of the target. By moving the antenna in fixed angular steps, and sampling the target echo at each step, a radar range image (also known as a synthetic map (SM) [1]) can be built up. The SM models the target area when converted into Cartesian coordinates as a 3D point cloud. These models will be our data sets.

All open pit mines have their own coordinate system in which mine features, survey points and maps are represented. The radar resolves targets using its own local coordinate system, unaware of where it is within the mine coordinate system. A georeferencing method is required to transform the radar coordinate system into the mine coordinate system [1].

The premise of this work is to georeference a mobile mine surveying radar using a three-dimensional model alignment algorithm. Existing alignment algorithms are to be investigated as possible solutions to our problem, ensuring that the algorithm ultimately chosen converges to alignment within minutes of operation time, provides a heading accuracy of 2 degrees and an absolute position accuracy of 6 metres, considers surfaced and unsurfaced models, is implementable in a non-interpreted programming language without the use of platform-specific third-party libraries, is implementable as a cross-platform application, compatible with Microsoft Windows XP and Red Hat Linux and is implementable while easily accommodating possible integration with existing GNU C++ and Delphi 7 software.

The heading and absolute positions accuracies were chosen based on the radar's antenna beamwidth as well as the user required accuracy in general.

Model alignment has long been a key problem in the fields of computer vision, robotics and surveying and can be explained as follows: given two free-form 3D models, estimate the optimal rigid body motion which best aligns the models, minimising the distance between them. By aligning the 3D model sourced from the radar to an already existing model covering the same mine area, we indirectly determine the position and heading of the radar as well as how to merge all points into a common coordinate frame. This is useful, because having all surveyed points – irrespective of how they were sourced – in a common coordinate frame is much more intuitive than having them spatially uncorrelated [1]. Mine surveying radars have traditionally been georeferenced using a triangulation technique called survey resectioning [1], a time consuming process where both models are placed within the common coordinate frame by tying the models together with known targets or beacons. Survey resectioning requires surveying knowledge as well as access to a theodolite, an expensive high precision optical instrument used for measuring horizontal and vertical angles to the known targets. Due to the limitations, this alignment method is not always practical or accurate if not performed correctly by the operator.

The problem addressed by this study is that of georeferencing a ground-based mobile mine surveying radar using a software-implementable geometric algorithm. The position and heading of the radar is determined by aligning a radar range image (also known as a synthetic map (SM)) with an already georeferenced three-dimensional (3D) model. The algorithm which will align two models, one obtained from the mine surveying radar, to a template model referenced within a common mine coordinate frame. The template model will typically be sourced from another sensory device such as terrestrial laser scanner (or manual global positioning system (GPS) measurements) and will be referenced within the common coordinate frame. The template model could also have been sourced from a surveying radar, where the points have previously been placed in the common coordinate frame.

We present a practically implementable algorithm useful for this alignment problem. The algorithm allows for three degrees of translation freedom as well as one degree of rotation freedom, requiring only an estimated radar position for full alignment. The algorithm will overcome the existing shortcomings of radar georeferencing which are that it [1] is time consuming, requires physical user interaction, requires expensive hardware namely a total station (theodolite), cannot be done remotely and sometimes needs to be performed under extreme weather conditions.

Currently many general alignment algorithms exist. Some of their shortcomings are that they take too long to align models, require prealignment prior to alignment, are difficult to implement, sometimes only work for surfaced models and are often too generalised in the

sense that they sometimes consider unnecessary complexity [1].

The subject of literature covered in this study includes point cloud and surface alignment as well as other useful geometric algorithms. We first investigate studies related to motion parameter estimation between corresponding models (e.g. [4], [5], [6], [7]). We then cover least squares estimation of motion parameters between models including the very influential iterative closest point (ICP) algorithm introduced by Besl and McKay [7]. Then we consider studies showing methods of establishing correspondence between models which are different in size (e.g. [7], [6]). Next we consider the resampling of surface models and how that it is useful in model aligment [4]. We also cover the effect of repeated motion estimation and alignment [7]. Many existing alignment algorithms require models to be prealigned. We consider different methods of prealignment (e.g. [8], [9], [10]). Next we consider ways of reducing computational time during alignment using optimised data storage (e.g. [4], [7], [11]). We also cover studies which addressed issues such as special alignment cases [12]. Lastly we consider the different types of data sets used in the study.

1.2 Contributions

In accomplishing the objectives, a number of contributions were made:

- The development of a software implementable alignment scheme which does not require full prealignment, useful for georeferencing a mine surveying radar.
- The development of a resampling scheme to control the number of points used to represent a 3D model and for obtaining the maximum accuracy out of surfaced models through surface resampling.

1.3 Thesis overview

A literature study is provided in Chapter 2. We consider the various existing alignment algorithms. They are carefully scrutinised for complexity, ease of implementation, time of convergence and algorithm prerequisites.

In Chapter 3, the plausibility of the study is tested. Included in the chapter is the design of the alignment system and the limitations which have to be overcome. Some of the limitations of the problem present themselves. Firstly, models which are not prealigned tend to fail convergence during iterative correspondence alignment. We show how this limitation can be overcome with a new unique global alignment (also referred to as prealignment) algorithm. Secondly, points in the point set are sometimes unevenly distributed. We show how this limitation can be overcome by using surface resampling. Thirdly, sometimes it

is preferred for discrete point clouds to be surfaced. We show how this limitation can be overcome by a set of Delaunay triangulation algorithms.

Chapter 4 covers the software implementation of the design described in Chapter 3.

Chapter 5 is devoted to testing the system. Testing is performed on the new global alignment algorithm as well as the system as a whole.

The concluding chapter, Chapter 6, summarises the achievements of this work and suggests further research topics.

Chapter 2

Existing alignment algorithms

2.1 Introduction

In the chapter to follow, we look at various studies related to 3D model alignment. We cover the following useful concepts:

- The correction of motion between two models.
- Cost functions as a way to evaluate alignment progess and termination.
- Least squares estimation of motion parameters.
- How to establish point correspondence between models.
- The resampling (or interpolation) of surface models.
- The effect of repeated correspondence alignment.
- The different prealignment methods for iterative correspondence alignment (ICA).
- How to reduce computational time during alignment.
- Special alignment cases.
- The properties of the data sets used in the study.

2.2 Correcting rigid body motion

In the past, several efforts have been made concerning the registration or alignment of 3D point clouds. The alignment of models involves the minimisation of the spatial difference between them. One model remains stationary (template set), whereas a second model (search set) is manouvered spatially to best align with the first model. In order to minimise

this difference, the search set is repositioned to best align with the template set. Many authors (see e.g. [5], [6], [7], [4]), describe the problem of aligning rigid 3D point clouds (or models), as the estimation of the translation movement, as well as the movement due to rotation, often referred to as the rigid body motion parameters.

The rigid motion parameters are used to construct an acute transformation function, which aligns the search set with the template set. There are various ways of expressing this transformation. Arun and Blostein [5] use rotation matrix $\boldsymbol{R}$ and vector $\boldsymbol{T}$ to define the rotation and translation of one point $\boldsymbol{p}$ as

$$\hat{\boldsymbol{p}} = \boldsymbol{R}\boldsymbol{p} + \boldsymbol{T}$$

where $\boldsymbol{R}$ is an orthonormal 3×3 rotation matrix and column vectors $\hat{\boldsymbol{p}}, \boldsymbol{p}$ and $\boldsymbol{T}$ are of size 3×1. Rotations done via a rotation matrix are a fairly common method of transformation, but have certain disadvantages. Others (see e.g. [7], [4]) have chosen to make use of quarternions to define a rotation transformation. Quarternions (also known as Euler parameters) extend the complex number system into four dimensions. Euler showed that any rotation can be described by a rotation of a certain magnitude about a fixed axis. This means that the full space of 3D rigid body rotations can be described in four dimensions. A valid quarternion set describes the axis of rotation as a 3D vector and a magnitude of rotation.

A quaternion rotation transformation of magnitude β around fixed axis

$$\boldsymbol{n} = \begin{pmatrix} x_n, & y_n, & z_n \end{pmatrix}$$

can be described as

$$\begin{aligned} \boldsymbol{R} &= (q_4^2 - \mathbf{q}^T\mathbf{q})\boldsymbol{I}_3 + 2\mathbf{q}\mathbf{q}^T + 2q_4\boldsymbol{K}(\mathbf{q}) \\ &= \begin{bmatrix} (q_4^2 + q_1^2 - q_2^2 - q_3^2) & 2(q_1q_2 - q_3q_4) & 2(q_1q_3 + q_2q_4) \\ 2(q_1q_2 + q_3q_4) & (q_4^2 - q_1^2 + q_2^2 - q_3^2) & 2(q_2q_3 - q_1q_4) \\ 2(q_1q_3 - q_2q_4) & 2(q_2q_3 + q_1q_4) & (q_4^2 - q_1^2 - q_2^2 + q_3^2) \end{bmatrix} \end{aligned}$$

where

$$\boldsymbol{q} = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} = \begin{bmatrix} \sin\left(\frac{\beta}{2}\right)\boldsymbol{n} \\ \cos\left(\frac{\beta}{2}\right) \end{bmatrix} = \begin{bmatrix} \mathbf{q} \\ q_4 \end{bmatrix}$$

and

$$\boldsymbol{K}(\mathbf{q}) = \begin{bmatrix} 0 & -q_3 & q_2 \\ q_3 & 0 & -q_1 \\ -q_2 & q_1 & 0 \end{bmatrix}.$$

Quaternions, and quaternion algebra provide certain advantages over transformation using rotation matrices: The representation of a quaternion set requires only 4 scalars instead of 9, as in the case with rotation matrices, making it a more compact form. The quaternion transformation is not susceptible to gimbal lock, a condition where the rotation of a rigid body cannot be completed in a smooth continuous motion. Quarternion transformations are also far less susceptible to round-off errors.

Some methods [13] of motion estimation make use of dual quartenions. This is done so that the transformation can be handled as a homogeneous function. Zhang [4] describes a full rigid transformation using dual quaternions $\boldsymbol{s} = \begin{bmatrix} s_1 & s_2 & s_3 & s_4 \end{bmatrix}^T$ and $\boldsymbol{q} = \begin{bmatrix} q_1 & q_2 & q_3 & q_4 \end{bmatrix}^T$ as

$$\boldsymbol{Rp} + \boldsymbol{T} = \boldsymbol{W}(\boldsymbol{q})^T \boldsymbol{s} + \boldsymbol{W}(\boldsymbol{q})^T \boldsymbol{Q}(\boldsymbol{q}) \boldsymbol{p}$$

where

$$\boldsymbol{W}(\boldsymbol{q}) = \begin{bmatrix} q_4 & q_3 & -q_2 & q_1 \\ -q_3 & q_4 & q_1 & q_2 \\ q_2 & -q_1 & q_4 & q_3 \\ -q_1 & -q_2 & -q_3 & q_4 \end{bmatrix},$$

$$\boldsymbol{Q}(\boldsymbol{q}) = \begin{bmatrix} q_4 & -q_3 & q_2 & q_1 \\ q_3 & q_4 & -q_1 & q_2 \\ -q_2 & q_1 & q_4 & q_3 \\ -q_1 & -q_2 & -q_3 & q_4 \end{bmatrix}.$$

Despite the differences in transformation method, we prefer the rotation matrix method of transformation. Transformations using a rotation matrix are more intuitive and conversion into homogeneous coordinates makes for unnnecessary overhead. Also, the rounding errors made due to trigonometric functions are insignificant. The possibility of gimbal lock is also very small given that the correction of rotation will mostly be for only one Euler angle.

2.3 The best distance metric for alignment

In order to determine the progress in alignment, a metric is needed to measure the spatial difference between the template and the search model. This quantitive spatial difference will determine the quality of the fit. Besl and McKay [7], as well as Zhang [4] recommend using the mean-square Euclidean distance between models as a cost function to determine the quality of alignment. This metric is sensitive to outliers giving large errors more weight compared to small errors. Also, compared to root mean squared distances, mean squared distance is less expensive computationally.

2.4 Least squares estimation of the rigid body parameters

Least squares is an estimation method for overdetermined systems. Least squares means that the overall solution minimises the sum of the squares of the errors made in solving every single equation of the system. Arun and Blostein [5] estimate the motion parameters $\boldsymbol{R}$ and $\boldsymbol{T}$ by minimising a cost function

$$\boldsymbol{\Gamma}^2 = \sum_{i=1}^{N} \left\| \boldsymbol{p}_i^{'} - (\boldsymbol{R}\boldsymbol{p}_i + \boldsymbol{T}) \right\|$$

where $\boldsymbol{p}_i$ is a point from the search set and $\boldsymbol{p}_i^{'}$ is a point from the template set. They suggest maximising the trace of $(\boldsymbol{RH})$ (where $\boldsymbol{H}$ is the decoupled correlation matrix between the sets) using singular value decomposition (SVD). The optimal rotation is formed by the multiplication of the unitary matrices of the decomposition. Horn's [14] estimation of the optimal rotation is similar. He, however, also estimates a scaling factor by minimising the cost function

$$\boldsymbol{\Gamma}_s^2 = \sum_{i=1}^{N} \left\| \boldsymbol{p}_i^{'} - (s\boldsymbol{R}\boldsymbol{p}_i + \boldsymbol{T}) \right\| .$$

Scaling is not a factor for our specific problem, and hence shall not be considered. Horn's method also differs by using eigendecomposition, as apposed to SVD, to estimate the optimal rotation. He finds the rotation matrix which maximises function

$$\boldsymbol{\Upsilon} = \sum_{i=1}^{N} \boldsymbol{p}_i^{'} \cdot (\boldsymbol{R}\boldsymbol{p}_i) . \tag{2.1}$$

Besl and McKay [7] similarly estimate the ideal rotation by maximising (2.1) because it represents the dot product sums of all vectors in the sets. When the sets are ideally rotated,

the dot product sums will be maximised [14]. This ideal rotation will result in an overall system solution which minimises the sum of the squares of the errors, based on Arun and Blostein's minimising cost function. For our final implementation we use Besl and McKay's estimation method since we only require the maximum eigenvector. Decomposition of the maximum eigenvector is easier to implement than full SVD decomposition. We use an iterative Jacobi method to estimate the maximum eigenvector. See Appendix A for more detail.

2.5 The methods of correspondence

Many of the motion estimation algorithms presented (see e.g. [5], [14], [13], [15]), only work when the template and the search set are of the same size and when the correspondence between point sets are known. That is to say, there is an one-to-one mapping for the points in the template set to that of the points in the search set (Figure 2.1). Non-correspondence is when that mapping is unknown.

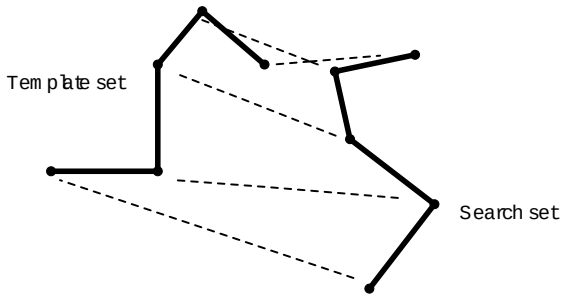


Figure 2.1: The correspondence mapping between two 2D point sets is indicated by the dotted lines in this figure. Note how the sets are of the same size, and differ only by spatial rotation and translation.

Obtaining correspondence between a template and search set is not always trivial, especially if the search set is a spatial subset of the template set. We consider two popular correspondence methods, namely closest point correspondence and correspondence through normal shooting.

2.5.1 Closest point correspondence

We consider the simple case of aligning two curve point sets (Figure 2.2(a)). The template set has roughly double the number of points when compared to the search set. Obtaining the rigid body parameters will be extremely difficult considering that we do not know which points from the template set to pair with the points in the search set. Besl, McKay

and Zhang ([7], [4]) suggest picking correspondence based on spatial similarity. Correspondence is established by a closest point search between the points in the search set and the template set. They consider a template set point closest to a search set point to be the best possible association. Figures 2.2 (a) and (b) show the sets before and after closest point correspondence. This method works well if models are relatively well aligned prior to alignment.

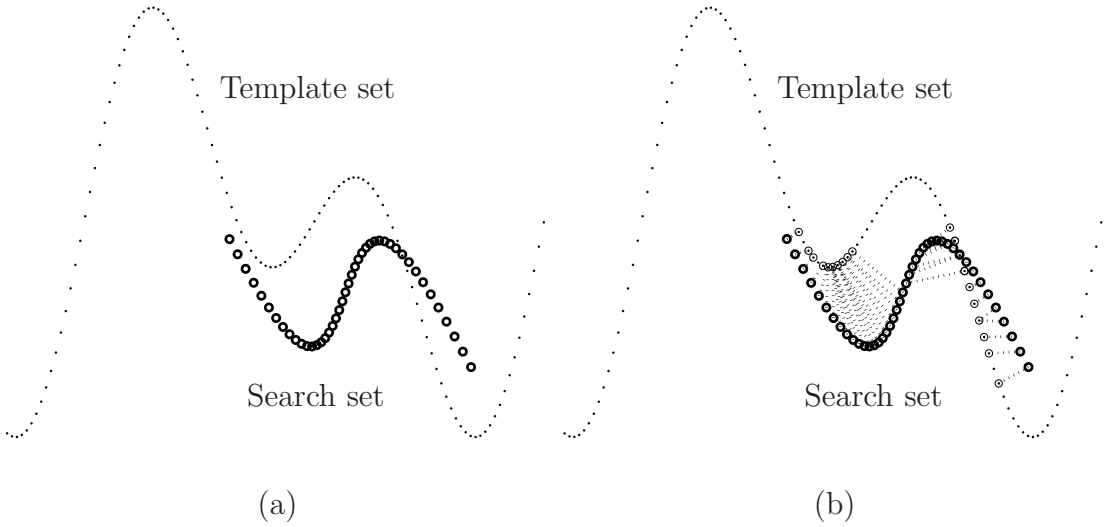


Figure 2.2: (a) Two misaligned curve point sets. (b) Correspondence between curve sets based on a nearest neighbour search.

2.5.2 Normal shooting correspondence

Chen and Medioni [6] suggest a more elaborate method to address correspondence. Their method, unlike closest point correspondence, can only be applied to meshed surface model sets. The search set's points are projected orthogonally onto the template set. The positions where surface normals successfully intercepted the model are used to determine the closest point associated with the original search set point (Figure 2.3). This method ensures that correspondence is made in a unified direction and favours alignment of very smooth surface models. Testing successful interception of mesh polygons is computationally expensive, especially when the meshes consist of elaborate polygon structures. This method, like closest point correspondence, works best when the models are reasonably aligned prior to alignment.

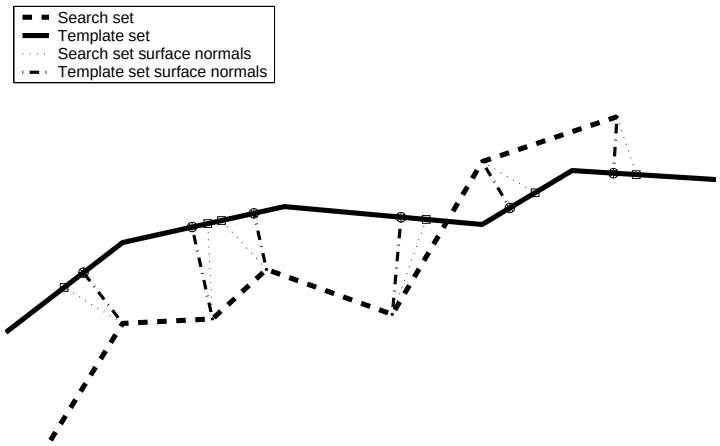


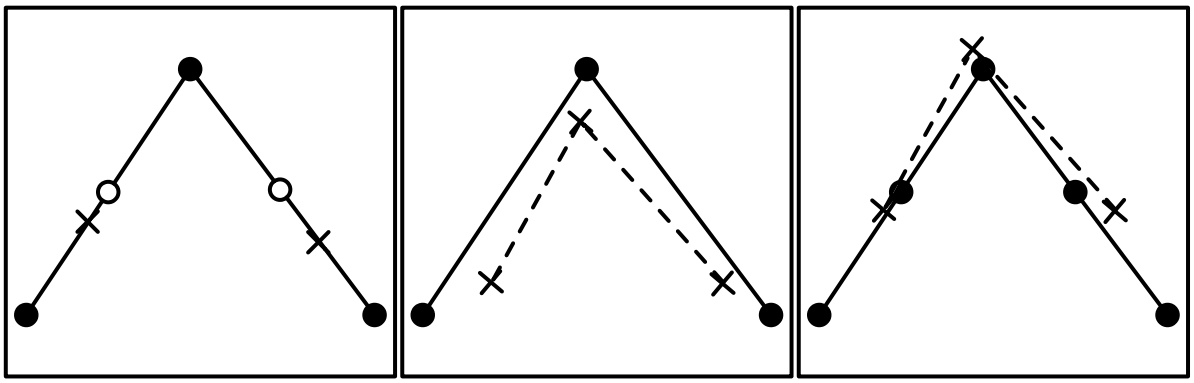
Figure 2.3: With normal shooting the surface normals are projected orthogonally onto the template set in order to establish correspondence.

2.5.3 Conclusion

We have compared two methods of correspondence in the previous sections. The one most useful to this study is the closest point correspondence algorithm. It can be applied to both surfaced and unsurfaced models. It can also be easily optimised by using specialised data structures as described in Appendix B [4]. For the purposes of aligning a mine surveying radar, closest point correspondence will be the best correspondence method, since the radar only exports unsurfaced sets.

2.6 Resampling of a meshed surface

The concept of resampling for alignment is discussed by Zhang [4]. His algorithm focusses on alignment of 3D curves, but is still very much applicable to the problem 3D model alignment. He suggests that when closest point correspondence is used, that resampling of the curves leads to better alignment accuracy. We show a graphical example taken from [4]. Figure 2.4 (a) shows two 2D lines. The black points are the vertices which connect the lines. These points will form the template set. If those lines are resampled, then they form two new points shown as white points circled in black. From the template set, we also sample the search set marked by crosses instead of circles. The aim is to align the template and the search set with the best possible accuracy. Figure 2.4 (b) shows the result of the alignment when just the original control points were availible for correspondence. Figure 2.4 (c) shows the result of the alignment when the newly sampled points were also used as possible correspondence points. The alignment is more accurate, matching the original more closely than the alignment shown in (b). We describe a resampling method for surfaced 3D



(a) (b) (c)

Figure 2.4: Figures taken from Zhang [4], which illustrate the improved matching through resampling. (a) The original sets when perfectly aligned. (b) Without resampled points, correspondence is only possible to the original control points. (c) The accuracy of the alignment is improved when there are points available for correspondence.

models in Chapter 3.

2.7 Repeated application of rigid body correction

Chen and Medioni [6] describe an algorithm which improves alignment through multiple repetitions of motion parameter estimation and transformation. Besl and McKay [7] describe the same algorithm, commonly referred to as iterative closest point (ICP) alignment. They also supply a proof that iterative alignment provides monotonically improving convergence of the spatial difference between sets. The process is repeated until the change in spatial difference falls below a certain threshold. Besl and McKay present the following algorithm:

1. Compute the closest point correspondence between the template and the search set.
2. Estimate the motion parameters between the search set and the newly defined subset from the template set which is the closest points.
3. Apply the transformation (constructed from the motion parameters) to the search set.
4. Terminate the alignment when the change in spatial difference falls below a certain threshold.
5. Repeat from step 1.

A drawback to iterative alignment is that the search set needs to be reasonably aligned with the template set prior to alignment for it to converge to the correct local minimum. During closest point correspondence, only the nearest points will be considered for alignment, and

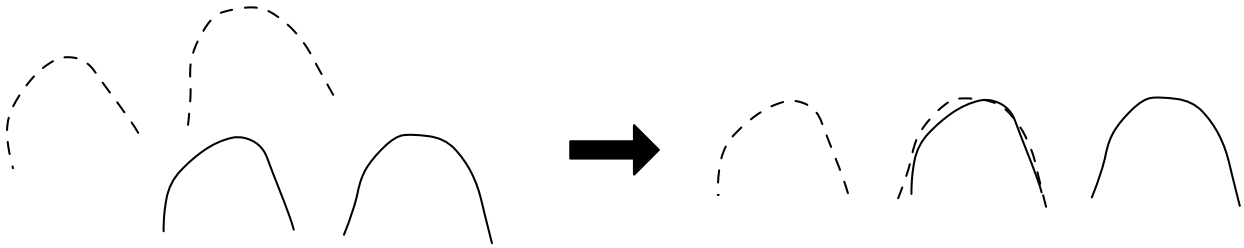


Figure 2.5: A graphic taken from Zhang [4]. Iterative correspondence alignment converges to the closest local minimum, which is not neccessarily the optimal one.

so convergence to a distant (but possibly correct) local minimum is very unlikely (Figure 2.5).

For this reason, prealignment of models is still an active research topic. The section to follow covers some of the attempts that have been made to the handle the problem of prealignment before iterative correspondence alignment.

2.8 Prealignment

Murino, Ronchetti, Castellani and Fusiello [16] introduced a preprocessing technique which provides coarse alignment prior to using the iterative correspondence alignment. This is also referred to as global alignment or prealignment. 3D skeletons are extracted from the data sets. The skeletons are then matched after which the alignment can be completed using iterative closest point alignment. They describe an extraction algorithm which extracts skeletons for tubular 3D models. They distinguish between pipes and joints, where a joint is an intersection point of many pipes. Their algorithm works well for tubular data sets but not for mostly flat surface sets such as radar synthetic maps, so it will not be considered. More detailed information on skeleton extraction in general can be found in [9].

Another prealignment method was presented by Lucchese, Doretto and Cortelazzo [17]. They introduced an original method to determine the motion parameters between sets using frequency domain analysis. The Fourier transform allows the decoupling of the estimate of the rotation parameters from the estimate of the translation parameters. The concept can be most easily explained using a 2D analogy taken from [8]. Consider an ideal windowed image of the famous Lenna picture (See figure 2.6 (a)). The same image rotated by 35 degrees can be seen in figure 2.6 (b).

(a) (b)

Figure 2.6: (a) An image windowed around the center of the image. (b) The same windowed image rotated around the centre by 35 degrees.

Let $\boldsymbol{l}_1(\boldsymbol{x})$ be the original image, and $\boldsymbol{l}_2(\boldsymbol{x})$ the rotated version of $\boldsymbol{l}_1(\boldsymbol{x})$ such that $\boldsymbol{l}_2(\boldsymbol{x}) = \boldsymbol{l}_1\left(\boldsymbol{R}(\theta)^{-1}\boldsymbol{x}\right)$, where $\boldsymbol{x} = (x, y)$ and $\boldsymbol{x} \in \mathbb{R}^2$. $\boldsymbol{l}_1(0,0)$ denotes the centre of the image. Let the two dimensional discrete Fourier transform be denoted as

$$\boldsymbol{L}_i(\boldsymbol{k}) = \boldsymbol{F}[\boldsymbol{l}_i(\boldsymbol{x})] = \iint_{-\infty}^{+\infty} \boldsymbol{l}_i(\boldsymbol{x})\, e^{-j2\pi\boldsymbol{k}^T\boldsymbol{x}} d\boldsymbol{x}.$$

The Fourier transform property of use here, is that the rotation in the spatial domain is also present in the frequency domain as

$$\boldsymbol{L}_2(\boldsymbol{k}) = \boldsymbol{L}_1\left(\boldsymbol{R}(\theta)^{-1}\boldsymbol{k}\right) e^{-j2\pi\boldsymbol{k}^T\boldsymbol{R}(\theta)\boldsymbol{T}},$$

where $\boldsymbol{T} = (x_t, y_t, z_t) \in \mathbb{R}^3$ is the translation parameters and

$$\boldsymbol{R}(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}.$$

As a result the spectrum has the following property

$$|\boldsymbol{L}_2(\boldsymbol{k})| = \left|\boldsymbol{L}_1\left(\boldsymbol{R}(\theta)^{-1}\boldsymbol{k}\right)\right|, \quad \text{and} \quad \angle\boldsymbol{L}_2(\boldsymbol{k}) = -2\pi\boldsymbol{k}^T\boldsymbol{R}(\theta)\boldsymbol{T}$$

which ensures that the rotation is decoupled from the translation movement. They define the Fourier transformed components in terms of a polar reference system as

$$\boldsymbol{M}_i(\rho, \varphi) = |\boldsymbol{L}_i(\rho\cos(\varphi), \rho\sin(\varphi))| \quad \text{for } i = 1, 2$$

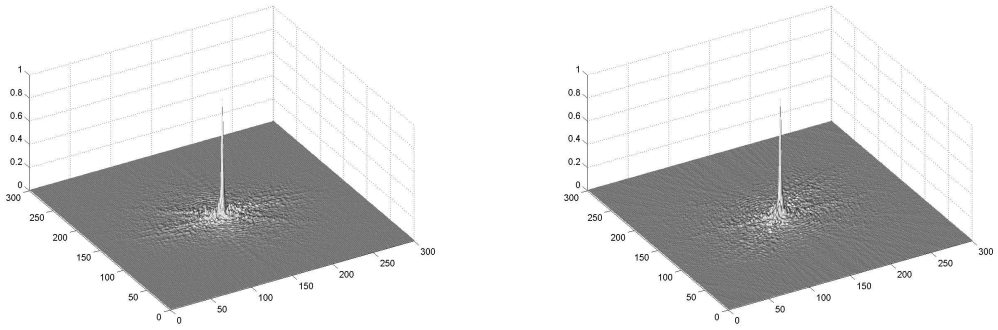
where

$$x = \rho\cos(\varphi) \quad \text{and} \quad y = \rho\sin(\varphi).$$

Presented in that form, the angle of rotation can be described in terms of spectrum functions as

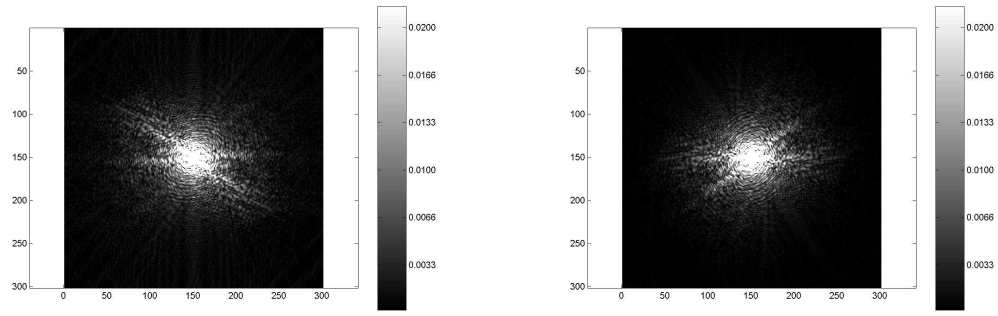
$$\boldsymbol{M}_1(\rho, \varphi) = \boldsymbol{M}_2(\rho, \varphi - \theta),$$

where θ is the difference planar rotation, and in this specific example, equal to 35 degrees. Displayed in figures 2.7 (a) and (b) are the three dimensional spectrum plots of the images of $\boldsymbol{l}_1(\boldsymbol{x})$ and $\boldsymbol{l}_2(\boldsymbol{x})$. Figures 2.8 (a) and (b) are the two dimensional spectrum plots of the same images.



(a) (b)

Figure 2.7: (a) The amplitude components of the first lenna image shown in three dimensions. (b) The amplitude components of the second rotated lenna image shown in three dimensions.



(a) (b)

Figure 2.8: (a) The amplitude components of the first lenna image shown in two dimensions. (b) The amplitude components of the second rotated lenna image shown in two dimensions.

Calculate the radial projections of functions $\boldsymbol{M}_1(\rho,\varphi)$ and $\boldsymbol{M}_2(\rho,\varphi)$ as

$$\boldsymbol{p}_i(\varphi) = \int_0^{\infty} \boldsymbol{M}_i(\rho,\varphi)\, d\rho \quad \text{where } i = \{1,2\}.$$

The rotation becomes evident when the radial projections of $\boldsymbol{p}_1(\varphi)$ and $\boldsymbol{p}_2(\varphi)$ are normalised and graphed. The maximum peaks of the two projections are seperated by exactly 35 degrees (Figure 2.9).

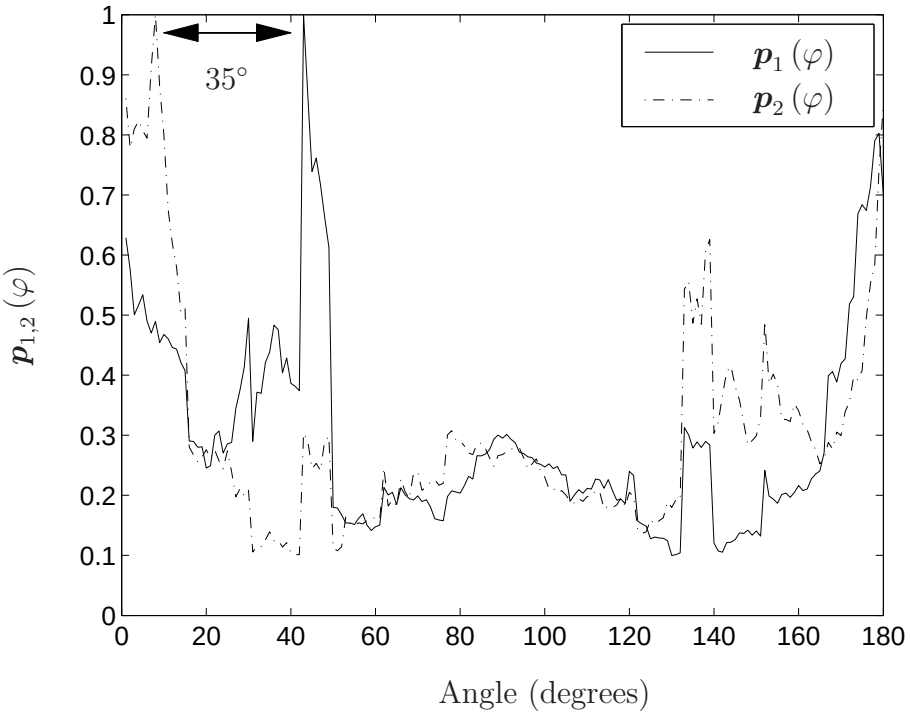


Figure 2.9: The maximum peaks of the radial projections are seperated by exactly 35 degrees of angular difference.

They extend the same concept into three dimensions. Models used in their study were surveyed using a range camera which captures data in the form of implicit surfaces. Let $\boldsymbol{q}_1(\boldsymbol{x})$ be the template model, and $\boldsymbol{q}_2(\boldsymbol{x})$ the rotated version of $\boldsymbol{q}_1(\boldsymbol{x})$ such that $\boldsymbol{q}_2(\boldsymbol{x}) = \boldsymbol{q}_1\left(\boldsymbol{R}(\theta)^{-1}\boldsymbol{x}\right)$, where $\boldsymbol{x} = (x,y,z)$ and $\boldsymbol{x} \in \mathbb{R}^3$. They denote the three dimensional discrete Fourier transform as

$$\boldsymbol{L}_i(\boldsymbol{k}) = \boldsymbol{F}[\boldsymbol{l}_i(\boldsymbol{x})] = \int_{\mathbb{R}^3} \boldsymbol{l}_i(\boldsymbol{x})\, e^{-j2\pi \boldsymbol{k}^T \boldsymbol{x}} d\boldsymbol{x},$$

where $\boldsymbol{l}_i(\boldsymbol{x})$ is the windowed equivalent of $\boldsymbol{q}_i(\boldsymbol{x})$. Similarly to the two dimensional method, the rotation and translation parameters are seperated using the following property of the

discrete Fourier transform

$$\boldsymbol{L}_2(\boldsymbol{k}) = \boldsymbol{L}_1\left(\boldsymbol{R}(\theta)^{-1}\boldsymbol{k}\right) e^{-j2\pi \boldsymbol{k}^T \boldsymbol{R}(\theta)\boldsymbol{T}},$$

where $\boldsymbol{k} = (k_x, k_y, k_z) \in \mathbb{R}^3$. Similar to the two dimensional example the rotation can be isolated using the following property

$$|\boldsymbol{L}_2(\boldsymbol{k})| = \left|\boldsymbol{L}_1\left(\boldsymbol{R}(\theta)^{-1}\boldsymbol{k}\right)\right|.$$

The next steps of the algorithm further calculates the axis of rotation and also the magnitude of rotation. The algorithm is completed when the translation is calculated on the basis that once the rotation is known, the following must hold true

$$\boldsymbol{l}_2(\boldsymbol{R}(\theta)\boldsymbol{x}) = \boldsymbol{l}_1\left(\boldsymbol{R}(\theta)^{-1}\boldsymbol{R}(\theta)\boldsymbol{x} - \boldsymbol{T}\right) = \boldsymbol{l}_1(\boldsymbol{x} - \boldsymbol{T}).$$

Overall there are limitations to this method. The alignment scheme was originally intended for the use of aligning data sets which were scanned using a range camera. A typical model would be scanned from various directions in order to cover the whole surface of the object. The partial views would be similar in size, where each view would have sufficient overlap. In our case the template model will usually cover a much larger area and the search could be a much smaller area compared to the template set, and so the area of overlap will be completely unknown. Also, the scheme is only applicable for data sets structured as voxels. The pre-processing required for converting a point cloud set or a surface model to a voxel set is resource-intensive.

Johnson and Hebert [10, 18, 19] first introduced a registration method using spin images. Spin images are invariant to rigid transformation and, as such, a very useful approach to model registration. A 3D model, presented in the form of a polygonal mesh, is transformed into a 2D image using a spin image function. This 2D representation is referred to as the spin image. Because the image encodes the coordinates of points on the surface of an object with respect to the local basis, it is a local description of the global shape of the object and is invariant to rigid transformations. Spin images are created using an oriented point, a three dimensional point with an associated direction. This is the center of the local basis. The oriented point is defined on a surface mesh point using its 3D position and the normal from its associated surface. Normals are calculated by fitting a plane through the points connecting triangulated facelets. The normals of the mesh should be orientated towards the outside prior to spin image creation. The images generated for two corresponding points, one on the search set and one on the template set, will be similar, and so oriented points can be matched based on a comparison of their associated images. The transformation of

an orientated point is done through a transformation function

$$\boldsymbol{S}(\boldsymbol{x}) \rightarrow (a,b) = \left(\sqrt{\|\boldsymbol{x}-\boldsymbol{p}\|^2 - (\boldsymbol{n}\cdot(\boldsymbol{x}-\boldsymbol{p}))^2},\ \boldsymbol{n}\cdot(\boldsymbol{x}-\boldsymbol{p}) \right).$$

Once all the points on the template set and the search set have been converted into spin images, the complexity of alignment is reduced to two dimensions. Their method is, however, limited to surfaced models only. It can also be computationally expensive for sets containing of a large number of points since a spin image needs to be generated for each point. Their algorithm requires that the triangle facelet resolution be controlled. The resolution of a triangle mesh determines the amount of surface detail the mesh contains and is closely related to the number of points, edges and faces in the mesh. Thirdly, when using data sets that represent a mostly flat surface – like in our case – the distiction between spin images might be too small to match successfully.

Silva et al. [20] presented an alignment method using genetic algorithms (GAs). GAs handle a set of solutions, where the set undergoes a process similar to natural evolution. They suggest registration of two images by finding the geometric transformation through a pose-space search, rather than the correspondence based search of ICP algorithms. The search space of geometric transformations contains solutions that can be used to align two views. GAs are computational models of natural evolution in which stronger individuals are more likely to be the winners in a competitive environment. The general principle underlying GAs is to maintain a population of possible solutions. Each possible solution is refered to as an individual. The population is subjected to an evolutionary procedure until some criteria can be satisfied, in our case the best possible transformation. Unlike correspondence alignment techniques, such as ICP, prealignment is not required for the GA to converge a global minimum. Their algorithm is, however, very computationally expensive. They state that a GA is a stochastic method and is generally time-consuming. One way to speed up computation is by using parallelisation, which in our case, is not an option, because our hardware platform does not allow for it easily.

Due to the limitations of the above mentioned methods, it was decided to design a prealignment algorithm from first principles. This prealignment algorithm will be sharply focussed on our specific alignment problem. The design is presented and described in the next chapter.

2.9 Reducing alignment processing time

When using an alignment algorithm making use of repeated application of rigid body correction (e.g. ICP), the majority of processing time is spent searching for correspondence

between sets. The traditional linear closest point (nearest neighbour) algorithm has a lookup time for one point, of $O(N)$, where N is the size of the template set. This lookup time can be reduced by making use of an abstract data type (ADT) to store the points. Correctly constructed, the ADT should split the search space, making lookup more efficient. Zhang [4] makes use of a k-dimensional binary search tree (sometimes abbreviated as kd-tree) data structure to optimise the lookup processing. He describes a recursive lookup function, but no details on how the tree should be constructed. We use kd-tree optimisation in our implementation. More detail on kd-tree optimisation can be found in Appendix B.

Besl and McKay [7] sugget that alignment time can be reduced by the linearisation and prediction of the motion parameters at each iteration of the ICP alignment. This approach, however, can lead to overshoot estimation of the motion parameters and only works well when the changes in motion are minor.

In order to reduce the time spent during correspondence processing, Masuda et al. [11] introduced random sampling of the points in the search set. The direct intention of this is more to counter the effect of outliers, but also has the added benefit of speeding up the correspondence process. We apply this technique to our aligment algorithm since it is easy to implement and maintain.

2.10 A special alignment case

Chetverikov and Svirko [12] describes an iterative alignment algorithm which handles the special case where the search set is not a complete subset of the template set. They describe an alignment algorithm for overlapping sets. The algorithm steps are as follows:

1. Calculate the correspondence between the points of the template and the search set based on a closest point metric. Also compute the squared distances between corresponding points, denoted as d_i^2.

2. Sort d_i^2 in ascending order, and select N of the least values.

3. Compute the optimal motion parameters for the N selected points.

4. Apply the transformation to the search set.

5. Go to step 1, or stop the process if any of the stopping conditions were satisfied.

The selection of the N overlapping correspondence points must be done very carefully as it will critically effect the accuracy of the final alignment. For our specific problem, the search set will always be a spatial subset of the template set and so this special case will not be considered.

2.11 Data sets used in the study

2.11.1 Digital terrain model

A digital elevation model (DEM) is a digital representation of ground surface topography or terrain and is also widely known as a digital terrain model (DTM). A DTM can be structured as a raster (a grid of squares) or as a triangular irregular network (TIN). The latter is a vector based representation made up of irregularly distributed nodes and lines with three dimensional coordinates that are arranged in a network of non-overlapping triangles. These triangles form a mesh structure representing a surface terrain or landscape (see Figure 2.10). The DTMs presented for this study are TIN structured and sourced either from terrestrial laser scanners or hand-surveyed data. All DTM points (or vertices) are in a left-handed Cartesian coordinate system and use metres as a unit of measurement. In the study the template model is refered to as the DTM. All DTMs are already in the common coordinate frame and as such remain stationary during alignment [1].

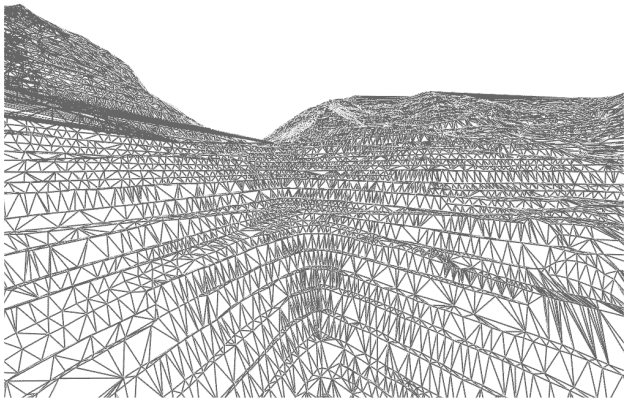


Figure 2.10: A typical triangulated irregular network (TIN) structured digital terrain model (DTM) representing an opencast mining pit wall rendered as a wire frame.

2.11.2 Synthetic map

The term synthetic map (SM) is used to describe the three-dimensional data sourced from the mobile mine surveying radar. Mine surveying radars are used to scan the sloped walls of opencast mining pits. The sloped wall of the pit is scanned by sweeping the antenna position in a regularly spaced angular grid covering the area of interest. The angular position of the radar antenna, combined with the propagated range data, forms a three-dimensional representation of the sloped pit wall (see Figure 2.11). SMs are typically less detailed than DTMs, often containing noisy points as well as outliers. SMs by default are in a local left-handed Cartesian coordinate system and so require alignment (or georeferencing) in order

to be useful. For the purposes of the study the SMs will be in either an arbitrary local coordinate frame or within the common coordinate frame. The SMs already placed within the common coordinate frame will also be used as a template models. However, when the SM model is still in an arbitrary frame, it will be used as the search model, requiring motion parameter adjustment to merge with the common mine coordinate system [1].

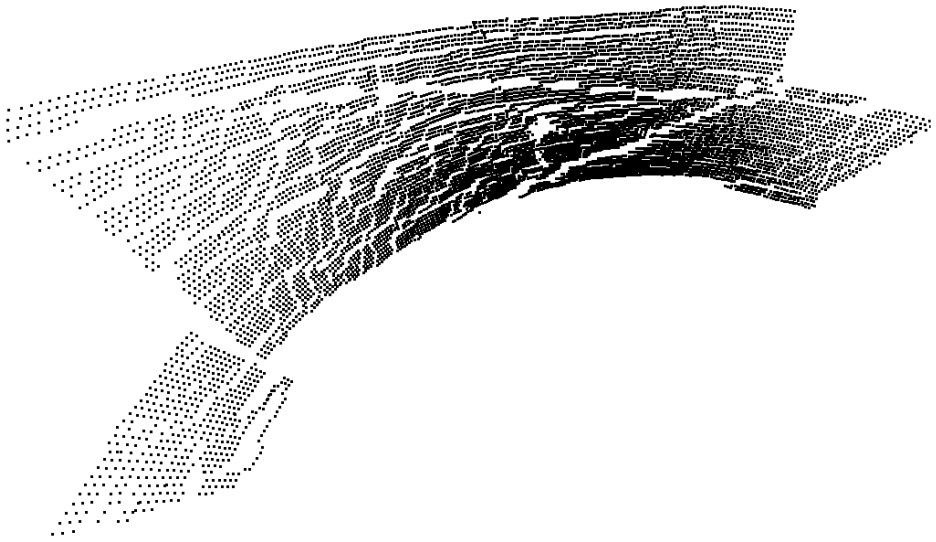


Figure 2.11: A rendered image of a synthetic map (SM) point cloud sourced from a mine surveying radar.

2.12 Conclusion

We have examined various studies pertaining to three-dimensional model alignment. We listed the algorithms which might be useful for our specific alignment application, some of which are covered in more detail in the sections to follow. We also listed the data set types applicable to the study. In the next chapter we consider the design of our georeferencing scheme.

Chapter 3

System design

3.1 Introduction

In the chapter to follow we design a georeferencing system based on three-dimensional model alignment. The methods and strategies described in Chapter 2 will form the basis of our design. The design can be simplistically described as a high level flow chart (Figure 3.1). The bulk of the chapter to follow, will invlove map conditioning, global alignment and then finally, iterative correspondence alignment. However, before any design is attempted, we first establish a baseline of the typical spatial differences achievable with the current georeferencing method. We also list below, the sections which are to follow.

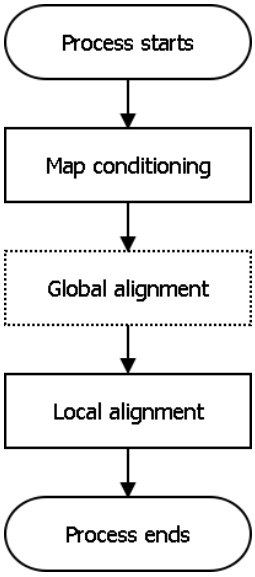


Figure 3.1: A high level flow diagram of steps needed for aligning two three-dimensional data sets.

Establishing a baseline of typical spatial differences

By replacing an existing mine surveying radar georeferencing method, we aim to achieve at least similar spatial differences currently obtainable. In the first section of our system design, we perform tests to establish a baseline of typical spatial differences expected for synthetic map (SM) on digital terrain model (DTM) alignment. The detail is discussed in section 3.2.

Rigid body motion estimation

From the baseline comparison we determined a set of typical spatial difference values. We also established that surface resampling improves the spatial differences measured between sets. In the next step of our design, we consider how to correct the spatial difference between two misaligned three-dimensional models. Such a correction involves the estimation of the rigid body parameters. We show the steps necessary for such an estimation. We also consider the limitations of this estimation. The detail is described in section 3.3.

Iterative correspondence alignment

The process of correspondence, estimation of rigid body parameters and alignment can be repeated in order to further reduce the spatial difference between misaligned models, as described in Chapter 2. Besl and McKay [7] proved this. We conduct a simulation to verify this. As part of the simulation the most commonly known limitation of the iterative closest point (ICP) becomes evident. Iterative correspondence alignment (ICA) does not always convergence to the correct local minimum. A prealignment algorithm is required to ensure converge to the correct local minimum. The detail is described in section 3.4.

Tessellation methods

Next in our design we consider map conditioning. As part of our design it might be helpful to transform discrete point clouds into surfaced models. We propose three Delaunay tessellation methods useful for the application of discrete SM tessellation. The detail is described in section 3.5.

Resampling algorithm

In section 3.2 we show that linear surface resampling improves closest point correspondence. We propose a resampling algorithm useful for our design. We show that by resampling surfaced maps we can control the number of points in a set and also ensure that points are evenly distributed geometrically. The detail is described in section 3.6.

A global alignment algorithm

In order to overcome the limitation of incorrect convergence using ICA, we propose a global alignment algorithm. The algorithm corrects for four degrees of freedom. Once the models have been globally aligned, iterative correspondence alignement can be applied in order to avoid false convergence. The detail is described in section 3.7.

A complete mine surveying georeferencing scheme

The design is completed using algorithms described so far. A complete georeferencing scheme is described using map resampling, global alignment and iterative correspondence alignment.

3.2 Establishing a baseline of typical spatial differences

3.2.1 Introduction

As stated before, mobile surveying radars currently make use of an existing method of georeferencing to place the radar within the mine coordinate frame. Once the radar is georeferenced, the 3D models generated by the radar will overlap the 3D models provided by the mine. Due to the limitations of the surveying radar, the alignment is not perfect. In this section we aim to quantify the average Euclidean error made when typical map pairs are compared spatially using alignment done through the existing georeferencing method. The statistics of such a comparison will provide a benchmark to which we can compare the results of the alignment using our new software based algorithm. Introduced in this section are the two three-dimensional data sets which are aligned using the existing georeferencing method. The three-dimensional models sourced from the radar will be refered to as the data or search sets, and the models already in the mine coordinate frame, as the template sets.

As mentioned in Chapter 2, 3D model alignment has long been a prominent field of study. Alignment of 3D models is also sometimes referred to as range image registration. The idea is simple: A template (base) model remains stationary while a search model – initially in an arbitrary coordinate frame – is maneuvered spatially to best fit the template model (Figure 3.2). When we have satisfactory alignment between the models, we consider the two models to be in the same coordinate frame. A metric for satisfactory alignment can differ depending on the cost function. Usually the models are considered aligned when the spatial difference is reduced to a minimum.

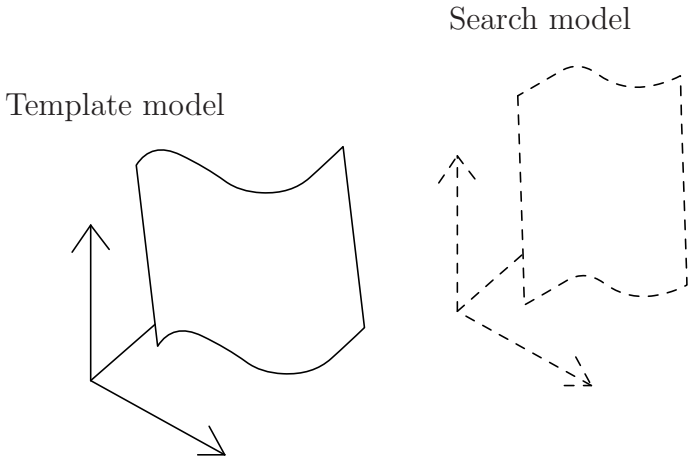


Figure 3.2: Two 3D models start off in their own coordinate frames. Alignment is done to merge the coordinate frames into a single common coordinate frame.

3.2.2 Notation

Here we describe the notation used throughout the study. A three-dimensional point (or vertex) structured as a horizontal vector is denoted as

$$\boldsymbol{p} = (x, y, z).$$

The search set is denoted as

$$\boldsymbol{P} = \{\boldsymbol{p}_i\}, \quad \forall\, i \in \{1, \ldots, N_P\}$$

where N_P is the number of points in the set. The template set is defined as

$$\boldsymbol{W} = \{\boldsymbol{w}_j\}, \quad \forall\, j \in \{1, \ldots, N_W\}.$$

A set of triangle points (or vertices) is defined as

$$\boldsymbol{T} = \{\boldsymbol{t}_q\}, \quad \forall\, q \in \{1, \ldots, N_T\}$$

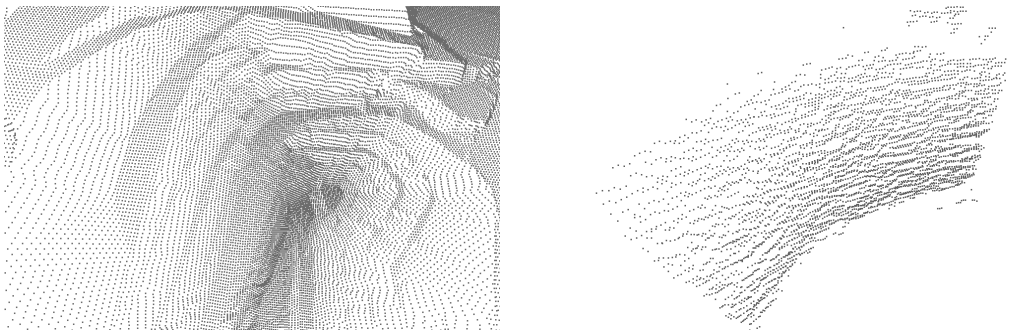
where each $\boldsymbol{t}$ is defined as

$$\boldsymbol{t} = \{(x_1, y_1, z_1), (x_2, y_2, z_2), (x_3, y_3, z_3)\}.$$

3.2.3 Prealigned comparison

The alignment of two geometric models involves the minimisation of their spatial difference. SMs in general are limited in geographical detail when compared to DTMs representing the

same geographical area. Usually a DTM's features – such as roads and sloped benches – appear more detailed and distinct when compared to an equivalent SM (Figure 3.3). In this section we determine the spatial differences for a few real-world data sets – when they are already aligned using the traditional total station method – in order to establish a baseline of achievable alignment quality.



(a) (b)

Figure 3.3: (a) The slope benches are clearly distinguishable in most DTMs due to a higher level of detail compared to most synthetic maps. (b) A typical synthetic map is far less detailed when compared to most DTMs.

The spatial difference comparison involves data from seven different geographical locations, each with a DTM and a SM covering the same area. The model pairs were sampled from the same opencast mine pit and share overlapping features such that the SM is a spatial subset of the DTM. All the pairs were aligned prior to the comparison using a traditional triangulation algorithm. The DTMs presented have varying degrees of TIN resolution (see Table 3.1). The spatial resolution is quantified as the average distance to the closest point for every point in the set. The number of points contained in the SMs are listed in Table 3.2. The spatial resolution is also indicated. In the two sections to follow we compare the models based on triangle corner point correspondence as well as correspondence with the DTM after surface resampling.

Table 3.1: The properties of the DTMs used in the spatial difference test.

DTM	No. of points	No. of triangles	Spatial coverage			Spatial resolution		Avg triangle area
			ΔX (km)	ΔY (km)	ΔZ (km)	Average (m)	Std. dev. (m)	(m^2)
A	93652	23413	1.06	0.39	0.11	3.28	1.80	16.61
B	310856	77714	0.77	1.25	0.36	2.15	2.79	16.67
C	189380	47345	1.49	0.73	0.24	3.39	2.41	27.34
D	470288	117572	2.91	3.50	0.51	3.43	3.37	67.13
E	824056	206014	1.49	2.76	0.36	3.49	2.06	19.32
F	771220	192805	0.75	1.04	0.20	0.42	0.36	0.23
G	114588	28647	1.39	1.56	0.42	5.87	4.16	72.99

Table 3.2: The properties of SMs used in the spatial difference test.

SM	No. of points	Spatial coverage			Spatial resolution	
		ΔX (km)	ΔY (km)	ΔZ (km)	Average (m)	Std. dev. (m)
A	9444	0.57	0.26	0.10	2.36	0.44
B	7000	0.34	0.65	0.17	3.83	0.44
C	10827	1.06	0.55	0.18	3.50	1.08
D	4354	0.39	0.69	0.22	5.46	0.79
E	8374	0.27	0.53	0.18	1.51	0.64
F	1098	0.69	0.50	0.13	4.41	2.65
G	16683	0.89	1.25	0.60	4.81	1.62

3.2.4 Triangle corner point correspondence test

The DTMs used in this study are TIN structured and form a mesh surface of non-overlapping triangles. We determine the spatial proximity of a Cartesian point to the model by calculating the Euclidean distance to the nearest point in the model. This nearest point will be a triangle corner point, since all the points form part of the triangle edges. For the tests the SM points are spatially compared with the DTM points in order to quantify the average Euclidean distance between them. For each SM point a corresponding closest point in the DTM is found, qualifying the correspondence method as a nearest neighbour point search. We present search point set

$$\boldsymbol{P} = \{\boldsymbol{p}_i\}, \quad \forall\, i \in \{1, \ldots, N_P\} \tag{3.1}$$

prealigned with model point set

$$\boldsymbol{W} = \{\boldsymbol{w}_j\}, \quad \forall\, j \in \{1, \ldots, N_W\} \tag{3.2}$$

where both sets are $\mathbb{R}^3$. $\boldsymbol{W}$ denotes the DTM and search set $\boldsymbol{P}$ the SM. The Euclidean distance $\mathbf{d}(\boldsymbol{r}_1, \boldsymbol{r}_2)$ between two general vertices $\boldsymbol{r}_1 = (x_1, y_1, z_1)$ and $\boldsymbol{r}_2 = (x_2, y_2, z_2)$ is

$$\boldsymbol{d}(\boldsymbol{r}_1, \boldsymbol{r}_2) = \|\boldsymbol{r}_1 - \boldsymbol{r}_2\| = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}. \tag{3.3}$$

The closest distance between the general point $\boldsymbol{p}$ and the point set $\boldsymbol{W}$ is

$$\mathbf{d}(\boldsymbol{p}, \boldsymbol{W}) = \min(\mathbf{d}(\boldsymbol{p}, \boldsymbol{w}_j))$$

Adapting this function, we derive a nearest neighbour root mean square (RMS) error function to quantify the error in model similarity and define it as

$$\mathbf{f}(\boldsymbol{P}, \boldsymbol{W}) = \sqrt{\frac{1}{N_p} \sum_{i=1}^{N_p} \mathbf{d}(\boldsymbol{p}_i, \boldsymbol{W})^2}. \tag{3.4}$$

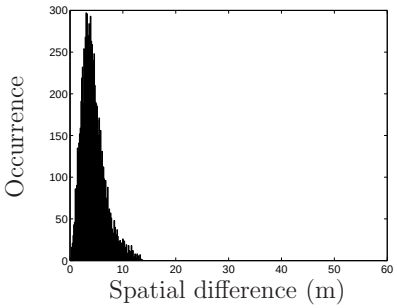
For the comparison, (3.4) is applied to all the search set pairs in Tables 3.1 and 3.2.

Results

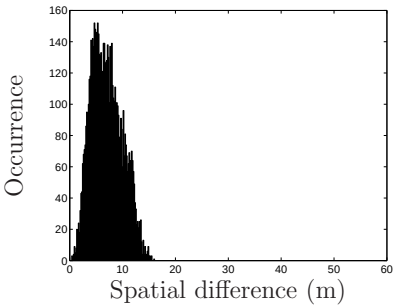
The results indicate (Table 3.3 and Figure 3.4) that the RMS spatial difference between the sets is in the order of metres, despite the perfect alignment of the models prior to comparison. The abnormally high maximum distance of test G is due to the search set not being a perfect spatial subset of the template set. The maximum distance recorded ranges over an order of magnitude. The results confirm that the SM data is limited in accuracy and only serves as an approximated model of the geographic landscape being sampled. The radar beam of the radar, unlike a laser, is not narrow and so sometimes covers multiple targets causing noisy measurements. Another factor in the model discrepancies is the size of the triangles making up the DTM meshes. Since correspondence is limited to association with corner points, we would hope that the mesh triangles are as small as possible, reducing the space unavailable for point correspondence between triangle points. In the section to follow we present a resampling scheme which will improve point correspondence by resampling new points in between triangle corner points.

Table 3.3: The results of the spatial difference comparison when comparing the SM point sets to the DTM point sets using triangle corner point correspondence.

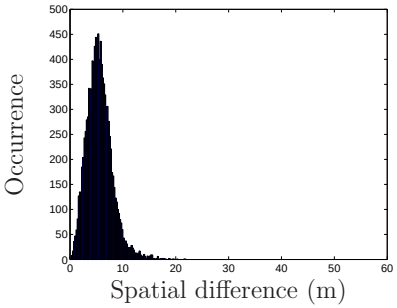
Comparison	Spatial difference		Min. distance	Max. distance
	RMS (m)	Std. deviation (m)	(m)	(m)
A	4.92	2.24	0.15	13.74
B	7.58	2.92	0.38	16.06
C	6.08	2.43	0.26	21.94
D	14.48	7.18	0.52	53.66
E	3.14	1.40	0.05	14.98
F	5.84	5.23	0.04	43.03
G	12.36	8.79	0.32	287.94



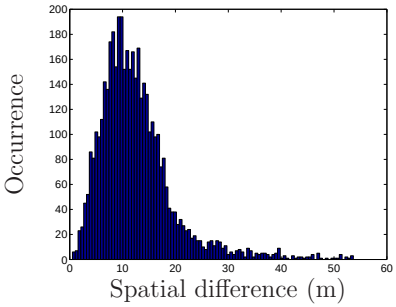
(a) Comparison A



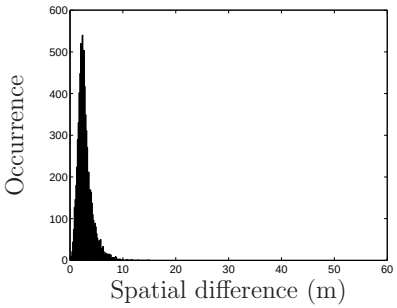
(b) Comparison B



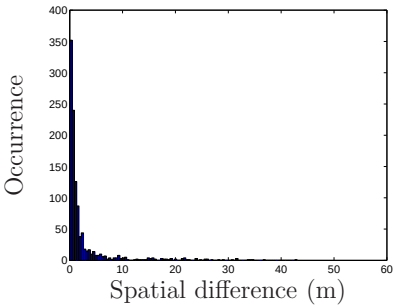
(c) Comparison C



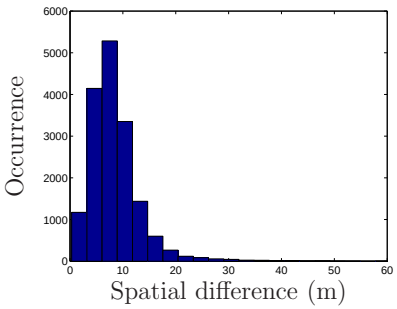
(d) Comparison D



(e) Comparison E



(f) Comparison F

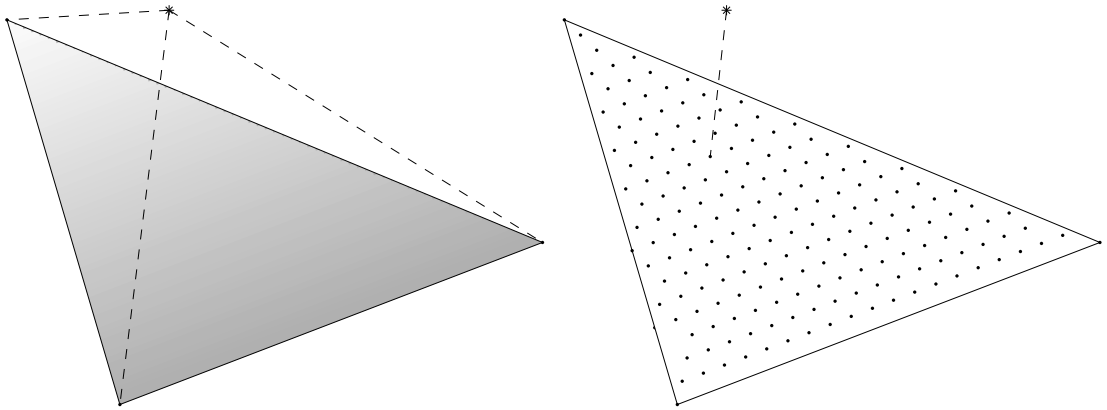


(g) Comparison G

Figure 3.4: The test histograms showing the occurrence of spatial difference between the DTM and SM sets using triangle corner point correspondence.

3.2.5 Projected resampling correspondence test

Depicted in Figure 3.5 (a) is an example of how an SM point is compared spatially with the corner points of a mesh triangle. An SM point may be close to the triangle surface but, because of corner point correspondence, the true spatial difference remains overestimated. We suggest resampling (or interpolating) the TIN mesh (see Figure 3.5 (b)), thereby enhancing the point dispersion, making the point set more representative of a piecewise linear surface.



(a) (b)

Figure 3.5: (a) A geometric comparison between a SM point and the corner points of a triangle face. (b) A geometric comparison between a SM point and a set of resampled points.

The point set making up the DTM is denoted as

$$\boldsymbol{W} = \{\boldsymbol{w}_j = (x_{w_j}, y_{w_j}, z_{w_j})\}, \quad \forall\, j \in \{1, \ldots, N_W\}. \tag{3.5}$$

The triangle mesh set is denoted as

$$\boldsymbol{T} = \{\boldsymbol{t}_q\}, \quad \forall\, q \in \{1, \ldots, N_T\} \tag{3.6}$$

such that $N_T = {}^{N_W}/_3$ and

$$\boldsymbol{t}_q = \{(\boldsymbol{w}_{3(q-1)+1}, \boldsymbol{w}_{3(q-1)+2}, \boldsymbol{w}_{3(q-1)+3})\}, \quad \forall\, q \in \{1, \ldots, N_T\}. \tag{3.7}$$

For the following test the DTMs are resampled by evaluating new points based on the mesh as a piecewise linear surface. Consider a point set $\boldsymbol{G} = \{\boldsymbol{g}_m\}$ in $\mathbb{R}^2$ where points form a two-dimensional equispaced grid when projected onto the horizontal plane. Each grid point

$$\boldsymbol{g}_m = (x_{g_m}, y_{g_m}), \quad \forall\, m \in \{1, \ldots, N_G\} \tag{3.8}$$

is evaluated on the condition that it should reside within the two dimensional bounds of the triangle mesh convex hull. Grid points residing within the triangle mesh will be included into the final resampled point set denoted as

$$\boldsymbol{R} = \{\boldsymbol{r}_k\}, \quad \forall\, k \in \{1, \ldots, N_R\}. \tag{3.9}$$

Grid points are projected onto the mesh at an angle perpendicular to the horizontal plane. The barycentric coordinates [21]

$$\begin{aligned} d_1 &= \begin{vmatrix} x_{g_m} & y_{g_m} & 1 \\ x_{3(q-1)+2} & y_{3(q-1)+2} & 1 \\ x_{3(q-1)+3} & y_{3(q-1)+3} & 1 \end{vmatrix}, \\ d_2 &= \begin{vmatrix} x_{3(q-1)+1} & y_{3(q-1)+1} & 1 \\ x_{g_m} & y_{g_m} & 1 \\ x_{3(q-1)+3} & y_{3(q-1)+3} & 1 \end{vmatrix} \quad \text{and} \\ d_3 &= \begin{vmatrix} x_{3(q-1)+1} & y_{3(q-1)+1} & 1 \\ x_{3(q-1)+2} & y_{3(q-1)+2} & 1 \\ x_{g_m} & y_{g_m} & 1 \end{vmatrix} \end{aligned} \tag{3.10}$$

are used to determine if the resampled points reside within the triangle mesh where $\forall\, m \in \{1, \ldots, N_G\}$ and $\forall\, q \in \{1, \ldots, N_T\}$. The barycentric coordinates are a set of triple numbers proportional to the signed distances of a point to the sides of the triangle. They are normalised so that they represent the areas of the subtriangles formed by the test point. The coordinates give the areas of the subtriangles normalised by the area of the original triangle. In the case where all three barycentric coordinates are negative (or all positive, depending on the corner point order) $\hat{g}_m \in \boldsymbol{R}$ where $\hat{g}_m = (x_{g_m}, y_{g_m}, z_{g_m})$ and

$$z_{g_m} = a_1 + a_2 x_{g_m} + a_3 y_{g_m} \tag{3.11}$$

where the planar parameters [21] are set as

$$\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 1 & x_{3(q-1)+1} & y_{3(q-1)+1} \\ 1 & x_{3(q-1)+2} & y_{3(q-1)+2} \\ 1 & x_{3(q-1)+3} & y_{3(q-1)+3} \end{bmatrix}^{-1} \begin{bmatrix} z_{3(q-1)+1} \\ z_{3(q-1)+2} \\ z_{3(q-2)+3} \end{bmatrix} \quad \forall\, q \in \{1, \ldots, N_T\}. \tag{3.12}$$

The average spatial difference between the SM and the resampled set is calculated as

$$\mathbf{f}(\boldsymbol{P}, \boldsymbol{R}) = \sqrt{\frac{1}{N_p} \sum_{i=1}^{N_p} \boldsymbol{d}(\boldsymbol{p}_i, \boldsymbol{R})^2}. \tag{3.13}$$

For the test the DTMs are resampled using various degrees of grid spacing. A finer grid spacing is conducive to an increase in the number of points making up the resampled point set, thereby increasing the number of possible correspondence points. Grid spacings of 10m, 6m, 3m, 2m and 1m are used for the comparison.

Results

The results indicate that a finer grid spacing is conducive to improved spatial similarity between the models (see Figure 3.6). The standard deviations remain mostly constant for the chosen grid spacings. The probability of matching geographical features common to both models is greatly improved by applying pre-processed resampling. Table 3.4 shows the improvement in spatial difference between the models when a 2m resample grid spacing is used. The values can be identified in Figure 3.6 as the spatial differences corresponding to a 2m grid spacing.

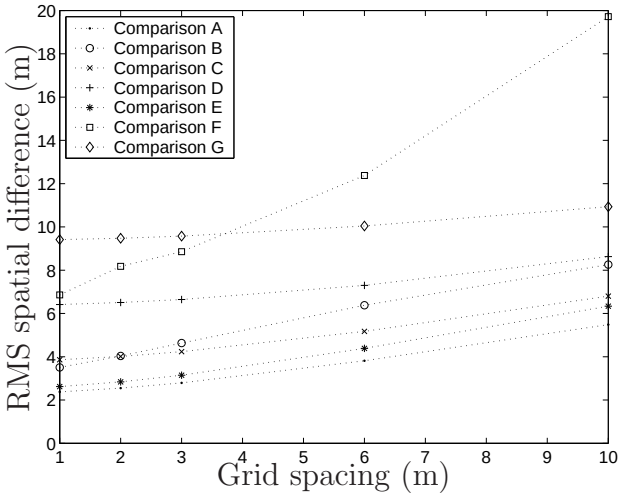


Figure 3.6: The RMS spatial difference graphed against grid spacings 10m, 6m, 3m, 2m and 1m taken from the resampled correspondence comparison.

Table 3.4: The improvement in RMS spatial difference when comparing the corner point correspondence tests to the resampled correspondence tests using a 2m grid spacing.

Test (#)	Less error (%)
A	48.275
B	46.761
C	33.973
D	55.094
E	9.5827
F	-40.102
G	23.327

3.2.6 Conclusion

Results indicate that spatial differences (as high as 10m) remain between the models even when they are perfectly aligned. This is due to the SMs being less detailed than the template DTMs. The noisy SM points also contribute to this discrepancy. Another factor is poor point correspondence because of triangle corner point association. This spatial discrepancy can be significantly reduced by resampling the DTMs prior to alignment, creating new points which can be used for point correspondence.

3.3 Rigid body motion estimation

3.3.1 Introduction

Non-scalable rigid bodies have six degrees of spatial freedom. That is to say, the freedom of independent translation on the frame axes as well as the freedom of independent rotation around those axes. Model alignment involves the estimation of the rigid body motion parameters which best reduce the spatial difference between two sets. In this section we describe how to estimate the motion parameters from two point sets of equal size.

3.3.2 The estimation algorithm

Let us consider rotation around the frame axes (Figure 3.7) namely the roll (X-axis) rotation θ, the pitch (Y-axis) rotation ϕ and the heading (Z-axis) rotation ψ. Euler's theorem states that any rotation can be given as a composition of rotations about three axes and thus can be presented by a 3×3 rotation matrix. We define the individual orthogonal rotation matrices as

$$\boldsymbol{R}_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix},$$

$$\boldsymbol{R}_y = \begin{bmatrix} \cos\phi & 0 & \sin\phi \\ 0 & 1 & 0 \\ -\sin\phi & 0 & \cos\phi \end{bmatrix} \quad \text{and}$$

$$\boldsymbol{R}_z = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

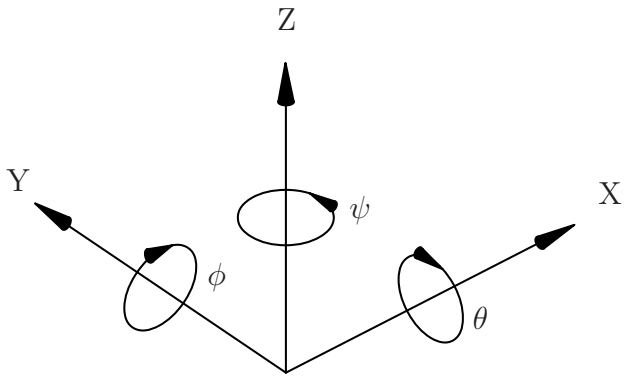


Figure 3.7: A right handed coordinate system.

Combined they become

$$\boldsymbol{R} = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix} = \boldsymbol{R}_x\boldsymbol{R}_y\boldsymbol{R}_z$$

$$= \begin{bmatrix} (\cos\phi\cos\psi) & (-\cos\phi\sin\psi) & (\sin\phi) \\ (\sin\theta\sin\phi\cos\psi + \cos\theta\sin\psi) & (-\sin\theta\sin\phi\sin\psi + \cos\theta\cos\psi) & (-\sin\theta\cos\phi) \\ (-\cos\theta\sin\phi\cos\psi + \sin\theta\sin\psi) & (\cos\theta\sin\phi\sin\psi + \sin\theta\cos\psi) & (\cos\theta\cos\phi) \end{bmatrix}$$

The order of rotation is important and to avoid confusion this study consistently makes use of the rotation order described above, namely rotation around X-axis, then rotation around Y-axis and finally rotation around Z-axis. Decomposition of the Euler angles from the composite rotation matrix can then be derived as

$$\theta = \arctan2(-R_{23}, R_{33}), \quad \phi = \arcsin(R_{13}) \quad \text{and} \quad \psi = \arctan2(-R_{12}, R_{11}).$$

Besl and McKay [7] describe the same composite rotation matrix as $\boldsymbol{R}$, but as a function of quaternion (as described in Chapter 2) values

$$\hat{\boldsymbol{R}}(\boldsymbol{q}) = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1 q_2 - q_0 q_3) & 2(q_1 q_3 + q_0 q_2) \\ 2(q_1 q_2 + q_0 q_3) & q_0^2 + q_2^2 - q_1^2 - q_3^2 & 2(q_2 q_3 - q_0 q_1) \\ 2(q_1 q_3 - q_0 q_2) & 2(q_2 q_3 + q_0 q_1) & q_0^2 + q_3^2 - q_1^2 - q_2^2 \end{bmatrix}. \tag{3.14}$$

Besl and McKay show in [7] the following set of steps up to (3.19) the extraction of the motion parameters from paired (corresponding) point sets: Let $\boldsymbol{P} = \{\boldsymbol{p}_i\}$ be a point set to be aligned with template point set $\boldsymbol{X} = \{\boldsymbol{x}_i\}$, where $N_x = N_p$ and where each point $\boldsymbol{p}_i$ corresponds to the point $\boldsymbol{x}_i$ with the same index. Both sets are row matrices in $\mathbb{R}^3$ where the centres are given by

$$\boldsymbol{\mu}_p = \frac{1}{N_p} \sum_{i=1}^{N_p} \boldsymbol{p}_i \quad \text{and} \quad \boldsymbol{\mu}_x = \frac{1}{N_x} \sum_{i=1}^{N_x} \boldsymbol{x}_i,$$

calculated as the mean values. The mean is subtracted from the point set in order to align the centre of mass of the two sets. The cross-covariance matrix of sets $\boldsymbol{P}$ and $\boldsymbol{X}$ is given by

$$\boldsymbol{\Sigma}_{px} = \frac{1}{N_p} \sum_{i=1}^{N_p} \left[\left(\boldsymbol{p}_i - \boldsymbol{\mu}_p\right)^T \left(\boldsymbol{x}_i - \boldsymbol{\mu}_x\right) \right] \tag{3.15}$$

with dimension 3×3. From (3.15) construct anti-symmetric 3×3 matrix

$$\boldsymbol{A} = \boldsymbol{\Sigma}_{px} - \boldsymbol{\Sigma}_{px}^T. \tag{3.16}$$

From $\boldsymbol{A}$ construct column vector

$$\boldsymbol{\Delta} = \begin{bmatrix} A_{23} & A_{31} & A_{12} \end{bmatrix}^T. \tag{3.17}$$

Construct symmetric 4×4 matrix

$$\boldsymbol{Q}(\boldsymbol{\Sigma}_{px}) = \begin{bmatrix} \operatorname{tr}(\boldsymbol{\Sigma}_{px}) & \boldsymbol{\Delta}^T \\ \boldsymbol{\Delta} & \boldsymbol{\Sigma}_{px} + \boldsymbol{\Sigma}_{px}^T - \operatorname{tr}(\boldsymbol{\Sigma}_{px}) \boldsymbol{I}_3 \end{bmatrix} \tag{3.18}$$

where $\boldsymbol{I}_3$ represents a 3×3 identity matrix and $\operatorname{tr}(\boldsymbol{\Sigma}_{px})$ represents the trace of matrix $\boldsymbol{\Sigma}_{px}$. The rotation parameters required for alignment reside within the column space of (3.18). It is shown by Horn [14] that the column space – calculated using the principal eignenvector –

results in a least squares solution of the overdetermined system using evalution cost function $\mathbf{\Gamma}^2 = \sum_{i=1}^{N} \left\| \boldsymbol{x}_i^{'} - (\boldsymbol{R}\boldsymbol{p}_i + \boldsymbol{T}) \right\|$. The unit eigenvector $\boldsymbol{q}_R = \begin{bmatrix} q_0 & q_1 & q_2 & q_3 \end{bmatrix}^T$, corresponding to the principal eigenvalue of matrix $\boldsymbol{Q}(\boldsymbol{\Sigma}_{px})$, is selected for optimal estimation of the rotation parameters. Singular value decomposition (SVD) is ideal to decompose $\boldsymbol{Q}(\boldsymbol{\Sigma}_{px})$ into $\boldsymbol{Q}(\boldsymbol{\Sigma}_{px}) = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T$. See Appendix A for details on how to estimate the principal eigenvector using an easily implementable Gram–Schmidt Jacobi method. Construct unit eigenvector $\boldsymbol{q}_R = \begin{bmatrix} U_{11} & U_{21} & U_{31} & U_{41} \end{bmatrix}^T$ which corresponds to the first column vector of matrix $\boldsymbol{U}$. The optimal translation column vector is given by

$$\boldsymbol{q}_T = \boldsymbol{\mu}_x^T - \hat{\boldsymbol{R}}(\boldsymbol{q}_R)\boldsymbol{\mu}_p^T, \tag{3.19}$$

where function $\boldsymbol{R}(\boldsymbol{q}_R)$ refers to (3.14). Finally, alignment of set $\boldsymbol{P}$ with set $\boldsymbol{X}$ is done as

$$\boldsymbol{P}^{'} = \left\{ \boldsymbol{p}_i \hat{\boldsymbol{R}}(\boldsymbol{q}_R)^T + \boldsymbol{q}_T \right\}, \quad \forall\, i \in \{1, \ldots, N_P\}.$$

3.3.3 Realignment test

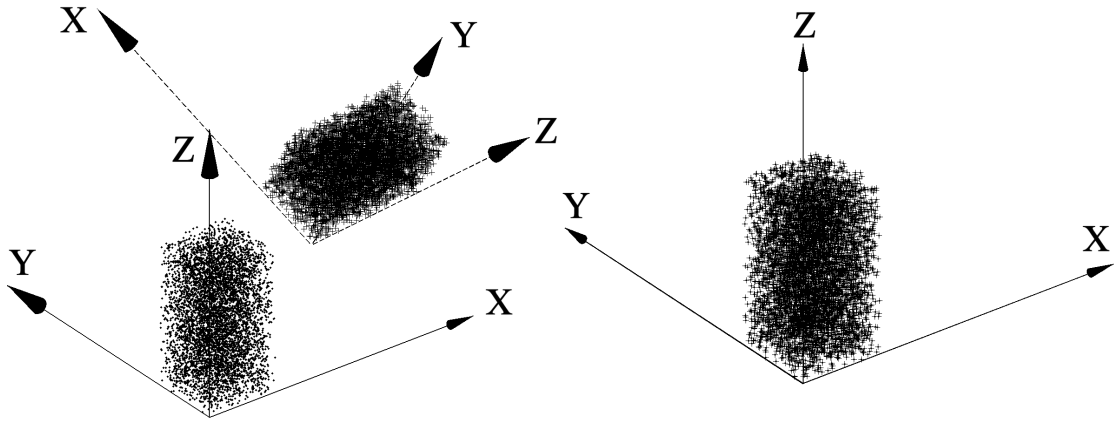
The algorithm's capability to estimate motion parameters is limited to applications where the two sets are of equal size, the points have known correspondence and where the noise discrepancy between sets is within tolerance [5]. We consider the effect of non-compliance to the suggested use of the algorithm by three tests. We consider three possible scenarios: Ideal correspondence between sets, ideal correspondence with noise added to one of the sets and then non-correspondence between sets. The test involves two point clouds shaped as cuboids. Both sets contain 6000 randomly sampled points and have spatial dimensions of 1 $\times$ 1 $\times$ 2. Both sets are identical except for differing in spatial orientation and translation (Figure 3.8 (a)). For all three scenarios we estimate the motion parameters and realign the models.

Perfect correspondence alignment test

For the first test the sets have perfect correspondence. The alignment resulted in perfect realignment (Figure 3.8 (b)). The alignment is so effective that the template model is not even visible on the figure because of the overlap. After alignment the residuals are essentially zero. The following residuals remain, as seen in Table 3.5.

Table 3.5: The motion parameter errors which remain after alignment.

Angle of rotation error			Translation error		
θ (deg)	ϕ (deg)	ψ (deg)	X	Y	Z
-7.10e-14°	-4.26e-14°	5.68e-14°	-1.39e-14	1.66e-16	-5.99e-15



(a) (b)

Figure 3.8: (a) Two cuboid data sets prior to alignment, rendered with coordinate frame axes. (b) The same two cuboid data sets after alignment.

Perfect correspondence with noise influence alignment test

For the second test we add normally distributed noise to point set $\boldsymbol{P}$ to test the robustness of the algorithm against zero average noise. The test is repeated five times, each time with a different noise level. Noise with standard deviations of 0.1, 0.25, 0.5, 1 and 2 was used. Figure 3.9 shows the alignment of the sets when noise with standard deviation 0.5 is used.

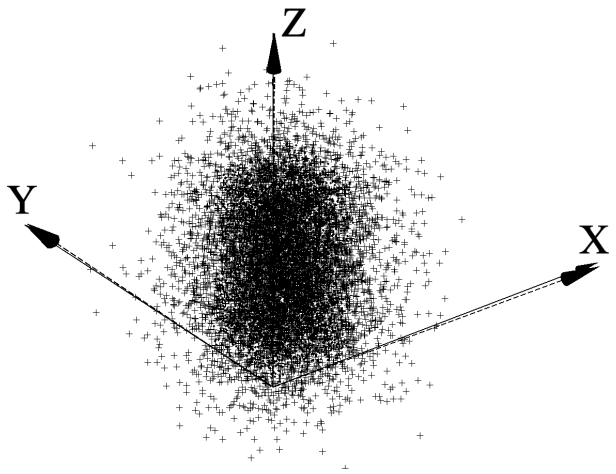


Figure 3.9: The alignment of the set with added noise.

The influence of noise on the alignment becomes more apparent as we increase the spread of the probability density function (Table 3.6).

Table 3.6: The motion parameter error for different noisy sets after alignment.

Noise std. dev.	Angle of rotation error			Translation error		
	θ (deg)	ϕ (deg)	ψ (deg)	X	Y	Z
0.1	0.160°	0.034°	0.018°	0.0009	0.0031	0.0002
0.25	0.636°	0.234°	0.994°	0.0061	0.0036	0.0003
0.5	1.984°	0.597°	2.156°	0.0013	0.0123	0.0333
1	4.489°	0.601°	4.363°	0.0011	0.0300	0.0131
2	18.982°	3.216°	20.632°	0.0233	0.0892	0.0597

From the results we see that noise is a factor in the accuracy of estimation of the motion parameters. Noise factors up to 0.5 produce heading errors of 2 degrees or less, which is within the specified heading error limit stipulated in Chapter 1. This shows us that the estimation algorithm is fairly robust, even when the sets are noisy.

Non-correspondence alignment test

For the last test we consider the scenario where the correspondence between the sets is not known. This is an important test because correspondence between point sets is very rarely known prior to alignment. For the test we consider the scenario where the correspondence is randomly sampled. Figure 3.10 shows the sets after alignment. Visually the quality of alignment is much worse when compared to Figure 3.9. Comparing the residuals of this test

to the residuals from the worst noise influenced test we see that non-correspondence has more influence on the quality of alignment than noise.

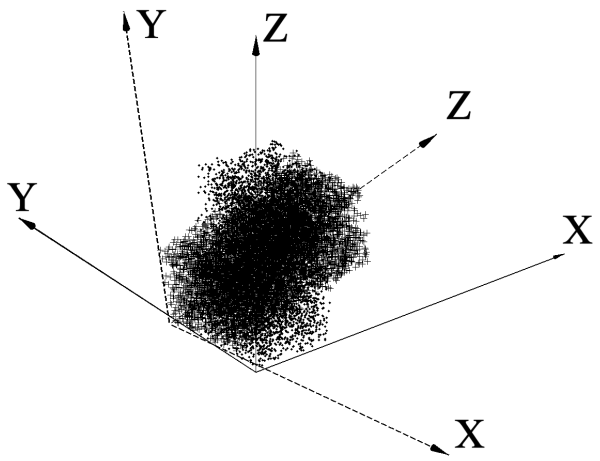


Figure 3.10: Incorrect alignment because of non-correspondence between points.

3.3.4 Conclusion

The results of the tests indicate that estimating the rigid body motion parameters between two misaligned point sets is easy, as well as accurate, provided the correspondence is known. Even with noisy sets the accuracy of the estimation is still fairly good. However, when the sets share no correspondence, the estimation becomes inaccurate to the point of being unusable. A correspondence method is needed for our design. We use the closest corresponding correspondence, as described in section 2.5, because it is the most suited for our application.

3.4 Iterative correspondence alignment

3.4.1 Introduction

Besl, McKay and others (see e.g. [7], [6], [5]) showed that the spatial difference between two three-dimensional models can be reduced by estimating the motion parameters between them. They also show that the spatial difference can be reduced even further if the process is repeated (Figure 3.11). In the section to follow we describe iterative correspondence alignment.

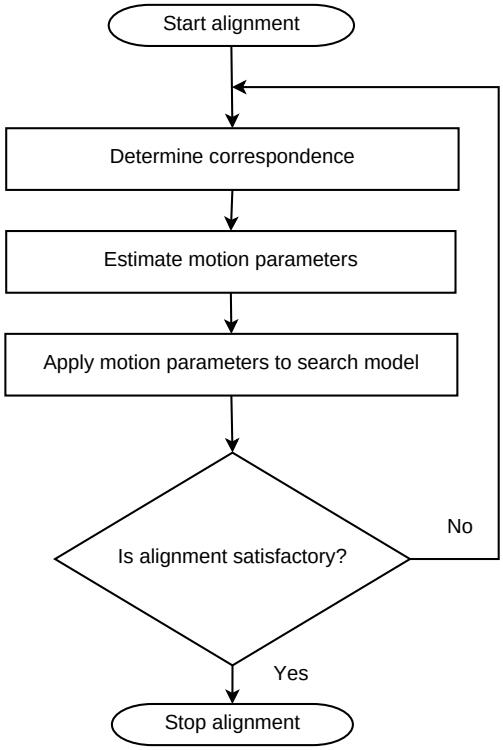


Figure 3.11: The process of motion parameter estimation and application can be repeated to even further reduce the spatial difference between models.

3.4.2 The effect of iterative alignment

Consider two corresponding point sets $\boldsymbol{P} = \{\boldsymbol{p}_i\}$ and $\boldsymbol{Y} = \{\boldsymbol{y}_i\}$ where $\forall\, i \in \{1, \ldots, N_P\}$ and $N_P = N_Y$. Correspondence, as mentioned before, refers to how the sets are of equal size and sequenced in order. To align the sets we aim to reduce the average spatial difference

$$e = \frac{1}{N_P} \sum_{i=1}^{N_P} \|\boldsymbol{y}_i - \boldsymbol{p}_i\|^2 \tag{3.20}$$

to a minimum. We estimate the rigid body motion parameters of the sets using the least-squares algorithm from section 3.3. By applying these motion parameters to set $\boldsymbol{P}$ the average spatial difference after alignment is now

$$d = \frac{1}{N_P} \sum_{i=1}^{N_P} \|\boldsymbol{y}_i - \boldsymbol{R}(\boldsymbol{p}_i) - \boldsymbol{t}\|^2 \tag{3.21}$$

where function $\boldsymbol{R}$ is the rotation and $\boldsymbol{t}$ the translation. Since the motion parameters are based on the least squares fit of the sets it can be said that $d \leq e$. This is a statement supported by Besl and McKay [7]. They go even further by saying that, by iteratively repeating the correspondence and alignment process the average spatial difference will converge mono-

tonically. Consider the alignment of two point sets, $\boldsymbol{X}$ and $\boldsymbol{P}$. They are not corresponding sets and as such $N_X \neq N_P$. $\boldsymbol{P}$ is a spatial subset of $\boldsymbol{X}$. $\boldsymbol{X}$ will remain stationary and $\boldsymbol{P}$ will be rotated and translated to best align with $\boldsymbol{X}$. Using the closest point algorithm presented in section 2.5, the goal is to determine the new correspondence set $\boldsymbol{Y}_1$ such that $N_{Y_1} = N_P$ and $\boldsymbol{Y}_1 \subset \boldsymbol{X}$. The average spatial difference prior to alignment is

$$e_1 = \frac{1}{N_P} \sum_{i=1}^{N_P} \|(\boldsymbol{y}_i)_1 - \boldsymbol{p}_i\|^2 . \tag{3.22}$$

If we repeat the process of correspondence and alignment k times, the average spatial differences become

$$e_k = \frac{1}{N_P} \sum_{i=1}^{N_P} \|(\boldsymbol{y}_i)_k - (\boldsymbol{p}_i)_k\|^2 \quad \text{and} \quad d_k = \frac{1}{N_P} \sum_{i=1}^{N_P} \|(\boldsymbol{y}_i)_k - (\boldsymbol{p}_i)_{k+1}\|^2 . \tag{3.23}$$

where $\boldsymbol{P}_{k+1} = \boldsymbol{R}_k(\boldsymbol{P}_k) + \boldsymbol{T}_k$ is the rotation and translation of the current iteration (indicated by integer value k). Besl and McKay [7] further prove that

$$0 \leq d_{k+1} \leq e_{k+1} \leq d_k \leq e_k \text{ for all } k. \tag{3.24}$$

We verify the proof using a simulation. Figure 3.12 depicts a three-dimensional view of two point sets prior to alignment. The sets were misaligned by a rotation of 11 and 5 degrees around two of the axes. Iteratively applying correspondence and alignment results in an ideal fit since the smaller set was originally sampled from the template set. Figure 3.13 depicts the average spatial difference e_k during each iteration of the alignment. The spatial error decreases monotonically until convergence, as proven by Besl and McKay analytically [7].

Figure 3.12: A three-dimensional view of two point sets prior to alignment. The larger template set remains stationary whilst the smaller set (indicated by crosses) is rotated and translated to best align with the template set.

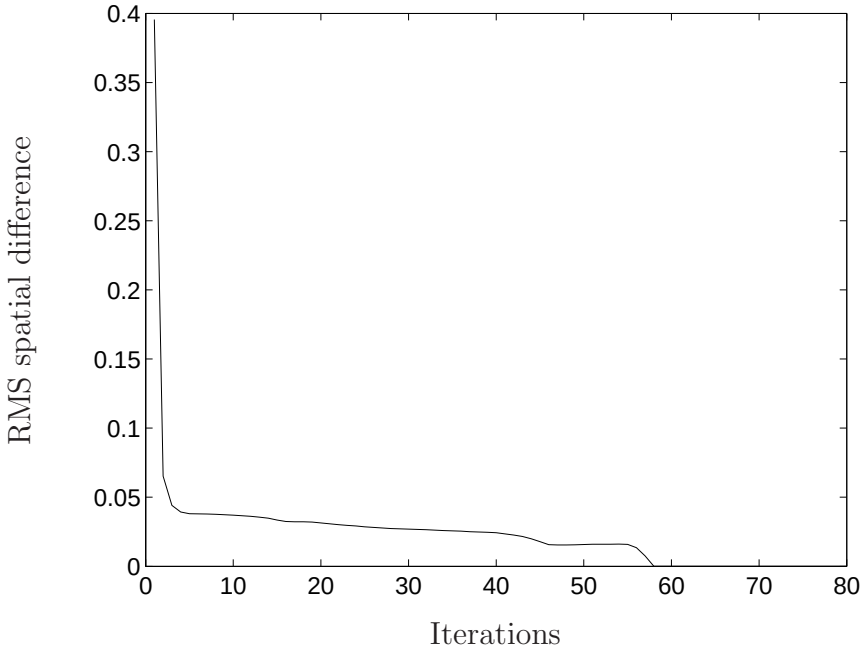


Figure 3.13: A graph of the RMS spatial difference during each iteration of the alignment.

The limitations of iterative alignment

Iterative alignment ensures convergence to a localised position. Figure 3.14 shows such an example of correct localised alignment. Correct convergence is, however, dependent on the initial alignment parameters. Consider the same two point sets used previously. For the next test the sets start off misaligned by rotation angles of 18 and 15 degrees around two of the axes. The error in rotation prior to alignment is 7 and 10 degrees higher compared to the previous test. Alignment of the sets fails and the search set converges to an incorrect localised position (Figure 3.15). Despite the misalignment, the average spatial difference still converges monotonically (Figure 3.16).

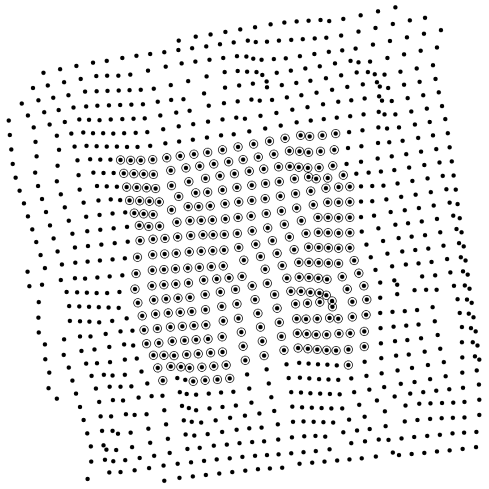


Figure 3.14: The data sets from Figure 3.12 after correct alignment.

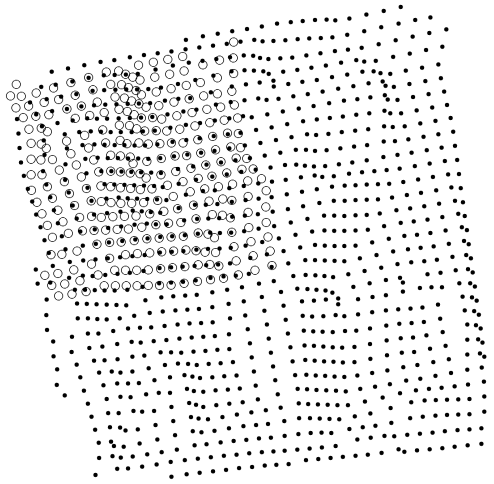


Figure 3.15: The data sets from Figure 3.12 after incorrect alignment.

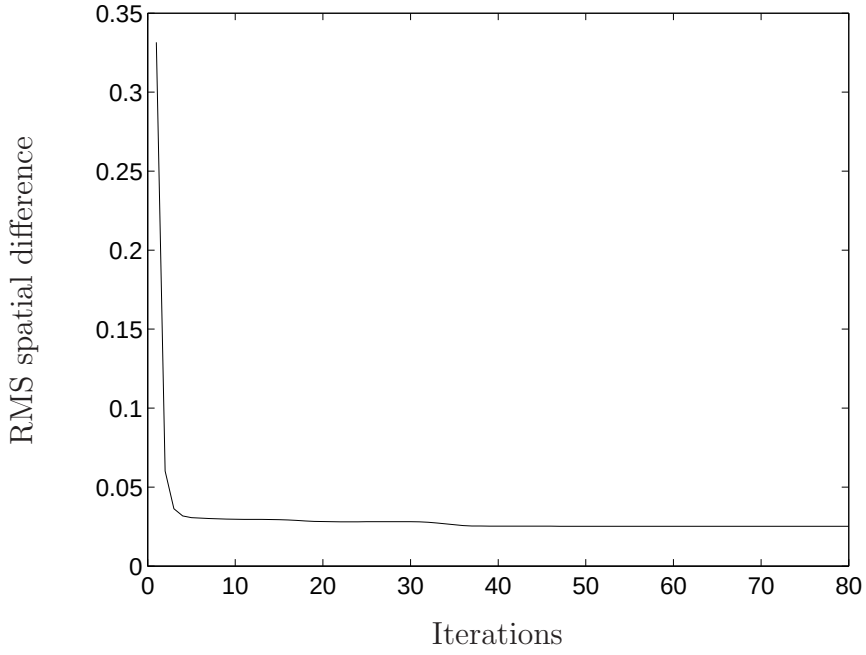


Figure 3.16: A graph depicting the RMS spatial difference for the incorrect alignment.

From this we see that iterative correspondence alignment (ICA) is limited to when models are reasonably aligned prior to ICA. The alignment is considered unsuccessful if the average spatial difference after a number of iterations still remain greater than a predefined threshold.

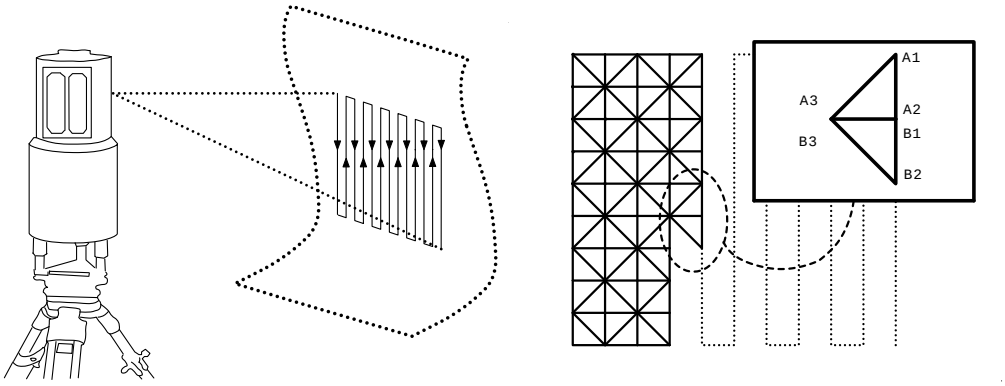
3.4.3 Conclusion

Iterative alignment is a very commonly used method of model alignment [22]. Using a simulation we have confirmed Besl and McKay's statement that monotonic convergence is guaranteed for alignment to a local optimum. Correct alignment is, however, very sensitive to the initial alignment parameters.

3.5 Synthetic map tessellation methods

3.5.1 Introduction

Survey imaging hardware such as terrestrial lasers and mine surveying radars, scan points according to a fixed angular grid which makes it possible to mesh points in real-time (Figure 3.17 (a) and (b)). Facelets are constructed by connecting neighbouring points arranged in a deterministic scan line pattern. Having the points meshed is convenient but does require more overhead, and so users usually have the option to export scan data either in a meshed or unmeshed format. As part of this study we concern ourselves with sets which could be meshed or unmeshed. In the sections to follow we describe computationally inexpensive meshing methods useful for when point sets are unmeshed or when the points are in random order.



(a) (b)

Figure 3.17: (a) The terrestrial laser scans points according to fixed angular scan lines, making meshing easier to manage. (b) An example of how meshing is done when points are arranged according to a fixed angular grid.

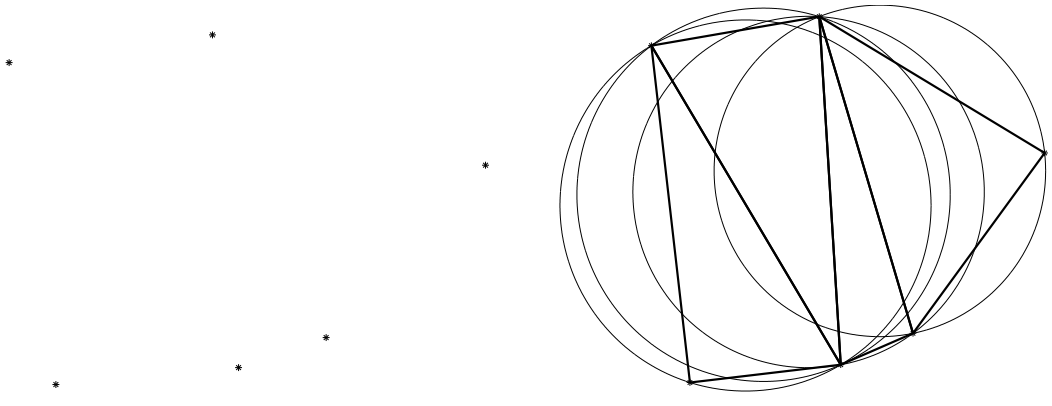
3.5.2 Triangulated irregular network meshing

Meshing involves connecting existing data points to form non-overlapping polygonal facelets which best model a piecewise linear surface. Meshing can technically be done using any

polygonal shape, but the triangle form is the most practical and the most widely used (e.g. [21], [23]). The data structures required to store triangle faces are easier to implement compared to the structures required for other polygonal facelets. Triangulated meshes are also referred to as triangulated irregular networks (TINs).

A very popular and efficient method of TIN generation is with Delaunay triangulation (e.g. [21], [24]). The Delaunay algorithm connects the points of the sets to form non-overlapping triangles in such a way that all triangles comply with the Delaunay constraint. For two-dimensional space the Delaunay constraint states that the circumcircle of a model triangle may not contain other points. By complying with the Delaunay constraint, thin and small angled triangles are avoided, producing a very well distributed TIN. The Delaunay constraint was originally defined for two-dimensional space but easily extends to work in n-dimensional spaces, or in our case three-dimensional space.

The three-dimensional Delaunay constraint states that no points are allowed to be within the circumsphere of a triangle surface piece connecting three points. The sets relevant to this study, however, model thin geographical surfaces and render as piecewise linear surfaces instead of closed primitive shapes, and therefore we apply two-dimensional Delaunay triangulation instead of three-dimensional Delaunay triangulation. Figure 3.18 (a) and (b) depicts a typical TIN generation example using a two dimensional Delaunay triangulation. In the section to follow we propose three different TIN generation methods using two-dimensional Delaunay triangulation. There are many Delaunay triangulation algorithms available. It is shown that the divide and conquer [25] or the sweep line [26] algorithm are very efficient, both having $O(n \log n)$ computational time [27]. The sweep line algorithm is, however, marginally slower than the divide and conquer algorithm.



(a) (b)

Figure 3.18: (a) A set of two-dimensional points randomly dispersed on the horizontal plane. (b) A triangulated network generated using the Delaunay triangulation algorithm on the same set of points. The Delaunay constraint ensures that thin and small angled triangles are avoided.

3.5.3 Efficient meshing methods using two-dimensional Delaunay triangulation

The three proposed meshing methods are described in the sections to follow. They will be compared through a test, where the three methods are applied on the same data set. The Delaunay triangulation method for which the minimum triangle angle is maximised the most, and the maximum angle is minimised the lowest, will be considered the most optimal [21].

Horizontal plane projection

Typically, both the terrestrial laser and the slope stability radar will be levelled prior to being operated. Also, most open pit mines are sloped so as to avoid slope failures, and therefore, are very seldom completely vertical. Given this, the template sets will generally have more spatial variation in the horizontal plane than in the vertical plane. Therefore, a way of triangulation is to ignore the height component of the set, effectively projecting the points onto the XY plane where two-dimensional triangulation can be applied (Figure 3.19 (a)). We propose the following method:

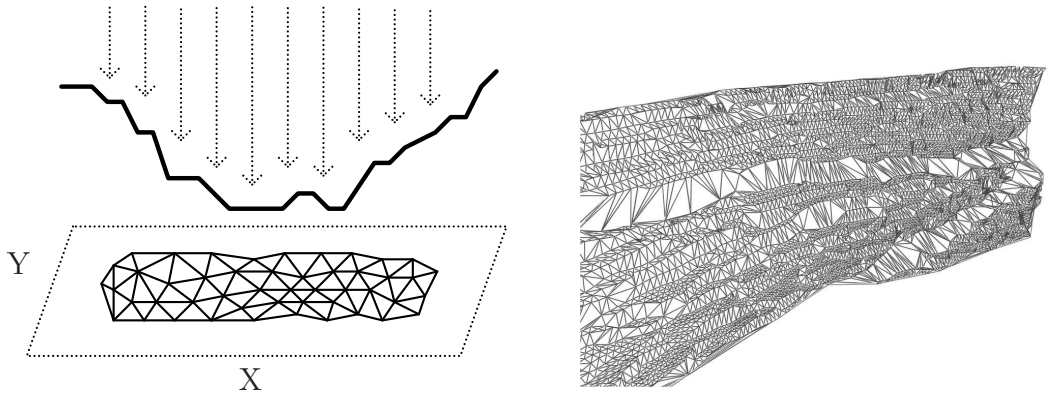
Consider 3D point set

$$\begin{aligned}\boldsymbol{P} &= \{\boldsymbol{p}_i\} = \{(x_i, y_i, z_i)\}, \quad \forall\, i \in \{1, \ldots, N_P\} \\ &= \begin{bmatrix} P_{11} & P_{12} & P_{13} \\ P_{21} & P_{22} & P_{23} \\ \vdots & \vdots & \vdots \\ P_{(N_p)(1)} & P_{(N_P)(2)} & P_{(N_P)(3)} \end{bmatrix}.\end{aligned}$$

Apply two dimensional Delaunay triangulation on the horizontal components of the set namely the first and second column of $\boldsymbol{P}$ as

$$\boldsymbol{T} = \text{delaunay}_{2D}\left(\begin{bmatrix} P_{11} & P_{21} & \cdots & P_{(N_P)(1)} \end{bmatrix}, \begin{bmatrix} P_{12} & P_{22} & \cdots & P_{(N_P)(2)} \end{bmatrix}\right).$$

Matrix $\boldsymbol{T}$ with dimension $3 \times N_T$ now contains the indices of points taken from $\boldsymbol{P}$ which form a set of triangle facelets. Figure 3.19 (b) depicts a typical result. This method of triangulation requires minimal overhead compared to the methods which follow. An overall acceptable result is only achievable when the set is spatially fairly flat and the variation of the height component is small when compared to the other components. The next meshing method addresses this limitation.



(a) (b)

Figure 3.19: (a) Ignoring the height component of the points results in the orthogonal projection of the points onto the XY-plane where we apply two-dimensional Delaunay triangulation. (b) A typical triangulated model of an open-pit sloped wall using two-dimensional Delaunay triangulation on the projected points.

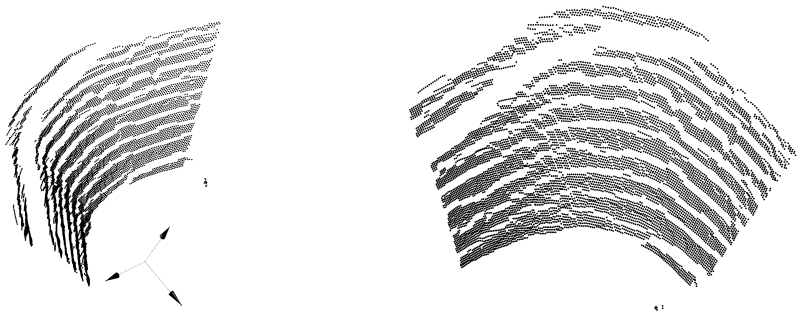
Principal component projected meshing

Another method of triangulation applies two-dimensional Delaunay triangulation on the Cartesian components with the greatest geometrical spread. This can be done using principal component analysis, also known as the Karhunen–Loève transformation. The data set is decorrelated and rotated so as to ensure the greatest independent spread across the Cartesian axes (Figure 3.20 (a) and (b)). Two-dimensional Delaunay triangulation is then applied on the axes with the greatest variation.

The following steps apply:

Consider 3D point set

$$\boldsymbol{P} = \{\boldsymbol{p}_i\}, \quad \forall\, i \in \{1, \ldots, N_P\}.$$



(a) (b)

Figure 3.20: (a) An almost vertical model with the principal components indicated by the arrowed axes. (b) The same model "flattened" by rotating the model with the principal component unit vectors.

Calculate the mean value as

$$\boldsymbol{\mu}_p = \frac{1}{N_P} \sum_{i=1}^{N_P} \boldsymbol{p}_i.$$

Convert points into mean-deviation form as

$$\hat{\boldsymbol{P}} = \begin{bmatrix} \boldsymbol{p}_1 - \boldsymbol{\mu}_p \\ \boldsymbol{p}_2 - \boldsymbol{\mu}_p \\ \vdots \\ \boldsymbol{p}_{(N_P)} - \boldsymbol{\mu}_p \end{bmatrix}.$$

Calculate covariance matrix

$$\boldsymbol{S} = \frac{1}{N_P - 1} \hat{\boldsymbol{P}} \hat{\boldsymbol{P}}^T.$$

Use singular value decomposition to determine the eigenvectors as

$$\boldsymbol{S} = \boldsymbol{R}\boldsymbol{D}\boldsymbol{V}^T.$$

Since $\boldsymbol{S}$ is a square symmetric matrix, eigendecomposition can also be used, but singular value decomposition is prefered because singular vectors are automatically sorted based on the variation of each component. $\boldsymbol{R}$ is a 3×3 orthogonal rotation matrix such that $\det(\boldsymbol{R}) = \pm 1$. Apply the rotation to form a new set as

$$\boldsymbol{Y} = \boldsymbol{P}\ \boldsymbol{R} = \begin{bmatrix} Y_{11} & Y_{12} & Y_{13} \\ Y_{21} & Y_{22} & Y_{23} \\ \vdots & \vdots & \vdots \\ Y_{(N_Y)(1)} & Y_{(N_Y)(2)} & Y_{(N_Y)(3)} \end{bmatrix}.$$

Apply two-dimensional Delaunay triangulation as

$$\boldsymbol{T} = \text{delaunay}_{2D} \left(\begin{bmatrix} Y_{11} & Y_{21} & \cdots & Y_{(N_P)(1)} \end{bmatrix}, \begin{bmatrix} Y_{12} & Y_{22} & \cdots & Y_{(N_P)(2)} \end{bmatrix} \right).$$

The last column of $\boldsymbol{Y}$ is ignored as it contains the orthogonal component with least amount of variation.

Point-of-origin Delaunay meshing

Point-of-origin meshing – our proposed method – should be used in the cases where the survey origin is known. Points are easily sorted according to the original survey pattern

when viewed from the surveying origin. We propose the following algorithm:

Consider 3D point set

$$\boldsymbol{P} = \begin{bmatrix} \boldsymbol{p}_1 & \boldsymbol{p}_2 & \cdots & \boldsymbol{p}_{N_P} \end{bmatrix}^T = \begin{bmatrix} (x_1, y_1, z_1) & (x_2, y_2, z_2) & \cdots & (x_{N_P}, y_{N_P}, z_{N_P}) \end{bmatrix}^T$$

constructed as a row matrix. The survey origin $C = (x_C, y_C, z_C)$ is known and defined within the same coordinate frame as $\boldsymbol{P}$. Place points within a local coordinate frame as

$$\hat{\boldsymbol{P}} = \begin{bmatrix} \boldsymbol{p}_1 - \boldsymbol{C} \\ \boldsymbol{p}_2 - \boldsymbol{C} \\ \vdots \\ \boldsymbol{p}_{N_P} - \boldsymbol{C} \end{bmatrix} = \begin{bmatrix} \hat{\boldsymbol{p}}_1 \\ \hat{\boldsymbol{p}}_2 \\ \vdots \\ \hat{\boldsymbol{p}}_{N_P} \end{bmatrix} = \begin{bmatrix} \hat{x}_1 & \hat{y}_1 & \hat{z}_1 \\ \hat{x}_2 & \hat{y}_2 & \hat{z}_2 \\ \vdots & \vdots & \vdots \\ \hat{x}_{N_P} & \hat{y}_{N_P} & \hat{z}_{N_P} \end{bmatrix}.$$

Define the Cartesian to spherical conversion function

$$(\theta, \phi, r) = \text{cart2sph}\,(x, y, z)\,, \tag{3.25}$$

with the return values defined as

$$\begin{aligned} \theta &= \text{atan2}\,(y, x)\,, \\ \phi &= \text{atan2}\left(z, \sqrt{x^2 + y^2}\right) \quad \text{and} \\ r &= \sqrt{x^2 + y^2 + z^2}. \end{aligned}$$

Convert $\hat{\boldsymbol{P}}$ into spherical coordinates as

$$\boldsymbol{A} = \begin{bmatrix} \text{atan2}\,(\hat{y}_1, \hat{x}_1) \\ \text{atan2}\,(\hat{y}_2, \hat{x}_2) \\ \vdots \\ \text{atan2}\,(\hat{y}_{N_P}, \hat{x}_{N_P}) \end{bmatrix} = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_{N_P} \end{bmatrix}$$

and

$$\boldsymbol{B} = \begin{bmatrix} \text{atan2}\left(\hat{z}_1, \sqrt{\hat{x}_1^2 + \hat{y}_1^2}\right) \\ \text{atan2}\left(\hat{z}_2, \sqrt{\hat{x}_2^2 + \hat{y}_2^2}\right) \\ \vdots \\ \text{atan2}\left(\hat{z}_{N_P}, \sqrt{\hat{x}_{N_P}^2 + \hat{y}_{N_P}^2}\right) \end{bmatrix} = \begin{bmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_{N_P} \end{bmatrix}.$$

Unwrap the azimuth angles of $\boldsymbol{A}$, to remove 2π jumps from adjacent points as

$$\hat{\boldsymbol{A}} = \text{unwrap}(\boldsymbol{A}).$$

Apply two-dimensional Delaunay triangulation on the corrected azimuth angles and the elevation angles as

$$\boldsymbol{T} = \text{delaunay}_{2D}(\hat{\boldsymbol{A}}, \boldsymbol{B})$$

where $\boldsymbol{T}$ is of dimension $N_T \times 3$ and contains the indices of the vertices making up the triangles. Figure 3.21 shows the result of the algorithm on a typical point set.

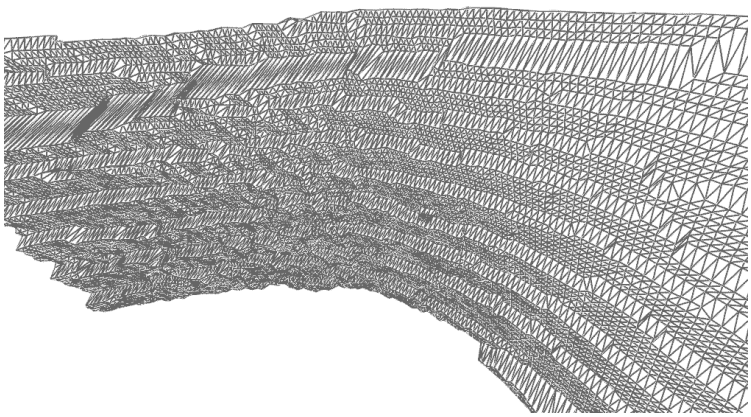


Figure 3.21: A model surface triangulated from the survey origin viewpoint.

3.5.4 Proposed method comparison

We test the proposed methods by applying them on a typical SM. The mean minimum and maximum triangle angles are calculated, as shown in Table 3.7. It is shown in [21], that the minimum triangle angle should be maximised and the maximum triangle angle should be minimised. The results show that point-of-origin meshing produces the best quality mesh.

Table 3.7: Shown here are the mean minimum and maximum angles of the triangles produced by the three proposed methods.

Method	Mean minimum angle (degrees)	Mean maximum angle (degrees)
Horizontal plane projected	20.93°	123.28°
Principal component projected	42.87°	78.85°
Point-of-origin	43.71°	74.45°

3.5.5 Conclusion

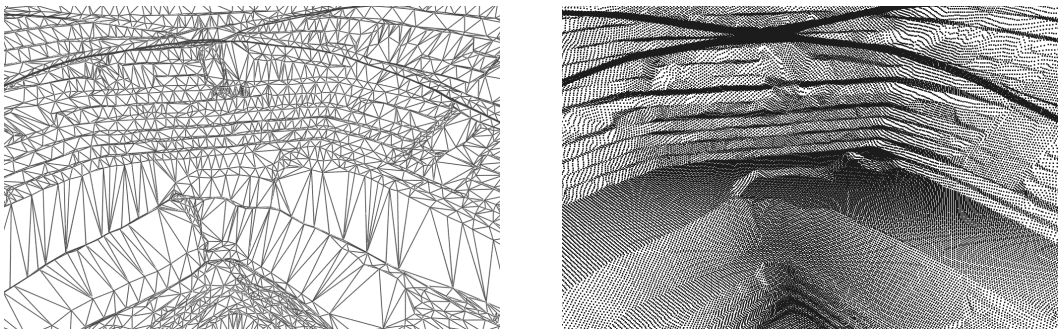
There are various ways of meshing model point clouds. We suggest TIN meshing as it the most widely used, and can easily be done using a Delaunay triangulation algorithm. The models used in this study are all non-overlapping meshed surfaces and therefore two dimensional Delaunay triangulation is the most practical. Deciding which two components to use for the Delaunay triangulation depends on the information available to us. Delaunay triangulation on the spherical converted coordinates produce the best quality mesh when the model survey origin is known to us. Principal component projected meshing should be used in cases where the survey origin is not known.

3.6 Point control with map resampling

3.6.1 Introduction

TIN DTMs usually have a very uneven distribution of points [1]. DTM areas with many, small triangles will have a higher concentration of points compared to the areas where the triangles are larger and cover a bigger area (Figure 3.22 (a)). Areas of lower concentration are far less likely to pair with a point from the search set during correspondence and as such could affect the quality of alignment.

Also, DTM areas with a high concentration of points could also slow down the rate of convergence given that the complexity of correspondence is now higher than needs be. We suggest resampling meshed models using a piecewise linear interpolation technique which ensures that points are evenly dispersed (Figure 3.22 (b)) and that the rate of convergence is optimised by controlling the density of the DTM. We propose a simple resampling algorithm which interpolates TIN models to a customisable density in the next section.



(a) (b)

Figure 3.22: (a) A section of DTM with triangle facelets of different sizes. Some areas require fewer triangles to describe the features with enough detail. (b) The same section resampled to ensure uniform point dispersion.

3.6.2 Bounded edge projected resampling

Consider meshed DTM point set $\boldsymbol{P} = \{\boldsymbol{p}_i\} = \{(x_i, y_i, z_i)\}, \quad \forall i \in \{1, \ldots, N_P\}$. The triangle facelets are sequenced in order from the points in $\boldsymbol{P}$ such that the collection of triangles $\boldsymbol{T} = \{\boldsymbol{t}_q\}$, $\forall q \in \{1, \ldots, {}^{N_P}/_3\}$ where

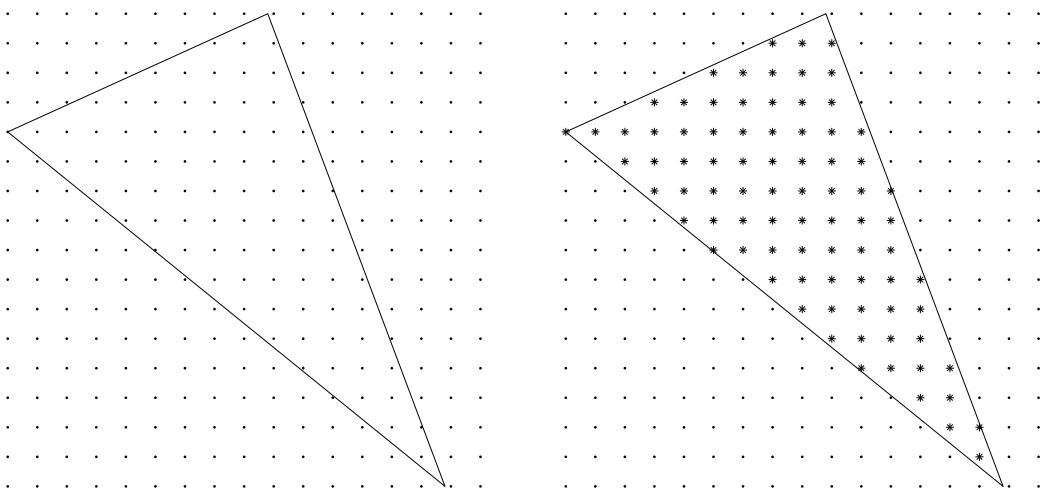
$$\begin{aligned}
\boldsymbol{t}_q &= \begin{bmatrix} t_{q11} & t_{q12} & t_{q13} \\ t_{q21} & t_{q22} & t_{q23} \\ t_{q31} & t_{q32} & t_{q33} \end{bmatrix} \\
&= \begin{bmatrix} \boldsymbol{p}_{3(q-1)+1} \\ \boldsymbol{p}_{3(q-1)+2} \\ \boldsymbol{p}_{3(q-1)+3} \end{bmatrix} \\
&= \begin{bmatrix} x_{3(q-1)+1} & y_{3(q-1)+1} & z_{3(q-1)+1} \\ x_{3(q-1)+2} & y_{3(q-1)+2} & z_{3(q-1)+2} \\ x_{3(q-1)+3} & y_{3(q-1)+3} & z_{3(q-1)+3} \end{bmatrix}.
\end{aligned}$$

For each triangle $\boldsymbol{t}_q \, \forall \, q \in \{1, ..., {}^{N_P}/_3\}$ where $N_T = {}^{N_P}/_3$ do the following:

Record the horizontally projected spatial bounds of each triangle as

$$\begin{aligned}
x_{min} &= \min(\{t_{q11}, t_{q21}, t_{q31}\}), \\
x_{max} &= \max(\{t_{q11}, t_{q21}, t_{q31}\}), \\
y_{min} &= \min(\{t_{q12}, t_{q22}, t_{q32}\}) \quad \text{and} \\
y_{max} &= \max(\{t_{q12}, t_{q22}, t_{q32}\}).
\end{aligned}$$

In order to make sure that all areas of the map are covered, we test all the points which project onto the triangle facelet within the recorded spatial bounds (Figure 3.23 (a)).



(a) (b)

Figure 3.23: (a) A bounding grid is created around each facelet based on the horizontally projected spatial limits of each triangle. All the points within the grid need to be tested for possible inclusion (b) The starred points qualify to be included in the final resampled point set because they fall within the triangle facelet.

From the spatial bounds construct a set of possible test points as

$$\begin{aligned} x_{start} &= x_{min} - \text{modulo}(x_{min}, x_{step}), \\ y_{start} &= y_{min} - \text{modulo}(y_{min}, y_{step}), \\ x_{end} &= x_{max} + \text{modulo}(x_{max}, x_{step}) \quad \text{and} \\ y_{end} &= y_{max} + \text{modulo}(y_{max}, y_{step}). \end{aligned}$$

The two-dimensional step sizes x_{step} and y_{step} allow us to control how dense the final resampled set will be. Set up matrices

$$\begin{aligned} \boldsymbol{C}_x &= \{x_{start},\ x_{start} + x_{step},\ x_{start} + 2x_{step},\ \ldots,\ x_{end}\} \quad \text{and} \\ \boldsymbol{C}_y &= \{y_{start},\ y_{start} + y_{step},\ y_{start} + 2y_{step},\ \ldots,\ y_{end}\}. \end{aligned}$$

Construct two-dimensional Cartesian product set

$$\boldsymbol{C} = \{c_j\} = \boldsymbol{C}_x \times \boldsymbol{C}_y, \forall\, j \in \left\{1, ..., \left(N_{C_x} N_{C_y}\right)\right\},$$

where N_{C_x} and $N_{C_y}$are the number of points in sets $\boldsymbol{C}_x$ and $\boldsymbol{C}_y$ respectively. The points in $\boldsymbol{C}$ represent all the points possibly contained within the current triangle facelet (Figure

3.23). Use the barycentric parameters [21]

$$d_1 = \begin{vmatrix} c_{j_x} & c_{j_y} & 1 \\ t_{q21} & t_{q22} & 1 \\ t_{q31} & t_{q32} & 1 \end{vmatrix}, \; d_2 = \begin{vmatrix} t_{q11} & t_{q12} & 1 \\ c_{j_x} & c_{j_y} & 1 \\ t_{q31} & t_{q32} & 1 \end{vmatrix} \quad \text{and } d_3 = \begin{vmatrix} t_{q11} & t_{q12} & 1 \\ t_{q21} & t_{q22} & 1 \\ c_{j_x} & c_{j_y} & 1 \end{vmatrix} \tag{3.26}$$

to evaluate all the points within the grid (Figure 3.23 (a)). Points that qualify using the following conditional function using the previous barycentric parameters are positioned within the triangle facelet

$$\mathrm{f}(d_1, d_2, d_3) = [((d_1 \geq 0) \wedge (d_2 \geq 0) \wedge (d_3 \geq 0))] \; \vee \; [((d_1 \leq 0) \wedge (d_2 \leq 0) \wedge (d_3 \leq 0))]\,. \tag{3.27}$$

Operator $\vee$ denotes logic disjunction function "or", and operator $\wedge$ denotes logic conjunction function "and". The starred points in Figure 3.23 (b) fall within the triangle facelet, and as such, have qualifying barycentric values. Qualified points require a height component that is calculated as

$$z = a_1 + a_2 c_{j_x} + a_3 c_{j_y}$$

where the planar parameters [21] are set as

$$\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 1 & t_{q11} & t_{q12} \\ 1 & t_{q21} & t_{q22} \\ 1 & t_{q31} & t_{q32} \end{bmatrix}^{-1} \begin{bmatrix} t_{q13} \\ t_{q23} \\ t_{q33} \end{bmatrix}.$$

Finally we calculate the height component as

$$\begin{aligned} z &= \begin{bmatrix} 1 & c_{j_x} & c_{j_y} \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}, \\ &= \begin{bmatrix} 1 & c_{j_x} & c_{j_y} \end{bmatrix} \begin{bmatrix} 1 & t_{q11} & t_{q12} \\ 1 & t_{q21} & t_{q22} \\ 1 & t_{q31} & t_{q32} \end{bmatrix}^{-1} \begin{bmatrix} t_{q13} \\ t_{q23} \\ t_{q33} \end{bmatrix}. \end{aligned} \tag{3.28}$$

The qualified points of each triangle facelet are grouped together to form the final resampled set. The algorithm is given in pseudocode form in algorithm 1.

Algorithm 1 A bounded edge piecewise linear resampling algorithm.

```
for q = 1 to N_T do                                  ▷ Iterate through all the triangles
    for c_jx = x_start to x_end do                   ▷ Iterate through all the possible x points
        for c_jy = y_start to y_end do               ▷ Iterate through all the possible y points
            d_1, d_2, d_3 ← Equation 3.26
            if f(d_1, d_2, d_3) then
                Include point into the final point set
                z ← Equation 3.28
            end if
        end for
    end for
end for
```

3.6.3 Point control example

To test the effect of different resample spacing we apply the interpolation scheme to a TIN DTM. Grid spacings 32m, 22m, 12m and 7m are used (Figure 3.24 (a), (b), (c) and (d)). Wider grid spacing results in less detailed, less dense point sets. A finer grid spacing results in more detail but require more points to describe the same area.

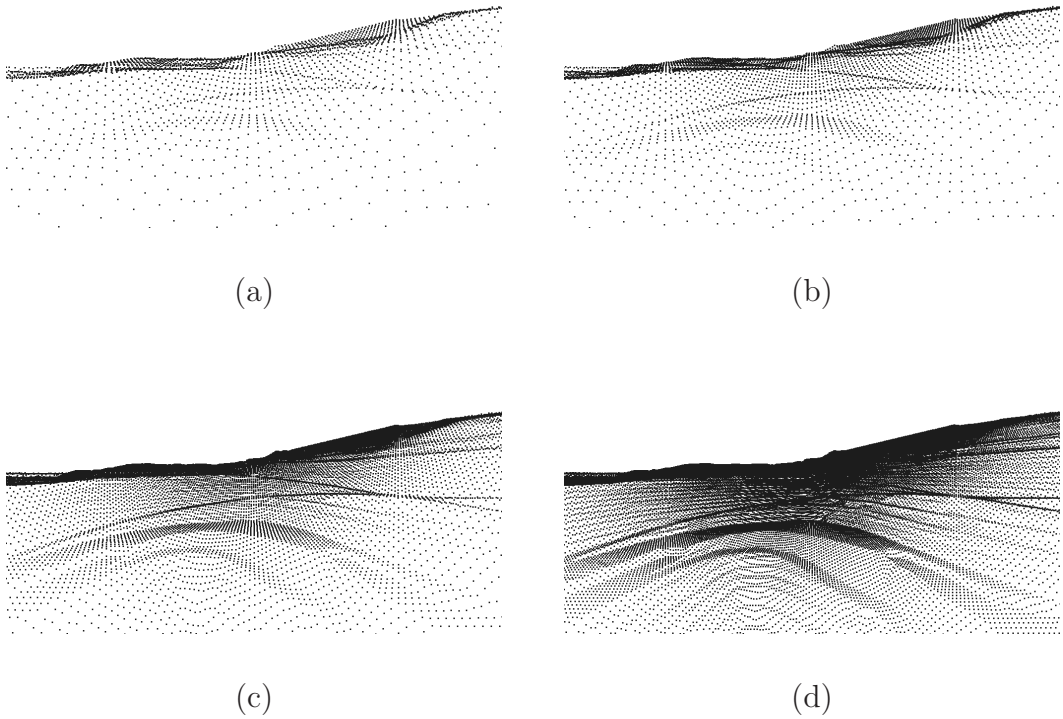


Figure 3.24: (a) A DTM resampled with 32 metre grid spacing. (b) The same DTM resampled with 22 metre grid spacing. (c) The same DTM resampled with 12 metre grid spacing. (d) The same DTM resampled with 7 metre grid spacing.

3.6.4 Conclusion

Correspondence is a critical step during alignment and DTMs with unevenly distributed points or DTMs of insufficient point density can lead to less than ideal correspondence. One method of ensuring evenly distributed points and a customisable density is by resampling TIN DTMs using a piecewise linear interpolation technique. This will aid the correspondence process where point density is low, and speed up the alignment in the cases where the density is too high.

3.7 A global alignment scheme for synthetic maps with four degrees of freedom

3.7.1 Introduction

Previously we discussed iterative correspondence alignment (ICA) and how it is used to align two three-dimensional sets. ICA is a general solution and depending on how correspondence is made, can be applied to meshed or unmeshed three-dimensional models with six degrees of motion freedom. ICA is, however, in general limited in application since models which are not reasonably aligned prior to ICA are very unlikely to converge. In many studies, this is referred to as global alignment or prealignment (see e.g. [16], [20], [17]). In this section we cover a global fitting scheme which will align two model sets well enough so that ICA will converge to the correct local position. The scheme is designed specifically for the alignment of a synthetic map (SM) with either another SM or with a digital terrain model (DTM). The scheme estimates spatial motion parameters of four degrees of freedom, that is to say translation and heading correction. A set of tests are done to verify the accuracy of the algorithm. These tests are presented in Chapter 5.

3.7.2 Synthetic maps

The purpose of this study is to show that a SM can be aligned with another SM or DTM when they cover the same area. Because of the limitations of a general alignment solution (especially considering processing time and implementability) we examine the specific conditions applicable to our problem. The SMs used in this study are obtained from a mobile mine surveying radar. The radar is placed on a stable high point to ensure unobstructed perpendicular coverage of the mine slope (Figure 3.25 (a)). The radar is levelled prior to operation which means that the roll and pitch angles will always match that of the DTM (or previously obtained SM), reducing the alignment complexity from six to four degrees of freedom.

Before operation the radar operator defines an area of the slope wall for surveying. This is also refered to as a scan region. The scan region is marked out in the shape of a trapezium with the four corner points defined in azimuth and elevation angles. Figure 3.25 (b) depicts an example scan region viewed from the radar origin. The scan region is scanned in a sweeping motion, assigning a range value to each grid point as the antenna beam passes over it. When the scan is finished the spherical coordinate values of each point (azimuth angle, elevation angle and range) are converted into three dimensional Cartesian coordinates, creating a 3D SM point cloud.

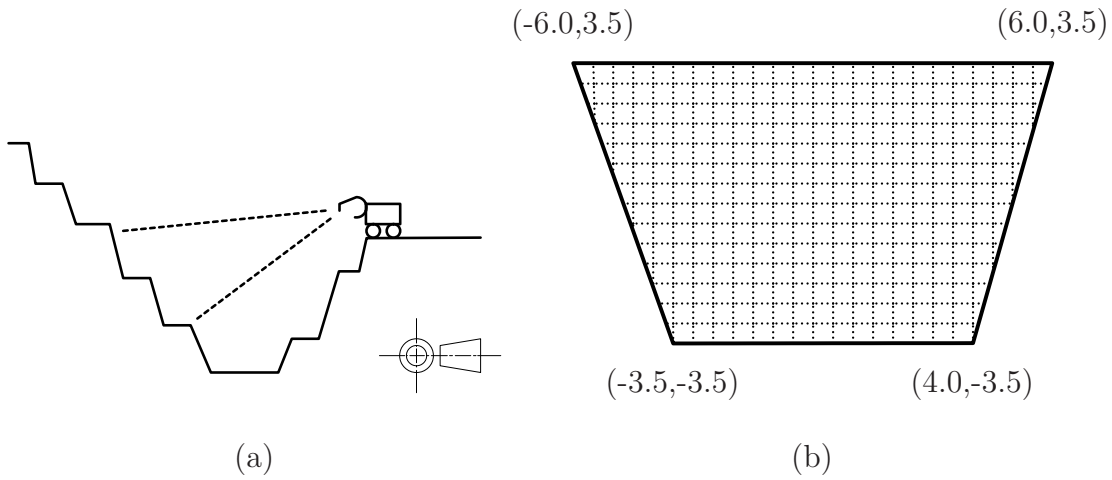


Figure 3.25: (a) The radar is levelled prior to scanning. A section of the mine slope is defined and is refered to as a scan region. (b) A scan region is defined by a trapezium shape where the corner points are in angular coordinates (azimuth and elevation angles, respectively).

In order for the SM point cloud to be useful in a geographical and surveying sense, it will have to be placed within the global or common coordinate frame used by that specific mine. This is done by either georeferencing the radar prior to operation or by aligning the radar point cloud with a template map defined in the global (common) coordinate frame. For the study we concern ourselves with the latter. By aligning the SM with the DTM we indirectly determine the radar position (origin) and heading (Figure 3.26). In the sections to follow we explain and test a prealignment algorithm specifically designed for SM point cloud alignment.

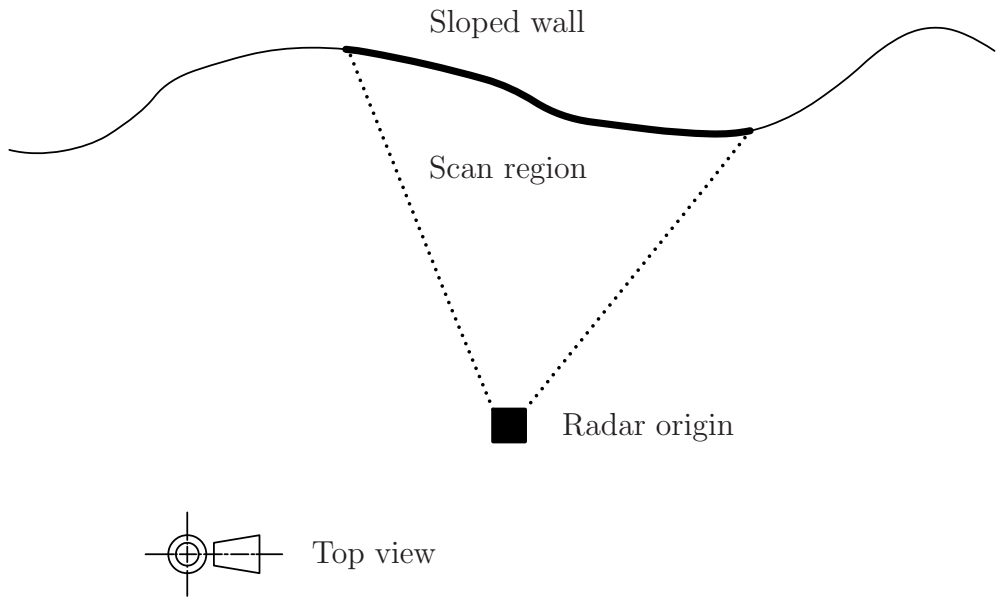


Figure 3.26: The scan region is aligned with the model of the mine slope. The radar position and heading is then calculated after alignment.

3.7.3 Prealignment algorithm

So far we have shown that ICA guarantees convergence provided that the models are reasonably aligned beforehand. In this section we describe an algorithm which aligns a radar point cloud to another radar point cloud or DTM well enough for ICA to converge to the correct local position. To speed up the processing time of the algorithm some user interaction is required in the form of an estimated radar position. The estimated radar position is used to limit the search space, thereby decreasing the required processing time.

The algorithm takes as input the template set, the search set, the estimated radar position as well as some predefined tolerance values. The DTM or base radar point cloud is the template set which we want to align to, and remains stationary. The SM point cloud is the search set (in the form of a scan region) and is rotated and translated to best align with the template set. The algorithm finds the best fit of the scan region (given its SM properties) upon the template set. The estimated radar position as well as tolerance values ensure that unrealistic alignment positions are disregarded during the process.

3.7.4 An overview of the algorithm

Synthetic maps are defined using trapezium shapes where the first point in the set will be the upper left corner of the shape and the last point the bottom right corner of the shape. We denote the SM point set as $\boldsymbol{D}$. The first point is denoted as $\boldsymbol{d}_1$, and the last point as $\boldsymbol{d}_{N_D}$. Deducting these two point for one another as $\boldsymbol{d}_{\text{cross}} = \boldsymbol{d}_{N_D} - \boldsymbol{d}_1$, results in a vector which

spans the SM in a downward direction (Figure 3.27). Vector $\boldsymbol{d}_{\text{cross}}$ describes the maximum space which the SM could take up when placed over the template set $\boldsymbol{M}$. By fitting the scan region shape on all possible points in the template set, a prealignment solution is guaranteed. Such a brute force solution is however very expensive computationally, especially when working under a time constraint. The processing time can however be reduced by eliminating false solutions as early as possible. The elimination process can be done in five conditional steps. After each step the points which fail the conditional requirements will be removed from the template set.

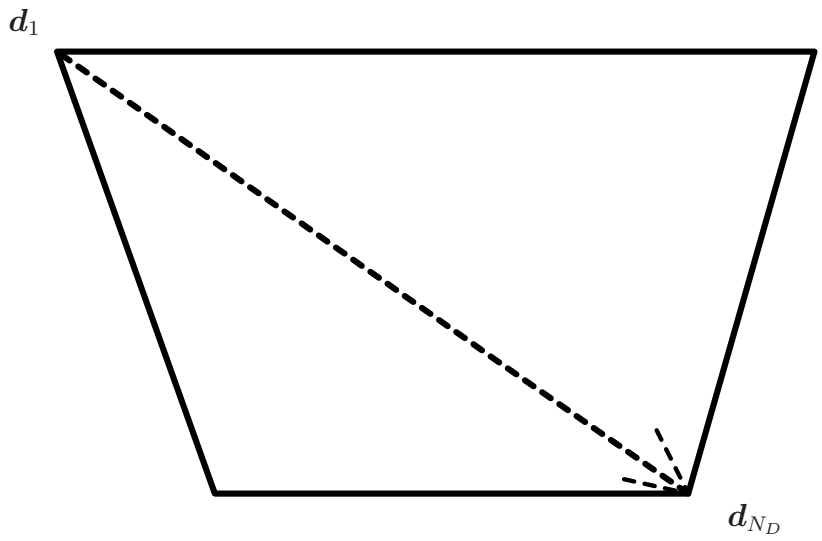


Figure 3.27: A vector drawn from the first to the last point in synthetic map point set $\boldsymbol{D}$.

Conditional requirement 1 : Elimination based on height difference

The radar is always levelled prior to operations which means that the height component of the SM point set will remain constant for all possible heading angles. As stated before, the first and last point of the SM defines its spatial bounds. By deducting the height components of these two points we determine the exact height spread (denoted as $z_{d_{\text{cross}}}$) the search set has (Figure 3.28). The first – and least computationally expensive – conditional requirement is to evaluate the points in the template set $\boldsymbol{M}$ to have a similar height component difference as that of $\boldsymbol{D}$. Unlike the other conditionals, this conditional requires no squaring or square root calculations, making it the least expensive computationally. The height difference of vectors calculated from points in $\boldsymbol{M}$ must fall within a certain tolerance (denoted as $z_{\text{tolerance}}$) of $z_{d_{\text{cross}}}$ to meet the conditional requirement. Points which fall outside the conditional requirement are removed from the set. Processing continues with the remaining points of $\boldsymbol{M}$ to be evaluated by the second conditional requirement.

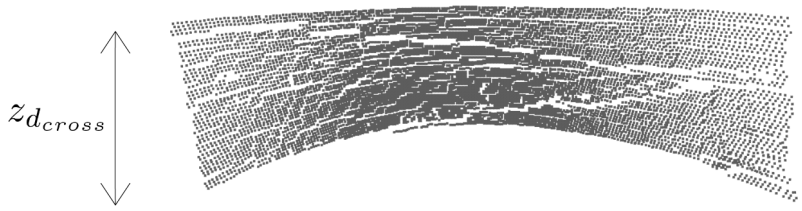


Figure 3.28: A SM with difference in height indicated by the vertical vector $z_{d_{\text{cross}}}$.

Conditional requirement 2 : Elimination based on SM cross vector length

The second conditional requirement compares the length of vectors calculated from template set $\boldsymbol{M}$ with that of the length of cross vector $\boldsymbol{d}_{\text{cross}}$. All the point pairs in the template set are then filtered so as to have the same vector length as that of the SM cross vector (Figure 3.29). Again the conditional requirement allows for a tolerance, denoted as $r_{\text{tolerance}}$. Processing continues with the remaining points of $\boldsymbol{M}$ to be evaluated using the third conditional requirement.

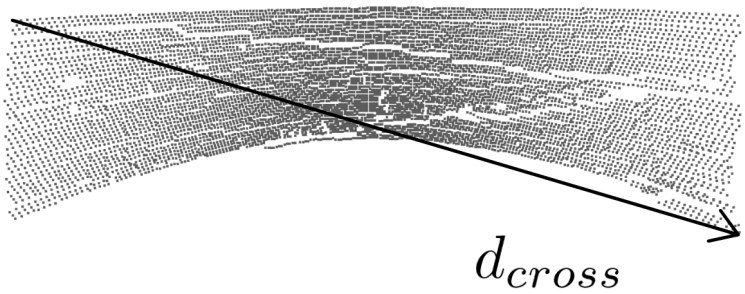


Figure 3.29: A SM with vector $\boldsymbol{d}_{\text{cross}}$ drawn from the first to the last point in the set.

Conditional requirements 3 and 4 : Elimination based on estimated centre to corner point vector lengths

Vectors are calculated between the estimated centre $\boldsymbol{c}_R$ and the remaining points of $\boldsymbol{M}$. The lengths of these vectors must be similar in length to that of vectors calculated between the original local radar centre $\boldsymbol{c}_0$ and corner points $\boldsymbol{d}_1$ and $\boldsymbol{d}_{N_D}$ (Figure 3.30). The third conditional requirement compares length $\|\boldsymbol{m}_A - \boldsymbol{c}_R\|$ to $\|\boldsymbol{d}_1 - \boldsymbol{c}_0\|$, and the fourth conditional requirement compares length $\|\boldsymbol{m}_B - \boldsymbol{c}_R\|$ to $\|\boldsymbol{d}_{N_D} - \boldsymbol{c}_0\|$. The vector lengths must be similar, within tolerance value $c_{\text{tolerance}}$. Processing continues with the remaining points of $\boldsymbol{M}$ to be evaluated by the fifth and final conditional.

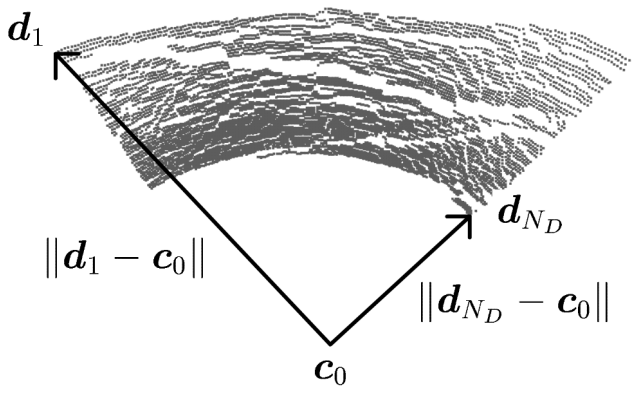


Figure 3.30: The points in the template set are evaluated so that distances between the estimated radar centre $\boldsymbol{c}_R$ and the matching corner points fall with a certain tolerance.

Conditional requirement 5 : Elimination based on new estimated centre

SM points $\boldsymbol{d}_1$ and $\boldsymbol{d}_{N_D}$ are positioned in the global coordinate frame before the fifth and last conditional requirement is to be evaluated (Figure 3.31). The points are first corrected in heading angle and then corrected in absolute position. Figure 3.32 is a three-dimensional depiction of how the SM corner points are placed over template set point pair $\boldsymbol{m}_A$ and $\boldsymbol{m}_B$. Note that labels $\boldsymbol{d}_1$ and $\boldsymbol{m}_A$ represent the same point. The new estimated centre $\boldsymbol{e}_R$ is calculated using the original local radar centre and the difference in position between the two frames. The final conditional is then evaluated, ensuring that the newly estimated centre $\boldsymbol{e}_R$ fall within a radius (of length $c_{\text{tolerance}}$) of estimated centre $\boldsymbol{c}_R$ that was provided by the user (Figure 3.32).

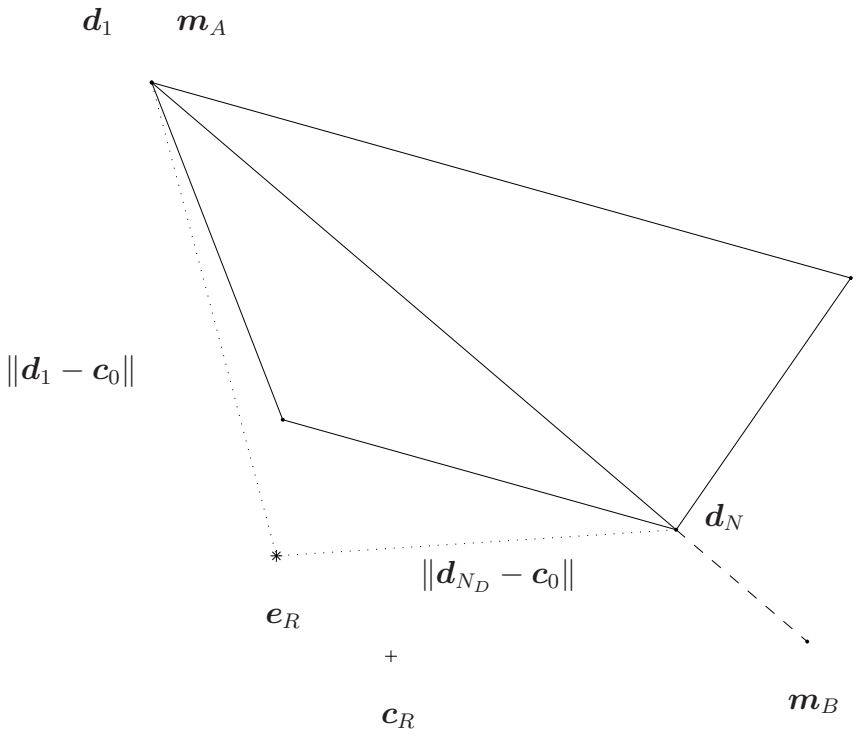


Figure 3.31: Points $\boldsymbol{d}_1$ and $\boldsymbol{d}_{N_D}$ are corrected in heading and position in order to be placed on top of points $\boldsymbol{m}_A$ and $\boldsymbol{m}_B$.

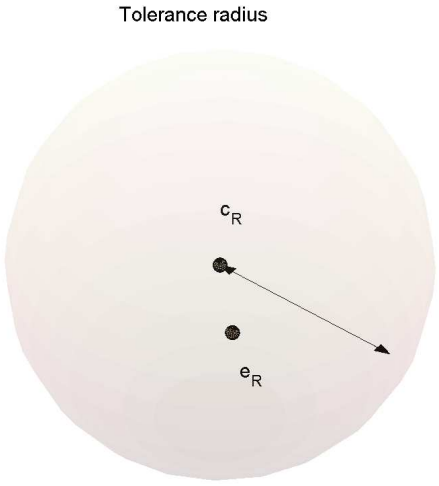


Figure 3.32: The newly estimated centre must be within a certain radial tolerance relative to the user supplied initial centre.

Determine best alignment fit based on remaining points

Points that passed the fifth and final conditional requirement represent all the possible matches that remain. Points from $\boldsymbol{D}$ will now be much more thoroughly compared to the original unfiltered template set $\boldsymbol{M}$. $\boldsymbol{D}$ is corrected in heading and absolute position as $\boldsymbol{D}_{\triangle\theta} = \mathbf{R}(\triangle\theta)\,\boldsymbol{D} + \boldsymbol{t}$, where function $\mathbf{R}(\triangle\theta)$ represents a rotation matrix which corrects the heading angle and $\boldsymbol{t}$ the correction in absolute position. A subset of random points are uniformly selected from $\boldsymbol{D}_{\triangle\theta}$ as $\boldsymbol{D}_{\gamma} = \text{random}(\boldsymbol{D}_{\triangle\theta})$, where random( ) has a uniform distribution. For set $\boldsymbol{D}_{\gamma}$ find the closest points in the original unfiltered template set $\boldsymbol{M}$ and calculate the RMS spatial difference. The RMS spatial difference will be stored and compared later when all iterations have ended. The alignment heading and absolute position difference associated with the lowest RMS spatial difference will be considered the best possible fit of $\boldsymbol{D}$ onto $\boldsymbol{M}$.

3.7.5 The algorithm in mathematical steps

We define the template set as

$$\boldsymbol{M} = \{\boldsymbol{m}_j\} = \left\{\left(x_{m_j}, y_{m_j}, z_{m_j}\right)\right\} \quad \forall\, j \in \{1, \ldots, N_M\} \quad \text{and}$$

the search set (SM) as

$$\boldsymbol{D} = \{\boldsymbol{d}_i\} = \{(x_{d_i}, y_{d_i}, z_{d_i})\} \quad \forall\, i \in \{1, \ldots, N_D\}\,.$$

$\boldsymbol{M}$ is in a geographical global (common) coordinate system, whereas $\boldsymbol{D}$ is in a local coordinate system centred around the radar frame origin $\boldsymbol{c}_0 = \{(x_0, y_0, z_0)\} = \{(0, 0, 0)\}$. Define an initial radar centre point, within the geographical coordinate frame as

$$\boldsymbol{c}_R = (x_{c_R}, y_{c_R}, z_{c_R})\,.$$

This coordinate will be provided by the user's initial estimate. As part of the algorithm a new radar centre will be calculated, defined as

$$\boldsymbol{e}_R = (x_{e_R}, y_{e_R}, z_{e_R})\,.$$

Vector $\boldsymbol{d}_{\text{cross}}$ which spans the scan region outline is calculated from two of the corner points (Figure 3.33). We calculate this cross vector from the first and last points of set $\boldsymbol{D}$ denoting it as

$$\boldsymbol{d}_{\text{cross}} = \boldsymbol{d}_{N_D} - \boldsymbol{d}_1 = (x_{d_{\text{cross}}}, y_{d_{\text{cross}}}, z_{d_{\text{cross}}})\,,$$

where $\boldsymbol{d}_1$ is the upper left point and $\boldsymbol{d}_{N_D}$ the bottom right point (Figure 3.33).

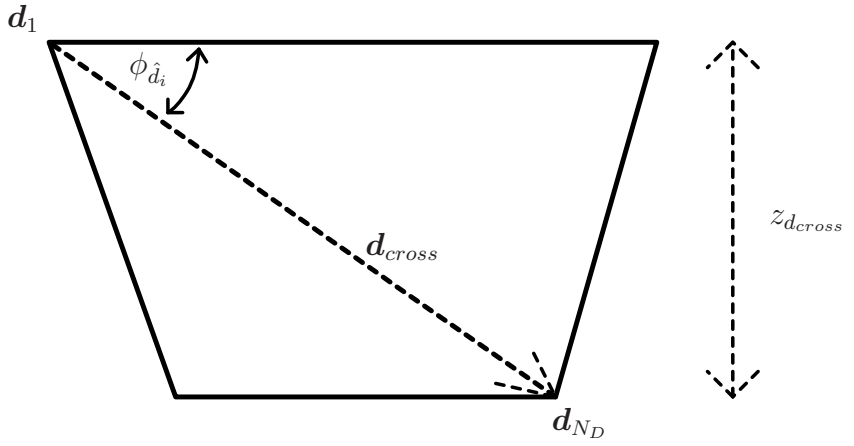


Figure 3.33: A scan region viewed from the radar origin. The vector spanning the scan region outlines, is calculated from the opposite corners of the scan region.

Calculate the elevation angle of vector $\boldsymbol{d}_{\text{cross}}$ as

$$\phi_{d_{\text{cross}}} = \text{atan2}\left(z_{d_{\text{cross}}},\ \sqrt{x^2_{d_{\text{cross}}} + y^2_{d_{\text{cross}}}}\right).$$

Calculate the length of vector $\boldsymbol{d}_{\text{cross}}$ as

$$r_{d_{\text{cross}}} = \|\boldsymbol{d}_{\text{cross}}\|.$$

Iterate through template set and evaluate conditional requirements

All possible template set point combinations are considered during the alignment process. In order to match the SM shape onto the template set, all combinations of possible corner point pairs must be considered. Define template point pair

$$\boldsymbol{m}_A = \{(x_{m_A}, y_{m_A}, z_{m_A})\} \quad \text{and} \quad \boldsymbol{m}_B = \{(x_{m_B}, y_{m_B}, z_{m_B})\} \quad \forall\, A, B \in \{1, \ldots, N_M\},$$

where $A \neq B$. From template points $\boldsymbol{m}_A$ and $\boldsymbol{m}_B$ define template cross vector

$$\boldsymbol{m}_{B-A} = \boldsymbol{m}_B - \boldsymbol{m}_A.$$

We hope to find the point pair $\boldsymbol{m}_A$ and $\boldsymbol{m}_B$ which most closely matches the SM cross vector $\boldsymbol{d}_{\text{cross}}$. The template set is efficiently iterated as

Algorithm 2 The template set is efficiently iterated using the following algorithm.

for $k = 1$ to N_M **do**
  $\boldsymbol{m}_A \leftarrow \boldsymbol{m}_k$
  for $j = (k+1)$ to N_M **do**
    $\boldsymbol{m}_B \leftarrow \boldsymbol{m}_j$
    $\boldsymbol{m}_{B-A} \leftarrow \boldsymbol{m}_B - \boldsymbol{m}_A$
    Evaluate conditional requirements
  end for
end for

From $\boldsymbol{m}_{B-A}$ calculate delta values

$$\begin{aligned}
\triangle x_{B-A} &= x_{m_B} - x_{m_A}, \\
\triangle y_{B-A} &= y_{m_B} - y_{m_A}, \\
\triangle z_{B-A} &= z_{m_B} - z_{m_A} \text{ and} \\
\triangle r_{B-A} &= \|\boldsymbol{m}_B - \boldsymbol{m}_A\| .
\end{aligned}$$

Define tolerance values $z_{\text{tolerance}}$, $r_{\text{tolerance}}$ and $c_{\text{tolerance}}$. The rejection criteria of the algorithm is dependent on five conditionals which are functions of the tolerance values. Four out of the five conditions are:

- Conditional requirement 1 : evaluate $\triangle z_{B-A}$ as

$$(z_{d_{\text{cross}}} - z_{\text{tolerance}}) < \triangle z_{B-A} < (z_{d_{\text{cross}}} + z_{\text{tolerance}}) . \tag{3.29}$$

- Conditional requirement 2 : evaluate $\triangle r_{B-A}$ as

$$(r_{d_{\text{cross}}} - r_{\text{tolerance}}) < \triangle r_{B-A} < (r_{d_{\text{cross}}} + r_{\text{tolerance}}) . \tag{3.30}$$

- Conditional requirement 3 : evaluate $\|\boldsymbol{m}_A - \boldsymbol{c}_R\|$ as

$$(\|\boldsymbol{d}_1 - \boldsymbol{c}_0\| - c_{\text{tolerance}}) < \|\boldsymbol{m}_A - \boldsymbol{c}_R\| < (\|\boldsymbol{d}_1 - \boldsymbol{c}_0\| + c_{\text{tolerance}}) . \tag{3.31}$$

- Conditional requirement 4 : evaluate$\|\boldsymbol{m}_B - \boldsymbol{c}_R\|$ as

$$(\|\boldsymbol{d}_{N_D} - \boldsymbol{c}_0\| - c_{\text{tolerance}}) < \|\boldsymbol{m}_B - \boldsymbol{c}_R\| < (\|\boldsymbol{d}_{N_D} - \boldsymbol{c}_0\| + c_{\text{tolerance}}) . \tag{3.32}$$

More steps are required in order to evaluate the fifth and final conditional requirement. SM point $\boldsymbol{d}_1$ is to be corrected in heading angle and absolute position before the final conditional

requirement can be evaluated. Since the radar is always levelled prior to operation, the heading can be determined as the difference in azimuth Euler components between the template and the search set. Get current horizontal angle as

$$\theta_{m_{B-A}} = \text{atan2}\left(y_{m_{B-A}}, x_{m_{B-A}}\right). \tag{3.33}$$

Calculate the difference in heading angle as

$$\triangle\theta = \theta_{\hat{d}_{\text{cross}}} - \theta_{m_{B-A}}. \tag{3.34}$$

Correct the corner point's heading as

$$\boldsymbol{d}_1^{'} = \begin{bmatrix} \cos(\triangle\theta) & \sin(\triangle\theta) & 0 \\ -\sin(\triangle\theta) & \cos(\triangle\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{d_1} \\ y_{d_1} \\ z_{d_1} \end{bmatrix}. \tag{3.35}$$

Correct the local radar centre's heading as

$$\boldsymbol{c}_0^{'} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \cos(\triangle\theta) & \sin(\triangle\theta) & 0 \\ -\sin(\triangle\theta) & \cos(\triangle\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}. \tag{3.36}$$

The SM represents a rigid body and as such any point can be used the determine the difference in absolute position (Figure 3.34). Determine the difference in absolute position as

$$\boldsymbol{t} = \begin{bmatrix} \triangle x \\ \triangle y \\ \triangle z \end{bmatrix} = \boldsymbol{m}_A - \boldsymbol{d}_1^{'}. \tag{3.37}$$

Calculate the estimated radar centre $\boldsymbol{e}_R$ as

$$\boldsymbol{e}_R = \begin{bmatrix} x_{e_R} \\ y_{e_R} \\ z_{e_R} \end{bmatrix} = \boldsymbol{c}_0^{'} + \boldsymbol{t} = \begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix} + \begin{bmatrix} \triangle x \\ \triangle y \\ \triangle z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} \triangle x \\ \triangle y \\ \triangle z \end{bmatrix} \tag{3.38}$$

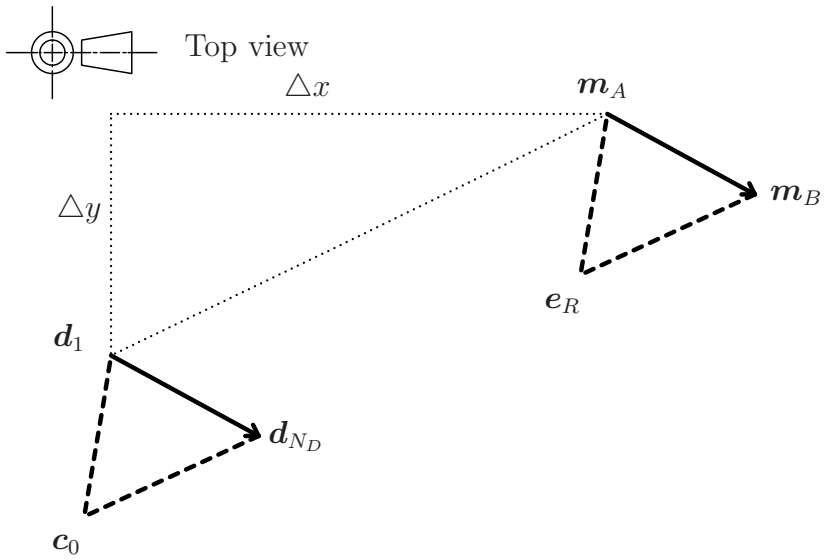


Figure 3.34: The radar position difference between the coordinate frames is determined by subtraction of the Cartesian components of any two corresponding points, as in this case with $\boldsymbol{d}_1$ and $\boldsymbol{m}_A$.

We can now evaluate the fifth and final conditional requirement:

- Conditional requirement 5 : evaluate

$$\|\boldsymbol{e}_R - \boldsymbol{c}_R\| < (c_{\text{tolerance}}). \tag{3.39}$$

All the points which pass this conditional will be used in the next processing steps.

Determine best alignment fit

Rotate all the SM points in order to correct the heading as well as the absolute position

$$\boldsymbol{D}_{\triangle\theta} = \begin{bmatrix} \cos(\triangle\theta) & \sin(\triangle\theta) & 0 \\ -\sin(\triangle\theta) & \cos(\triangle\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \boldsymbol{D} + \boldsymbol{t}. \tag{3.40}$$

Pick a subset of random points from $\boldsymbol{D}_{\triangle\theta}$ using a uniform distribution such that

$$\boldsymbol{D}_\gamma = \text{random}(\boldsymbol{D}_{\triangle\theta}) \tag{3.41}$$

and

$$\boldsymbol{D}_\gamma \subset \boldsymbol{D}_{\triangle\theta} \text{ where } N_{D_\gamma} \ll N_D. \tag{3.42}$$

For set $\boldsymbol{D}_\gamma$, find the closest points in set $\boldsymbol{M}$ and calculate the RMS spatial difference. The RMS error is stored as

$$dist_{\text{error}} == \sqrt{\frac{1}{N_{D_\gamma}} \sum^{N_{D_\gamma}} \min\left(\|\boldsymbol{D}_\gamma - \boldsymbol{M}\|\right)}. \tag{3.43}$$

After all iterations

The heading and absolute position associated with the lowest stored RMS error are regarded as the final alignment parameters. Reaching the end of the algorithm, the template set fit with lowest spatial difference is picked as the optimal global alignment position.

3.7.6 The algorithm steps explained in pseudocode

Algorithm 3 The global alignment algorithm.

```
for k = 1 to N_M do
    m_A ← m_k
    for j = (k + 1) to N_M do
        m_B ← m_j
        m_{B−A} ← m_B − m_A
        △z_{B−A} ← z_{m_B} − z_{m_A}
        if cond(△z_{B−A}) then
            △r_{B−A} ← ‖m_B − m_A‖
            if cond(△r_{B−A}) then
                if z_A < z_B then
                    swap (m_A, m_B)
                end if
                if cond(‖m_A − c_R‖) then
                    if cond(‖m_B − c_R‖) then
                        Calculate estimated radar centre using Eq. 3.33 to 3.38
                        if cond(‖e_R − c_R‖) then
                            Determine final estimated radar centre using Eq. 3.40 to 3.43
                        end if
                    end if
                end if
            end if
        end if
    end for
end for
Best estimate is result with minimum error
```

3.7.7 Conclusion

We have explained a global alignment algorithm which aligns SM point clouds to either other SM point clouds or DTMs. The algorithm considers motion complexity of four degrees of freedom. The algorithm requires that one model be a spatial subset of the other. An estimated model origin point is used to eliminate incorrect alignment positions, which speeds up the algorithm. Several tests are done to verify the accuracy of the algorithm (Chapter 5). Results indicate a spatial difference of less than 8 metres in all cases (Table 5.1). Six out of the seven tests resulted in a heading accuracy of less than 2 degrees and three out of the seven tests resulted in position errors of less than 6 metres. In the section to follow we combine this prealignment algorithm with iterative correspondence alignment to form a complete alignment scheme suitable for radar georeferencing.

3.8 A complete radar georeferencing scheme

3.8.1 Introduction

One of the biggest problems of iterative correspondence alignment is that it requires the models to be reasonably aligned prior to alignment. In the previous section we described a global alignment algorithm which provides some alignment, requiring only an estimated starting point. This algorithm provides reasonable alignment quality, but the alignment quality can be improved even further by using iterative correspondence alignment (ICA) to complete the process. In this section we describe a fully implementable georeferencing scheme useful to georeference a mobile mine surveying radar. This full solution is kept as versatile as possible by allowing for alignment on surfaced DTMs as well as a non-surfaced georeferenced SMs. In this Chapter we also show that by applying the iterative correspondence alignment after the global fit, we improve the quality of the final alignment sufficiently for the determined position and heading to be within our predefined specification as stipulated in Chapter 1. To verify the accuracy of the georeferencing scheme several tests are done. They are presented in Chapter 5.

3.8.2 The effect of position and heading error

The radar illuminates the mine slope with a torch-like antenna beam. The radar's 3 dB beamwidth is 2.5 degrees in both azimuth and elevation angles [1]. For a correctly georeferenced radar the targets will appear where we expect them to be when viewed in the mine coordinate frame. However, when the radar is incorrectly georeferenced, the targets will be positioned incorrectly. Inaccuracy in the absolute position of the radar results in an offset shift between the radar targets and the mine model. Inaccuracy in the heading of the radar

results in a rotation error between the radar targets and the mine model. Between these two inaccuracies, the latter is more of a concern since the error scales linearly with the target range, while the offset is independent of the target range. Figure 3.36 illustrates a scenario where the heading angle is off by 45 degrees. At a distance of 100 metres the target position error is only 100 metres, but for a distance of 300 metres the error is 300 metres.

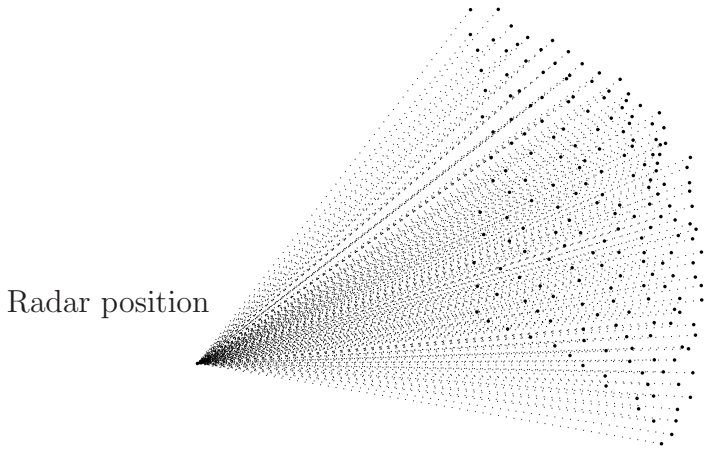


Figure 3.35: Synthetic maps (SMs) are resolved in spherical coordinates and so the frame origin is always known to us.

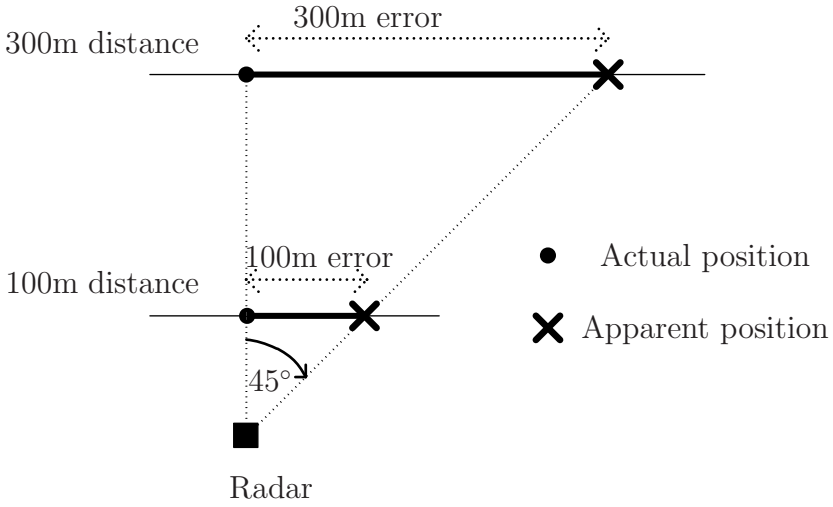


Figure 3.36: The error in distance of a target when the radar is misaligned due to a heading error of 45 degrees.

Range (m)	Diameter (m)
100	4.36
250	10.90
500	21.81
1000	43.62
1500	65.43
2000	87.24

Table 3.8: The diameter coverage of a 2.5 degree antenna beam at different ranges.

The radar can sight targets as far as a few kilometers and therefore we have to take into account the target position error due to range. Table 3.8 shows the beam diameter at different ranges. Radars do, however, have the ability to resolve multiple targets within its illuminated beam. The targets will however resolve with weaker amplitude when appearing on the edge of the beam, but will still be visible. For this reason we consider 2 degrees accuracy in heading angle, which is smaller than the beam width, adequate for the application.

3.8.3 Correcting rigid body motion for four degrees of freedom only

Since the intended use of the algorithm is to georeference an already levelled radar, we can reduce the complexity of the process by correcting for four degrees of freedom only, instead of six degrees of freedom. By reducing the complexity we speed up convergence and also enhance the quality of the final fit. During rigid body correction for six degrees of freedom we construct a 3×3 rotation matrix

$$\boldsymbol{R} = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2\left(q_1 q_2 - q_0 q_3\right) & 2\left(q_1 q_3 + q_0 q_2\right) \\ 2\left(q_1 q_2 + q_0 q_3\right) & q_0^2 + q_2^2 - q_1^2 - q_3^2 & 2\left(q_2 q_3 - q_0 q_1\right) \\ 2\left(q_1 q_3 - q_0 q_2\right) & 2\left(q_2 q_3 + q_0 q_1\right) & q_0^2 + q_3^2 - q_1^2 - q_2^2 \end{bmatrix}$$

$$\boldsymbol{R} = \begin{bmatrix} (\cos\phi\cos\psi) & (-\cos\phi\sin\psi) & (\sin\phi) \\ (\sin\theta\sin\phi\cos\psi + \cos\theta\sin\psi) & (\sin\theta\sin\phi\sin\psi + \cos\theta\cos\psi) & (-\sin\theta\cos\phi) \\ (-\cos\theta\sin\phi\cos\psi + \sin\theta\sin\psi) & (\cos\theta\sin\phi\sin\psi + \sin\theta\cos\psi) & (\cos\theta\cos\phi) \end{bmatrix}.$$

See section 3.3 for more detail on the matrix construction. Since we only want to correct the heading angle we define

$$\psi = \text{arctan2}(-R_{12}, R_{11}) = \text{arctan2}(\cos\phi\sin\psi, \cos\phi\cos\psi).$$

Since we assume the pitch and roll angles to be zero, we let $\theta = 0°$ and $\phi = 0°$. From heading angle ψ construct new rotation matrix

$$\boldsymbol{R}_z = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Matrix $\boldsymbol{R}_z$ will now be used to change the heading angle during each iteration.

3.8.4 The full georeferencing scheme

In this section we describe the flow of steps leading to final alignment. Some user interaction is required, and depending on the template map type, the alignment accuracy and the number of iterations, the flow of the procedure will be different. The full alignment scheme takes into account that the template map may be either a DTM or a georeferenced SM. A chart which describes the flow process is presented below in Figure 3.37.

The alignment is started by choosing a georeferenced template map with which to align. Surface resampling is done only if a surfaced DTM model map is chosen. A prealignment fit will be attempted to provide the initial alignment. The scheme will only continue if the root mean square value of the spatial difference between the two models is below a certain tolerance value. If the spatial difference is above that tolerance value, then it is unlikely that the iterative alignment will converge correctly and so the procedure is aborted.

If the prealignment is within tolerance, the procedure continues. Once again we consider if a surfaced DTM was used for the alignment. If yes, then the DTM is again resampled, this time with a finer grid spacing upon which we proceed with the iterative correspondence alignment. The correspondence alignment will be iterated until the spatial difference converges or until the maximum allowed number of iterations is reached.

After the iterative correspondence alignment the RMS spatial difference is evaluated. If the spatial difference is greater than the allowed tolerance then the complete alignment is considered unsuccessful, and the user will have to georeference the radar with the manual georeferencing method.

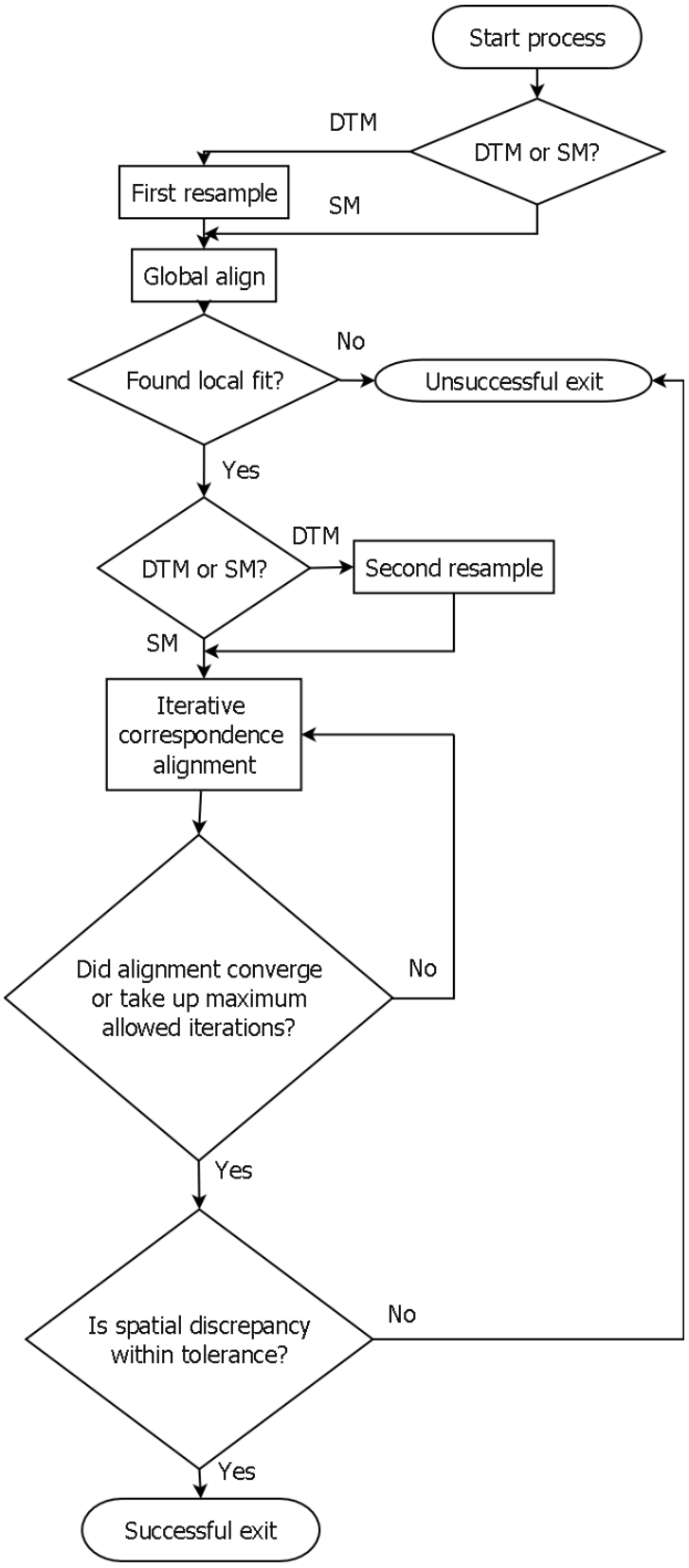


Figure 3.37: A flowchart describing the alignment process. All scenarios are considered.

3.8.5 Convergence and successful alignment

Convergence is reevaluated at each iteration of correspondence alignment. Models are considered converged when the difference in RMS spatial difference rms_k between two continuous iterations is less then a predefined constant, where k is the iteration number. Alignment continues while

$$rms_{k+1} - rms_k < s_{tolerance}$$

where $s_{tolerance}$ is predefined tolerance constant. The second convergence condition is when the maximum number of iterations $max_{iterations}$ is reached. After iterative correspondence alignment is stopped, we evaluate if the alignment was a success. The alignment is considered successful when the RMS spatial difference of the models falls below a certain threshold. The complete system tests performed (Chapter 5) indicate that in general the scheme will align the models well enough for the radar to be georeferenced within our stipulated accuracy specification.

3.8.6 Conclusion

A full alignment scheme was described which can be applied to model maps which are either surfaced or unsurfaced. The full scheme was explained as a flow of processes. Iterative correspondence alignment is applied directly after the prealignment using the proposed global alignment algorithm. We also defined alignment convergence. The next Chapter describes the implementation of the design.

Chapter 4

System implementation

4.1 Introduction

As a requirement of the study, two software applications were developed. One application is to be used for SM on SM alignment and another for SM on DTM alignment. Details of the software implementation are described in this chapter. We first explain the implementation methodology and then the implementation steps. Design choices explained include such things as development tools, data structures and methods of testing. The software is broken down and explained as a flow of functions.

4.2 Development methodology

Reutech coding standards were used as a guideline for the implementation. This allowed for consistent layout and consistent variable and function naming. The classic waterfall development model is used. This model was chosen because there was only one person doing development. Also the waterfall model is the most natural to the development process. The waterfall development cycle has the following steps:

- Requirements specification
- Design
- Construction
- Integration
- Testing

4.3 Requirement specification

For the scope of the project a piece of software is required that satisfies the following requirements:

- The software must be cross-platform, compatible with at least Microsoft Windows XP and Red-Hat Linux 9 on a x-86 target machine.
- The real-time execution time of the software must not exceed a few minutes.
- The software must be easily integratable with programming languages GNU C++ and Delphi 7.
- The software should only use standard or cross-platform libraries.
- The software must be executable from the command line without a graphical user interface.
- The software must handle DTM and SM files with a great number of points, in the order of hundreds of thousands.

It was decided to split the software into two parts to handle the alignment of DTMs and SMs as template sets separately. Two executable applications were developed. The one aligns a SM to a DTM. Another application aligns a SM to an already georeferenced SM. In the subsections to follow we describe the differences between these two alignment procedures.

4.3.1 SM on DTM alignment

The software takes as input the DTM template map and the SM model. The other input parameters are:

1. The number of random match points used during the global fit. The default is 150 points.
2. The centre point tolerance in metres. The default is 50 metres.
3. The azimuth radar angle of the first (upper leftmost) point in the search model. The default is 0 degrees.
4. The rough resampling grid spacing which will be used during the global fit. The default is 7 metres.
5. The fine resampling grid spacing which will be used during the iterative correspondence alignment. The default is 3 metres.

6. The maximum number of correspondence alignment iterations. The default is 100 times.

7. The estimated radar position.

Upon successful alignment, the newly calculated heading and absolute position is printed to screen.

4.3.2 SM on SM alignment

The software takes as input the georeferenced SM template model and the SM search model. The other input parameters are:

1. The number of random match points used during the global fit. The default is 150 points.

2. The centre point tolerance in metres. The default is 50 metres.

3. The azimuth radar angle of the first (upper leftmost) point in the search model. The default is 0 degrees.

4. The maximum number of correspondence alignment iterations. The default is 100 times.

Upon successful alignment, the newly calculated heading and absolute position is printed to screen.

4.4 Design

In this section we consider several design choices. They include:

- Development tools
- Extra library
- Data structures
- Software architecture

Development tools

Considering the development restrictions mentioned in 4.3, we exclude the following development tools:

- Any interpreted language such as Python, Ruby or Java because run-time execution is too slow.
- Languages only compatible in Microsoft Windows such as C#.

The GNU C++ programming language provides the following:

- It uses standard libraries available on most Linux distributions including Red-Hat Linux 9.
- It can be compiled under Microsoft Windows using free compilers such as MinGW (Microsoft GNU for Windows) or Cygwin.
- It can be compiled under Microsoft Windows into a dynamically linked library (DLL) for integration with already existing Delphi 7 software.
- It includes the standard template library (STL) which provides generic handling of abstract data types. This library includes functions such as sorts and searches which will shorten development time.
- The developer is most skilled in this programming language.
- It provides faster execution time compared to an interpreted language such as Java [28].

GNU C++ was ultimately chosen as the compiler of choice, given the above mentioned points. The software can be built for a Microsoft Windows XP platform using version 3.4.2 of the MinGW (Microsoft GNU for Windows) C++ compiler. For the Linux platform, the software can be built using version 3.2.2 of the GNU C++ compiler. The software has not been tested on other versions of the compilers, but it is very likely that it will still be compatible.

The build utility *make*, is used for the compilation and linking of the source into executables. Two separate make files were used to build the software. One for building for Microsoft Windows XP platform, and another for the Linux platform.

Extra libraries

A third-party C++ kd-tree library written by Guy Shechter [29] is used in the software. The library is statically linked after compilation. The library compiles for both Windows and Linux. The license states that the library may be used freely as long as a copyright notice is placed in the source and binary form of the software.

Data structures

The data sets used for the study contain a large number of points and need to be handled adequately. When map files are loaded at start up, the following apply:

- The maps will be loaded into memory only once.
- Points will be inserted into the data structure sequentially without removal.
- The number of points in the file is known prior to loading it.
- The maps will sometimes have to be randomly accessed during processing.
- Maps might need to be displayed graphically and therefore must be randomly accessible in constant time.

Given the properties just mentioned, an array structure is the best design choice for storing the maps in memory for the following reasons:

- It provides random access in constant time.
- It provides the most optimised loading time when the number of elements are known.

The C++ STL provides such an array structure, in the form of a vector template. This data structure will be used throughout the source code.

Software Architecture

The software is split into several source files, namely two application source files and a common library source file. The two application source files are ultimately compiled into executables. The common library module contains all the algorithmic functions needed for the study. The software consists out of the following source files (including the kd-tree library):

Table 4.1: The software implementation has the following source files.

C++ source file	C++ header file
CommonLib.cpp	CommonLib.h
sm2sm.cpp	–
sm2dtm.cpp	–
kdtree_common.cc	kdtree_common.h

4.5 Construction

All the functions coded are original, except those already included in the standard libraries or the kd-tree library. The functions were placed in a common library file. Additional to the library file, two applications were developed which both share the common set of processing functions (Figure 4.1). The application used for SM on SM alignment is called "sm2sm" and the application used for SM on DTM alignment called "sm2dtm".

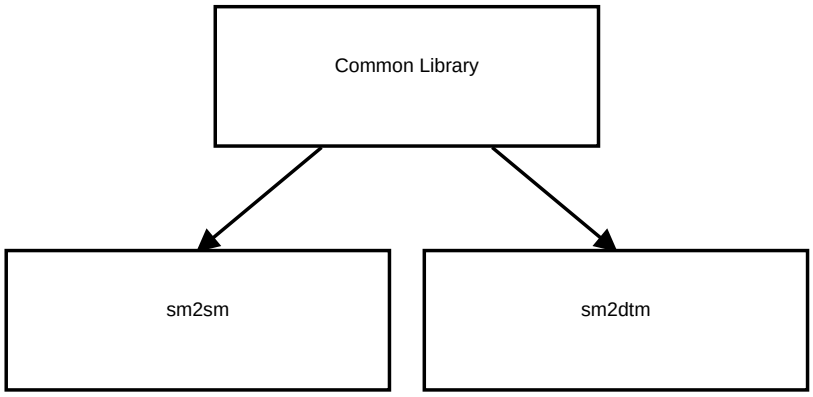


Figure 4.1: The two applications needed for the study share a common set of functions.

The common library file contains all the functions used by both the applications. The common library is compiled into an object file and linked into the executable application during compilations. The functions are defined in a header file and included in the executable source. The functions in the common library are for calculating:

- The determinant of a 3×3 matrix.
- The inverse of a 3×3 matrix.
- The result of adding a scalar to a $N \times 3$ matrix.
- The result of subtracting a scalar to a $N \times 3$ matrix.
- The transpose of a 4×4 matrix.

- The norm of a 4×1 matrix.
- The dot product of a 4×1 matrix.
- The mean of a $N \times 3$ matrix.
- The cross covariance matrix of a $N \times 3$ matrix.
- A 4×4 rotation matrix using Euler angles.
- The QR decomposition of 4×4 matrix.

Other functions which uses the above listed mathematical functions were also present. They are:

- Resample surfaced data set.
- Read drawing exchange format (DXF) file into standard template library (STL) vector.
- Save STL vector to comma separated variable (CSV) file.
- Read exported SM points into STL vector.
- Process global alignment between STL vector sets.
- Process ICP alignment between STL vector sets.
- Do RMS comparison between STL vector sets.

Application sm2sm

Sm2sm is the application used for SM on SM alignment. The two main input parameters of the application is the template model in the form of a SM and the search model, also in the form of a SM. The program has the following flowchart:

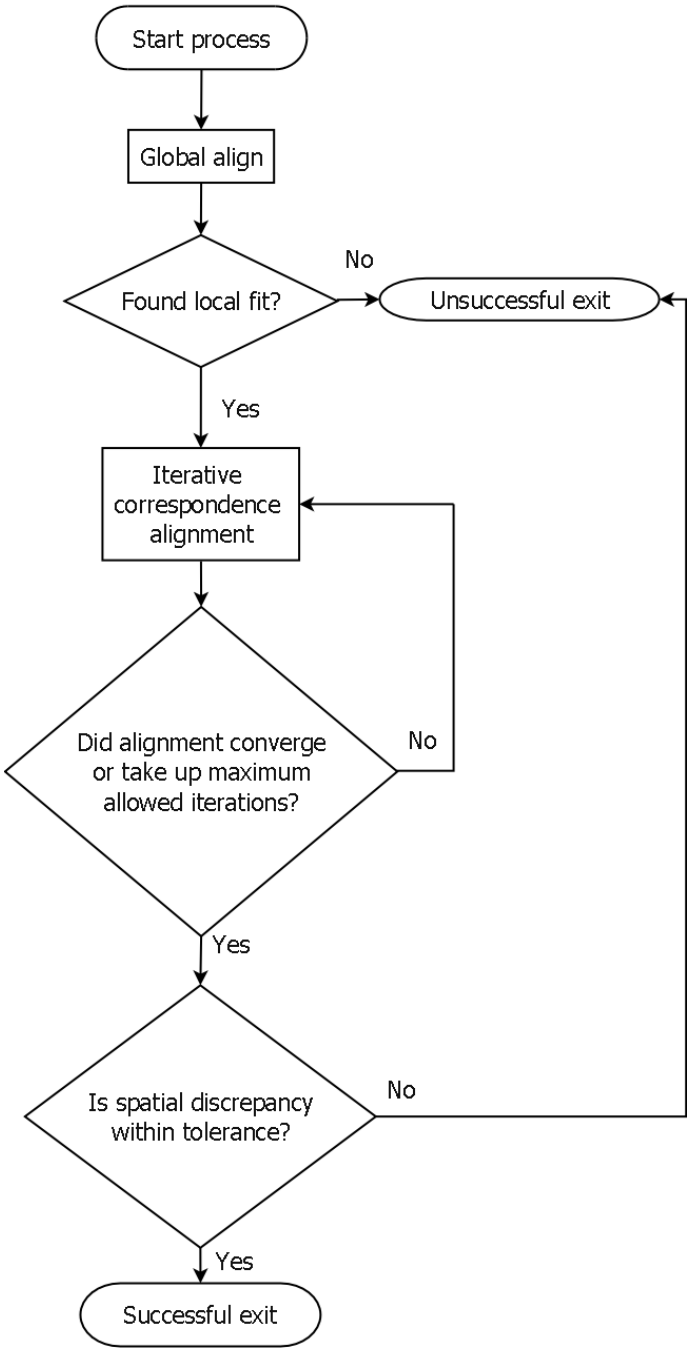


Figure 4.2: The flowchart of the sm2sm application used for SM on SM alignment.

The sm2sm executable takes the following parameters from the command line:

- The template model in the form of a SM (gis) file.

- The search model in the form of a SM (gis) file.
- The number of match points to be used during global alignment.
- The centre tolerance used during global alignment, specified in metres.
- The azimuth reference angle of the template model, specified in degrees.
- The maximum number of ICP iterations used during ICA alignment.

To build the program in Windows, the following should be typed in the command line:

```
make -f Makefile.win
```

To build the program in Linux, the following should be typed in the command line:

```
make -f Makefile.linux
```

Application sm2dtm

Sm2dtm is the application used for SM on DTM alignment. The two main input parameters of the application are the template model in the form of a DTM and the search model in the form of a SM. The application has the following flowchart:

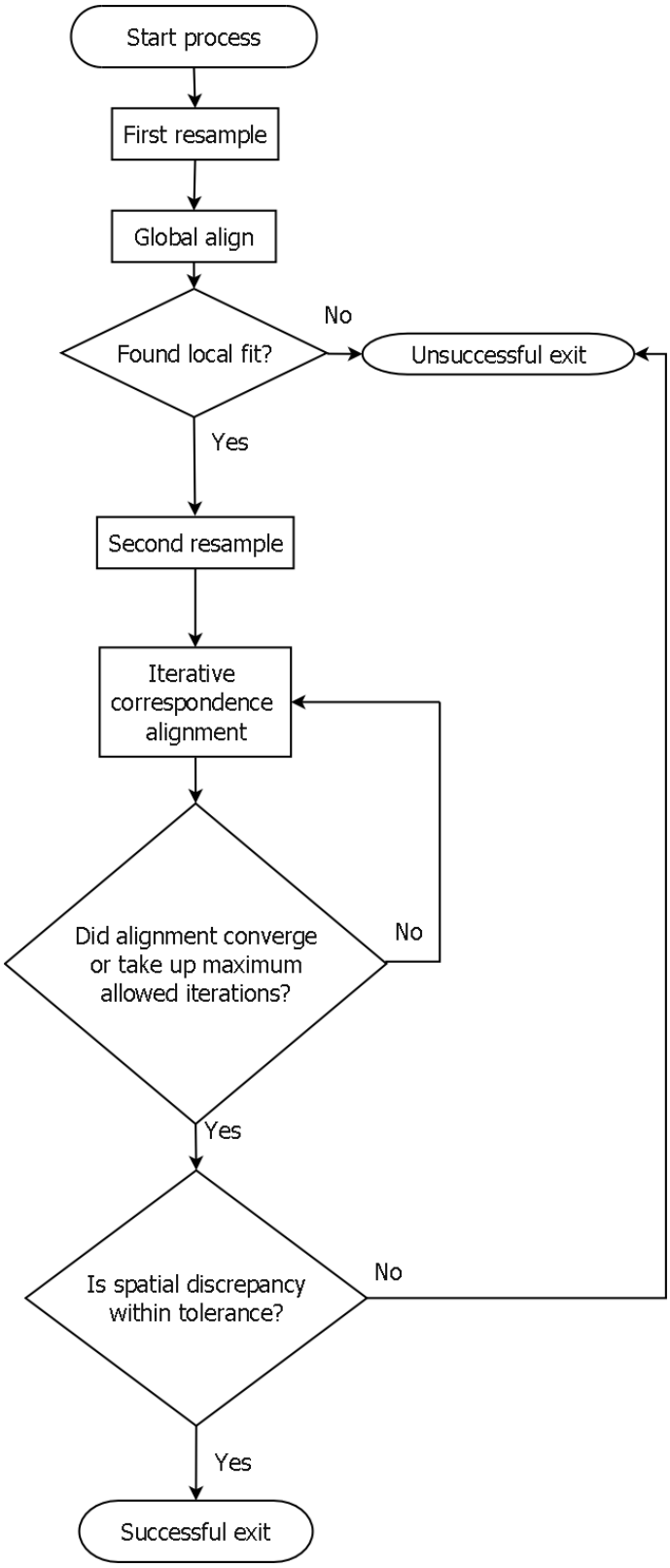


Figure 4.3: The flowchart of the sm2dtm application used for SM on DTM alignment.

The sm2dtm executable takes the following parameters from the command line:

- The template model in the form of a DTM (dxf) file.
- The search model in the form of a SM (gis) file.
- The number of match points to be used during global alignment.
- The centre tolerance used during global alignment, specified in metres.
- The azimuth reference angle of the template model, specified in degrees.
- The global alignment resample grid spacing, specified in metres.
- The ICA alignment resample grid spacing, specified in metres.
- The maximum number of ICP iterations used during ICA alignment.
- The x coordinate of the estimated centre point, specified in metres.
- The y coordinate of the estimated centre point, specified in metres.
- The z coordinate of the estimated centre point, specified in metres.

To build the program in Windows, the following should be typed in the command line:

```
make -f Makefile.win
```

To build the program in Linux, the following should be typed in the command line:

```
make -f Makefile.linux
```

Application MapDisplay

A third application was also developed. This application was not a requirement of the study. It is used to graphically display the sets either before or after alignment. Models are gendered using the OpenGL library as well as the OpenGL utility toolkit (Figure 4.4). The application takes as inputs the template set and the data set. Models are displayed as wireframes. Surfaced models are also displayed using resampled points.

4.6 Integration

Existing GNU C++ software under Red-Hat Linux 9

Integration into existing GNU C++ software requires minimal provision since both are written in GNU C++. The library with the functions is compiled into an object file and linked into the existing software.

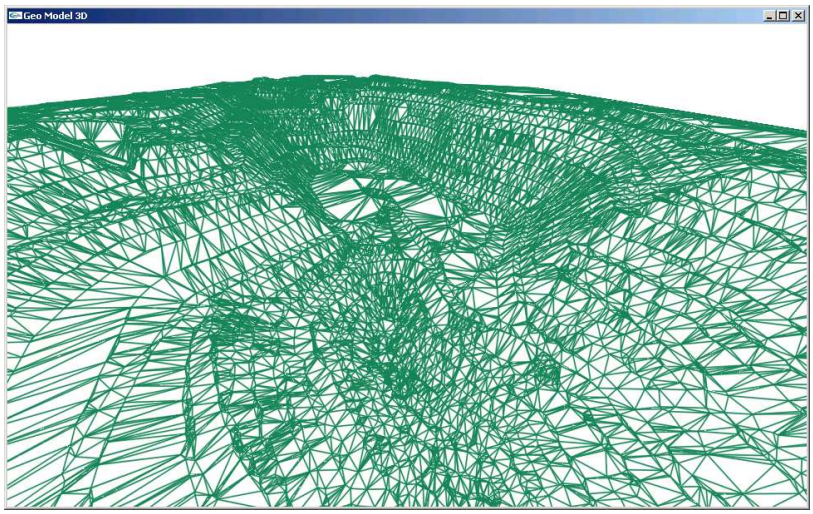


Figure 4.4: MapDisplay, an OpenGL viewer application used for viewing models.

Existing Delphi 7 software under Microsoft Windows

Integration into Delphi 7 software require a few more provisions:

- No operating specific calls must be used since this will prevent cross-platform compatibility.
- Software is compiled into a dynamic library which can then be linked to the Delphi software. This compilation can be done using free Windows C++ compilers such as MinGW and Cygwin.

4.7 Testing

In this section we describe the procedure for the testing of the software. Different levels of testing are used to ensure overall robustness and consistency.

Unit testing

Unit testing verifies the functionality of specific sections of code at the functional level. The unit tests form the bulk of the tests that are done. The mathematical functions listed in Section 4.5 were unit tested during development. The unit tests are executed in the command line as a special input parameter. Typing “sm2sm test” or “sm2dtm test” will start the unit tests. The list of functions testes are printed to the screen. The unit test will stop if one the functions fail. Math functions are tested to have the same results as functions executed using the MATLAB programming language.

Table 4.2: A list of batch scripts used for system testing.

Test	Batch script
A	AlignRun_A_Full.bat
B	AlignRun_B_Full.bat
C	AlignRun_C_Full.bat
D	AlignRun_D_Full.bat
E	DeployRun_E_Full.bat
F	DeployRun_F_Full.bat
G	DeployRun_G_Full.bat

System testing

The complete system was tested by doing alignment tests on data for which the correct alignment was known. These tests are described in Chapter 5 as part of the system evaluation. During development unrealistic results normally indicated faults in the code. The tests were automated using batch scripts, executed on Microsoft Windows XP platform. Table 4.2 lists the batch scripts used for executing the tests. The system was also tested by inspecting the alignment visually using the OpenGl MapDisplay viewer application. Faults in the code were sometimes detected in the cases where models rendered abnormally.

Regression testing

Regression testing is done by periodically re-running the unit and system tests previously described.

4.8 Conclusion

The waterfall model of development was applied to the software required for the study. All the specified requirements were met. Two pieces of software of were designed, constructed and tested. The outcome was predictable development time and minimal code fixes.

Chapter 5

System testing

5.1 Introduction

The georeferencing scheme is tested in two parts. The first set of tests are done in order to evaluate the proposed global alignment algorithm in isolation. The second set of tests are done to test the georeferencing system as a whole.

5.2 Global alignment tests

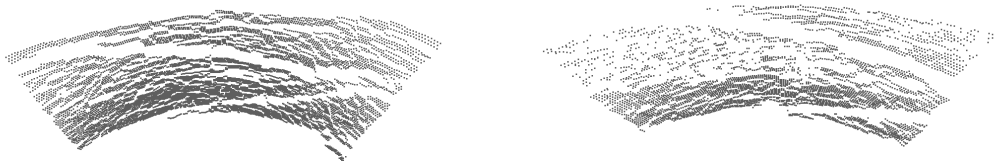
Alignment tests are performed to verify the accuracy of the prealignment algorithm. Seven tests are done. Note that these are seven new tests, not to be confused with the data sets used in section 3.2.3. The tests are performed on typical data sets. After each test the spatial difference is calculated to verify the quality of alignment. We also measure the error in heading as well as the resultant difference between the calculated and actual radar position. The actual radar position is determined using the standard method of georeferencing using a triangulated resectioning method as described in Chapter 1. The user estimated centre is 20 metres away from the actual radar centre in all test cases. We graph the spatial differences in all cases using histograms. The algorithm tolerance values, for all tests, are set to the following: A radius tolerance $r_{\text{tolerance}}$ of 5 metres. A centre tolerance $c_{\text{tolerance}}$ of 30 metres. A delta height tolerance $z_{\text{tolerance}}$ of 2 metres. N_{D_γ} is set to 150 points for all cases. For the first four tests (A to D) we use unsurfaced SMs as the template sets. For tests E to G we use resampled TIN DTMs as the template sets. Note that comments on the results will only be presented after all the results (Section 5.2.2).

5.2.1 Test results

Test A - SM alignment to SM

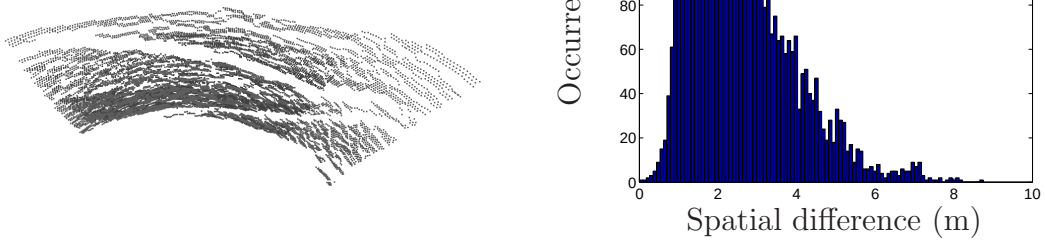
For test A we use a template set (Figure 5.1 (a)) sourced from a SM and a search set (Figure 5.1 (b)) also sourced from a SM. Both sets are very similar in size and as such represent a simple alignment example. The template set is not a surface model and as such is not resampled prior to the alignment. The template set contains 8336 points and the search set 4379. The spatial bounds for the template set in the Cartesian axes are $\triangle x = 230$m, $\triangle y = 559$m and $\triangle z = 88$m. The spatial bounds for the search set in the Cartesian axes are $\triangle x = 140$m, $\triangle y = 411$m and $\triangle z = 72$m.

Figure 5.2 (a) shows the models after alignment. Figure 5.2 (b) shows the spatial difference as a histogram.



(a) (b)

Figure 5.1: Test A (a) The template set prior to alignment. (b) The search set prior to alignment.



(a) (b)

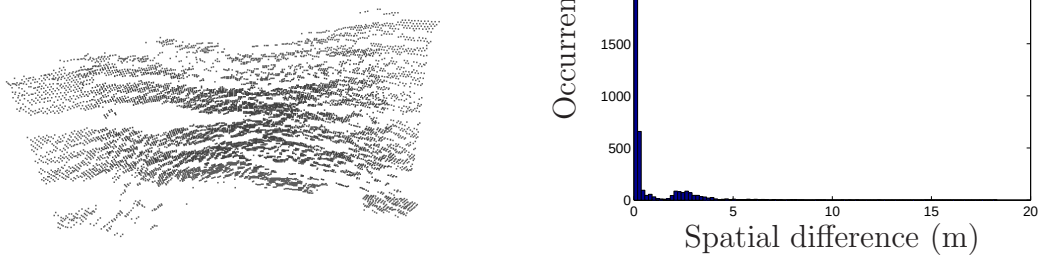
Figure 5.2: Test A (a) Both sets after alignment. (b) Spatial difference histogram.

Test B - SM alignment to SM

For test B we again have a template set and a search set both sourced from SMs. No resampling is applied given that the template set is not a surfaced model. The template set contains 7215 points and the search set 4276. The spatial bounds for the template set in the Cartesian axes are $\triangle x = 388$m, $\triangle y = 783$m and $\triangle z = 163$m. The spatial bounds for the search set in the Cartesian axes are $\triangle x = 243$m, $\triangle y = 564$m and $\triangle z = 90$m.

(a) (b)

Figure 5.3: Test B (a) The template set prior to alignment. (b) The search set prior to alignment.



(a) (b)

Figure 5.4: Test B (a) Both sets after alignment. (b) Spatial difference histogram.

Figure 5.4 (a) shows the models after alignment. Figure 5.4 (b) shows the spatial difference as a histogram.

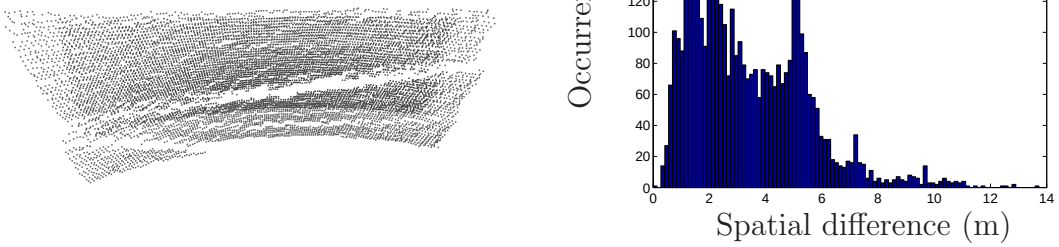
Test C - SM alignment to SM

For test C we again have a template set (Figure 5.5 (a)) and a search set (Figure 5.5 (b)) both sourced from SMs. No resampling is applied given that the template set is not a surfaced

model. The template set contains 6785 points and the search set 4221. The spatial bounds for the template set in the Cartesian axes are $\triangle x = 949$m, $\triangle y = 374$m and $\triangle z = 177$m. The spatial bounds for the search set in the Cartesian axes are $\triangle x = 691$m, $\triangle y = 244$m and $\triangle z = 132$m.

(a) (b)

Figure 5.5: Test C (a) The template set prior to alignment. (b) The search set prior to alignment.



(a) (b)

Figure 5.6: Test C (a) Both sets after alignment. (b) Spatial difference histogram.

Figure 5.6 (a) shows the models after alignment. Figure 5.6 (b) shows the spatial difference as a histogram.

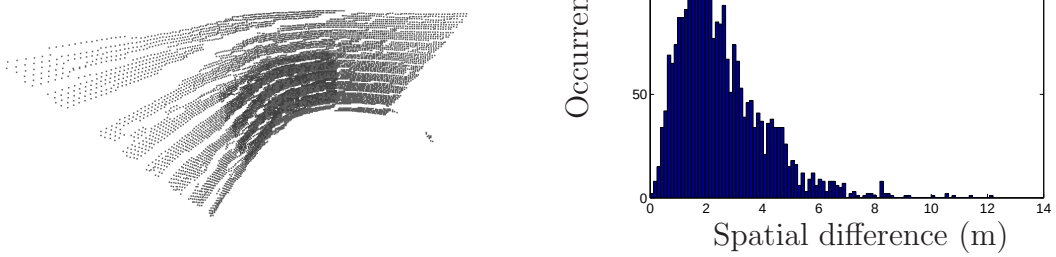
Test D - SM alignment to SM

For test D we again have a template set (Figure 5.7 (a)) and a search set (Figure 5.7 (b)) both sourced from SMs. No resampling is applied given that the template set is not a surfaced model. The template set in this case is much larger than the search set. The template set contains 8834 points and the search set 2788. The spatial bounds for the template set in

the Cartesian axes are $\triangle x = 654$m, $\triangle y = 357$m and $\triangle z = 211$m. The spatial bounds for the search set in the Cartesian axes are $\triangle x = 328$m, $\triangle y = 118$m and $\triangle z = 108$m.

(a) (b)

Figure 5.7: Test D (a) The template set prior to alignment. (b) The search set prior to alignment.



(a) (b)

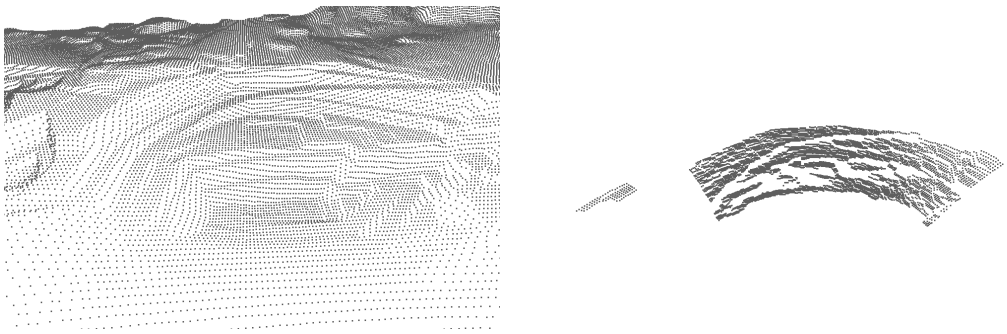
Figure 5.8: Test D (a) Both sets after alignment. (b) Spatial difference histogram.

Figure 5.8 (a) shows the models after alignment. Figure 5.8 (b) shows the spatial difference as a histogram.

Test E - SM alignment to DTM

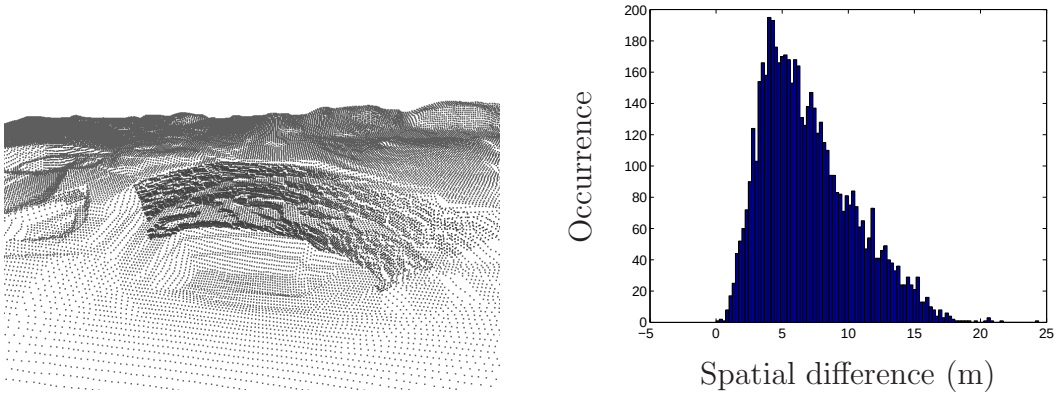
For test E the template set (Figure 5.9 (a)) is sourced from a TIN DTM and the search set (Figure 5.9 (b)) from a SM. The template set is resampled using a 7m grid spacing. The resampling results in evenly dispersed point distribution for the template set. In this case the search set is much smaller in size when compared to the template set. The template set contains 786704 points and the search set 5567. After resampling the template set contains 59830 points. The spatial bounds for the template set in the Cartesian axes are

$\triangle x = 1350$m, $\triangle y = 2175$m and $\triangle z = 336$m. The spatial bounds for the search set in the Cartesian axes are $\triangle x = 330$m, $\triangle y = 1055$m and $\triangle z = 120$m.



(a) (b)

Figure 5.9: Test E (a) The template set prior to alignment. (b) The search set prior to alignment.



(a) (b)

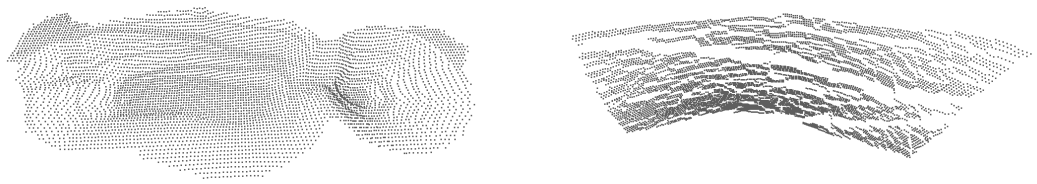
Figure 5.10: Test E (a) Both sets after alignment. (b) Spatial difference histogram.

Figure 5.10 (a) shows the models after alignment. Figure 5.10 (b) shows the spatial difference as a histogram.

Test F - SM alignment to DTM

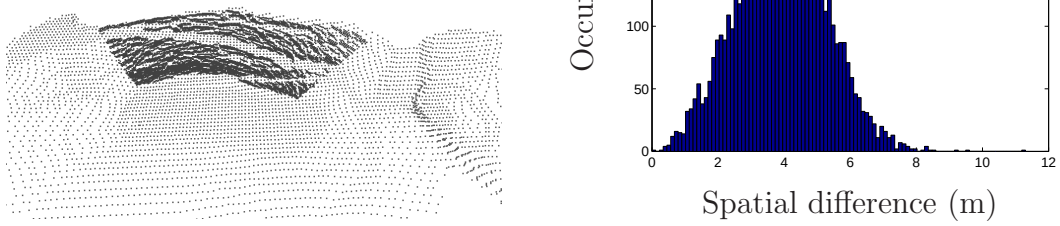
For test F the template set (Figure 5.11 (a)) is again sourced from a TIN DTM and the search set (Figure 5.11 (b)) from a SM. The template set is resampled using a 7m grid spacing. The resampling results in evenly dispersed point distribution for the template set. The template set contains 93652 points and the search set 6125. After resampling the template set contains 5949 points. The spatial bounds for the template set in the Cartesian

axes are $\triangle x = 393$m, $\triangle y = 1056$m and $\triangle z = 112$m. The spatial bounds for the search set in the Cartesian axes are $\triangle x = 180$m, $\triangle y = 524$m and $\triangle z = 75$m.



(a) (b)

Figure 5.11: Test F (a) The template set prior to alignment. (b) The search set prior to alignment.



(a) (b)

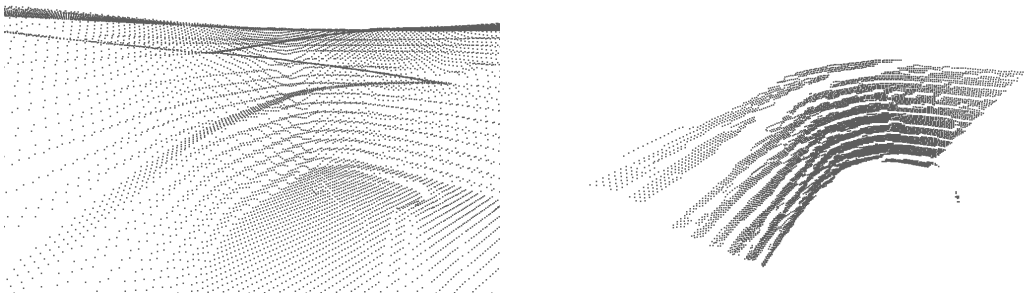
Figure 5.12: Test F (a) Both sets after alignment. (b) Spatial difference histogram.

Figure 5.12 (a) shows the models after alignment. Figure 5.12 (b) shows the spatial difference as a histogram.

Test G

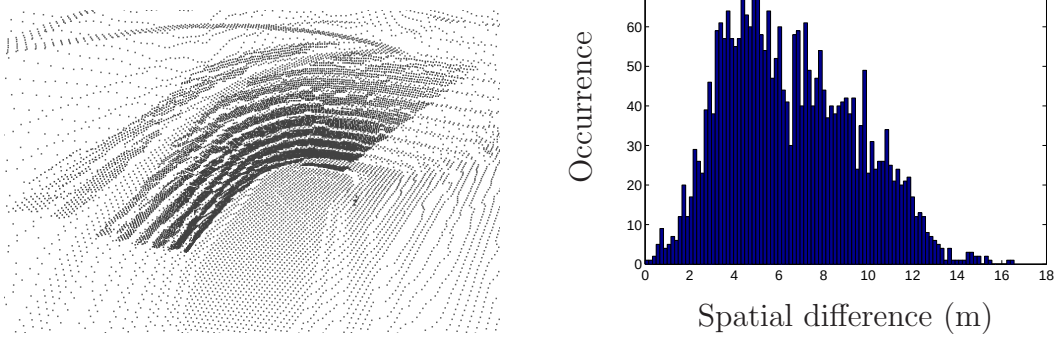
For test G the template set (Figure 5.13 (a)) is sourced from a TIN DTM and the search set (Figure 5.13 (b)) from a SM. The template set is resampled using a 7m grid spacing. The resampling results in evenly dispersed point distribution for the template set. The template set contains 802256 points and the search set 8834. After resampling the template set contains 44766 points. The spatial bounds for the template set in the Cartesian axes are

$\triangle x = 2193$m, $\triangle y = 1211$m and $\triangle z = 366$m. The spatial bounds for the search set in the Cartesian axes are $\triangle x = 654$m, $\triangle y = 357$m and $\triangle z = 211$m.



(a) (b)

Figure 5.13: Test G (a) The template set prior to alignment. (b) The search set prior to alignment.



(a) (b)

Figure 5.14: Test G (a) Both sets after alignment. (b) Spatial difference histogram.

Figure 5.14 (a) shows the models after alignment. Figure 5.14 (b) shows the spatial difference as a histogram.

5.2.2 Test results

The search set aligned to correct local positions for all the tests covered above, tests A to D with unsurfaced SM template sets and tests E to G with resampled DTM template sets. The results of the tests are listed in table 5.1.

Table 5.1: The results of the prealignment tests A to G.

Test	Heading error	Centre position difference	Spatial difference		Run-time
			RMS	Std. deviation	
	(deg)	(m)	(m)	(m)	(sec)
A	1.02°	9.60	2.96	1.33	165
B	0.03°	0.51	1.58	1.41	13
C	0.31°	8.11	3.96	2.04	9
D	1.88°	3.42	2.89	1.49	6
E	4.01°	15.41	7.82	3.49	33
F	1.22°	4.04	4.12	1.34	5
G	2.23°	12.59	7.16	2.92	15

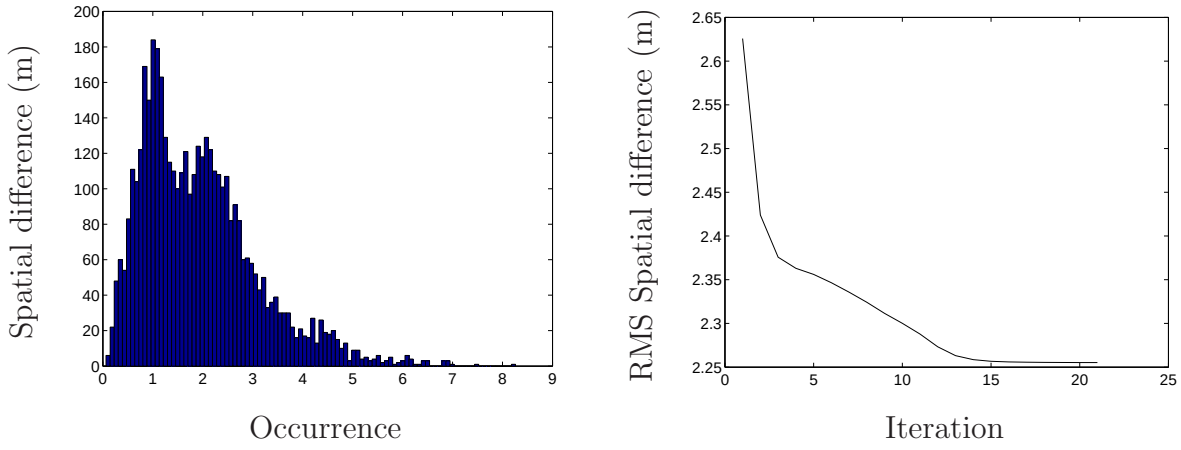
For most of the tests the alignment resulted in a heading error of less than 2 degrees. For the seven tests conducted, the position errors average 7.67 metres. All the tests resulted in RMS spatial difference errors of less than 8 metres. The errors in spatial difference average 4.36 metres. Although the tests done in section 3.2.3 were from different data sets, a comparison can still be made. The spatial errors are similar, indicating that the alignment, for most cases, is converging to the correct local minimum.

5.3 Full alignment tests

5.3.1 Introduction

In this section we repeat the tests from section 5.2. Sets are aligned using the full alignment scheme presented in figure 3.37. For the tests, we verify that the full scheme – as described in this Chapter 3 – meets the georeferencing requirements stipulated in Chapter 1. For all tests we render histograms showing final spatial difference as well as graphs which plot RMS spatial difference versus alignment iteration count. The iteration counts are just for the ICA alignment. Parameters used in the tests include the maximum number of iterations $max_{iterations} = 100$ and the convergence constant $s_{tolerance} = 0.0001$m. Note that comments on the results will only be presented after all the results.

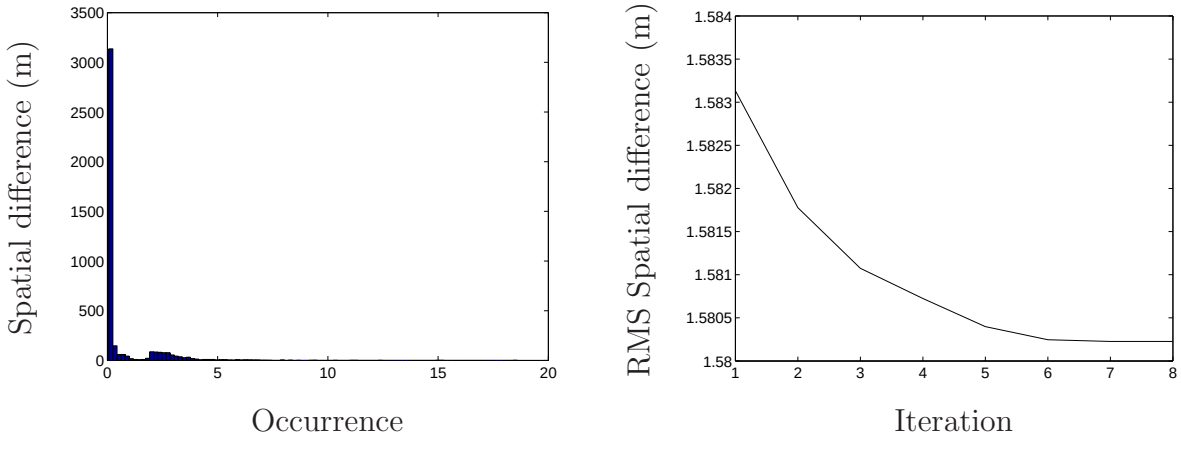
Test A - SM alignment to SM



(a) (b)

Figure 5.15: Test A (a) The spatial differences as a histogram. (b) The RMS spatial difference plotted against iterations.

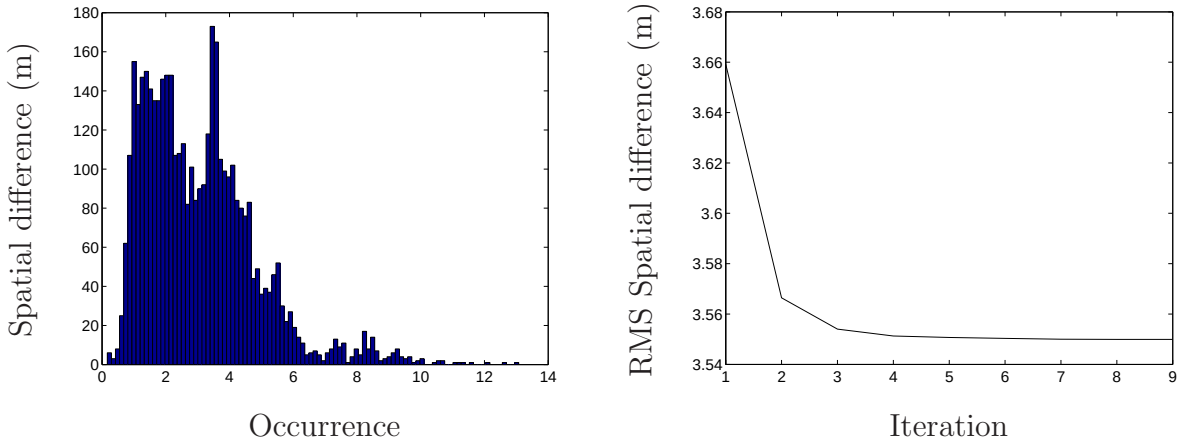
Test B - SM alignment to SM



(a) (b)

Figure 5.16: Test B (a) The spatial differences as a histogram. (b) The RMS spatial difference plotted against iterations.

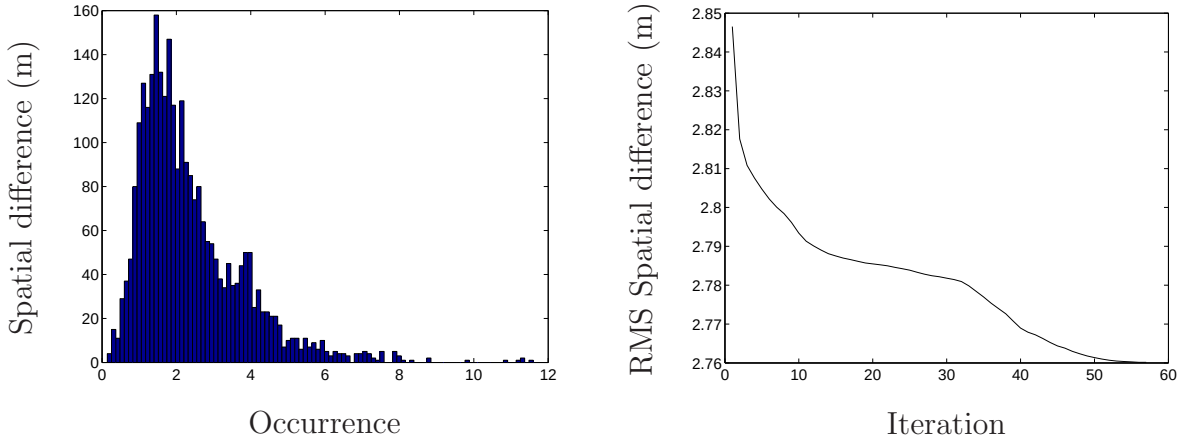
Test C - SM alignment to SM



(a) (b)

Figure 5.17: Test C (a) The spatial differences as a histogram. (b) The RMS spatial difference plotted against iterations.

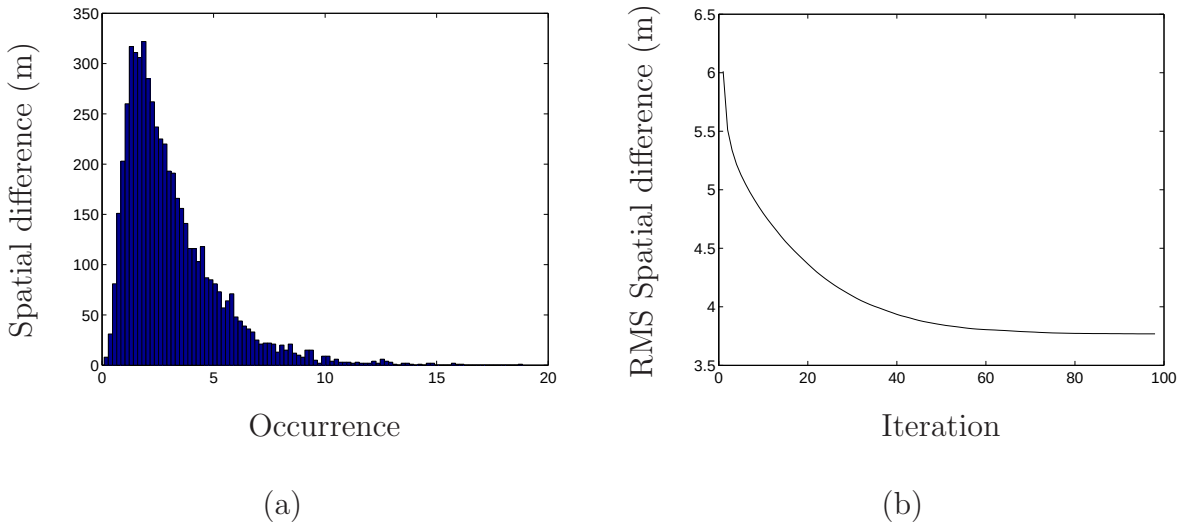
Test D - SM alignment to SM



(a) (b)

Figure 5.18: Test D (a) The spatial differences as a histogram. (b) The RMS spatial difference plotted against iterations.

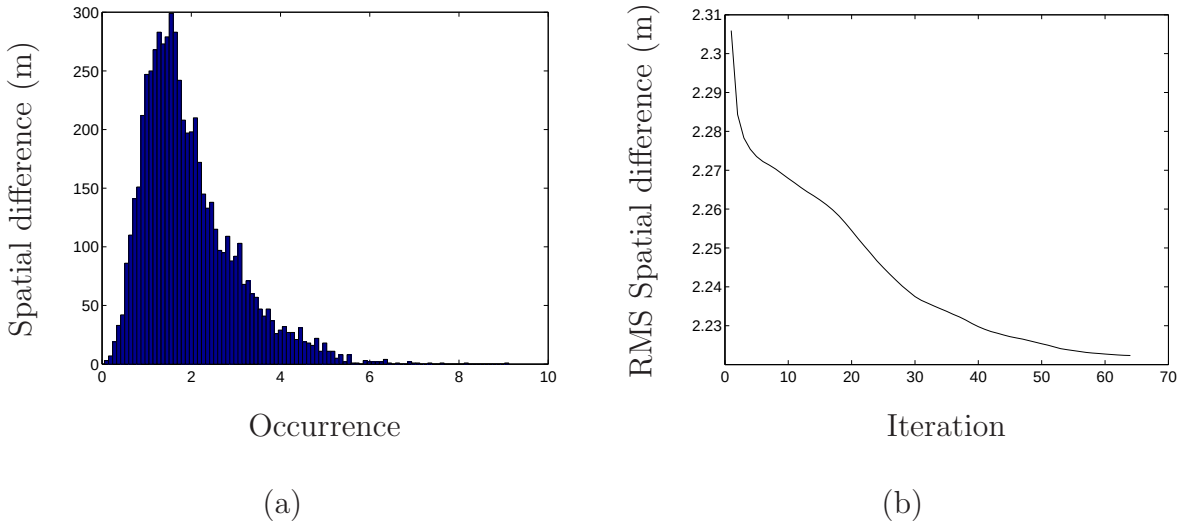
Test E - SM alignment to DTM



(a) (b)

Figure 5.19: Test E (a) The spatial differences as a histogram. (b) The RMS spatial difference plotted against iterations.

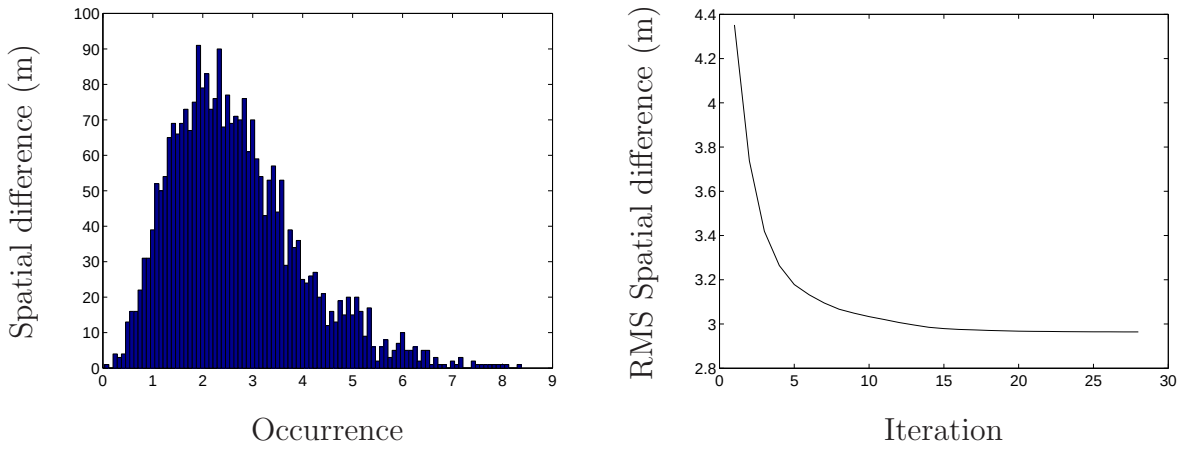
Test F - SM alignment to DTM



(a) (b)

Figure 5.20: Test F (a) The spatial differences as a histogram. (b) The RMS spatial difference plotted against iterations.

Test G - SM alignment to DTM



(a) (b)

Figure 5.21: Test G (a) The spatial differences as a histogram. (b) The RMS spatial difference plotted against iterations.

5.3.2 Test results

The results of the tests indicate that 6 out of 7 tests showed improvement in the quality of alignment compared to the tests done with global alignment only (Table 5.2). Graphs 5.15-5.21 (b) show that iterative correspondence alignment improved the spatial difference in all cases. Table 5.3 shows the percentage of improvement of the heading accuracy as well as the percentage of improvement in the radar centre position: 4 out of 7 tests indicate an improvement in heading accuracy and 6 out of 7 tests indicate an improvement in position accuracy. Tests done on resampled sets (E to G) resulted in a greater improvement in the absolute position error compared to the tests done on unsurfaced sets

Table 5.2: The results for tests using the full alignment scheme. The same data sets from section 5.2 were used.

Test	Iterations	Heading error (deg)	Centre pos. difference (m)	Spatial difference			Run-time (sec)
				RMS (m)	RMS imprv. (m)	Std.dev. (m)	
A	21	0.02°	6.57	2.25	0.71	1.16	195
B	8	0.07°	0.53	1.58	0.00	1.39	21
C	9	0.35°	6.37	3.55	0.41	1.77	17
D	57	1.19°	2.10	2.76	0.13	1.41	51
E	98	0.99°	2.07	3.77	4.05	2.13	636
F	64	1.41°	1.22	2.22	1.9	1.04	120
G	28	0.86°	3.97	2.96	4.2	1.29	62

Table 5.3: The percentage of improvement for the tests compared to the results from section 5.2. The table also shows the final heading and position error as a percentage of the required accuracy specification.

Test	Heading error		Centre position difference	
	Per. improvement (%)	As % of 2 deg. (%)	Per. improvement (%)	As % of 6m (%)
A	98.04	1.00	31.56	109.50
B	-133.33	3.50	-3.92	8.83
C	-12.90	17.50	21.45	106.17
D	36.70	59.50	38.60	35.00
E	75.31	49.50	86.57	34.50
F	-15.57	70.50	70.05	20.17
G	61.43	43.00	68.47	66.17

5.4 Conclusion

The complete design was tested on surfaced and unsurfaced models. The tests performed indicate that we have met the requirements stipulated in Chapter 1. All the tests passed the specified heading requirement. 5 out of 7 tests passed the specified allowable position error. The maximum run-time was 11 minutes, which is within the specified allowable operation time.

Chapter 6

Conclusions and recommendations

6.1 Research results

6.1.1 Project objectives

The first objective of this study was to investigate existing alignment algorithms as possible solutions to our problem. Chapter 2 examined some of the literature covering model alignment and related topics. In the same chapter, the pertinent local alignment algorithms were considered and some chosen for our design. It was established that least squares estimation is a very common method of three-dimensional model alignment, and that it would function well in our design. The literature did, however, also show that a global alignment algorithm would be required for our design. A global alignment algorithm was proposed and described in Chapter 3. The first objective was achieved in full.

The second objective was to develop a complete georeferencing scheme which considers surfaced and unsurfaced models, converges to alignment within minutes of operation time, and provides a heading accuracy of 2 degrees and an absolute position accuracy of 6 metres. As part of the design, a baseline was established for which the expected spatial differences were calculated. From the literature, iterative least squares alignment was chosen as a local alignment algorithm, and shown to work well for that function. A global alignment algorithm was proposed, forming a very important contribution to the study. It is now possible to georeference the radar using an estimated centre position only. Another contribution to the design was a map resampling scheme which conditions maps to ensure that points are distributed evenly. It also provides the ability to control the detail level of the template set, which in turn speeds up the georeferencing process. The ability to resample maps is another important contribution to the georeferencing system. The second objective was achieved in full, to the effect that a full georeferencing system was developed. It can also be noted that the design described can also be applied to other alignment scenarios, and not

only radar alignment. The solution is general enough to be useful for many other alignment applications, such as laser model alignment. Chapter 3 describes this design.

The third objective was to implement a complete georeferencing scheme as a cross-platform application, in a non-interpreted programming language without the use of platform specific third party libraries. Chapter 4 describes the implementation steps. The applications created compile under Microsoft Windows XP and Red Hat Linux, using the same set of source code. Integration with existing software can be done very easily by compiling the source into a dynamically linked library (DLL) file under Windows, or by compiling the source into a dynamic shared object (DSO) file under Linux. This will allow the routines to be linked into existing software dynamically for both GNU C++ and Delphi 7. The software is a very important contribution to the study. It focuses directly on the requirement of georeferencing a mine surveying radar. It was shown that the software is robust and easy to build for the required platform. Its simple, but effective design, makes possible modification easy. The objective was achieved in full.

Through tests conducted in Chapter 5 on surfaced and unsurfaced models, we have shown that the alignment converges within minutes. Furthermore, we have also shown that for all tests the heading error was less than 2 degrees, and for 5 out of the 7 tests, the absolute position error was less than 6 metres. The tests also resulted in the same order of spatial difference as recorded in the baseline results. Tests that preformed poorly are contributed to template sets of insufficient resolution, and to the lack of distinguishable features in the geographical makeup of the sets. Most of the DTMs used in the study were sourced from hand surveyed points. It is possible that the georeferencing accuracy of the system could be maximised when DTMs sourced from laser are used, as they are known to produce very high resolution models. Also, as stated before, ideally, the template set, as well as the search set should contain a sufficient amount of distinguishable geographical features. Sets of cylindrical shape should be avoided, and ideally the search set should cover as wide an area as possible. The objective was achieved to the point where most of the tests passed the stipulated accuracy limit.

6.2 Further work

Despite the careful consideration that went into the research, design and implementation of this georeferencing system, there are some things that could be improved upon:

The proposed global alignment algorithm could be expanded to be more robust against outlier points. Currently the algorithm will align poorly if the search set corner points, happen to be outliers. The algorithm could be modified to use alternate points when the corner points are considered to be outliers. The algorithm can also be improved upon

by making use of an improved random selection criterion. Currently the algorithm picks a number of randomly selected points to evaluate the quality of the fit. This selection could be done more strategically, which in turn could possibly improve the final alignment quality. The proposed global alignment algorithm could also be replaced by a more general alignment solution. A genetic algorithm, trained for alignment is a good example for such a replacement. We discuss one such genetic algorithm in Chapter 2. A more general solution might make it possible to align for six degrees-of-freedom instead of only four. Such a general solution might, however, result in a significant increase in overall run-time.

The process of correspondence is a major factor in the run-time of the local alignment algorithm used in this study (see e.g. [7], [4], [6]). The run-time can be decreased significantly if the correspondence is done more efficiently, given that it is repeated so many times. Local alignment can also be expanded to be more robust by considering partially overlapping data sets.

The algorithms used in this study rely heavily on linear algebra functions. The linear algebra functions used in the software were, however, all written from first principles in C++. Optimising these functions may result in a decrease in alignment time. Otherwise, optimised mathematical libraries can be considered for Windows or Linux respectively. Two sets of source code may be required, if this option is to be investigated.

Graphics processing unit (GPU) parallelisation libraries are now readily available for development. The proposed global alignment algorithm could, in theory, be optimised using GPU parallelisation. The template set, could for instance, be split into several parts, and then for each part a separate process can be instantiated. When processing is completed, each process reports back with its best alignment position. The process which produces the lowest spatial difference will be considered to be optimal.

6.3 Summary

The objectives were achieved, with the project culminating in the successful implementation of a mine surveying radar georeferencing scheme. The core of the scheme is largely applicable to mine surveying radar, although it can be applied to more general alignment problems. In achieving the objectives, the envisioned contributions were also realised.

Firstly, a software implementable alignment scheme was developed which does not require full prealignment, useful for georeferencing a mine surveying radar. Secondly, a resampling scheme was developed useful in controlling the number of points used to represent a three-dimensional model and for obtaining the maximum accuracy out of surfaced models through surface resampling.

Bibliography

[1] REUTECH RADAR SYSTEMS, "Private communications", 2008-2010.

[2] M.I. Skolnik, *Introduction to Radar Systems*, McGraw-Hill, Tokyo, international student edition edition, 1962.

[3] S. Kingsley and S. Quegan, *Understanding Radar Systems*, SciTech, Mendham, 1999.

[4] Zhengyou Zhang, "Iterative point matching for registration of free-form curves and surfaces", *International Journal of Computer Vision*, vol. 13, no. 2, pp. 119–152, October 1994.

[5] K. S. Arun, T. S. Huang, and S. D. Blostein, "Least-squares fitting of two 3-d point sets", *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 9, no. 5, pp. 698–700, September 1987.

[6] Y. Chen and G. Medioni, "Object modeling by registration of multiple range images", 1991, pp. 2724–2729 vol.3.

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

[8] L. Lucchese, G.M. Cortelazzo, and C. Monti, "High resolution estimation of planar rotations based on fourier transform and radial projections", in *Circuits and Systems, 1997. ISCAS '97., Proceedings of 1997 IEEE International Symposium on*, 9-12 1997, vol. 2, pp. 1181 –1184 vol.2.

[9] R. Giannitrapani and V. Murino, "Three-dimensional skeleton extraction by point set contraction", in *Image Processing, 1999. ICIP 99. Proceedings. 1999 International Conference on*, 1999, vol. 1, pp. 565 –569 vol.1.

[10] A. E. Johnson, *Spin-images: A representation for 3-d surface matching*, PhD thesis, Carnegie Mellon University, 1997.

[11] T. Masuda and N. Yokoya, "A robust method for registration and segmentation of multiple range images", 1994, pp. 106–113.

[12] D. Chetverikov, D. Svirko, D. Stepanov, and P. Krsek, "The trimmed iterative closest point algorithm", in *In International Conference on Pattern Recognition*, 2002, pp. 545–548.

[13] M. W. Walker, L. Shao, and R. A. Volz, "Estimating 3-d location parameters using dual number quaternions", *CVGIP: Image Underst.*, vol. 54, no. 3, pp. 358–367, 1991.

[14] B. K. P. Horn, "Closed-form solution of absolute orientation using unit quaternions", *Journal of the Optical Society of America A*, vol. 4, no. 4, pp. 629–642, 1987.

[15] O. D. Faugeras and M. Hebert, "A 3-d recognition and positioning algorithm using geometrical matching between primitive surfaces", in *IJCAI'83: Proceedings of the Eighth international joint conference on Artificial intelligence*, San Francisco, CA, USA, 1983, pp. 996–1002, Morgan Kaufmann Publishers Inc.

[16] V. Murino, L. Ronchetti, U. Castellani, and A. Fusiello, "Reconstruction of complex environments by robust pre-aligned icp", in *In Proc. Third Int. Conf. on 3-D Digital Imaging and Modeling*, 2001, pp. 187–194.

[17] L. Lucchese, G. Doretto, and G. M. Cortelazzo, "A frequency domain technique for range data registration", *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 11, pp. 1468–1484, November 2002.

[18] A. E. Johnson and M. Hebert, "Surface matching for object recognition in complex 3-d scenes", *Image and Vision Computing*, vol. 16, pp. 635–651, 1998.

[19] A. E. Johnson and M. Hebert, "Using spin images for efficient object recognition in cluttered 3d scenes", *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 21, no. 5, pp. 433–449, 1999.

[20] Bellon Olga R.P. Boyer Kim L. Gotardo Paulo F. U. Silva, Luciano, "Low-overlap range image registration for archaeological applications", June 2003, vol. 1, pp. 9 –9.

[21] Zhilin Li, Qing Zhu, and Chris Gold, *Digital Terrain Modeling: Principles and Methodology*, CRC Press, 2005.

[22] A. Gruen and D. Akca, "Least squares 3d surface and curve matching", *ISPRS Journal of photogrammetry and remote sensing*, vol. 59, pp. 151–174, 2005.

[23] J. Lee, "Comparison of existing methods for building triangular irregular network, models of terrain from grid digital elevation models", *International journal of geographical information systems*, vol. 5, no. 3, pp. 267–285, 1991.

[24] P. Fernandes, P. Girdinio, M. Repetto, and G. Secondo, "Refinement strategies in adaptive meshing", *Magnetics, IEEE Transactions on*, vol. 28, no. 2, pp. 1739 –1742, mar. 1992.

[25] Leonidas Guibas and Jorge Stolfi, "Primitives for the manipulation of general subdivisions and the computation of voronoi", *ACM Trans. Graph.*, vol. 4, no. 2, pp. 74–123, April 1985.

[26] S. Fortune, "A sweepline algorithm for voronoi diagrams", in *SCG '86: Proceedings of the second annual symposium on Computational geometry*, New York, NY, USA, 1986, pp. 313–322, ACM Press.

[27] P. Su and R. Drysdale, "A comparison of sequential delaunay triangulation algorithms", *Computational Geometry*, vol. 7, no. 5-6, pp. 361 – 385, 1997, 11th ACM Symposium on Computational Geometry.

[28] L. Prechelt, "Comparing java vs. c/c++ efficiency differences to inter-personal differences", 1999.

[29] G. Shechter, "http://www.mathworks.com/matlabcentral/fileexchange/4586", 2010.

[30] D. Lay, *Linear algebra and its applications*, Pearson Education, Inc., United States of America, third international edition edition, 2003.

[31] J. L. Bentley, "Multidimensional binary search trees used for associative searching", *Commun. ACM*, vol. 18, no. 9, pp. 509–517, September 1975.

[32] David Simon, *Fast and Accurate Shape-Based Registration*, PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, December 1996.

Appendix A

Gram–Schmidt Jacobi iterative method

A.1 Introduction

A general way to calculate the principal eigenvector of a symmetric square matrix is by using singular value decomposition (SVD). The eigenvectors are contained within the column vectors of unitary matrix $\boldsymbol{U}$, when $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T$ and $\boldsymbol{A} = \boldsymbol{A}^T$ [30]. However, to implement a singular value decomposition algorithm from first principles is unnecessary when the principal eigenvector is the only required unknown. Presented here – as shown in [30] – is an alternate method for calculating the principal eigenvector, by using an iterative Gram–Schmidt Jacobi process, useful for when the subject matrix symmetric and of size $N \times N$.

A.2 The proposed method

The method is repeated iteratively. For each iteration of the Gram–Schmidt process an orthonormal basis is constructed. Our requirement is for three dimensions, but the process can be applied easily for N dimensions. Given a $N \times N$ subject matrix $\boldsymbol{A} = \begin{bmatrix} \boldsymbol{x}_1 & \cdots & \boldsymbol{x}_p \end{bmatrix}$, construct column vectors given by

$$\boldsymbol{v}_1 = \boldsymbol{x}_{1,} \tag{A.1}$$

$$\boldsymbol{v}_2 = \boldsymbol{x}_2 - \frac{\boldsymbol{x}_2 \cdot \boldsymbol{v}_1}{\boldsymbol{v}_1 \cdot \boldsymbol{v}_1}\boldsymbol{v}_1, \tag{A.2}$$

$$\boldsymbol{v}_3 = \boldsymbol{x}_3 - \frac{\boldsymbol{x}_3 \cdot \boldsymbol{v}_1}{\boldsymbol{v}_1 \cdot \boldsymbol{v}_1}\boldsymbol{v}_1 - \frac{\boldsymbol{x}_3 \cdot \boldsymbol{v}_2}{\boldsymbol{v}_2 \cdot \boldsymbol{v}_2}\boldsymbol{v}_2 \quad \text{and} \tag{A.3}$$

$$\boldsymbol{v}_p = \boldsymbol{x}_p - \frac{\boldsymbol{x}_p \cdot \boldsymbol{v}_1}{\boldsymbol{v}_1 \cdot \boldsymbol{v}_1}\boldsymbol{v}_1 - \frac{\boldsymbol{x}_p \cdot \boldsymbol{v}_2}{\boldsymbol{v}_2 \cdot \boldsymbol{v}_2}\boldsymbol{v}_2 - \ldots - \frac{\boldsymbol{x}_p \cdot \boldsymbol{v}_{p-1}}{\boldsymbol{v}_{p-1} \cdot \boldsymbol{v}_{p-1}}\boldsymbol{v}_{p-1}. \tag{A.4}$$

Construct orthonormal basis

$$\boldsymbol{P} = \left[\begin{array}{ccc} \frac{\boldsymbol{v}_1}{\|\boldsymbol{v}_1\|} & \cdots & \frac{\boldsymbol{v}_p}{\|\boldsymbol{v}_p\|} \end{array} \right]. \tag{A.5}$$

Provided that subject matrix $\boldsymbol{A} = \boldsymbol{A}^T$, the eigenvectors are iteratively calculated as

$$\boldsymbol{A}_1 = \boldsymbol{A} \quad \text{and} \quad \boldsymbol{A}_{k+1} = \boldsymbol{P}_k^{-1} \boldsymbol{A}_k \boldsymbol{P}_k. \tag{A.6}$$

$\boldsymbol{P}_k$ is a orthonormal basis, and so $\boldsymbol{P}_k^{-1} = \boldsymbol{P}_k^T$. The eigenvectors are calculated iteratively as

$$\boldsymbol{D}_{k+1} = \boldsymbol{D}_k \boldsymbol{A}_{k+1} \quad \text{where} \quad \boldsymbol{D}_1 = \boldsymbol{I}_3. \tag{A.7}$$

After convergence the first column of $\boldsymbol{D}$ will contain the principal eigenvector, and A_{11} the corresponding eigenvalue [30].

A.3 Conclusion

The principal eigenvector and corresponding eigenvalue is calculated using a minimal set of steps. This method is easy to implement, compared to implementation of an SVD algorithm.

Appendix B

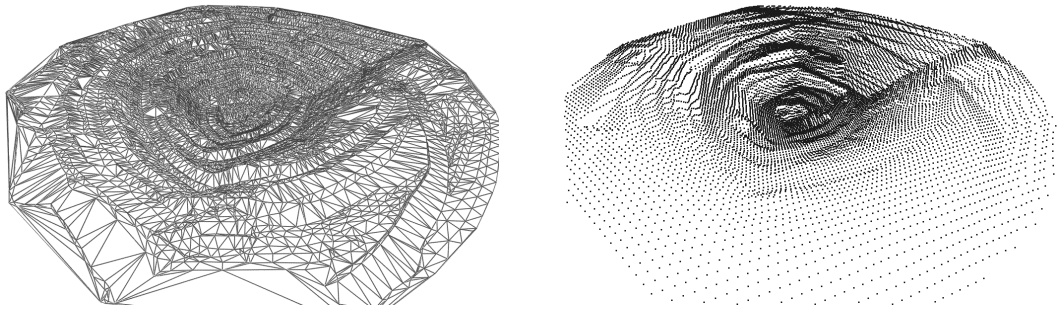
Accelerating closest point correspondence

B.1 Introduction

In Chapter 3 we introduced various methods of point correspondence. Most of them have limited application to this problem since they require meshed models. From the correspondence algorithms mentioned, closest point correspondence (CPC) is the most practical since it also works on unmeshed point clouds. CPC is also the simplest algorithm to implement since it requires only the most basic of mathematical functions. CPC is, however (like all correspondence algorithms), computationally expensive and takes up the majority of the execution time during the alignment process. In this chapter we examine computational implications of the linear CPC algorithm and how to optimise it.

B.2 Linear closest point

The linear closest point algorithm is the most basic form of the closest point algorithm since no pre-processing is necessary. The spatial difference is computed using either the euclidean or the euclidean squared distance metric. The test point is compared to all the points in the template set and therefor has a complexity of $\boldsymbol{O}(N_X)$. If we compare all the points in the search set with all the points in the template set, then the complexity order is $\boldsymbol{O}(N_X N_D)$, where N_X is the size of the template set and N_D is the size of search set. For the purpose of aligning DTMs and SMs we would expect the alignment to converge within the order of minutes. It is typical for DTMs and radar range images to contain thousands or even hundreds of thousands of points. In this section we examine the implications involved with large point sets on the linear CPC algorithm. We conduct several tests to compare processing times.



(a) (b)

Figure B.1: (a) A typical TIN DTM rendered with triangle facelets. (b) A resampled set of the same DTM with 10 metre grid spacing.

Linear closest point correspondence test

In order to establish a baseline of typical execution times for linear closest point correspondence, we do a test. For the test we do the following:

We create a template point set using a typical TIN DTM (Figure 3.3 (a)) and define it as

$$\boldsymbol{X} = \{\boldsymbol{x}_i\} \quad \forall\, i \in \{1, \ldots, N_X\}\,.$$

We resample $\boldsymbol{X}$ using various grid spacings, defining the resampled sets as

$$\boldsymbol{Y}_\alpha = \text{resample}(\boldsymbol{X}, \alpha) \quad \forall\, \alpha \in \{60, 40, 20, 10, 8, 6, 4\}\,,$$

where α indicates the grid spacing. The size of resampled set N_{Y_α} is inversely proportional to the grid spacing such that

$$N_{Y_{60}} < N_{Y_{40}} < N_{Y_{20}} < N_{Y_{10}} < N_{Y_8} < N_{Y_6} < N_{Y_4}.$$

Figure 3.3 (b) depicts the resampled set $\boldsymbol{Y}_{10}$, resampled with a grid spacing of 10 metres. In order to perform a correspondence test, a second set of points is required. We generate such a set by randomly selecting points from the resampled set. The new set is rotated and translated so as to simulate a practical alignment scenario. For the test we assume that the search set $\boldsymbol{S}_\alpha$ has 20% the number of points contained in $\boldsymbol{Y}_\alpha$.

From each $\boldsymbol{Y}_\alpha$ select a subset of points such that

$$\boldsymbol{S}_\alpha \subseteq \boldsymbol{Y}_\alpha \quad \text{and} \quad N_{S_\alpha} = 0.2 \times N_{Y_\alpha}.$$

The elements of $\boldsymbol{S}_\alpha$ are picked randomly from $\boldsymbol{Y}_\alpha$ using an uniform distribution function. Translate and rotate $\boldsymbol{S}_\alpha$ to be more representative of a practical alignment scenario. The

order magnitude of the CPC algorithm during each alignment iteration is $\boldsymbol{O}(N_{S_\alpha} N_{Y_\alpha})$. The CPC algorithm is executed once, simulating one iteration of alignment. We perform all tests on an Intel Pentium 4 3.0 GHz machine with 512 MB of RAM. The algorithm is implemented using the GNU C++ programming language calling standard glibc functions. The Linux application gprof is used to profile the processing times.

Table B.1: The processing times of the closest point correspondence algorithm on point sets of various sizes. Larger sets require longer processing time.

Grid spacing α (m)	N_{Y_α}	N_{S_α}	Linear search time (s)
60	429	85	<0.01
40	963	192	0.02
20	3864	772	0.17
10	15483	3096	4.19
8	24180	4836	10.48
6	43002	8600	33.26
4	96742	19348	164.77

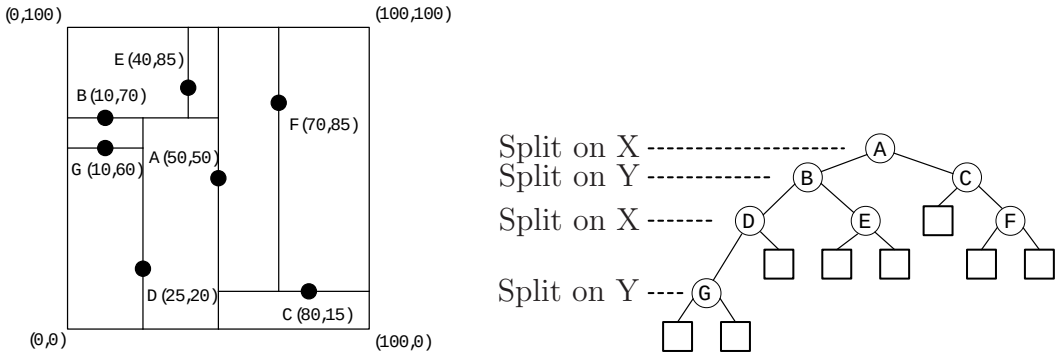
Refer to Table B.1 for the results of the CPC tests. The processing times for the smaller sets are insignificant (grid spacings 60 to 20), taking less than a second, but as the sets become larger (grid spacings 10 to 4) the processing times are prolonged to seconds and even minutes. We must also consider that the processing times are only for one iteration of alignment. In the section to follow we examine a spatial subdivision technique which will accelerate the CPC algorithm and reduce processing times.

B.3 kd-tree space partitioning

In this section we introduce a very effective spatial search method involving a divide-and-conquer spatial partitioning technique which reduces the search complexity from $\boldsymbol{O}(N_{S_\alpha} N_{Y_\alpha})$ to $\boldsymbol{O}(N_{S_\alpha} \log N_{Y_\alpha})$ [31]. The model point set $\boldsymbol{Y}_\alpha$, is pre-processed into an abstract data structure (ADS). The ADS is a kd-tree (also known as the k-dimensional tree) . A kd-tree is a special case of the binary tree structure (BTS). It has been shown [32] that the kd-tree storage structure accelerates the alignment process, reducing the processing time. kd-tree acceleration is a general search solution which can be applied to multi-dimensional data sets, but is especially effective for three-dimensional data sets. kd-tree optimisation involves two steps: kd-tree construction and tree traversal (searching).

During construction, the BTS is filled by bisecting the point set into spatial subsets and inserting them into the various branches of the tree structure. The second and final step involves a search algorithm which traverses the kd-tree in search of the closest point.

We briefly explain kd-tree construction using a two dimensional example taken from [31]. Consider point set $\boldsymbol{X} = \{(50,50), (80,15), (10,70), (70,85), (25,20), (40,85), (10,60)\}$. The two-dimensional space is bisected according to the split dimension (determined by the current tree depth) and placed within the BTS (Figure B.2 (a) and (b)). After the kd-tree construction, a recursive search algorithm is used to search for the closest point. See [31] for additional information.



(a) (b)

Figure B.2: (a) A two-dimensional example taken from [31] showing a spatially divided point set. (b) The kd-tree structure of the same example. The splitting dimension is determined by the current tree depth.

kd-tree accelerated closest point correspondence test

We perform a series of tests using the same sets as in Table B.1. The results of the tests can be seen in Table B.2. The processing times of the kd-tree accelerated searches for the resampled sets with grid spacings 4 to 20 were significantly reduced when compared to the equivalent linear search times.

Table B.2: The processing times of the linear closest point correspondence algorithm as well as the kd-tree accelerated closest point search.

α (m)	N_{Y_α}	N_{S_α}	Linear search time (s)	kd-tree const. time (s)	kd-tree search time (s)	Time diff. (%)
60	429	85	<0.01	<0.01	<0.01	–
40	963	192	0.02	<0.01	<0.01	–
20	3864	772	0.17	0.01	0.03	23.5
10	15483	3096	4.19	0.06	0.31	8.83
8	24180	4836	10.48	0.11	0.98	10.4
6	43002	8600	33.26	0.19	2.88	9.23
4	96742	19348	164.77	0.4	16.56	10.3

B.4 Conclusion

The CPC algorithm is computationally expensive, especially when applied to very large point sets. Typical DTM sets will require minutes of processing time when processed by a linear CPC algorithm. The CPC algorithm can however be significantly accelerated using a kd-tree storage structure. The tree construction requires some overhead and additional processing time but is still insignificant when compared to the computational expense of the linear CPC algorithm.