NTNU – Trondheim
Norwegian University of
Science and Technology

Comparison of Large Surfaces from Scanned 3-D Objects

Vilius Kazakauskas

Master of Science in Computer Science
Submission date: June 2014
Supervisor: Theoharis Theoharis, IDI
Co-supervisor: Christian Schellewald, IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

I dedicate this thesis to Laimute Zofija Jakštiene

Abstract

Registration of three–dimensional objects has vast applications in many areas of science. This research is linked to the international PRESIOUS project, concerning measurement of erosion effects on cultural heritage objects. The data in question is several magnitudes larger than found in research so far, often reaching several hundred million points. This thesis presents a novel approach to registration of such large data sets by co–dividing the sets into corresponding parts and registering them locally. The same division algorithm can be used to precisely measure the distance between the data sets. We test experimentally how the local registration results can be used to obtain an optimal global transformation for the entire data set.

Sammendrag

Registrering av tredimensjonale gjenstander har mange burksområder i ulike vitenskapsgrener. Denne avhandlingen er knyttet til det internasjonale PRESIOUS-prosjektet, som jobber med å måle effekten av erosjon på kulturminneobjekter. Vi arbeider med data som er langt større enn det som finnes i tidligere forskning på registrering. Denne avhandlingen presenterer en ny metode for registrering av slike store datasett ved å dele de i samsvarende deler og registrere de lokalt. Den samme inndelingsalgoritmen kan brukes til nøyaktig måling av avstanden mellom datasettene. Vi tester eksperimentelt hvordan de lokale registreringsresultatene kan brukes for å finne en optimal global transformasjon for hele datasettet.

Preface

This thesis presents work done at the Norwegian University of Science and Technology (NTNU) over the course of two semesters. I would like to thank my advisors Theoharis Theoharis and Christian Schellewald for supplying the task at hand and valuable and restless guidance during the project.

I would also like to thank my family – Laima Kazakauskiene, Kestutis Kazakauskas and Ramunas Kazakauskas for continuous love and support during this period of time.

Table of Contents

Chapter 1

Introduction

The research in this thesis is linked to the PRESIOUS project, which concerns measuring and simulation of erosion in cultural heritage objects. Real world objects are scanned, using a 3–D scanner and saved in digital form. New scans of the same objects are performed after a sufficient amount of time has passed, usually in the order of a year. The result is two digital representations of a 3–D shape, where one of them has been changed by erosion due to environmental effects. The goal is to find a way to measure these changes precisely, such that both the extent and location of erosion can be found automatically, given only the two scans of a cultural object.

The two scans of the object are performed manually and at different points in time. We can therefore not expect the scanning process to start at the exact same position both times. Furthermore, if the object is sufficiently large, or if it is part of a larger construct, the scans do not necessarily contain the entire surface area. In this case, we cannot expect the two scans of an object to represent exactly the same surface – they may not overlap completely. It may also be that one of the scans covers a smaller part of the object's surface, while the other one covers the whole object. For these reasons, the two digital versions must be aligned correctly, before we can measure the actual differences between them. This problem of alignment is called registration.

1.1 Motivation

1.1.1 Registration

Surface registration is a central problem in computer vision, where two data sets that each describe a 3–D shape in different coordinate systems are aligned via a transformation. The goal is to optimize the transformation, such that the shapes match as closely as possible. The equivalence of the shapes can then be determined by a distance metric.

Registration algorithms have a vast number of applications in many areas of science. (Huang et al., 2007) describe an algorithm used for several tasks within medical image analysis. One such task can be correction of deformation of 3-D MR brain images due to brain shift, which has a big impact on neuro–navigation systems. (Ferrant et al., 2001) Another application is matching of PET and MRI data, since the scans cannot be performed with the exact same patient positioning. (Mangin et al., 1992) (Guest et al.) describe a registration algorithm for matching images taken of a patient's face before and after surgery, which requires the registration to cope with significant differences in the data to be matched. (Chen et al., 2009) propose use of automatic surface registration in inverse engineering and manufacturing industry, to ensure satisfactory mechanical components with regard to their design specifications. Other uses of registration are found in matching of point clouds produced by modern sensors, recognition of objects in 3-D scenes, computer-aided design and computer vision systems. (Yamany et al., 1999) Thus, optimization techniques for registration have been widely studied during the last few decades, producing novel algorithms and new application areas.

1.1.2 Data Size

Erosion is an effect that works slowly on many types of materials. A 3–D scan that is meant to capture the results of erosion after a year on for example a marble statue, must have very high precision, resulting in a large amount of data points. This introduces difficulties in processing, aligning, measuring and even storing the digital data. In fact, we have not found any previous registration literature that deals with objects of this size, as will be elaborated upon in chapter 2. The exact size of our data is given in section 4.1.

There exists previous work that recognizes this problem and refers to the data sets in their applications as ”large”, but the solutions given still do not scale to our purposes. This will be elaborated upon in section 2.4. For clarity, we define a ”large data set” in this thesis as a data set that can not be wholly contained in memory on a modern computer.

1.2 Structure of Thesis

The thesis is divided into four chapters.

1. The first chapter gives the motivation and background for the work. Here we explain the terminology used later in the thesis.
2. The second chapter is an overview of the field of registration and how the previous works relate to our case of large data sets. We present the popular iterative closest point (ICP) algorithm and many of its variants, as well as looking into other optimization techniques.
3. The third chapter is divided into two parts. The first part describes our octree–based technique for dividing a large data set and looking at the registration problem locally. We propose a method for acquiring the global transformation from the information obtained in the local registrations. The second part describes our method for accurate measurement of distance between two large data sets, using the octree division algorithm from part one. We show that the method is sound.
4. Chapter four describes the results from our experimental runs and includes a discussion of said results, as well as ideas for future work.

1.3 Background

1.3.1 Transformation Types

A *Euclidean* transformation transforms vectors in $\mathbb{R}^n$, such that distances between every pair of points is preserved. This is a special case of *rigid transformations*, which work in any vector space, with the same constraint. Every rigid transformation is a combination of rotations, translations and reflections. If we exclude reflections, the transformation preserves the orientation of objects and is called a *proper rigid transformation*.

Non–rigid transformations, like shearing and scaling do not preserve distances between point pairs. Erosion transforms a real world object in a non–rigid way and there has been done research on non–rigid registration. This is however not relevant for us, since we wish to measure the erosion and must preserve the differences due to erosion while registering the objects.

All our work is done in three–dimensional Euclidean space and we wish to preserve point pair distances and orientation during registration. Thus, whenever we refer to ”Euclidean transformation” or ”rigid transformation” in this thesis, it should be understood as *proper Euclidean transformation*.

1.3.2 Registration Process

Let d be a function which gives the distance between two three-dimensional objects and let S and T be two such objects. The registration problem is defined as finding the transformation which applied to S produces the object S' such that $d(S', T)$ is as small as possible. As explained above, in our case we wish the transformation to be a proper Euclidean transformation in three–dimensional space, which means that it consists of a rotation and a translation.

Expressing the rotation as a 3x3 matrix R and the translation as a 3x1 vector **t**, registration searches for R and **t**, such that $d(\{R\mathbf{s} + \mathbf{t}\}_{\mathbf{s}\in S}, T)$ is as small as possible. We can view each point s in S as a 4x1 vector $\mathbf{s}' = (\mathbf{s}, [1])$. Then, we can combine the rotation and translation into one 4x4 matrix A and try to minimize $d(\{A\mathbf{s}'\}_{\mathbf{s}'\in S'}, T)$. Regardless of the parameter presentation chosen, we see that there are six degrees of freedom – three for the rotation and three for the translation. Thus, registration can be viewed as a six–dimensional search problem for the rotation and translation parameters.

The object that is transformed is often called the source, while the object being matched against is called the target. Still, one of the cornerstone articles in the field (Besl and Mckay, 1992), uses the term *data* for the source and *model* for the target. The term *template* has also been used instead of target. The terminology we use in this thesis is source and target, often denoted S and T in mathematical notation.

1.3.3 Optimization

Registration can be cast as a non-linear optimization problem and many well known optimization techniques are applicable for the problem. (Tam et al., 2013) These techniques vary in performance and applicability based on the geometric properties of the 3-D surface to be matched. Matching a smooth sphere for example, may benefit from different optimizations than a flat surface with many small bumps and ridges. Noise and outliers provide further difficulties for the registration process, especially when the data is collected by a 3–D scanner or other types of sensors. In many cases, the amount of data may be too large to handle in a straightforward approach, as is the case with high resolution 3–D scans of large objects.

Constraints on the data, like features, saliency, regularization and search constraints can all be used in the optimization process, so different registration approaches may be applicable based on the available description of the data. Other algorithms, like iterative closest point (ICP) work only with the geometry of the data, though it can be combined with other approaches. In our case, we will have access to only the geometry of the objects.

1.3.4 Bounding Box

A minimum bounding box of a set of points in three–dimensional space is the box with the least area such that all the points lie within it. Such a box can be represented by its corner points. An axis–aligned minimum bounding box must have edges parallel to the axes of the coordinate system. An arbitrarily aligned minimum bounding box can have any orientation. An axis–aligned bounding box can be represented by just two points in space, corresponding to any of its corners that lie diagonally across each other.

1.3.5 Octree

An octree is a data structure where every node in the tree has exactly eight children. If we let each node represent a rectangular portion of space, then each node divides that space into eight pieces called *octants*. Each of these octants is the space represented by one of the node's children. Thus every node has a bounding box that encloses its space. The point of subdivision is the center point of the bounding box. See figure 1.1 for a visualization of an octree.

1.3.6 Centroid Point

When rotating an object, it is of relevance what point the object is rotated around. This can for example be the origin in the coordinate system, or the object's centroid point. Given an object $S = \{\mathbf{s}_i\}$ with N_S data points, the object's centroid point, also called its center of mass is defined as

$$\mathbf{c} = \frac{1}{N_S} \sum_{i=1}^{N_S} \mathbf{s}_i$$

Note that the centroid point is not necessarily the same as the center point of the object's bounding box.

The usual way of rotating an object around its centroid point, is to first translate it to the origin, rotate it around the origin and then translate it back to the original position:

$$S' = \{R(\mathbf{s} - \mathbf{c}) + \mathbf{c}\}_{\mathbf{s} \in S}$$

This technique allows us to work with rotations around any point in space. It is important to keep track of which point a rotation was performed around, when working with transformation parameters.

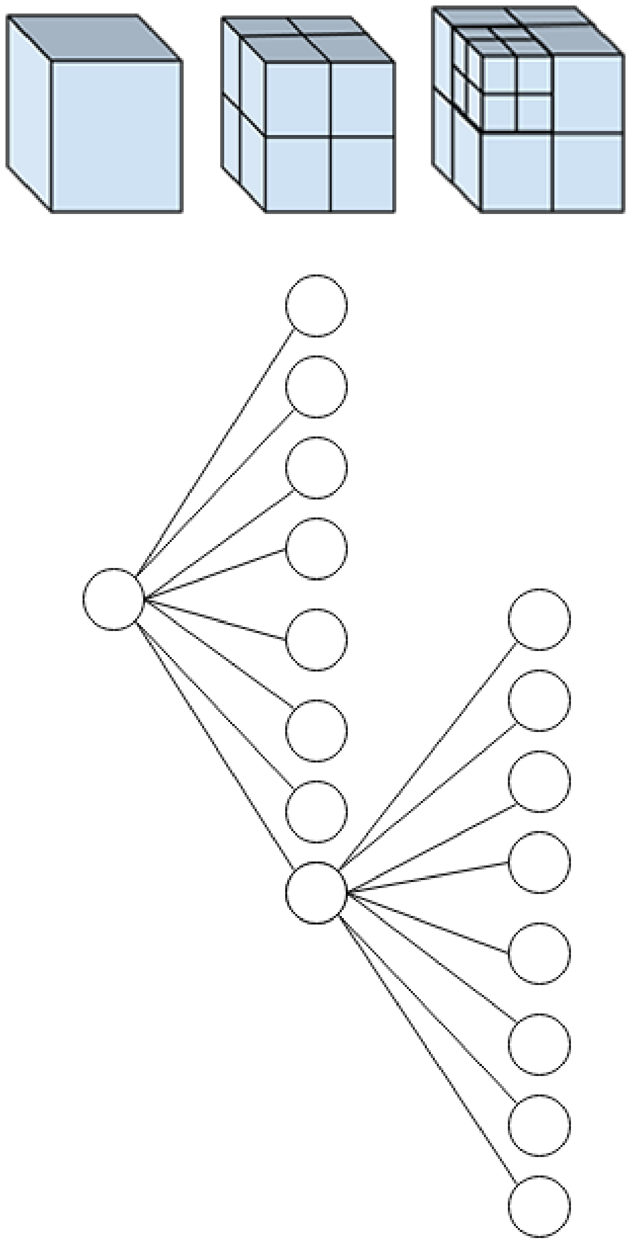


Figure 1.1: Sub–divisions of space corresponding to octree level.

Chapter 2

Overview of Field

This chapter provides an overview of the current field of 3–D registration. We begin by describing similar overviews done by other authors, before delving into the ICP algorithm, which is a cornerstone of the registration field. We proceed by describing the different ICP variants and finally other optimization alternatives for the registration problem.

2.1 Previous Overviews

Several comparisons and overviews of different registration techniques have been done throughout the years. Most of these focus either only on rigid or non–rigid registration, where the former is of relevance in our case.

Many studies focus on variants of the ICP algorithm, which is described in section 2.2. (Rusinkiewicz and Levoy, 2001) classify the variants based on their differences in six stages of the ICP algorithm:

- Selection of points to match in the data.
- Matching of points between the data sets.
- Weighting of the matched pairs.
- Rejection of some matched pairs.
- Error metric used.
- Technique for minimization of the error.

Furthermore, the algorithm variants are run on data sets with different geometric properties – a smooth wave–like landscape with added Gaussian noise, a fractal landscape with all levels of detail and a flat Gaussian noise landscape, with deep grooves in the middle. The study focuses on speed of convergence and provides many test results, but not all algorithm variants are run on all the test cases and the testing procedure is not always made clear. The paper uses speed of convergence as the judging criteria of the different algorithms.

(Pomerleau et al., 2013) point out the problem of choosing a fitting ICP variant between the dozens available for implementation purposes. They propose a protocol for comparison between ICP variants and use the protocol to compare the well known point–to–point and point–to–plane distance metrics used in ICP (see section 2.2.1).

(Salvi et al., 2007) take a look at different techniques for both initial coarse registration and fine registration, once an approximation is available. The paper focuses on the case of data with noise and outliers and is accompanied by a Matlab toolbox with several of the described techniques.

A review of geometric shape correspondence techniques by van Kaick et. al. focuses on space-time registration, where non-rigid surfaces to be matched may vary in time. (van Kaick et al., 2011) The paper also looks at semantic shape analysis, which requires knowledge-driven shape correspondence. The objective is to recognize parts of the shape and their functionality, so they can be matched even when the shape has changed over time. An example of this is matching a doll figure with movable parts in different positions. Several methods are reviewed and classified according to data representation, objective function and solution approach.

2.1.1 Relation to Data Fitting

Tam et. al., who give an overview of both rigid and non–rigid techniques, focus on the relation of registration to the field of data fitting in hopes of acquiring novel perspectives of the problem. (Tam et al., 2013) Data fitting is simply the process of finding a mathematical function that fits to a set of data points. For this purpose, they first define registration as an optimization problem. Let $P = \{\mathbf{p}_1, ..., \mathbf{p}_{N_1}\}$ and $Q = \{\mathbf{q}_1, ..., \mathbf{q}_{N_2}\}$ be two overlapping point sets. Then the objective function in the optimization problem is:

$$E = E_{data} + E_{reg}$$

$$E_{data} = \sum_i \|\mathbf{q}_i - \chi(\mathbf{a}, \mathbf{p}_i)\|^2,$$

such that $\{\mathbf{p}_i, \mathbf{q}_i\}$ are corresponding points, and χ is a transformation of $\mathbf{p}_i$ using parameter vector $\mathbf{a}$ such that E_{data} is minimal. The goal is to find the optimal parameters $\mathbf{a}$. In slightly less abstract terms, the parameter vector $\mathbf{a}$ will often describe a rotation matrix and a translation vector, while the transformation χ simply applies this matrix and vector to the point $\mathbf{p}_i$. E_{reg} represents any additional constraint we wish to impose on the registration and is called a regularization term.

(Tam et al., 2013) then give the following definition of a data fitting problem. Given a set of data points $\mathbf{p}_1, ..., \mathbf{p}_N$, $\mathbf{p}_i = (x_i, y_i, z_i)$, we wish to find the parameter vector $\mathbf{a}$ such that the function $z = f(\mathbf{a}, x, y)$ describes the data in the best possible way. The fitting error is measured by:

$$E = \sum_i^N \|z_i - f(\mathbf{a}, x_i, y_i)\|^2 + E_{reg}$$

We immediately see the similarities between the above definitions of registration and data fitting. The paper points out three similarities, which they call model selection, correspondences and constraints, and optimization.

Model selection refers to the choice of functions χ in the case of registration and f in the case of data fitting. For rigid transformation, the only example given by (Tam et al., 2013) is the Euclidean transformation, consisting of a rotation and a translation, which we described in section 1.3.1.

Correspondence refers to the matching of points between the two data sets in case of registration, while it is already given in the data point (x_i, y_i, z_i) in case of data fitting. Thus, this is more of a dissimilarity between the two fields, than a "core interwoven component of their relation", as stated by (Tam et al., 2013). The paper still provides a thorough overview of the different constraints and techniques used to find these point matchings in the data sets P and Q.

The clearest similarity between registration and data fitting is that they are both cast as *optimization* problems in the above definitions. (Tam et al., 2013) discuss different optimization techniques to arrive at the best parameter vector a and set of point matchings Σ. The paper points out that when the optimization problem is continuous, the parameter vector and set of point matchings are determined simultaneously and iteratively. The soon to be familiar ICP algorithm falls under this classification.

(Tam et al., 2013) point out that all the above observations imply the possibility of solving registration problems with techniques primarily designed for data fitting. However, they note that not every registration technique fits into their framework, though most do.

2.2 Iterative Closest Point

Since its introduction in 1992 by Paul J. Besl and N. D. Mckay (Besl and Mckay, 1992), the Iterative Closest Point (ICP) algorithm has been widely employed when matching 3-D surfaces based on geometry alone. (Rusinkiewicz and Levoy, 2001) Different techniques have been developed that make use of constraints such as features, saliency, regularization and search constraints, and the use of these can often be combined with ICP. (Tam et al., 2013) We will therefore start with a description of the ICP algorithm, before continuing with other approaches used in the literature.

The ICP algorithm works iteratively and alternates between applying a rigid transformation to the data and assessing the mean-square error. Specifically, the ICP algorithm registers a source data shape with a target data shape by iterating the steps in table 2.1. Step 1 and 2 are the core of the algorithm and are often called the match–align steps.

1. **Match.** Associate points in the source data shape with their closest point in the target data shape.
2. **Align.** Estimate a rigid transformation that would align the above point pairs as closely as possible.
3. **Apply.** Apply the rigid transformation to the source data shape.
4. **Terminate.** Terminate the algorithm if the mean-square error change is below a specified threshold. Otherwise, iterate steps 1–4.

Table 2.1: Steps in the ICP algorithm

Data representation

The ICP algorithm is highly general when it comes to what data representations it can work with. (Besl and Mckay, 1992) provide the mathematics of computing minimum distance from a point to the following data representations, generalizing to n dimensions:

- Point sets
- Line segment sets
- Implicit curves
- Parametric curves
- Triangle sets
- Implicit surfaces
- Parametric surfaces

The source data set must be converted into a point set, but the target data set may be expressed as any of the above structures.

Matching data points

Given a point $\mathbf{s}_i$ in the source data set, (Besl and Mckay, 1992) provide the techniques necessary to calculate the distance between that point and the target data set. The closest point in the target $\mathbf{t}_i$ becomes the match of $\mathbf{s}_i$. Doing this for all the points in the source, we obtain the correspondences $\{\mathbf{s}_i, \mathbf{t}_i\}, i = 1, ..., N_S$, where N_S is the amount of points in the source.

Note that the ICP algorithm is meant to match data sets, where every data point in the source has a match in the target. This means that it cannot be applied as is to data sets that overlap only partially.

Aligning the matched point pairs

The transformation estimation uses a mean square cost function to determine the effectiveness of the matching:

$$e = \frac{1}{N_S} \sum_{i=1}^{N_S} \|\mathbf{t}_i - R\mathbf{s}_i - \Delta \mathbf{s}_i\|,$$

where $\{\mathbf{s}_i, \mathbf{t}_i\}, i = 1, ..., N_S$ are the point correspondences from the matching step, R is the rotation matrix and Δ is the translation vector.

The original ICP algorithm uses a quaternion–based technique to find the optimal rotation and translation for minimizing the mean square objective function e. (Horn, 1987) Let $\{\mathbf{s}_i, \mathbf{t}_i\}, i = 1..., N_S$ be the matched correspondences of points in the target and source data sets, after step 1 in the algorithm. Let $\boldsymbol{\mu}_T$ and $\boldsymbol{\mu}_S$ be the centroid points of the target and source, respectively. (Horn, 1987) then define the matrix

$$M = \sum_{i=1}^{N_S} (\mathbf{s}_i - \boldsymbol{\mu}_S)(\mathbf{t}_i - \boldsymbol{\mu}_T)' = \begin{bmatrix} M_{x,x} & M_{x,y} & M_{x,z} \\ M_{y,x} & M_{y,y} & M_{y,z} \\ M_{z,x} & M_{z,y} & M_{z,z} \end{bmatrix}$$

The matrix M contains the sums of products between point coordinates in the source and target data sets, indicated by the indices in the notation above. As (Horn, 1987) point out, this matrix contains all necessary information to find the rotation that minimizes the least squares distance error between the data sets. For this purpose, they define the following 4x4 symmetric matrix, where the nine degrees of freedom stem from the sums and differences of the elements in M (there is no 10th degree of freedom, since the diagonal elements of the matrix sum to 0):

$$N = \begin{bmatrix} d_1 & M_{y,z} - M_{z,y} & M_{z,x} - M_{x,z} & M_{x,y} - M_{y,x} \\ M_{y,z} - M_{z,y} & d_2 & M_{x,y} + M_{y,x} & M_{z,x} + M_{x,z} \\ M_{z,x} - M_{x,z} & M_{x,y} + M_{y,x} & d_3 & M_{y,z} + M_{z,y} \\ M_{x,y} - M_{y,x} & M_{z,x} + M_{x,z} & M_{y,z} + M_{z,y} & d_4 \end{bmatrix}$$

where

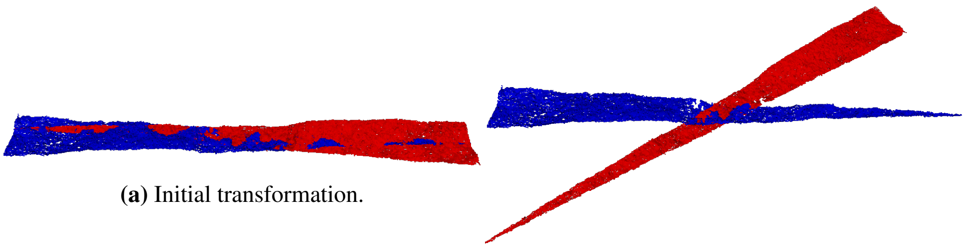
$$d_1 = M_{x,x} + M_{y,y} + M_{z,z},$$

$$d_2 = M_{x,x} - M_{y,y} - M_{z,z},$$

$$d_3 = -M_{x,x} + M_{y,y} - M_{z,z},$$

$$d_4 = -M_{x,x} - M_{y,y} + M_{z,z}$$

The optimal rotation expressed in quaternion form $\mathbf{q}_R$ is the eigenvector corresponding to the maximal eigenvalue of N. The optimal translation vector is given by
$T = \boldsymbol{\mu}_T - \mathbf{R}(\mathbf{q}_R)\boldsymbol{\mu}_S$, where $\mathbf{R}(\mathbf{q}_R)$ is the rotation matrix corresponding to $\mathbf{q}_R$. We refer to (Horn, 1987) for a more detailed derivation.



(a) Initial transformation.

(b) Hypothetical position during ICP, if it were to solve the problem.

Figure 2.1: **(a)** Initial position of a source (in red) to be matched with a target (in blue). **(b)** In order for the source to be iteratively transformed into the correct position, it has to first move through this position, which has a much higher distance value than in (a).

Termination

The algorithm terminates when the mean square error of the registration reaches a predetermined threshold. A mathematical proof of the convergence of ICP to a local minimum from any initial rotation and translation is given in (Besl and Mckay, 1992). However, there is no guarantee of converging to the global minimum. The selection of initial states of the source data is therefore a point of interest. In each iteration of the ICP algorithm, the average distance between the source and the target is reduced both by the matching of closest points step (step 1 in table 2.1) and the transformation estimation step (step 2 in table 2.1). This is the basis of the proof of ICP algorithm convergence given in the paper.

Performance

The aligning technique above is linear in the size of the source data set, as is the application of a transformation to the source data set. Finding the closest point in the target to a point in the source is linear in the size of the target data set. Doing this for each point in the source is therefore $O(N_T N_S)$, where N_T and N_S are the numbers of geometric entities in the target and data points in the source, respectively. Thus, the running time of the ICP algorithm is dominated by step 1 in table 2.1. (Besl and Mckay, 1992) Many ICP variants focus on reducing the running time of this step, or on executing the step fewer times.

Initial parameters and local minima

The ICP algorithm is highly dependable on initial parameter values in order to not get stuck in local minima. In fact, this is characteristic to most registration algorithms and an initial approximate transformation is often assumed to be available. (Pulli, 1999) Figure 2.1 shows an example of how the initial position of a source data set leads the algorithm to a local minimum.

A usual way to obtain the initial approximate transformation is interactively. In case of large data sets, one can first create a uniform sub–sampling of the data and then find an approximate transformation using the software of choice.

2.2.1 Iterative Closest Point Variants

There have been developed many variants of ICP that employ different strategies in several parts of the algorithm. (Rusinkiewicz and Levoy, 2001) point out that different implementations of ICP may vary in the procedures of all the following tasks:

- Selecting points in the target and source data sets
- Matching the selected points into pairs
- Weighting the resulting pairs
- Rejecting unwanted pairs
- Choice of error metrics
- Minimizing the error

Sub–sampling

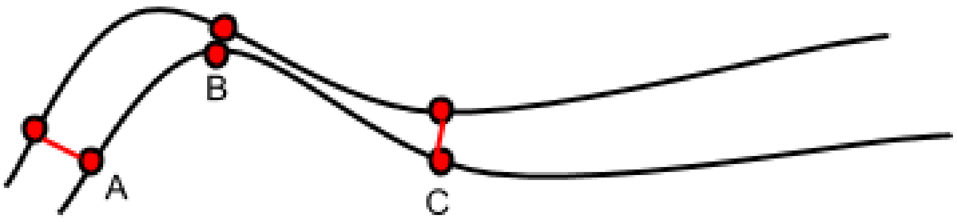
The original ICP paper by Besl and McKay propose using all points in the surface for matching. (Besl and Mckay, 1992) Real world data sets may be too large for this to be computationally feasible, since the algorithm iterates over the match–align steps many times. Note that the memory requirements of the algorithm are linear in the size of the target and source data sets. In this case sub–sampling of the data may be necessary.

By uniformly sub–sampling the data, a mesh hierarchy can be created, such that the meshes increase in detail as you go up in the hierarchy. ICP is then run on the lowest level mesh and the resulting transformation is used on the next level mesh. This process continues until the highest level mesh has been registered. (Turk and Levoy, 1994) Note that this approach still requires running ICP on the data at full resolution, though it has a good chance of decreasing the amount of iterations needed at that point.

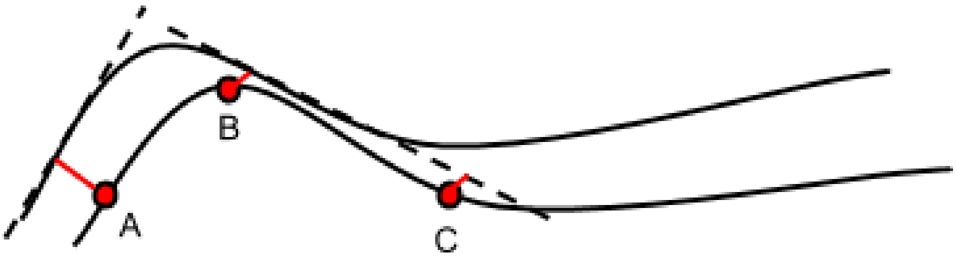
Another approach is simply sub–sampling the data randomly, with a new set of sampled points at each iteration of the algorithm. (Masuda et al., 1996) The drawback of this approach is that you cannot guarantee that all the information in the data is used before convergence.

Point–to–point and point–to–plane metrics

One of the main divisions of ICP variants is whether they use a point–to–point or point–to–plane error metric. The former computes the sum of squared distances between points, while the latter computes the sum of squared distances from each point to the plane containing the destination point, perpendicular to the destination point's normal. The different approaches are shown in Figure 2.2. Recent results indicate that point–to–plane has better overall performance across different registration cases. (Pomerleau et al., 2013) However, point–to–point can still perform better in certain cases, for example when matching shapes that are largely flat or have uniform curvature (e.g. a sphere). Point-to-point also handles constraints, such as matching different features of points, better than point–to–plane. (Pulli, 1999)



(a) Point–to–point



(b) Point–to–plane

Figure 2.2: **(a)** As point A is moved closer to its match, point B and C will receive new corresponding points in the next iteration. **(b)** As point A is moved closer to its match, point B and C ”slide” along their matching tangent plane, without changing correspondences.

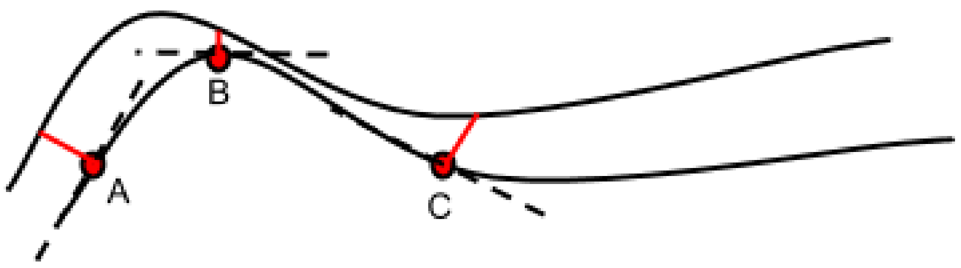
Matching points

The original ICP algorithm matches each source data point with the closest data point in the target. (Besl and Mckay, 1992) Yang and Medioni propose matching each source data point $\mathbf{p}$ with the intersection of the line normal to the source at $\mathbf{p}$ and the target data shape. However, this means that any points on the surface of the target may be used as correspondences for the source data points during the execution of the algorithm. Since many of these may not be the true corresponding points, it is possible that incompatible constraints arise and slow down the convergence of the algorithm. The solution is to approximate the target data shape, by using the point–to–plane metric described above. (Yang and Medioni, 1991) This approach is depicted in figure 2.3 for clarification.

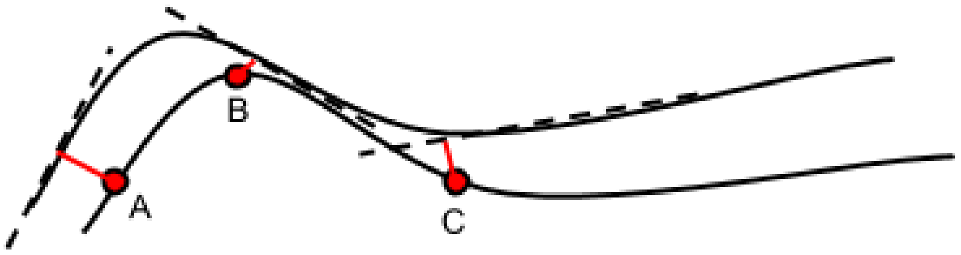
Note that using this approach, we are able to match a source data set to a target, without access to the true corresponding points in the target.

Overlapping data sets

The original ICP algorithm assumes that every point in the source has a match in the target, but this is not true for objects that overlap only partially. Still, we would like to be able to use ICP in the registration of such objects. A usual approach is to enforce a threshold such that any point pairs with distance above this threshold are ignored. (Pulli, 1999) The result is that the ICP algorithm only considers the points in the actually overlapping areas of the objects. One can also use a percentage–based threshold – keeping a certain percentage of the best matches. (Masuda et al., 1996)



(a) Matching points with intersection.



(b) Combined with point–to–plane metric.

Figure 2.3: **(a)** Each point is matched with the point of intersection between their line normal and the target. Line normals are shown in red. **(b)** The distance is measured between the point and the tangent plane of their match in (a). Distances are shown in red.

k–d tree optimization

As mentioned previously, the point-matching part of the ICP algorithm is the most computationally expensive one, where every closest point match runs in $O(N_S N_T)$ time, for N_S points in the source and N_T geometric entities in the target. Besl and McKay suggest using k-dimensional binary trees to optimize this step. (Besl and Mckay, 1992) The search for a closest point is then done in a binary tree, checking for which side of a hyperplane the point lies in at each node of the tree. This approach prunes away parts of the search space and results in an average time complexity of $O(N_S \log N_T)$ for computation of a closest point. A good demonstration can be found in (Zhang, 1994).

Caching

Another speedup of point-matching can be achieved by caching. Intuitively, if a pair of points are close to each other in an iteration of the algorithm, there is high probability that they will be close to each other in the following iteration also. By caching a small number of closest points in the target together with each source data point, one can limit the search for closest points in the next iteration to these small sets of points, instead of the whole data. (Simon, 1996) provides tests that can be done to ensure that the true closest point lies in this small subset of points. This technique can be used together with the *k*–d tree technique described above for maximal effect. Note that this is a big time–space trade–off, since several target data points must be saved for each source data point.

(Simon, 1996) also proposes using spatial proximity of points in the same manner – if two points in the source data set are close to each other, their matching points in the target should be close too. The paper does not however provide any further details of such a technique.

Point match rejection

When point-pairs have been matched, one has the choice of rejecting some amount of the pairs. Pairs may be rejected if they are further apart than some threshold distance or they are the worst $n\%$ matching pairs according to some metric. Godin et. al. propose the closest compatible point idea, where point pairs are only considered if they are compatible based on some invariant property of the points, like intensity and color. (Godin et al., 1994) Another choice is to reject point pairs with normal vectors that differ by more than 45 degrees. (Pulli, 1999) Rejection of pairs where either point is on a mesh boundary reduces the chance of wrong pairings if the overlap between the surfaces is incomplete. (Turk and Levoy, 1994)

Point match weighting

There are little examples in the literature of assigning different weights to point pairs in the error calculation, rather than constant weighting as in the original ICP algorithm. (Godin et al., 1994) uses a weight based on the Euclidean distance between the points and the compatibility of the points with regard to invariant intensity properties. If non-constant weighting of point-pairs is used, the ICP algorithm is no longer guaranteed to converge monotonically towards a minimum. (Godin et al., 1994) Rusinkiewicz and Levoy conclude that applying weights in this manner has a highly data-dependent effect and no consensus whether it is beneficial in the general case was reached. (Rusinkiewicz and Levoy, 2001)

Accelerated ICP

An accelerated ICP algorithm, which uses linear and quadratic extrapolation for the registration vectors is also proposed by Besl and McKay (Besl and Mckay, 1992). As the algorithm proceeds, it produces a sequence of registration vectors. The differences between each two consecutive vectors is a sequence of directions in the registration state space. If the angles between the last three of these directions are sufficiently small, the algorithm calculates a possible linear and parabola update to the registration vector sequence. The result of this acceleration is quick convergence, compared to other non-linear optimization methods. We refer to (Besl and Mckay, 1992) for the details of implementation.

Error minimization using SVD

The singular value decomposition (SVD) of a matrix M is a factorization $M = U\Sigma V^*$, such that U is a unitary matrix, Σ is a diagonal matrix with non–negative numbers and V^* is a unitary matrix. (Arun et al., 1987) propose an SVD–based algorithm for finding the rotation R and translation T that minimizes the least squares error once corresponding points between the source and target data sets have been matched. Let $\{\mathbf{s}_i, \mathbf{t}_i\}$, $i = 1, ..., N_S$ be the matched point correspondences. Both the target and source data points are first transposed, such that their centroid points lie at origin. We then have the point correspondences $\{\mathbf{s}'_i, \mathbf{t}'_i\}$. Viewing the points as column vectors, we calculate the cross–covariance matrix

$$H = \sum_{i=1}^{N_S} \mathbf{s}'_i \mathbf{t}'^T_i,$$

where the sum of the matrices is done element–by–element. As (Besl and Mckay, 1992) point out, this matrix is the same as the matrix M computed in the error minimization approach in (Horn, 1987) (see section 2.2). The next step is to find the singular value decomposition of H:

$$H = U\Sigma V^T$$

We can then find the rotation matrix $R = VU^T$ and translation vector $\Delta = \mathbf{s}_C - R\mathbf{t}_C$, where $\mathbf{s}_C$ and $\mathbf{t}_C$ are centroid points of the source and target points, respectively. We refer to (Arun et al., 1987) for derivation, proofs and some rare degenerate cases. The same paper also shows that the quaternion–based technique described in section 2.2 and the SVD–based technique have comparable running times.

2.3 Optimization Methods

As mentioned previously, registration can be looked at as a non–linear optimization problem, where one attempts to optimize the transformation of the data points, such that the error function is minimized. We have already introduced the quaternion based optimization technique in section 2.2 and the singular value decomposition based technique in section 2.2.1. In this section, we introduce other optimization methods found in the literature.

Geometric distance vs. largest common pointset

Objective functions are commonly either based on geometric distance between points or on the number of matching points between the data sets. (van Kaick et al., 2011) We recognize the former from the ICP algorithm. The latter, called Largest Common Pointset (LCP) attempts to find a correspondence for the largest possible subset of data points, via a transformation. A threshold value is used for deciding whether two points are close enough to be considered matching. A main distinction between the two approaches is that LCP handles partial matching naturally, while an objective function based on geometric distance will automatically consider all data points.

Optimization method types

There is a vast number of proposed optimization methods. (Tam et al., 2013) gives a thorough classification of the types of these methods which is briefly summarized here for reference, before expanding on techniques of interest.

- **Local Deterministic Optimization.** These methods try to locally minimize or maximize an objective function and are often efficient, but depend on good initialization. They also have a tendency to converge to a local minimum.
- **Global Deterministic Optimization.** The methods try to avoid local minima and find a global minimum. Thus they either perform a complete search, or relax the problem so a good approximation of a global solution can be found.
- **Stochastic Optimization.** These methods use statistics and probabilistic approaches. They can handle data sets with noise and outliers, as well as missing data.
- **Constrained Search.** These methods impose different constraints in order to limit the search space.

The above techniques can be combined, since they have different complementary strengths. Stochastic methods may work faster, but do not guarantee a globally optimal solution, given their nondeterministic nature. Therefore, they can be used to obtain an initial approximate alignment (coarse registration) for a local optimizer. This approach reduces the chance that the fine registration ends up in a local minimum and works well in practice. (Tam et al., 2013)

Local quadratic approximation

(Mitra et al., 2004) shows how to solve the registration problem using methods such as Newton iteration and gradient descent, which require computation of accurate derivatives of the objective function. The error landscape for the objective function is defined by a function that gives the squared distance from a data point to the model surface. Local quadratic approximants of this function are generated such that common optimization methods can be used to solve the problem. The proposed technique works as follows:

1. The objective function is the sum of quadratic approximants of the squared distance function for each point in the data set.
2. The rotation applied to the data points is first linearized (for small angles θ, $sin\theta \approx \theta$ and $cos\theta \approx 1$).
3. Find derivatives of the objective function.
4. Set derivatives to zero.
5. Solve resulting system of equations.
6. Apply transformation and iterate the technique until convergence.

For any point $\mathbf{x} = \begin{bmatrix} x & y & z \end{bmatrix} \in \mathbb{R}$, a local quadratic approximant has the form

$$\begin{aligned} F^{+}(\mathbf{x}) &= Ax^2 + Bxy + Cy^2 + Dxz + Eyz + Fz^2 + Gx + Hy + Iz \\ &= \begin{bmatrix} x & y & z & 1 \end{bmatrix} Q_{\mathbf{x}} \begin{bmatrix} x & y & z & 1 \end{bmatrix}^T , \end{aligned}$$

and approximates the squared distance from $\mathbf{x}$ to the target data shape. (Mitra et al., 2004) give two techniques for computing such functions. Under the small motion assumption mentioned in step 2 above, the general rotation matrix can be linearized as:

$$R = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 & 0 \\ \sin\gamma & \cos\gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\beta & 0 & \sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\approx \begin{bmatrix} 1 & -\alpha & \beta & 0 \\ \alpha & 1 & -\gamma & 0 \\ -\beta & \gamma & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} ,$$

where α, β and γ are rotations around the x, y and z-axes, respectively. Let $\mathbf{t} = \begin{bmatrix} t_x & t_y & t_z & 1 \end{bmatrix}$ be the registration translation vector and N_S the amount of points in the source data set $\{\mathbf{s}_i\}$. We now need to minimize the function

$$e = \sum_{i=1}^{N_S} (R\mathbf{s}_i + \mathbf{t}) Q_{\mathbf{s}_i} (R\mathbf{s}_i + \mathbf{t})^T ,$$

where $Q_{\mathbf{s}_i}$ is part of the local quadratic approximant as defined above and depends on $\mathbf{s}_i$. e depends on the six variables α, β, γ, t_x, t_y, t_z and by setting the corresponding partial derivatives to zero, we obtain a system of six linear equations, which can be solved using conventional methods.

Note that the point–to–point and point–to–plane ICP variants are special cases of the above framework. (Mitra et al., 2004)

Branch–and–bound

The above method is efficient, but can get stuck in local minima. There have been attempts to apply global optimization techniques to the registration problem, such that a global minimum can be found. If we can represent the solution of registration in a tree structure, branch–and–bound algorithms can be applied. One way to do this is to let each node be a match between a data point and a model point. Each path from the root to a leaf of the tree then represents a complete matching of the data sets. The technique then requires a lower bound on the cost function. In (Gelfand and Mitra, 2005), this method is used to compute the initialization for ICP. The paper suggests a tight and efficient lower bound for the cost function. Furthermore, the technique can be used to match only partially overlapping surfaces.

Relaxation methods

Several techniques relax the registration problem such that a solution close to the global minimum can be found. (Gold and Rangarajan, 1996) introduces a regularization term in the objective function such that it becomes convex. (Liu, 2007) introduces regularization terms in the objective function in order to apply an approximation of simulated annealing, called mean field annealing. The resulting function attempts to minimize the registration error, equalize mean field variables and maximize the overlapping area of the two data sets. (Boughorbel et al., 2010) acquires a smooth convex approximation of the objective function by summing a large number of Gaussian functions, resulting in another Gaussian function, which is convex in a sufficiently large neighborhood of the rotation and translation parameters, as per the paper. The paper proposes that this technique solves the problem of local convergence of ICP, without sacrificing registration accuracy. The technique is also good at handling noise in the data.

Non–deterministic methods

If the data to be matched is not too large, such that one affords to run the matching part many times, some interesting non–deterministic methods may be applied to registration. In brief, (Chow et al., 2004) describes how to construct a chromosome representation, fitness function and cross-over and mutation operators, such that registration may be solved as a genetic algorithm problem. As expected, the efficiency of this method is highly dependent on computation of the fitness function. (Sandhu et al., 2010) applies a particle filtering approach, where a particle represents an initial transformation of the data. The method runs local optimization (any method is applicable, for example ICP) on particles from a possible distribution and predicts new particles (transformations) in an iterative process. Strengths of the method is its ability to deal with noise and poor initialization.

2.4 Large Data Sets

The main problem with the original ICP algorithm when dealing with large data sets is that it requires all data points in the source and target to be read into memory. In our case, we have defined a ”large” data set as one where this is not possible.

The matching step in ICP can not be done in a straight forward manner, because for every point s in the source, we have to check every point in the target to find the closest one. Optimizations of this, such as k-d trees (subsection 2.2.1) still require the whole target data set to reside in memory at the same time. Most approaches in the literature that claim to deal with large data sets, try to minimize the amount of times the matching step is run, but in the end still require to run it at full resolution. As can be seen in sections 2.2.1 and 2.3, this includes ICP, the mesh hierarchy approach (Turk and Levoy, 1994) and matching with intersection points (Yang and Medioni, 1991). Some methods, like caching (Simon, 1996) and branch–and–bound (Gelfand and Mitra, 2005) have even larger space requirements than loading just the data points.

The alignment step in ICP has the same problem – if we cannot hold the point pairs in memory, we also cannot hold their corresponding distances in memory. This makes straightforward computation of the mean squared error impossible. The cross–covariant matrix needed in the common quaternion based (section 2.2) and singular value decomposition based (section 2.2.1) alignment methods is not computable, since it requires multiplication of matrices of size $3\text{x}N_S$ and $N_S\text{x}3$, where N_S is the amount of points in the source data set. The least quadratic approximation approach and all the relaxation methods mentioned in section 2.3 all require full resolution data in memory, while the non–deterministic methods have even more space overhead.

(Pulli, 1999) actually describe their algorithm running on data sets that fit our definition of "large". However, they work with multiple overlapping parts of the whole 3–D shape, that are to be registered together. Their approach registers the parts in pairs, and each registration creates constraints to use in the final registration of all the parts. Even though the final result is a large 3–D object, the proposed algorithm must have access to smaller parts that constitute that object.

In general, the registration literature focuses a lot on time–space trade–off in the benefit of time requirements. Most such methods are not interesting for our case. The approaches that acknowledge the challenge of large data sets still view it as a running time problem and try to minimize the number of iterations with the matching step in ICP, assuming that the matching step can in fact be executed. When this is not the case, there are few to none algorithms to pick from. Chapter 3 desrcibes our proposed solution both for registration and measuring distance between very large data sets.

Chapter 3

Comparing large data sets

3.1 Registration of Large 3–D Objects

3.1.1 Overview

The general 3–D registration problem can be described as follows. Given two three–dimensional objects, called the *source* and the *target*, we wish to find the transformation that minimizes the distance between them when applied to the source. As explained in section 1.3.1, we are interested in proper rigid Euclidean transformations.

The source and target may be identical, or they may overlap to some extent. Due to their initial orientation, the extent of this overlap is often not obvious. In our specific case, the target and source are results of 3–D scans of the same object at different times. The differences due to erosion over time are small, so the registration must be precise if we are to analyze them.

The scanned surfaces are represented by such large data sets, that the use of all the information concurrently is impossible. In fact, most computers are not able to load the whole object into memory. For this reason, we will need to use sub–sampling and local information.

3.1.2 Solution

Let the target T and the source S be the data sets we wish to register against each other. We wish to find the spatial transformation of S that minimizes the distance between S and T, according to some chosen metric. Due to the large nature of the data sets, even measuring the distance between them becomes a challenge. The current section deals with the registration problem, while section 3.2 describes a method for measuring the distance.

The proposed registration technique is summarized in table 3.1.

1. Perform rough registration on sub–sampled versions of S and T.
2. Apply the transformation found in step 1 to S, resulting in the data set S'.
3. Find the combined bounding box of S' and T.
4. Divide S' and T into two octree data structures on disk, using the bounding box found in step 3 to bound both root nodes. Let the octree of T have a fixed overlap between its subspaces.
5. Find matching leaf node pairs in the two octrees created in step 4.
6. Register the surface parts corresponding to the leaf node pairs found in step 5.
7. Now it is possible to:
   - Apply the registrations found in step 6 to the corresponding surface parts and proceed with distance measuring and analysis.
   - Try to find a global registration for S and T, using the local registration information found in step 6.

Table 3.1: Summary of the registration method.

Data representation

We assume the source and target object data is available on disk. The objects may be organized as point sets or sets of geometric entities comprised of points. The ordering of the geometric entities may be arbitrary. If the data is represented as a line segment set, extra case must be taken so a geometric face does not get split between two parts, once the division starts. The technique does not currently work with implicit and parametric surfaces.

Knowledge of the boundaries of the space to be subdivided is required. If this information is not present, a run through the original data to obtain the bounding boxes for both the source and the target must be done.

Rough alignment

The first task is to roughly align the two data sets S and T. For this purpose, any conventional registration method, such as ICP will suffice. The algorithm can be run with a uniform sub–sampling of the data sets, but often requires good initial parameters in order to not get stuck in a local minimum. (Pulli, 1999) Such initial parameters can be found with 3–D processing software and supplied manually by the programmer. The proposed approach is to uniformly sub–sample both data sets, so that they can be loaded with the software of choice. Let S^u and T^u be uniformly sub–sampled point sets from S and T, respectively. The sampling probability must be set so the size of the resulting objects is manageable for the tasks ahead. We align S^u to T^u in the software and extract the aligning transformation to use as initial parameters.

Given the initial parameters and a uniform sub–sampling of the data sets, we attempt to further align S^u and T^u with the registration method of choice. The resulting transformation must then be applied to each point of S as we read them from disk and store the transformed points back to disk as the new 3–D object S' (remember that the data sets are too large to hold in memory at once). This will often work well enough that the error is not visible to the human eye, but since we are interested in the erosion of the surfaces, the differences we are looking for may in fact be too small to see with the human eye. The sub–sampling approach to registration only uses some of the information of the data, while we are in fact interested in the displacement error of each and every point in the data sets. Unless we decrease the registration error as much as possible, we may not be able to differentiate it from the effects of erosion. For these reasons, we wish take into consideration all the available information and start dividing the data sets to look at them locally.

Bounding box combination

Before dividing the data sets S' and T into octrees, we wish to ensure that the two octrees represent the same part of space. In other words, we want the bounding boxes of the two root nodes to be identical. For this purpose, we choose the strictly larger bounding box of S' and T. If none of them is strictly larger than the other, we choose their combination.

More precisely, let the bounding box of S' be described by the eight corner points $B_{S'} = \{\mathbf{b}_{S'1}, ..., \mathbf{b}_{S'8}\}$ and the bounding box of T be described by the eight corner points $B_T = \{\mathbf{b}_{T1}, ..., \mathbf{b}_{T8}\}$. We choose as root bounding box B for both octrees the minimal axis–aligned bounding box of the point set $B_{S'} \cup B_T$.

Division of data

Given the data sets S', T and the bounding box B described above, we divide both data sets into octree representations, where each leaf node represents a file on disk containing a surface part of the whole object. Since we will be registering the surface parts at full resolution later, each file may not contain more than a user specified amount of data. The technique is summarized in table 3.2

We want the octree representing the data shape T to have a fixed overlap between its subspaces. In other words, if a node represents a space with center point $\mathbf{c}$ and radius $\mathbf{r}$, we want it to have the same center point, but radius $\mathbf{r} + \begin{bmatrix}\delta & \delta & \delta\end{bmatrix}'$, for some constant δ. This means that any point in T may be added to several leaf nodes in the octree. Modifications to the code for building the octree are straightforward, most notably we must add a series of checks to the procedure that decides which child to add a point to and allow it to add the point to multiple children.

This multiplies the running time of the algorithm by a constant, though the time complexity remains unchanged asymptotically.

```
FUNCTION divide_data
INPUT: data_file, bounding box $B$, max_points_per_file $m$
    1. root = new octree node
    2. set root's bounding box to $B$
    3. for each data point $\mathbf{p}$ in data_file:
    4.     input_point_in_tree(root, $\mathbf{p}$, $m$)

FUNCTION input_point_in_tree
INPUT: node root, point $\mathbf{p}$, max_points_per_file $m$
    1. n = leaf node for $\mathbf{p}$, from descendants of root
    2. if n.points_written < $m$:
    3.     write $\mathbf{p}$ to file represented by n
    4.     n.points_written++
    5. else:
    6.     split_points_to_children(n)
    7.     input_point_in_tree(n, $\mathbf{p}$, $m$)

FUNCTION split_points_to_children
INPUT: leaf node n
    1. create eight children for n
    2. for each point $\mathbf{p}$ in file represented by n:
    3.     c = correct child of n for $\mathbf{p}$
    4.     write $\mathbf{p}$ to file represented by c
    5. delete file represented by n
```

Table 3.2: Object division algorithm

Local registration

Once the data division is done, we can match leaf nodes in one octree with the corresponding leaf nodes in the other octree. There are cases, when no such match can be done, often because the target and source data sets do not overlap completely. Then, a leaf node may for example represent a surface part in the source that does not exist in the target. Such surface parts are of no interest to us.

Leaf nodes may also lack a match as a result of different scanning resolutions used when the data was collected. If one of the objects contains more points than the other, the division procedure could produce one octree that is much deeper on average than the other. In this case, a leaf node's match in the other octree may be a non–leaf node. Section 3.2.2 describes in detail how to match leaf nodes between the octrees in all possible cases.

Let $P_{S'}$ and P_T be surface parts represented by a match of leaf nodes $L_{S'}$ and L_T and belonging to S' and T, respectively. Since the leaf nodes have identical positions in their octrees and the octree roots have the same bounding box B, we know that both leaf nodes have the same center point $\mathbf{c}$. Let $\mathbf{r}_{S'}$ be the radius of the bounding box of $L_{S'}$. Then, the radius of the bounding box of L_T is $\mathbf{r}_T = \mathbf{r}_{S'} + \begin{bmatrix}\delta & \delta & \delta\end{bmatrix}'$.

During registration, we wish to find the closest neighbor in the target to every point in the source and find a transformation that brings all these neighbors as close together as possible. Let $P^*_{S'}$ be the surface part $P_{S'}$, but transformed in such a way that the distance between $P^*_{S'}$ and T is as small as possible. For every point $\mathbf{p}^*_i \in P^*_{S'}$, $i = 1, ..., N_P$, where N_P is the amount of points in $P^*_{S'}$, let $\mathbf{t}^*_i$ be its closest point in T. Let Γ be the transformation used on $P_{S'}$ to obtain $P^*_{S'}$, so that it transforms each point $\mathbf{p}_i \in P_{S'}$ to $\mathbf{p}^*_i \in P^*_{S'}$. Let the objective function for registration of $P_{S'}$ and T be

$$e_T = \frac{1}{N_P} \sum_{i=1}^{N_P} \|\mathbf{t}_i - \Theta(\mathbf{p}_i)\| \geq \frac{1}{N_P} \sum_{i=1}^{N_P} \|\mathbf{t}^*_i - \Gamma(\mathbf{p}_i)\|,$$

where Θ denotes some proper rigid transformation and for every point $\Theta(\mathbf{p}_i)$, $\mathbf{t}_i$ is its closest point in T. Similarly, let the objective function for registration of $P_{S'}$ and P_T be

$$e_{P_T} = \frac{1}{N_P} \sum_{i=1}^{N_P} \|\mathbf{t}_i - \Theta(\mathbf{p}_i)\| \geq \frac{1}{N_P} \sum_{i=1}^{N_P} \|\mathbf{t}^*_i - \Gamma(\mathbf{p}_i)\|,$$

where for every point $\Theta(\mathbf{p}_i)$, $\mathbf{t}_i$ is its closest point in P_T. We are interested in the conditions for when the latter equation can become an equality, because Γ was defined as the best possible transformation for the part $P_{S'}$. In other words, Γ is the answer to the registration problem for this surface part. We see that the condition for this situation is that the surface part P_T contains all the points $\mathbf{t}^*_i$. It follows that

> If for every $i = 1, ..., N_P$, $\mathbf{t}^*_i \in P_T$, then the objective function for registering $P_{S'}$ with P_T and the objective function for registering $P_{S'}$ with T have the same global minimum.

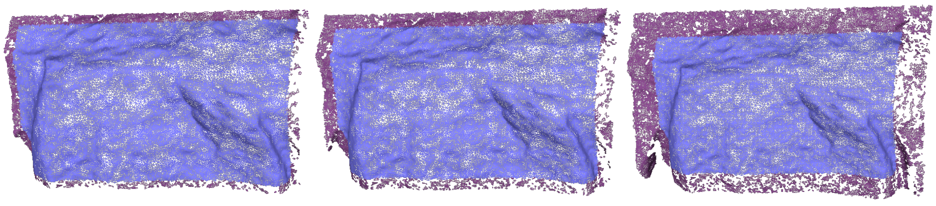
The consequence of this is that we can register the matching surface parts and find the global minimum, without the need of the whole target object. This makes it possible to perform registration at full resolution, by doing it part–by–part.

It is difficult to be sure that the condition above applies to the surface part in question. The probability of the condition being fulfilled rises with the quality of the rough registration between S and T and the size of the overlap constant δ. Figure 3.1 shows a visual comparison of different values of δ. In anticipation of future work in this area, the only method now seems to be experimentation with the overlap constant and manual analysis of the results.

Note that the above local registration technique often results in different surface parts finding different optimal transformations. This may point to the mentioned condition not being fulfilled for some of these parts or to the fact that a global registration of the whole object that is also optimal for all its surface parts does not exist. This will be the case if there were any problems with the scanning process, a lot of noise in the data or the object was somewhat wrongly ”glued” together from several scans. Our experience is that such factors are often present in real world data.

Application to global data sets

Once we have obtained the local transformations of the surface parts, we may try to use this information to register S' with T. Given a transformation from some part of S', we propose the following two methods:



(a) Overlap of 3 millimeters. **(b)** Overlap of 5 millimeters. **(c)** Overlap of 10 millimeters.

Figure 3.1: Visual comparison of different overlap constants when a surface part (in blue) is matched with its target (in pink).

1. **Direct application.** Apply the transformation to S'.
2. **Initial parameters.** Perform registration between uniform sub–samplings of S' and T, using any registration algorithm and the local transformation as initial parameters.

We expect the first method to work in perfect conditions – the local transformation is indeed globally optimal as described in the previous subsection and the effects of scanning errors and noise in the data sets S' and T are negligible. Unfortunately, such perfect conditions are rarely obtainable with real world data.

The second method hopes that the local transformation steers the new registration process into the global minimum. We expect this method to handle scanning errors and noise better than the first and it may even work if the local transformation is not globally optimal.

The data division process may result in a large amount of surface parts of the source and target objects, so it is a point of interest which local transformations we use to search for the global one. We have singled out some features of surface parts that may make them suitable candidates:

1. **Farthest part before local transformation.** Given two corresponding surface parts $P^i_{S'}$ and P^i_T of S' and T, we can measure the distance $d(P^i_{S'}, P^i_T)$ between them. The farthest part is the surface part $P^i_{S'}$, such that $d(P^i_{S'} P^i_T) \geq d(P^j_{S'}, P^j_T)$ for all $j \neq i$.
2. **Closest part before local transformation.** Similarly, the closest part is the surface part $P^i_{S'}$, such that $d(P^i_{S'}, P^i_T) \leq d(P^j_{S'}, P^j_T)$ for all $j \neq i$.
3. **Farthest part after local transformation.** Given two corresponding surface parts $P^i_{S'}$ and P^i_T of S' and T, and the resulting local transformation Θ, we can measure the distance $d(\Theta(P^i_{S'}), P^i_T)$ between them. The farthest part is the surface part $P^i_{S'}$, such that $d(\Theta(P^i_{S'}), P^i_T) \geq d(\Theta(P^j_{S'}), P^j_T)$ for all $j \neq i$.
4. **Closest part after local transformation.** Similarly, the closest part is the surface part $P^i_{S'}$, such that $d(\Theta(P^i_{S'}), P^i_T) \leq d(\Theta(P^j_{S'}), P^j_T)$ for all $j \neq i$.
5. **Biggest improvement after local transformation.** This is the surface part $P^i_{S'}$ such that $d(\Theta(P^i_{S'}), P^i_T) - d(P^i_{S'}, P^i_T) \geq d(\Theta(P^j_{S'}), P^j_T) - d(P^j_{S'}, P^j_T)$ for all $j \neq i$.

6. **Largest part.** The largest part is the surface part of S' that contains the most points.

7. **Combined corner parts.** The combination of corner parts of S' – these are the parts that contain the points closest to the corners of the bounding box of S'.

8. **Average transformation.** Instead of choosing just a couple surface parts, we use the transformation that is the average of all the local transformations obtained from the parts $P^i_{S'}$.

The farthest part before local transformation is the surface part that the rough registration had the worst effect on. In other words, the local minimum found by the rough registration does not give a good distance value for this surface part, and we hope that the proposed local transformation pushes the global registration into a better local minimum.

The closest part before local transformation is the surface part that the rough registration had the best effect on. We expect the proposed local transformation to be very similar to the one obtained after rough registration and have little further effect when applied globally.

The farthest part after local transformation is the one with the worst end result. This may mean that the surface part contains more noise or scanning errors than the other parts and may be of interest to analyze manually. The effect of applying its proposed local transformation globally is difficult to predict.

The closest part after local transformation is the one with the best end result. For this reason, one may expect good results when using its local transformation.

The surface part with biggest improvement after local transformation may often coincide with part number 1 in the list above. It is interesting, because the different distance value before and after local registration may point to there being a large difference between the resulting transformations too. Thus, there is a bigger chance that the local transformation guides the global registration into a different local minimum.

The largest part may be worth trying, simply because it contains more data than the other surface parts.

The combination of corner parts may counteract some scanning errors – if the source and target objects were ”glued” together from several scans, any error in the ”gluing” process could propagate outwards and become larger between parts on the object that are far away. Therefore, we combine the corner parts into one and try their proposed local transformation globally. If it is not possible to register the combined corners well enough, this may indeed point to the existence of such ”gluing” errors in the data.

The notion of somehow ”averaging” the local transformations to include all the available information in the solution is tempting, but presents several technical difficulties. In fact, ”an average rotation” is a concept that is difficult to define. Let the transformations be represented by a rotation, followed by a translation. We can extract the Euler angles from the rotation matrix and average them, thus creating a new rotation. The averaging of multiple translations is also straightforward. Let for example the rotation of surface part $P^i_{S'}$ be a rotation around the z–axis by α_i, then around the x–axis by β_i and then around the z–axis by γ_i. The first rotation around the z–axis of our average matrix would have angle

$$\alpha = \frac{1}{N}\sum_{i=1}^{N}\alpha_i,$$

where N is the amount of surface parts of S'. The problem is that the following rotations are affected by this z–axis rotation. For any surface part $P^i_{S'}$, its rotation around the x–axis by β_i has changed meaning, because it is performed after the z–axis rotation by α and not α_i. Hence, we are no longer averaging the local transformations of the surface parts, but something entirely else. Another way to view this, is that the pertinent information in the local transformations lies in the combination of the Euler angle rotations, and we lose this information if we average them one–by–one.

To circumvent these problems, we may try to express the local transformations in quaternion form. By use of spherical linear interpolation (Slerp) (Shoemake, 1985), we can trace a path between two quaternions through 3–D rotations. The average of two quaternions is then the rotation corresponding to the point in the middle of this path. However, this technique is meant for averaging only two quaternions. We refer to (Buss and Fillmore, 2001) for the generalization of Slerp and to (Alexa, 1997) for an approach to linear transformation approximation.

We have tested the two methods of global application by using the local transformations of several of the above parts in our data. The results of this are presented and discussed in chapter 4.

3.2 Measuring Distance Between Large 3–D Objects

3.2.1 Overview

We wish to measure the spatial distance between two 3–D objects whose data representation is too large to contain wholly in computer memory. The 3–D objects are assumed to be represented as sets of points. The data may be arranged in any way (triangles etc.), as long as we have access to the underlying point set. In addition to an overall distance between the objects, we are also interested in the per–point distance between each point in one of the objects to the other. We assume these objects have already been registered, meaning that any aligning transformations required before the distance measure have been performed.

Regardless of which metric is chosen for the task, surface points in one of the objects must somehow be matched with surface points in the other. This introduces a problem, because conventional methods to find the nearest neighbor of a point need access to the whole data for searching purposes. For example, a *k*-d tree provides logarithmic search times, but given a data set of size n, the space requirements are $O(n)$. In our case, we assume n to be so large, that this becomes unfeasible.

The proposed solution is to first split both 3-D objects into parts, and measure the distance between the resulting smaller parts conventionally. The distances between the parts can then be summed together and normalized over the total number of points in the large 3-D objects. This requires a procedure to divide both objects into small enough parts, without losing track of which corresponding pieces must be measured against each other.

1. Divide both surfaces into small enough parts.
2. Match the divided surface parts into pairs.
3. Calculate distance between each pair of surface parts.
4. Add and normalize results.

Table 3.3: Overview of distance measuring solution

It is also important that the resulting distance produced by this approach does not differ from the result that would be produced by a hypothetical machine with the capacity to measure the distance by using a conventional method.

The solution makes a couple of assumptions that will be explained thoroughly in the following sections. Mainly, we assume that only parts of the surfaces that overlap to some degree are of interest. In other words, we assume there is an upper threshold for the distance between the points of the surfaces and we disregard values above it. The other assumption of note is that the objects are dense – there are no large gaps between neighboring points. This constraint is usually fulfilled when the objects are the results of a 3–D scan.

Table 3.3 gives a summary of the solution. The following section starts with the mathematical background and elaborates on each point in the table in the subsections. Then follows a discussion of potential problems with this solution and alternative metrics to use for the distance calculations.

3.2.2 Solution

We will describe the approach in general terms with regard to the distance metric, before looking into specific ones. Let f be a function that takes a three dimensional point $\mathbf{x}$ and a three dimensional surface Y and outputs the distance between $\mathbf{x}$ and Y. We define the surface distance $d(X, Y)$ between data sets X and Y as the normalized sum of distances between each point of X and Y:

$$d(X, Y) = \frac{1}{N} \sum_{\mathbf{x} \in X} f(\mathbf{x}, Y),$$

where N is the number of data points in X.

Let S and T be aligned data sets. We wish to divide both the sets into parts denoted $S_1, S_2, ..., S_n$ and $T_1, T_2, ..., T_m$ respectively, such that no part contains more than some constant k points and

$$\bigcup_{i=1}^{n} S_i = S \qquad \text{and} \qquad \bigcup_{j=1}^{m} T_j = T$$

By working on single pairs of corresponding surface parts at a time and restricting the constant k, we can be certain that the memory requirements of the algorithm do not exceed the capacities of the computer available. We will be looping through each point in each part of the S data set, and we do not wish to count the distance of the same point multiple times, so we require that there is no overlap between the parts $S_1, S_2, ..., S_n$:

$$S_i \cup S_j = \emptyset \qquad \text{for } i \neq j$$

After the surface division process, we wish to find for each part $S_i \in \{S_1, S_2, ..., S_n\}$ a corresponding part $T_{j_i} \in \{T_1, T_2, ..., T_m\}$ such that for each point in S_i, T_{j_i} contains the point's true closest neighbor in T. We then loop through each part of S and sum the distances between its points and the corresponding part of T. If this condition is fulfilled, we have for $i = 1, 2, ..., n$:

$$f(\mathbf{x}, T_{j_i}) = f(\mathbf{x}, T) \qquad \text{for all } \mathbf{x} \in S_i$$

Let N_S be the number of points in S and N_{S_i} be the number of points in S_i for $i = 1, 2, ..., n$. Combining all conditions given so far, we have

$$\frac{1}{N_S} \sum_{i=1}^{n} \sum_{\mathbf{x} \in S_i} f(\mathbf{x}, T_{j_i}) = \frac{1}{N_S} \sum_{\mathbf{x} \in S} f(\mathbf{x}, T) = d(S, T)$$

and so we see that our approach gives the same result as a conventional distance measurer that does not split up the data sets beforehand.

Surface division

First, both surfaces S and T have to be split into smaller parts. As explained above, the following constraints must hold, in order for our approach to work and produce correct output:

1. Each part has no more points than some constant k
2. The parts are *collectively exhaustive* with respect to the surface being divided
3. The parts of surface S are *mutually exclusive*
4. We must be able to match pairs of surface parts from the two objects, such that they contain points in the same part of space
5. Within each pair, one part has to contain all of the true closest neighbors of the points in the other part

Constraints 1-3. The first three of these constraints are fulfilled when we use the octree surface division algorithm described in section 3.1.2 and set the maximum allowed points in an octree node to k. This restricts the size of each surface piece, and the pieces are collectively exhaustive and mutually exclusive, as a direct result of the octree algorithm.

Constraint 4. Constraint number four becomes challenging, if we naively apply the octree algorithm to both surfaces. The solution is to ensure that the root node of each octree represents the same bounding box in space. We can find the bounding boxes of each surface by simply looping through all points once as they are read from disk (remember that the surfaces are too large to hold in memory). Then we use the *minimum bounding box* of these two bounding boxes for both octree root nodes. In other words, if surface S has a bounding box represented by the set of corner points A and surface T has a bounding box represented by the set of corner points B, then we use the bounding box of $A \cup B$ as the space at the outermost level of both octrees. This will somewhat change the division of one or both surfaces and may add to the depth of the octrees, but gives the following useful results:

Let V and U be the octrees resulting from division of data sets S and T respectively. We enumerate the children of each non-leaf node from zero to seven and say that two paths α and β in the octrees are equivalent ($\alpha = \beta$), if their choices of child to travel to agree at each step. Then,

- If α is a path from the root node of V to node a and β is a path from the root node of U to node b, and $\alpha = \beta$, then a represents a (non-strict) subspace of the space represented by b.
- If α is a path from the root node of V to node a and β is a path from the root node of U to node b, and β is a *prefix* of α, then a represents a (strict) subspace of the space represented by b.

These observations allow us to find matching pairs of surface parts. The details of this procedure will be explained in section 3.2.2.

Constraint 5. Once we have matching parts, we want one of the parts to contain the true neighbors of the points in the other part, as per constraint number five above. This becomes problematic along the borders of the surface part, since we cannot be sure that the true neighbor of a border point does not lie in the bordering octree node.

This observation is troublesome, because to ensure ourselves that we have found the correct neighbor for a border point, we would have to find all the bordering octree nodes and check the distance to all points contained within them. This would have to be done for each border point in each octree leaf node and most importantly, since multiple surface parts are searched we no longer can ensure that the memory requirements are constrained by the constant k (constraint number one above), without submitting to an unsavory number of disk accesses per data point. The worst case scenario are border points that are near the center of the outermost bounding box of the octree, which adds all branches from the root node into consideration when searching for closest neighbors.

To avoid this, we require the space of each octree node of surface T to overlap with all of its neighboring spaces. We choose an overlap constant δ and expand the bounding box of each node with δ in all eight directions. When comparing a surface part S_i with its match T_{j_i}, T_{j_i} will then contain the nearest neighbor in T of any point in S_i, as long as the distance between them is less than δ. Thus, we satisfy constraint number five, but add the following constraint to our data:

$$f(\mathbf{x}, T) < \delta \qquad \text{for all } \mathbf{x} \in S$$

This will be fulfilled in cases where the data points are dense and in cases where point–to–point distances above some threshold are to be ignored. 3–D scanners often output data in the shape of densely packed small triangles and it is possible to set a maximum of how much change is done by erosion to a material surface over a certain time, so both of these points apply to the case described by this thesis.

The above means that each point may be added to multiple nodes during construction of the octree, so the parts $T_1, T_2, ..., T_m$ are no longer mutually exclusive. Note however, that all constraints and observations discussed above still hold.

Finding matches for surface parts

Once both surfaces S and T have been divided as detailed above, we wish to loop over the parts $S_1, S_2, ..., S_n$ and for each S_i, find the matching part T_{j_i}. For this purpose we use the octree representations of the two surfaces, but we disregard leaf nodes that contain no points. Then there is a bijection between the parts $S_1, S_2, ..., S_n$ and the leaf nodes in the octree of surface S. The same holds for surface T, though the surface parts do overlap, as described in the previous section.

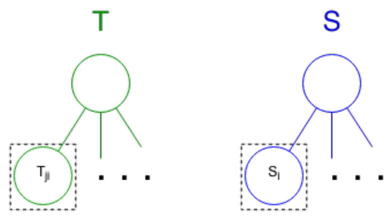
For each leaf node in the octree of surface S, we search the octree of surface T for a matching node. Let V and U be the octrees resulting from division of data sets S and T respectively. Let a be the leaf node representing the surface part S_i currently under inspection and α be the path from the root node of V to a. We use the same definition of equality of paths as in section 3.2.2. There are four possible cases of interest: (see also figure 3.2)

1. There is a path β from the root node of U to a *leaf node* b and $\alpha = \beta$.
2. There is a path β from the root node of U to a *leaf node* b and β is a prefix of α.
3. There is a path β from the root node of U to a *non-leaf node* b and $\alpha = \beta$.
4. There is no path in U equal to α and there is no path from the root node of U to a *leaf node* that is a prefix of α. In other words, none of the first three cases hold.

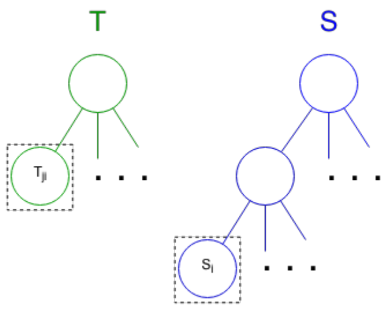
Case 1. The first case is straightforward. By observation 1 in section 3.2.2, node b represents the surface part T_{j_i} that matches S_i. The distance between them can be calculated using any 3–D distance algorithm of choice.

Case 2. By observation 2 in section 3.2.2, the bounding box of node a is fully contained in the bounding box of node b. Thus, node b represents the surface part T_{j_i} to match S_i, as in case 1.

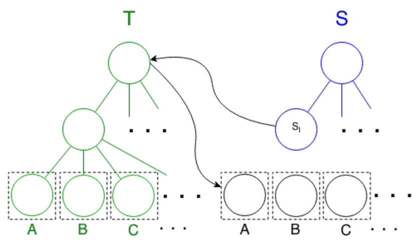
Case 3. Here the same observation is made about the bounding boxes of the nodes as in case 2, but the non–leaf node b does not directly represent any of the surface parts $T_1, T_2, ..., T_m$. In fact, it contains multiple of them, since the node is further split into children nodes. We cannot combine them into one part without possibly violating the memory constraint on our algorithm. The solution is to further divide S_i into equivalent parts and perform distance measuring on each of them. For each point in S_i, we use the already existing structure of T's octree and find which subspace of b it falls into. We end up with a partition of S_i where each part matches one of $T_1, T_2, ..., T_m$. Pseudo code follows:



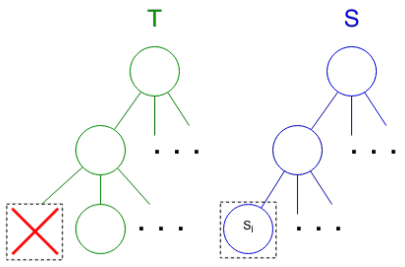
(a) Case 1.



(b) Case 2.



(c) Case 3.



(d) Case 4.

Figure 3.2: **(a)** S_i is matched with its equivalent node in T. **(b)** S_i is matched with a higher level node in T. **(c)** S_i is split into parts using T's octree and each part is matched with its equivalent leaf node in T. **(d)** S_i's equivalent node in T is empty, so there are no points close enough to measure against in T. S_i can be safely ignored.

```
1. M = empty map of leaf nodes to point sets
2. for each point $\mathbf{x}$ in $S_i$
3.   node = root node of U
4.   while node is not a leaf do
5.     leaf = node.find_correct_child($\mathbf{x}$)
6.   if node is a non-empty leaf then
7.     M[node] = M[node] $\cup$ $\mathbf{x}$
8. for each leaf node $\lambda$ in M
9.   calculate_distance($\lambda$.data_points, M[$\lambda$])
```

Note the conditional statement in line 6, resulting from the fact that we disregard leaf nodes that have no data points. If a point $\mathbf{x} \in S_i$ would fall into such a leaf node, then its distance to any point in T is larger than the threshold δ described in section 3.2.2 and should not be taken into consideration. This is a similar situation to case 4, described below.

Case 4. In this case, the 3–D object T simply has no points residing in the bounding box of node a. We are interested in points in T that have a distance to points in S_i less than the threshold δ. If such a point were to exist, it would have to be closer than δ to the bounding box of node a. Since the octree of T is built of subspaces using an overlap of δ in all directions, this point would force the creation of a node b, such that the path from the root node of V equals α. We see that either case 1 or case 3 would then be correct, which is a contradiction. Hence, there are no points $\mathbf{x} \in S_i$ such that $f(\mathbf{x}, T) < \delta$, so we can safely disregard all points in S_i from the distance measuring.

3.2.3 Distance Calculation

Since the parts $S_1, S_2, ..., S_n$ are mutually exclusive, we can safely sum the distance results from cases 1 - 3 from section 3.2.2 to a total sum, without counting the distance of any point multiple times. Since the parts are collectively exhaustive with regard to S, we can be sure that all points in S are part of the calculation, apart from those we wish to disregard. Hence, we have obtained the desired distance $d(S, T)$. The distance can also be normalized over the amount of points used in the calculation.

3.3 Potential Issues

3.3.1 Overlapping and Point Clusters

As explained in section 3.2.2, we use a constant overlap δ for the subspaces of T's octree, regardless of the size of the subspace itself. This can lead to strange situations, like a leaf node representing a space that was originally smaller than δ in one or more dimensions, but has been expanded more than twice its own size in that direction because of the overlap. We could make δ dependant on the size of the subspace, something like $\delta_x = 0.01T_{jx}$, where T_{jx} is the size of node T_j's bounding box in the x–direction, and the same for the other two dimensions. However, remembering constraint number 5 in section 3.2.2, we can no longer be sure that the distance calculation is correct. δ can now become arbitrarily small, so no leaf node of T is guaranteed to contain the true neighbors in T for the border points in its matching S node.

We conclude that δ must be kept as a constant. The only technical problem that can arise from this is during the construction of the octree. If there are geometrically very small clusters containing very many points in the object T, then it is possible that the construction of the octree ends up in an endless loop. This happens because the large amount of points calls for a division of the octree node, but the node's children are expanded to be as large or larger than their parent because of the δ overlap. Then each child contains the same amount of points as the parent node, so the algorithm tries to divide the children in the same way, ending up in a loop.

In order for this problem to arise, the bounding box dimensions of the point cluster cannot be much larger than δ, but still contain more points than our memory constraint k. In practice, this rarely occurs and is best dealt with in a case–by–case basis. Often, the cause will be a large amount of duplicate points, which can be removed before building the octree. The algorithm can be kept stable by enforcing a maximum depth of the octree and noting the cases when it wishes to exceed this depth for further examination by the programmer.

3.4 Distance Metrics

We have explained the distance measuring approach somewhat abstractly with regard to the function that actually gives us the distance in space. All that we require is a function f that takes a three dimensional point $\mathbf{x}$ and a three dimensional surface Y and outputs the distance between $\mathbf{x}$ and Y. In this section, we mention some alternatives for such a function and metrics to use for the distance between the 3–D surfaces themselves.

3.4.1 Sum of Point Distances

As explained in section 3.2.2, we define the distance between surface S and T as

$$d(S,T) = \frac{1}{N_S} \sum_{\mathbf{x} \in S} f(\mathbf{x}, T),$$

where N_S is the number of points in S. This has been assumed throughout the discussion so far. Note that this is not technically a metric, since it is not necessarily symmetric. To see this, imagine that S consists of only the origin, while T consists of two arbitrary points with different distances to the origin. Then, $d(S,T) \neq d(T,S)$, simply because T has more points than S. To solve this, we would have to define the metric as

$$d(S,T) = \min\{\frac{1}{N_S} \sum_{\mathbf{x} \in S} f(\mathbf{x}, T), \frac{1}{N_T} \sum_{\mathbf{y} \in T} f(\mathbf{y}, S)\}$$

or

$$d(S,T) = \max\{\frac{1}{N_S} \sum_{\mathbf{x} \in S} f(\mathbf{x}, T), \frac{1}{N_T} \sum_{\mathbf{y} \in T} f(\mathbf{y}, S)\}$$

This effectively doubles the computational work when calculating the distance, since we must run the algorithm twice, including splitting up the objects. The proposed approach is to use the distance function as defined previously, but be aware of the potential pitfalls. One should be careful not to compare the results of two distance calculations, where the roles of the objects have been ”reversed”. In other words, keep in mind that $d(S, T)$ and $d(T, S)$ are not measures of the same thing.

3.4.2 Hausdorff Distance

In our case of discrete data sets, we can define the Hausdorff distance between D and M as

$$d(S, T) = \max\{\max_{\mathbf{x}\in S} f(\mathbf{x}, T), \max_{\mathbf{y}\in T} f(\mathbf{y}, S)\},$$

assuming that $f(\mathbf{x}, Y)$ returns the distance between the point $\mathbf{x}$ and its closest point in Y. The Hausdorff distance is a well–defined metric on the space of 3–D objects and has the benefit that $d(S, T) = d(T, S)$. However, it again requires us to do double the computational work, as discussed in section 3.4.1.

Note that the Hausdorff distance basically measures the largest distance between closest neighbors found in the objects. This means that noise and erroneous scan data will more often than not be the deciding factor in the result. In our case, where we disregard distances over a certain threshold, the Hausdorff distance does not seem like a good choice. A possible scenario where it could be used, is to check that nearly identical objects have been registered correctly, such that no neighbor point pairs are too far away from each other. However, this requires data with very little or non–existent noise. Furthermore, the same goal can be obtained by using the normalized sum of point distances and requiring a result very close to zero.

3.4.3 ℓ_0 Distance

We define the ℓ_0 distance between S and T as

$$d(S, T) = \sum_{i=1}^{N_S} \#(i | f(\mathbf{s}_i, T) \neq 0),$$

where N_S is the amount of points in S and $f(\mathbf{s}_i, T)$ gives the distance between $\mathbf{s}_i$ and T, as before. In other words, the distance between S and T is the amount of points in S that have non–zero distance to T. This approach can make sense in the context of registration, since we are trying to align points and wish to maximize the amount of points that have zero distance from their target. The metric can be applied directly if we expect the points to match exactly. In real world sensor data however, we would need to define a threshold below which a distance counts as zero. This deals with the floating point data produced by for example 3–D scanners.

We refer to (Bouaziz et al., 2013) for more details about using the ℓ_0 norm as well as the ℓ_p norm for other values of $p \in [0, 1]$ for registration purposes.

3.4.4 Point Distance Function

The distance function $f(\mathbf{x}, Y)$ operates on parts of the surfaces after they have been split up. Since every surface part contains no more than some chosen constant amount of points, we can be certain that the calculation will not be too memory intensive. Let f return the distance from $\mathbf{x}$ to the closest point in Y. In order to avoid looping through all the points in Y every time, one can construct a k–d tree out of the points in Y, which takes $O(N_Y \log N_Y)$ time (N_Y being the number of points in Y). Each search then takes $O(\log N_Y)$ time.

The distance itself can be measured using the Euclidean or Euclidean squared metric. Let $\mathbf{x} = (x_1, x_2, x_3)$ and $\mathbf{y} = (y_1, y_2, y_3)$ be two three–dimensional points. Then the Euclidean distance between $\mathbf{x}$ and $\mathbf{y}$ is

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{3} (y_i - x_i)^2}$$

The Euclidean squared distance is

$$d(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{3} (y_i - x_i)^2$$

The Euclidean distance represents what we usually think of as distance and makes reasoning about the results as simple as possible for a human. If the units used by the 3–D scanner are millimeters for example, the result of an average distance of 3 millimeters between surface S and T immediately makes sense to us.

The Euclidean squared distance has the benefit of avoiding the computationally costly square root operation in the innermost loop of the distance algorithm and is therefore often the metric of choice. However, when reasoning about results, one must remember that an average Euclidean squared distance of j millimeters between two surfaces does not mean that they are j or $\sqrt{j}$ millimeters apart on average in reality. In conclusion, one should use the Euclidean squared distance when computational performance is of the essence, and the Euclidean distance when an exact answer in common distance units is required.

There are other metrics, like Chebyshev and different orders of Minkowski distance, but we do not benefit from applying these to our case.

If the surfaces we are measuring are represented as sets of triangles, then there may be a difference between the result of the point–to–point approach and the actual distance between them. For example, if a point $\mathbf{x} \in S$ lies on the line between its two closest points in T, we will obtain the distance from $\mathbf{x}$ to one of these neighbors in T. However, if we consider the line between the two neighbors to be part of the surface of T, then the distance between $\mathbf{x}$ and T is in fact zero. If we know that all the points are sufficiently densely packed, as may be the case in scanned objects, the error resulting from this will often be negligible.

Chapter 4

Results and Discussion

4.1 Data Sets

The experiments were run on real world data procured by the PRESIOUS project. The model T is a 3–D scan of the surface of a column approximately two meters in height and approximately 1.5 meters in diameter. It consists of 226 343 250 points, arranged into 75 447 840 triangles. The data S is a partial 3–D scan performed at a different time of one of the column's walls, approximately 75 centimeters tall and 30 centimeters wide. It consists of 292 386 669 points, arranged into 97 462 223 triangles. Sub–sampled versions of these data sets can be seen in figure 4.1.

Both data sets were combined from several scans, since it is difficult to scan a large object in one go.

The data S was first registered with the model T, using the singular value decomposition based ICP version described in section 2.2.1 with a point–to–point distance measuring approach and a uniform sampling of about 3 000 000 points each. The registration ran for a maximum of 200 iterations or until the error improvement from the last iteration became lower than 10^{-6}. Initial transformation parameters had to be found manually, using 3–D software and sub–sampled versions of S and T. The code was supplied by the libicp library, created by Andreas Geiger. (Geiger et al., 2012) The result of the coarse registration can be seen in figure 4.2.

Both T and S were then divided using the octree algorithm described in section 3.1.2, allowing a maximum of 1 000 000 triangles per surface part. The octree generated for the model T had a border overlap of 3 centimeters. This process produced a total of 262 parts for T and 282 parts for S.

After initial registration and division, each part of S was registered with its corresponding part in T. The registration was run for a maximum of 200 iterations, with the same error threshold of 10^{-6}, but this time done at full resolution. Corresponding parts were matched following the procedure in section 3.2.2.

The distances between the corresponding parts were measured using the mean Euclidean distance. The following pieces were chosen post rough registration as parts of interest:

Part 1: Farthest part. The part of S that had the largest distance to its corresponding part in T of all the pairs. It contained 35 937 points and had a distance of 7.58317 to its match.

Part 2: Closest part. The part of S that had the smallest distance to its corresponding part in T of all the pairs. It contained 294 points and had a distance of 0.119567 to its match.

Part 3: Largest part. The part of S that had the most points. It contained 2 957 853 points and had a distance of 1.17794 to its match.

Part 4: Corners. Two corner parts of S combined into one part. It contained 577 953 points and had a distance of 1.750950 to its match.

See also figure 4.3. We shall refer to these surface parts by their number in the above list.

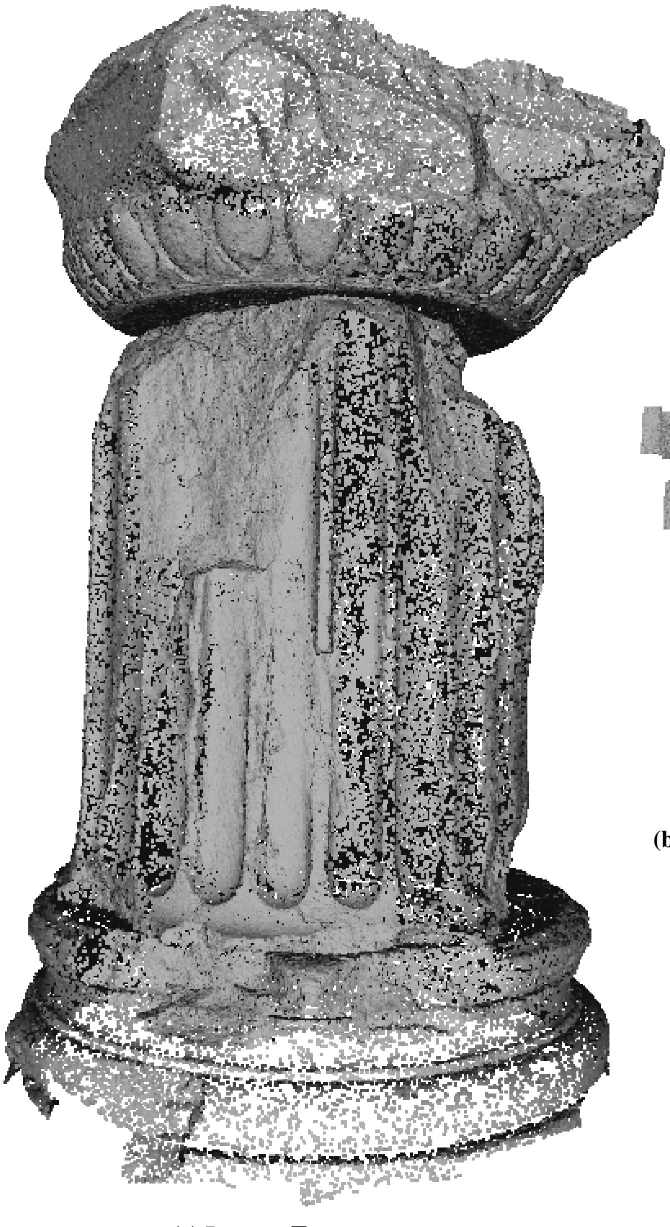
4.2 Potential Sources of Error

There are several potential sources of errors in a real world data set resulting from a 3–D scan. Noise may be induced by the 3–D scanner, resulting in data points that do not represent the actual object and disturb the registration procedure. Noisy data points that are above the distance threshold described in section 3.2.2 can be safely ignored, and will not have an effect on the results. However, noisy points that are similar to the real data points may still have an effect. If there is for some reason a lot of localized noise, its effect is hampered by the uniform sub–sampling done in the coarse registration step. However, it will have a much larger effect on the local registration done afterwards.

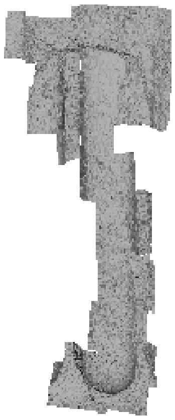
If the data set is ”glued” together from several surface scans, as it is in our case, the ”gluing” process may induce errors at the ”seams” of these individual scans. If two pieces of the object are combined together slightly wrong, this error propagates outward on the surface and has a higher effect far away from the erroneous ”seam”. Examination of corner pieces, such as part 4 described in section 4.1 may expose these errors, since the corners are relatively far away from each other. In presence of ”gluing errors”, we expect part 4 to have a comparatively worse distance result both after coarse registration and local registration.

4.3 Local Transformations

The results of local registrations on the surface parts can be seen in table 4.1. The initial distance column shows the distance between the surface parts and their matching part in T after coarse registration. The resulting distance column shows the same thing, but after local registration between the matching parts. The max. distance columns shows the largest distance found between neighboring point pairs after local registration. This is the same as the Hausdorff distance between the surface part and its match in T (see section 3.4.2 for definition).



(a) Data set T.



(b) Data set S.

Figure 4.1: (a) Sub–sampled 3–D scan of the whole column surface. (b) Another sub–sampled 3–D scan of a part of the same column. Images are to scale relative to each other.

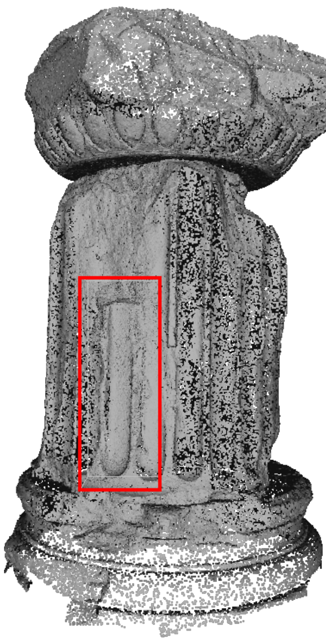
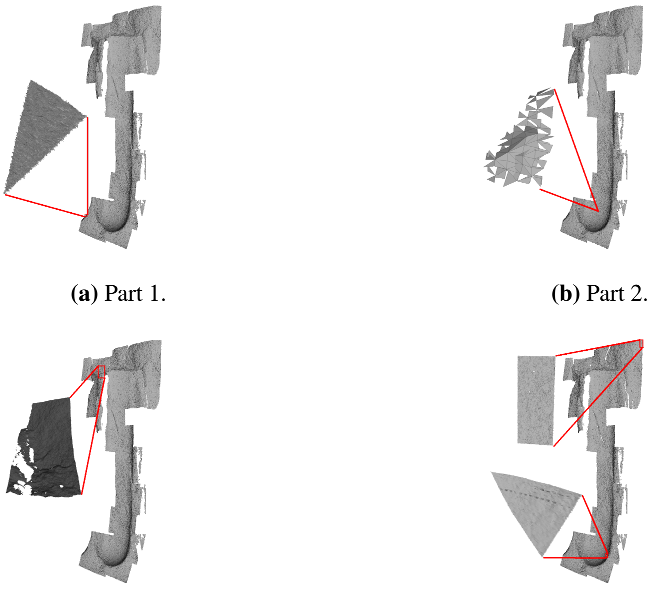


Figure 4.2: Result after coarse registration. The red rectangle shows the placement of data set S.



(a) Part 1. **(b)** Part 2.

(c) Part 3. **(d)** Part 4.

Figure 4.3: **(a)** Surface part with largest distance to its match. **(b)** Surface part with smallest distance to its match. **(c)** Surface part with most points. **(d)** Surface part consisting of two corners.

Surface part	Initial distance	Resulting distance	Max. distance
Part 1	7.583170	0.478686	1.084190
Part 2	0.119567	0.066857	0.147486
Part 3	1.177940	0.111948	0.854903
Part 4	1.750950	0.227744	1.287340

Table 4.1: Resulting average distances between surface parts after local registrations.

ICP iterations	Resulting distance	Improvement
0	68.081100	N/A
200	0.914787	67.166313
400	0.913825	0.000962

Table 4.2: Resulting average distance between S and T after different numbers of ICP iterations. The "Improvement" column shows improvement over the result in the previous row.

4.4 Application to Global Data Set

The local transformations of the parts of interest were applied in different ways to the whole data set S. The distance between the resulting transformed data set and the model T was measured using the procedure in section 3.2. The following experiments were run:

- **Direct application.** The local transformation is applied directly to the data set S.
- **Initial parameters.** Registration between S and T is run for 200 more steps, using the local transformation as initial parameters.

To be certain that any improvement is not simply a result of running more steps of the ICP algorithm, we ran registration between S and T for 200 more iterations using the identity transformation as initial parameters, and use that as the base of comparison. The results can be seen in table 4.2.

The results of the runs can be seen in table 4.3.

Surface part	Application	Global result (average distance)	Improvement over	
			200 ICP runs	400 ICP runs
Part 1	Direct	47.121000	-46.206213	-46.207175
	Initial parameters	0.913526	0.001261	0.000299
Part 2	Direct	37.240800	-36.326013	-36.326975
	Initial parameters	12.002600	-11.087813	-11.088775
Part 3	Direct	1.052460	-0.137673	-0.138635
	Initial parameters	0.913589	0.001198	0.000236
Part 4	Direct	1.294780	-0.379993	-0.380955
	Initial parameters	0.913718	0.001069	0.000107

Table 4.3: Results of testing local transformations globally.

4.5 Observations and Discussion

4.5.1 Local Registration

Results from local registration for some chosen surface parts can be seen in table 4.1. We immediately notice a clear distance improvement in all four cases. Part 1 in particular has a very large improvement over its initial distance value, which is much higher than for the other parts. Still, part 1 ends up with the worst result of the four. Since it had the farthest distance to its match out of all the surface pieces in S after coarse registration, we may assume that the sources of error described in section 4.2 had the most effect on this part. We conclude that analyzing a surface part with comparatively bad local registration result in a real world data set should be done while paying extra attention to possible errors in the data set.

Part 4 is a combination of surface parts that are far away from each other in the original data set S. Hence, if the ”gluing” error described in section 4.2 is an issue with the data set, we expect part 4 to have a comparatively high distance to its match, both before and after local registration, since the error would have ample time to propagate before both surface parts are glued to the final object. We see that this is somewhat true, with part 4 being the second worst performer of the group.

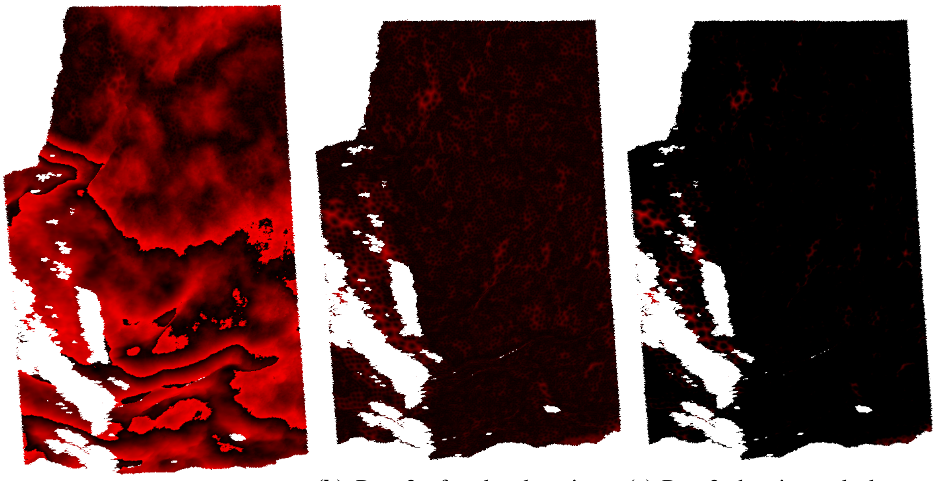
Part 2 and 3 both showed good and similar improvement after local registration, even though they are very different in size.

Under the assumption described in section 3.1.2 that the locally transformed surface parts contain the correct information about local differences between T and S, the surface parts are ready for analysis. The discrete point distances can for example be encoded as color information for visual inspection. Figure 4.4 shows this approach for part 3, where a more intense red color means a larger distance from T in that area. The surface parts are shown at full resolution.

4.5.2 Application to Global Data Set

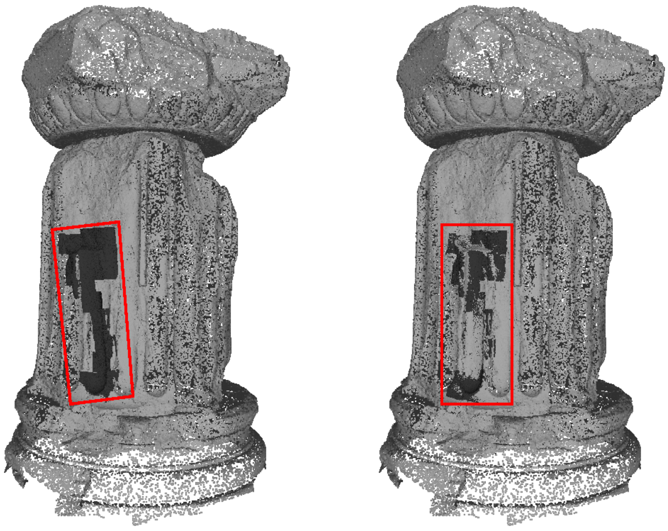
We see from table 4.3 that the application of local transformations to the global data set at best results in marginal improvements over additional iterations of the ICP algorithm, with identity transformation as initial parameters. The best result comes from using the local transformation of part 1 as initial parameters for ICP. This brings the points in S 0.000299 millimeters closer to T on average than if we don't use the local transformation. Interestingly, the worst performer was direct application of the same part's local transformation to S, which brought the points in S 46.207175 millimeters on average further away from T, than the basic ICP runs.

Overall, using local transformations as initial parameters for further global registration performs better than direct application to the global data set. This makes sense, since we can not expect the distance function of the whole surface to have the exact same local minima as the distance function for a smaller part of the surface in real world data. Any source of error, be it outlying points, noise or other problems with the scanning process would hinder this ideal scenario from happening. For visualization, a sub–sampling of part 1's results can be seen in figure 4.5. When running further global registration, we try



(a) Part 3 after coarse registration. **(b)** Part 3 after local registration. **(c)** Part 3 showing only largest distances.

Figure 4.4: Visualization of surface part distance. Black color denotes zero distance from T. **(a)** The most intense red color denotes a distance of 3.24 millimeters from T. **(b)** The most intense red color denotes a distance of 0.85 millimeters from T. **(c)** Same as (b), but only showing distances larger than 0.2 millimeters.



(a) Direct application. **(b)** Used as initial parameters.

Figure 4.5: Local transformation of part 1 applied to the global data set.

to minimize the effects of such errors, while hopefully benefiting from starting in the local minimum found in the smaller surface part's distance function.

Direct application of part 1's local transformation to S gives especially bad results. We see from table 4.1 that part 1 does not agree well with the coarse registration of S and T. Local registration therefore moved part 1 a larger distance than the other parts, and the same transformation then moves the whole data set S too far away from T.

Using the local transformations of the different surface parts as initial registration parameters performed very similarly, with the exception of part 2, which had much worse results than the others. We assume this is because of the low amount of data points in that surface part. With only 294 points taken into consideration, the local distance function may be too different from the global one, to find any meaningful correspondences between their local minima. After a certain threshold however, it seems that the amount of points has little effect. For instance, part 3 has more than 80 times as many points as part 1, but performs only slightly worse.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

We have presented a novel approach for registering of large 3–D data sets. The technique divides both the source and target into smaller surfaces and performs local registration. We have identified the conditions necessary for the local registration to be able to reach the same global minimum as the global registration. The same technique can be used for precise measurement of point distances between registered 3–D objects.

We have tested experimentally several methods of using transformations found in the local registration to find the global registration of the large data sets. The results and discussion of these experiments have been presented.

5.2 Future Work

We see several avenues for exploration in the future. Experimentation can be done on the size of sub–space overlap in the octree of the target data set in order to examine its effects on the local registrations of the surface parts. A point of interest is at what values of the overlap constant, combined with a measure of success of the initial rough registration, can we be sure that the objective function for local registration contains the desired global minimum. The answer to this question would completely validate the part–by–part approach to registration of large data sets.

Several different ways to use the information gained by the local registration of surface parts in order to find an optimal global transformation remain to be examined. One possibility is to use the generalization of spherical linear interpolation in (Buss and Fillmore, 2001) or the linear interpolation of transformations approach in (Alexa, 1997) to see if all the local transformations can be combined in a meaningful way for our purposes. What effect the averaging of transformations that correspond to local minima in the objective function of registration has is an exciting question for the future.

We believe there is more to be gleaned from the information lying in the local surface part registrations. By combining corner parts that are far away on the original surface, it may be possible to detect gluing errors in the data set, where the original object has been erroneously combined together from several parts. In general, there is research to be done about the possible explanations of two surface parts of the same rigid object having significantly different optimal registrations with another rigid object.

Bibliography

Alexa, M. (1997). Linear Combination of Transformations. pages 380–387.

Arun, K., Huang, T., and Blostein, S. (1987). Least-squares fitting of two 3-D point sets. *Pattern Analysis and ...*, PAMI-9(5):698–700.

Besl, P. J. and Mckay, N. D. (1992). A Method for Registration of 3-D Shapes. *Sensor Fusion IV: Control Paradigms and Data Structures*, 1611:586–606.

Bouaziz, S., Tagliasacchi, A., and Pauly, M. (2013). Sparse Iterative Closest Point. *Computer Graphics Forum*, 32(5):113–123.

Boughorbel, F., Mercimek, M., Koschan, A., and Abidi, M. (2010). A new method for the registration of three-dimensional point-sets: The Gaussian Fields framework. *Image and Vision Computing*, 28(1):124–137.

Buss, S. R. and Fillmore, J. P. (2001). Spherical averages and applications to spherical splines and interpolation. *ACM Transactions on Graphics*, 20(2):95–126.

Chen, Y., Hu, Y., and Ma, Z. (2009). Surface Matching Method for the Armed Inspection Robots. *2009 International Conference on Measuring Technology and Mechatronics Automation*, pages 144–147.

Chow, C. K., Tsui, H. T., and Lee, T. (2004). Surface registration using a dynamic genetic algorithm. *Pattern Recognition*, 37(1):105–117.

Ferrant, M., Nabavi, A., Macq, B., Jolesz, F. A., Kikinis, R., and Warfield, S. K. (2001). Registration of 3-D Intraoperative MR Images of the Brain Using a Finite-Element Biomechanical Model. *IEEE transactions on medical imaging*, 20(12):1384–97.

Geiger, A., Lenz, P., and Urtasun, R. (2012). Are we ready for autonomous driving? The KITTI vision benchmark suite. *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361.

Gelfand, N. and Mitra, N. (2005). Robust Global Registration. *Symposium on geometry ...*.

Godin, G., Rioux, M., and Baribeau, R. (1994). Three-dimensional registration using range and intensity information. *Photonics for . . .* , 2350(1994):279–290.

Gold, S. and Rangarajan, A. (1996). A graduated assignment algorithm for graph matching. *Pattern Analysis and Machine . . .* , 18(4):377–388.

Guest, E., Fidrich, M., Kelly, S., and Berry, E. Robust surface matching for registration. *Second International Conference on 3-D Digital Imaging and Modeling (Cat. No.PR00062)*, pages 169–177.

Horn, B. K. P. (1987). Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America A*, 4(4):629.

Huang, H., Shen, L., Zhang, R., Makedon, F., Saykin, A., and Pearlman, J. (2007). A novel surface registration algorithm with biomedical modeling applications. *. . . in Biomedicine, IEEE . . .* , 11(4):474–482.

Liu, Y. (2007). A mean field annealing approach to accurate free form shape matching. *Pattern Recognition*, 40(9):2418–2436.

Mangin, J., Frouin, V., and Bendriem, B. (1992). Nonsupervised 3d registration of pet and mri data using chamfer matching. *. . . Science Symposium and . . .* , pages 1262–1264.

Masuda, T., Sakaue, K., and Yokoya, N. (1996). Registration and integration of multiple range images for 3-D model construction. *Pattern Recognition, 1996., . . .* , pages 879–883.

Mitra, N. J., Gelfand, N., Pottmann, H., and Guibas, L. (2004). Registration of point cloud data from a geometric optimization perspective. *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing - SGP '04*, page 22.

Pomerleau, F., Colas, F., Siegwart, R., and Magnenat, S. (2013). Comparing ICP variants on real-world data sets. *Autonomous Robots*, 34(3):133–148.

Pulli, K. (1999). Multiview registration for large data sets. *Second International Conference on 3-D Digital Imaging and Modeling (Cat. No.PR00062)*, pages 160–168.

Rusinkiewicz, S. and Levoy, M. (2001). Efficient variants of the ICP algorithm. *3-D Digital Imaging and Modeling, ...*

Salvi, J., Matabosch, C., Fofi, D., and Forest, J. (2007). A review of recent range image registration methods with accuracy evaluation. *Image and Vision Computing*, 25(5):578–596.

Sandhu, R., Dambreville, S., and Tannenbaum, A. (2010). Point set registration via particle filtering and stochastic dynamics. *IEEE transactions on pattern analysis and machine intelligence*, 32(8):1459–73.

Shoemake, K. (1985). Animating rotation with quaternion curves. *ACM SIGGRAPH computer graphics*, 19(3):245–254.

Simon, D. (1996). Fast and accurate shape-based registration.

Tam, G. K. L., Cheng, Z.-Q., Lai, Y.-K., Langbein, F. C., Liu, Y., Marshall, D., Martin, R. R., Sun, X.-F., and Rosin, P. L. (2013). Registration of 3D point clouds and meshes: a survey from rigid to nonrigid. *IEEE transactions on visualization and computer graphics*, 19(7):1199–217.

Turk, G. and Levoy, M. (1994). Zippered polygon meshes from range images. *Proceedings of the 21st annual conference on Computer graphics and interactive techniques - SIGGRAPH '94*, pages 311–318.

van Kaick, O., Zhang, H., Hamarneh, G., and Cohen-Or, D. (2011). A Survey on Shape Correspondence. *Computer Graphics Forum*, 30(6):1681–1707.

Yamany, S., a.a. Farag, and El-Bialy, a. (1999). Free-form object recognition and registration using surface signatures. *Proceedings 1999 International Conference on Image Processing (Cat. 99CH36348)*, 2:457–461.

Yang, C. and Medioni, G. (1991). Object modelling by registration of multiple range images. *Image and vision computing*.

Zhang, Z. (1994). Iterative point matching for registration of free-form curves and surfaces. *International journal of computer vision*, 152(1994):119–152.