NTNU – Trondheim
Norwegian University of
Science and Technology

Comparing Datasets From Consecutively 3D-Scanned Objects

Kim Rune Solstad

Master of Science in Computer Science
Submission date: June 2015
Supervisor: Theoharis Theoharis, IDI
Co-supervisor: Christian Schellewald, IDI

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

Abstract

As time progress, environmental effects are slowly causing our cultural heritage to vanish. Registration of 3D-scanned datasets has applications in exposing how erosion is altering an object of cultural heritage. The Iterative Closest Point (ICP) algorithm, which iteratively matches the points in one dataset with those of another before minimizing the error between them, is usually applied for registration of datasets. In this thesis, we are concerned with adapting ICP for aligning 3D-scanned datasets to measure the erosion. We have done so by implementing different, both novel and published, modifications of the algorithm and testing their applicability on real world datasets. Implementation of a filtering procedure whose matching policy tightens as computation progress before minimizing the error yielded the best result. Using this filtering procedure, we are able to decrease the evaluation metric by up to 50% compared with using ordinary ICP. We present the result in a mesh with color codes indicating direction and impact of erosion.

Sammendrag

Med tidens gang sørger naturkreftene for at vår kulturarv sakte forsvinner. Registrering av 3D-scannede datasett har bruksområder innen eksponering av hvordan erosjon påvirker et objekt av kulturell verdi. Iterative Closest Point (ICP) algoritmen, som iterativt matcher punkter i et datasett med punkter i et annet datasett for så å minimalisere feilmarginen mellom de, er som oftest brukt for registrering av datasett. I denne avhandlingen vil vi utvikle en teknikk for måling av erosjon på kulturarvobjekter ved å tilpasse ICP for dette formål. Vi har utført oppgaven med å implementere forskjellige, både publiserte og nye, modifikasjoner av algoritmen for så å teste på datasett fra naturlige objekter. Implementasjon av en filtrerings prosedyre hvis filtrerings polise strammes inn etter som eksekvering progresserer før feilmarginen minimaliseres avkastet de beste resultatene. Ved hjelp av denne prosedyren er vi i stand til å senke evaluerings målet med 50% sammenliknet med vanlig ICP. Resultatene presenterer vi i et *mesh* med fargekoder for å indikere retning og påvirkning av erosjon.

Preface

This master thesis is the result of work performed over the course of one semester carried out at the Department of Computer and Information Science (IDI), at the Norwegian University of Science and Technology (NTNU). I would like to thank my supervisors Theoharis Theoharis and Christian Schellewald for their restless guidance during the project. I would also like to thank Vilius Kazakauskas for doing preparatory work for this project as part of his thesis.

Table of Contents

List of Tables

List of Figures

Glossary

CAD Computer Aided Design. 6

DGM Differential Geometry Measurer. 13, 14, 19, 32, 33, 36, 40, 46, 49, 54, 58, 67, 71, 77, 81, 87, 93

ICP Iterative Closest Point. 1, 5–8, 10, 14, 15, 25–27, 31–33, 35, 40, 49, 50, 58, 65, 66, 71

PCL Point Cloud Library. 18

query point The point in source that we are currently selecting candidates for. 15

SA Simulated Annealing. 8, 11, 26

source the dataset that will be transormed during registration. 1–3, 5–9, 11–14, 17, 18, 23, 25–28, 31, 33, 34, 62, 64, 91

SVD Singular Value Decomposition. 9, 10, 19, 29

target The dataset that is used as a reference during registration. 1–3, 5–12, 15, 17, 18, 23, 25–28, 31–34, 62, 64, 91

Symbols

$\vec{a}$, $\vec{b}$, $\vec{c}$ Symbols with an arrow pointing to the right represents vectors.

v_i The i'th coefficient of vector v

$0_{m \times n}$ A *zero matrix* with size $m \times n$, i.e. where all entries equals zero

$1_{m \times n}$ An *one matrix* with size $m \times n$, i.e. where all entries equals one

I_n An *identity matrix* with size $n \times n$

$\hat{a}$, $\hat{b}$, $\hat{c}$ A lower case letter with a hat represents a unit vector.

$\bar{v}$, $\bar{b}$, $\bar{c}$ A lower case letter with a bar represents the mean of a vector. The mean may be weighted.

A, B, C An upper case letter represents a matrix.

A^T, B^T, C^T An upper case letter with the power of T represents a transpose matrix.

$\boldsymbol{S}$ A $3 \times N$ matrix that represents the points in the source dataset.

$\boldsymbol{T}$ A $3 \times N$ matrix that represents the points in the target dataset.

Chapter 1

Introduction

Erosion of objects may be measured by computing the difference between two or more aligned datasets acquired from 3D scans of an object taken with a long enough period in between for environmental effects to alter the original object. Registration of datasets is done by estimating a rigid transformation that applied to one dataset, transforms it such that it is oriented relative to another dataset. The dataset being re-positioned, is referred to as the source S and the anchored dataset is referred to as the target T. For solving the registration problem the Iterative Closest Point algorithm, which we will focus on in this thesis, is used.

Solving the registration problem, apart from measuring erosion, has applications in many areas. In 3D Acqusition, acqusition of a dataset often has to be aquired in multiple sessions due to the sensor not being able percieve the whole object at once. In object recognition applications, the dataset might be aligned with a reference model to see if they are of the same object. Registration can also be used to to re-assemble fractured objects by aligning the fractured parts of the surface (Papaioannou et al., 2001).

The goal of this thesis is to study some modifications of the original ICP algorithm that have been proposed since the original paper (Besl and McKay, 1992) got published. We will also develop a procedure for registration that enables for precise measurement of erosion. We have acquired datasets from real-world objects scanned approximately a year apart. Among the scanned objects is a column from Elefsis (Greece) and the Nidaros Cathedral (Norway).

1.1 Structure of Thesis

This thesis is structured as follows

Chapter 1 presents the motivation, background and terminology for this work.

Chapter 2 contains the field study. The steps of the ICP algorithm is presented together with modifications proposed by researchers in the field.

Chapter 3 provides an overview of the implemented prototype.

Chapter 4 describes the evaluation metrics and results from experimenting with the prototype.

chapter 5 discuss the observations made during experimentation. Finally, the conclusion and future work is presented.

Appendix A enumerates the experiments run and all data generated while experimenting.

Appendix B is a manual for how to configure the prototype.

1.2 Nomenclature

Different articles tend to use different names for the datasets used for registration, none of theese seems super accurate. The most common names are source and target which is why we will use those names throughout this thesis. source is the term for the dataset that will be transformed and target is the name of the dataset that is used as a reference when transforming source. Most of the time, their symbols will be used. S for source and T for target. When searching for a point in T that corresponds to a certain point in S, that certain point is referred to as the query point. Candidate points are the points in T that are potential matches for the query point.

For the standard matrices $0_{m\times n}$ (all zeros), $1_{m\times n}$ (all ones) and $I_{m\times n}$ (identity), we will drop the subscripts ($m \times n$) as they can be determined from the context. A rigid transformation is usually referred to as a translation, rotation, a reflection or a combination of those. For this thesis, we will refer to a rigid transformation as a rotation R followed by a translation t as follows:

$$p' = R \cdot p + t \tag{1.1}$$

where p' is the point p with the transformation applied.

1.3 Background

1.3.1 Quaternions

Quaternions consist of a scalar s and a vector $\overrightarrow{v}$ (Theoharis, 2008)

$$q = (s, \overrightarrow{v}) = (s, x, y, z) \tag{1.2}$$

A rotation by an angle θ about an axis represented by the unit vector $\hat{n}$ is represented by

$$q = (cos\frac{\theta}{2}, sin\frac{\theta}{2}\hat{n}) \tag{1.3}$$

It is applied to a point p using the following equation

$$p' = q \cdot p \cdot q^{-1} = q \cdot p \cdot \bar{q} \tag{1.4}$$

The rotation matrix corresponding to a rotation represented by the unit quaternion $q = (s, x, y, z)$ is

$$R_q = \begin{bmatrix} 1 - 2y^2 - 2_z^2 & 2xy - 2sz & 2xz + 2sy & 0 \\ 2xy + 2sz & 1 - 2x^2 - 2z^2 & 2yz - 2sx & 0 \\ 2xz - 2sy & 2yz + 2sx & 1 - 2x^2 - 2y^2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.5}$$

Operations on quaternions such as addition and multiplication are explained in Theoharis (2008).

1.3.2 Transformations

A transformation is an operation that applied to a dataset changes its appearance. Commonly used in computer graphics is the affine transformation which preserves important geometric properties. Shear and scale are examples of affine transformations that are not rigid. Rigid transformations, which additionally preserve angles between lines, may consist of a rotation and translation.

1.3.3 Octree Datastructure

An octree data structure is a tree structure with a fixed number of child nodes like the binary tree, but instead of using two child nodes, eight are used. The data structure is commonly used recursively for splitting a 3-dimensional geometric space into eight spaces, one for each corner of the space.

1.3.4 K-D Tree Search

A k-d tree is a data structure for organizing points in k-dimensional space, i.e., 3-dimensional space in our case. The points are arranged in a binary tree where each level represents one of the k dimensions. The binary tree is constructed by choosing a plane parallel to the yz-plane that passes through a point p. This plane is parallel to the split dimension and cuts the space into two spaces. A left node and a right node is then obtained, one for each side of the plane. The split dimension advances to become a xz-plane and the same procedure continues. Construction of the K-D tree is done in $O(n \log n)$ time. We refer to Zhang (1994) for further explanation. This data structure allows for nearest point search that prunes away parts of the search space as search progresses resulting in average computation time of $O(N_S \log N_T)$ time where N_S is the number of points in S and N_T is the number of points in T.

1.3.5 Bounding Box

The bounding box of a dataset is the smallest possible box that encloses the dataset. If the edges are parallel to the axes of the coordinate system, we say that it is axis-aligned. Two points can represent axis-aligned bounding boxes, one for the corner closest to the origin and one for the corner most distant from the origin. A bounding sphere is an alternative to

the bounding box represented by a point and a radius. The axis-aligned bounding box is simple to visually represent whereas the bounding sphere computationally is the simplest to use for finding intersections.

Chapter 2

An Overview of the Field

There exist hundreds of implementations of the ICP algorithm (Pomerleau et al., 2013). Navigating the sea of published articles to locate implementations of ICP is certainly not a simple task, let alone reviewing and comparing them. The seminal work of Rusinkiewicz and Levoy (2001) has made the task somewhat simpler by comparing and describing a large number of them. They classify the variants as affecting one or more of the six stages of ICP.

1. Selecting sets of points in either mesh.
2. Matching selected points with samples in the other mesh.
3. Assigning weights to the matched pair.
4. Rejecting certain pairs.
5. Assigning an error metric based on point set.
6. Minimizing the error metric.

In this chapter, we aim to describe published modifications of the ICP algorithm. A section has been written for each step of the algorithm to explain the variations of that particular step. Modifications that affect multiple stages will be mentioned, with cross-references, in multiple sections. First, we will explain the original ICP algorithm.

2.1 Iterative Closest Point

Registration is the process of finding a rigid transformation that orients one movable dataset S relative to an anchored dataset T. In other words, the estimated rigid transformation applied to S, adjust its orientation such that matched points have the same position. The transformation is denoted as

$$S' = R * S + t \tag{2.1}$$

where R is the rotation matrix and $\overrightarrow{v}$ is the translation vector. ICP is an algorithm for registration of geometric data such as cloud data, CAD models, and planes, which was introduced by Besl and McKay (1992). ICP consists of the following steps

1. Match each point in S with the nearest point in T
2. Estimate the transformation that best aligns the points in S with the matching points in T by solving the least squares problem see Section 2.5.1.
3. Apply the transformation to S and evaluate the modified S using the objective function.
4. Repeat from the beginning using S' as S if the value returned by the objective function exceeds a predefined threshold. Terminate otherwise

2.2 Using a Subselection

A match is required for every point in source in the original ICP algorithm (Besl and McKay, 1992). Forcing a match for every point in S may cause an inferior location of the global minimum, especially for datasets that only partially overlap. There have been several proposals for different subselections of the datasets that may be used instead of including all points. The motivation for using a subset is to either increase accuracy or reduce computational cost.

Weik (1997) proposes using a selection of the points with the highest reliability. More information than just the geometric data is required for their approach to work, as the reliability is based per-sample color or intensity.

Turk and Levoy (1994) suggest accelerating the point selection step by uniformly subdividing the dataset and using a nearest point search that is constrained to search within the divided patches. The size of the divided patches will then be based on the distance threshold 2.4.

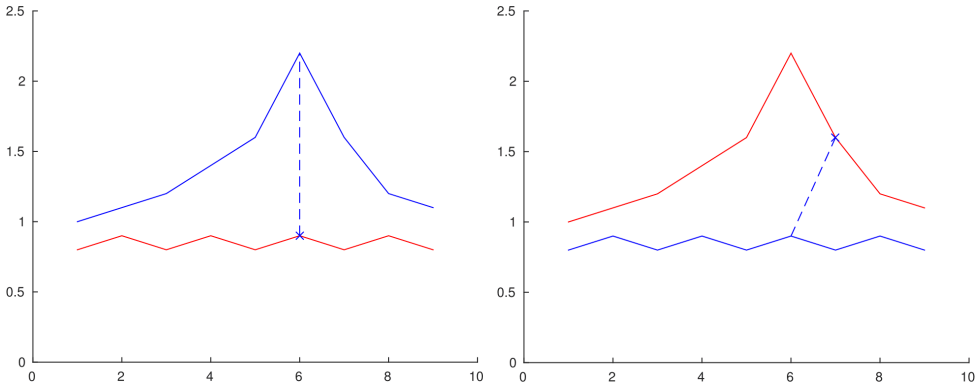
Masuda et al. (1996) recommend randomly selecting a subset of S before matching with the nearest point. A new random selection is generated after each iteration.

Rusinkiewicz and Levoy (2001) introduces an approach where they chose the points that give the largest distribution of normals. In their paper, uniform sampling, random sampling, and normal space sampling was tested showing little difference. They also tested the difference in convergence rate between sampling in one dataset versus sampling in both datasets. Little to no difference in convergence rate were detected indicating that different sampling strategies based on geometric data has little effect on the result.

2.3 Selecting Candidates

Candidates points are a selection of points in T that we consider good candidates for being matched with the query point, i.e. the point in S that are currently being matched. The candidates will usually be the n points in T that are closest to the target.

Besl and McKay (1992) matches with the point in T that has the shortest Euclidean distance from S_i . Their technique requires matching every point in S with a point in



(a) The points of the upper surface are being matched with the nearest point on lower surface

(b) The plot shows the same data, but the source and target properties have been swapped

Figure 2.1: The point that was matched in the left plot was matched with a different point, thus the asymmetry

T. Simon (1996) accelerate this step by using a K-D tree (see Section 1.3.4), which is common when searching for locations in a geometric space. He also optimizes this step by caching a predefined number of closest points as they are likely to be close also in the next iteration. An alternative to matching with the nearest vertex is to match with the closest point on the nearest plane (Chen and Medioni, 1991).

Most variants of ICP use the nearest point strategy. An exception is Chen and Medioni (1991)'s method, popularly called "Normal Shooting". "Normal Shooting" is an approach where a ray originating at the query point in S is shot in the direction of the vertex normal. The matching point is chosen as the point at which the ray intersects with the target surface. As there is no guarantee that T is present in the slant range, another ray should be shot in the opposite direction. Note that all of these matching schemes are asymmetric. As a consequence, we might end up with a different result if we swap the datasets before matching. This is illustrated in Figure 2.1.

2.4 Filtering Candidates

Godin et al. (1994) introduces the concept of using additional constraints when matching pairs with their *Iterative Closest Compatible Point* (ICCP) algorithm. The compatibility function C is used to determine the compatibility between two points. Only pairings of points for which $C > \tau C$, where τC is a hard threshold, is considered a match. They experimented with using normal vectors and point to point distances when computing the compatibility. Masuda et al. (1996) describes an alternative to the constant threshold introduced by Godin et al. (1994), namely the dynamic threshold in which the $n\%$ worst matches are rejected.

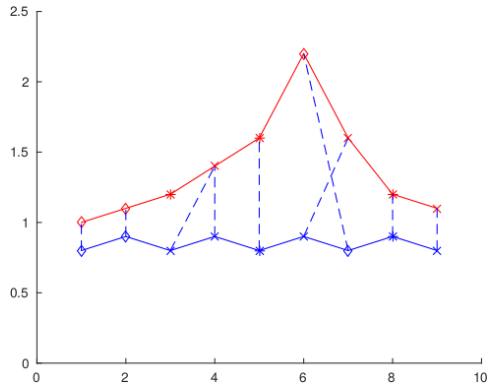


Figure 2.2: The points have been classified by three different point weights to avoid making matches with different classifications.

2.4.1 Penalizing Weights

Instead of rejecting points based on their compatibility, the compatibility may be used as weights that decide the pairs influence during estimation. The weighted least squares technique will then have to substitute the least squares technique used for estimating the transformation. Because usage of the term *'weight'* in the literature is ambiguous, we will refer to these kinds of weights as *penalizing weights*.

Penalizing weights based on the L_0 norm and the L_1 norm was compared by Marjanovic and Solo (2012) resulting in the lq norm where $0 < q < 1$. Bouaziz et al. (2013) formulates the registration problem as a minimization of the L_q norm and proposes the Sparse ICP algorithm. The q parameter is what controls the influence of outliers. A lower value of q increases robustness at the expense of computational effort. The negative effect Sparse ICP has on execution time triggered an improvement by Mavridis et al. (2015) with the introduction of Efficient Sparse ICP where Simulated Annealing, (SA, see Section 2.6, is used at the beginning of execution and later swapped with ICP to ensure convergence.

2.4.2 Point Weights

Some articles refer to weights as values that, assigned to a point, restricts it from being matched with points of different weights. We will in this thesis refer to such weights as *point weights*. Marinov and Zlateva (2010) aligns stars on scanned astronomical images by using stellar magnitudes as point weights. Thus, only the stars with the same magnitudes are deemed compatible. Another application for *point weights* is in facial recognition. Lin et al. (2006) describes a technique for pose invariant facial recognition by identifying the nose tip and assign it a different point weight than the rest of the face.

2.5 Error Minimization

In this step we estimate a rigid transformation which applied to S minimizes an error metric between S and T.

2.5.1 The Least Squares Technique

Besl and McKay (1992) minimizes the summed euclidean distance between the matching points of S and T. We refer to this metric as the L_2 norm.

$$\|v\|_2 := \sqrt{\sum_{i=1} v_i^2} \tag{2.2}$$

Where $v \in \mathbb{R}^3$ and v_i is the i'th coefficient of v. The summed square of residuals is given by

$$\tau = \sum_{i=1}^{N} (\lambda_i - \lambda_i')^2 \tag{2.3}$$

where $\lambda_i \in \mathbb{R}^3$ is the observed value and λ_i' is the fitted value. The transformation is estimated by substituting λ_i' with the transformation formula $\lambda_i' = S' = RS + t$ (Equation 2.1) and solving for the unknown coefficients R and t.

$$\tau = \sum_{i=1}^{N} \big(\lambda_i - (RS + t)\big)^2 \tag{2.4}$$

The most common algorithms for solving the least squares problem are described and compared by Eggert et al. (1997). They include a quaternion based method (Horn, 1987), a method using SVD Arun et al. (1987), a method using orthonormal matrices (Horn et al., 1988) and a method using dual quaternions (Walker et al., 1991). Eggert et al. (1997) thoroughly tested the algorithms in both ideal and noisy environments. Arun et al. (1987)'s SVD based method, which is also the oldest, outperformed the other procedures in terms of accuracy. SVD as a matrix operation is also widely available being supported by multiple applications including *Matlab, Octave, OpenCV, Eigen, and Libicp*. We first compute M using the following equation

$$M = \sum_{i=1}^{N_S} (s_i - \mu_S)(t_i - \mu_T)' = \begin{bmatrix} M_{x,x} & M_{x,y} & M_{x,z} \\ M_{y,x} & M_{y,y} & M_{y,z} \\ M_{z,x} & M_{z,y} & M_{z,z} \end{bmatrix} \tag{2.5}$$

Where $\{s_i, t_i\}, i = 1, ..., N_s$ are the corresponding keypoints in S and T, and $\mu_S \in \mathbb{R}^3$ and $\mu_T \in \mathbb{R}^3$ is the centroid points of S and T computed by

$$\mu = \frac{1}{N} \sum_{i=1}^{N} p_i \tag{2.6}$$

where N is the number of points in the dataset P and and $p_i \in \mathbb{R}^3$ is the i'th point in P. We then decompose M using SVD,

$$[U, S, V] = SVD(M) \tag{2.7}$$

The matrices U, S and V are derived from M such that $M = USV^T$. R can then be computed as follows:

$$R = VU^T \tag{2.8}$$

Lastly, t is computed by

$$t = -R \times \mu_T + \mu_S \tag{2.9}$$

2.5.2 Robust Least-Squares Techniques

An absence of outliers is assumed when using the original ICP algorithm. Thus, the least squares approach happens to be quite sensitive to outliers. This is addressed by Meer et al. (1991) which review several so-called robust data fitting techniques like utilizing RANSAC (Fischler and Bolles, 1981) and the least-median-of-squares (LMedS) method (Rousseeuw, 1984). The procedures were tested on datasets with up to 50% outliers. LMedS proved most efficient.

Also, minimizing the L_1 norm

$$\|v\|_1 := \sum |v_i| \tag{2.10}$$

as opposed to minimizing L_2 norm was deemed the more accurate option by Flöry and Hofer (2010).

2.5.3 Weighted Least Squares

For datasets with points of varying quality, weighted least squares can be used to avoid the fit being unduly influenced. The summed square of weighted residuals is given by

$$\tau = \sum w_i(\lambda_i - \lambda_i')^2 \tag{2.11}$$

Haralick et al. (1989) explains how the weighted least squares technique can be used by replacing λ_i' with the transformation equation 2.1.

$$\hat{y_i} = RS_i + t \tag{2.12}$$

R can be found by taking the single value decomposition of H, which is computed as follows

$$H = (h_1, h_2, h_3) \tag{2.13}$$

where

$$h_k = \sum_{i=1}^{N} w_i(T_{ik} - \bar{T_k})(S_i - \bar{S})^T \tag{2.14}$$

where k represents the spatial dimension such that T_{ik} is the value of the k'th dimension of the i'th point in T that corresponds to S_i. $\bar{S}$ and $\bar{T}$ is computed as follows:

$$\bar{S} = \frac{\sum_{i=1}^{N} w_i S_i}{\sum_{n=1}^{N} w_i}, \quad \bar{T} = \frac{\sum_{i=1}^{N} w_i T_i}{\sum_{n=1}^{N} w_i} \tag{2.15}$$

R can now be found by taking the SVD of H

$$UDV = svd(H) \tag{2.16}$$

$$R = V^T U^T \tag{2.17}$$

Once R has been obtained, t is easily computed as follows

$$t = \bar{T} - R\bar{S} \tag{2.18}$$

2.6 Simulated Annealing

Simulated Annealing SA (S. Kirkpatrick, 1983) is a search method inspired by the annealing process in metallurgy., In this process, metal is heated to a high temperature. In this state, the atoms move around until they settle in an ordered structure. When this ordered structure has occurred, we say that the metal has re-crystallized. After the metal has re-crystallized, the metal is cooled, and the physical properties of the metal have changed.

SA is supposed to be a good option for finding a global minimum in a deterministic search space with several local minimums. Traditional hill climbing algorithms advance to a neighbor that is perceived as a better solution based on some heuristic. When no better neighbor can be found, the algorithm terminates. SA considers multiple states and decides whether to advance based on a temperature τ and a probability function. When the temperature is high, advancement to a state that is worse than the current state is more likely. For the alignment problem, a state may be defined as a set of parameters that represents the transformation $C_0 = [x, y, z, \sigma, \rho, \gamma]$, where x, y and z are the parameters for movement and σ, ρ and γ are the parameters for rotation. The goal is to find the state C_g that represents the transformation that best align S with T.

These are the steps of the algorithm.

1. Choose a random initial state c and compute its energy
2. Lower temperature τ
3. Pick a random neighboring state c_r and compute its energy
4. Compute a probability value based on the new state and the temperature
5. If a randomly generated value exceeds the probability, set the current state c to c_r
6. Repeat from step 2 if τ is greater than 0.0

First, the state c is set to a random parameter vector representing the initial transformation. The energy e_c, is then computed with the energy function $e_c = e(c)$. The temperature τ is set to the first temperature within a temperature schedule. A set C, with the configurable length n, of random configurations is then generated. The energy e_i of each random configuration C_i , where $0 \leq i < n$, is then computed using the energy function $e_i = e(C_i)$. A probability function $p(e_i, e_c, \gamma)$, computes the chance of c advancing to the new state C_i. τ is then lowered to a new temperature and a new iteration with new random configurations begins. The algorithm converges when τ reaches zero.

2.7 Registration of Large Datasets

Kazakauskas (2014) defines a large dataset as *a dataset that is too big to be contained within the main memory.* He proposes a method for registration of large datasets consisting of four stages.

Rough registration Perform rough registration of subsampled versions of S and T. A good initial alignment of the datasets is presumed.

Sub division Divide the datasets into parts small enough for containment within the main memory of the computer and store each in a separate file on disk. The divided parts are referenced in the leaves of an octree data structure thus keeping track of the parts positions. The combined bounding box of the datasets is used for equal subdivision of the space. This way, a node representing a certain space in one octree has a path equal to the node representing the same space in the other octree. An overlap constant defines the size of the space that each leaf overlap adjacent leaves with. The purpose of the overlap constant is to eliminate the risk of excluding potential nearest points in neighboring sub surfaces during local registration. Each leaf node has a file on disk dedicated for storing its occupying points.

Piecewise Registration The leaves of S are matched with the leaves of T. Measures are taken to avoid matched nodes at different depths to poorly influence the registration. Then the surfaces of the corresponding octree nodes are registered. A combination of the resulting transformations is used as the global registration.

Distance Computation The Euclidean distance between the corresponding points is computed. A shade of gray to represent the distance is assigned the vertex. For similarity measure, he uses the summed Euclidean distance.

In both *Rough registration* and *Piecewise Registration* the original ICP algorithm(Besl and McKay, 1992) as implemented by Geiger et al. (2012) is used for registration.

Chapter 3

Method

This chapter explains the implementation of the Differential Geometry Measurer, DGM and the implementation of the fitting procedures. Code from Kazakauskas (2014) was used as a starting point. The datasets are assumed acquired from consecutive 3D-scans of an object taken approximately a year apart such that erosion has been able to alter it. Erosion will be displayed using a coloring scheme that indicates direction and movement of each point in S.

3.1 Differential Geometry Measurer

Differential Geometry Measurer (DGM) is the name of the software that has been developed. Its responsibilities are to measure the distance between two large datasets and to visually present the result. The function *performRegistration* (see Figure 3.1) performs registration in two steps, one rough and one fine, before presenting the result.

Our aim is to increase accuracy by improving the rough registration step of the pipeline. In our solution, we have modularized DGM and added optional configurations that let the user implement custom pipelines for registration and for displaying the results. Also added are tools for visual representation.

DGM		
+	source	: double *
+	target	: double *
+	preFitter	: Fitter *
+	postFitter	: Fitter *
+	presenter	: Presenter *
+	performRegistration(Matrix R, Matrix t)	: void

Figure 3.1: Simplified class diagram of DGM. The Fitter and Presenter data types are explained later in this chapter.

Fitter		
+	source	: double *
+	target	: double *
+	next	: Fitter *
+	fitAll(Matrix R, Matrix t)	: void
#	fit(Matrix R, Matrix t)	: void

Figure 3.2: All fitting modules are derived from this fitting class. The *fit* behavior performs a call to the *fitCurrent* behavior followed by a call to the *next* objects *fit* behavoir if initialized.

3.2 Fitting Procedures

We implemented a technique for fitting mesh-datasets that enables the user to create a pipeline consisting of several modules for registering the surface. We have implemented the following modules, all of which are derived from the Fitter class in Figure 3.2:

Empty Module A module that does nothing. Implemented for eliminating the time-consuming local registration step to get fast feedback when configuring DGM for a certain case.

Predefined Transformation Module Applies a predefined transformation to source. Added to enable a manually configured initial alignment without using other software.

ICP Module Registers source with the ICP algorithm. Our implementation of this module is vastly configurable; a subsection has been reserved for its explanation.

SA Module Uses Simulated Annealing for alignment. Added for cases where the initial alignment is insufficient and manual transformation is undesirable

Configurable parameters for the SA module include:

Starting Temperature High temperature represents high likelihood for advancement to a state with less energy.

Depreciation The value that the temperature is decreased by each iteration.

Successors The number of random states considered each iteration.

Random The random numbers may be set by the user for a more repeatable outcome.

As the name implies, the ICP module registers S with the ICP algorithm. Our implementation of this module is vastly configurable; a subsection has been reserved for its explanation.

3.3 ICP Module

The modules already described should provide sufficient initial registration for ICP. We have used Libicp as a starting point when developing this module. Libicp is an implementation of ICP created by Andreas Geiger, which we will use as our baseline for the primary

IcpModule : Fitter		
+	filter	: Filter *
-	fit(Matrix R, Matrix t)	: void

Figure 3.3: Class diagram of the ICP module

```
fit(Matrix R, Matrix t) {
  List<Point, Point> correspondences;
  foreach Point query in source do {
    /* Find a preconfigured number of candidate
     * points in target */
    Set<Point> candidates =
      select_candidates(query, target);
    /* Remove the bad candidate using a
     * preconfigured Filter */
    filter.remove_bad_candidates(query, candidates);
    /* if any candidates remains */
    if(length(candidates))
      /* add the first candidate and the query
       * to the set of correspondences */
      correspondences(query, first(cantidates);
  }
  /* compute the transformation that minimizes the
   * error between the correspondences */
  minimize_error(correspondences, R, t);
  /* if stopping criteria is NOT met */
  if( !isAcceptable(source * R + t, target) )
    fitCurrent(R, t);
}
```

Figure 3.4: Pseudocode of the ICP procedure. The preconfigured filter is explained in Section 2.4

fitting procedure. It is a clean implementation of ICP as proposed by Besl and McKay (1992) but with a KD-tree optimization. Libicp also comes with an implementation of matrices that we will use. Pseudocode of the ICP fitting procedure is shown in Figure 3.4.

3.4 Point Selection

As the literature study showed that sampling strategies had little influence on the result, the full target dataset is used when selecting candidate points in target. Potential corresponding points are found by selecting the n points in T that are closest to the query point. A K-dimensional binary tree optimization is used to speed up the point selection process. With an average computational complexity of $O(N_S log N_T)$, the chosen point selection procedure outperforms the other described section 2.3. Also, the nearest point search makes it easy to select more than one point, which is important for cases where we want to use a compatibility measure other than distance.

```
clean(query, candidates, iterator) {
  if(!isEnd(iterator, candidates)
    filter(query, candidates, iterator + 1);

  if(compute_compatibility(query, iterator)
      < getThreshold())
    erase(candidates, iterator);
}

remove_bad_candidates(query, candidates) {
  iterator = getStart(candidates);
  clean(query, candidates, iterator);
  if(nextFilter)
    nextFilter.clean(query, candidates);
}
```

Figure 3.5: Pseudocode of the filtering procedure. The language is object oriented and does calls by reference

3.5 Filter

In the literature, implementing a stricter matching policy yielded the biggest gains regarding accuracy. Most of the work has therefore been spent developing and testing procedures for filtering out bad candidates among the n selected candidates.

As input, the filter takes the query point p_q and the set of candidate points P_c. When called, the filter computes the compatibility between p_q and each candidate point p_{ci}. p_{ci} is removed if the compatibility between p_q and p_{ci} is inferior to what currently is set as the threshold. We have divided the filtering stage into two parts, namely the threshold computation and the compatibility measure to study modifications regarding them separately. Using a combination of filters have been enabled through a pointer in the filter that if initialized, is called to filter the remaining set of candidates after the current filter has finished. This way, a chain of filtering procedures may be configured. For pseudocode of the filtering procedure, see Figure 3.5. The remaining part of this section is reserved for describing the threshold and the compatibility.

3.5.1 Compatibility Measure

The filtering procedure makes use of the compatibility measure and the threshold see Section 3.5.2 when rejecting/accepting a pair. The compatibility is a value between 0 and 1 that tells us how applicable the pair is for registration. We have implemented three different compatibility measures.

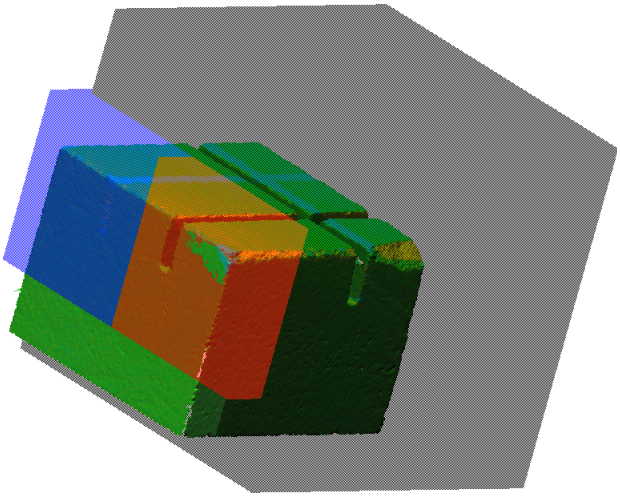


Figure 3.6: The Estone dataset with a filterbox in S (blue) and a corresponding filter box in T (red) and a third filterbox (black) that has no corresponding filter box, thus excludes all points within.

Compatibility Based on Distance

This compatibility measure returns a value based on the Euclidean distance between p_q and p_{ci}.

$$C_d(p_q, p_{ci}) = \|\widehat{d(p_1, p_{ci})}\|^2 j \tag{3.1}$$

Compatibility Based on Predefined Areas

This compatibility measure returns a binary value based on the position of p_q and p_{ci} relative to predefined areas. If both p_q and p_{ci} is either outside or inside their respective predefined areas, the value one is returned. Otherwise, the value 0 is returned. Two areas may be configured, one for S and one for T which are represented by their radius r_{bS}, r_{bT} and their positions p_{bS} and p_{bT} respectively. We define this compatibility as:

$$C_p(p_q, p_{ci}) = A(r_{bS}, p_{bS}, p_1) \equiv A(r_{bT}, p_{bT}, p_{ci}) \tag{3.2}$$

where A returns 1 or 0 based on whether the point p is inside or outside its respective box.

$$A(b_r, p_b, p) = d_2(p_b, p) < b_r \tag{3.3}$$

Where d_2 computes the Euclidean distance between the points p_b and p.

The bounding spheres axis-aligned bounding boxes are used for representation when displayed on an image (see Figure 3.6). Note that a filter box assigned to the S dataset needs to be re-oriented for each iteration because the S dataset gets re-oriented according to the newest transformation. The orientation of source's filter box is updated by passing the rotation matrix and the translation vector to the box filter through the update function.

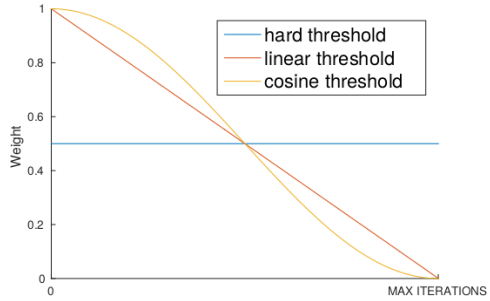


Figure 3.7: The available thresholds visualized

Compatibility Based on Normal Vectors

This compatibility measure computes the angle θ between the normal vector of the source point N_{S_i} and the normal vector of the candidate point in target N_{T_i}

$$C_n(p_q, p_{ci}) = \theta(\widehat{N(p_q), N}(p_{ci})) \tag{3.4}$$

where $N(p_q)$ returns the normal vector of p_q and

$$\theta(N_S, N_T) = \cos^{-1}\left(\frac{N_{S_i} \cdot N_{T_i}}{|N_{S_i}| * |N_{T_i}|}\right) \tag{3.5}$$

If θ exceeds a certain threshold, the point gets rejected. The nearest point with an acceptable direction of the normal vector gets selected as a match. If the normal vectors are unavailable, they may be computed using PCL.

3.5.2 Threshold

The remaining component of the filtering procedure yet to explain is the threshold. The threshold is a value for which the compatibility of the pair can not exceed without being rejected. We have implemented the *hard threshold* proposed by Godin et al. (1994) and introduced the concept of depreciation to it. By decreasing the threshold during registration, we can start with a loose policy for accepting matches that tighten up as the computation progresses. We have implemented two types of depreciation, namely the *linear depreciation* and the *cosine depreciation*. The original threshold has no depreciation, i.e.; the threshold does not change over the course of computation. The depreciations are visualized in Figure 3.7.

3.6 Error Minimization

In the filtering stage, work is done to prevent outliers from being included in the error minimization procedure. Because of this there is no need for the implementation of a robust error minimization. In theory, minimizing the L_0 distance between S and T is

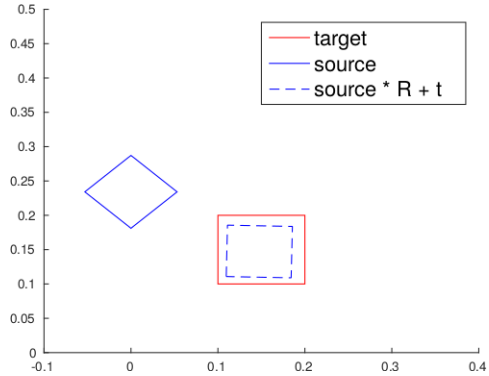


Figure 3.8: The least squares approach works well on downscaled objects

favorable over minimizing the L_2 distance. The same effect is achieved by minimizing the L_2 distance if a threshold for max distance allowed between two corresponding points is set. The SVD based approach proposed by Arun et al. (1987) was chosen as the error minimization technique as it outperformed the alternatives in the study of Eggert et al. (1997). A quick test using Matlab demonstrate its applicability for datasets that have changed size presumably because of erosion, see Figure 3.8.

3.7 Presentation

We have developed two programs for presenting the results of DGM, one concerning how the erosion is displayed, and one for displaying a small area of a large dataset in full resolution.

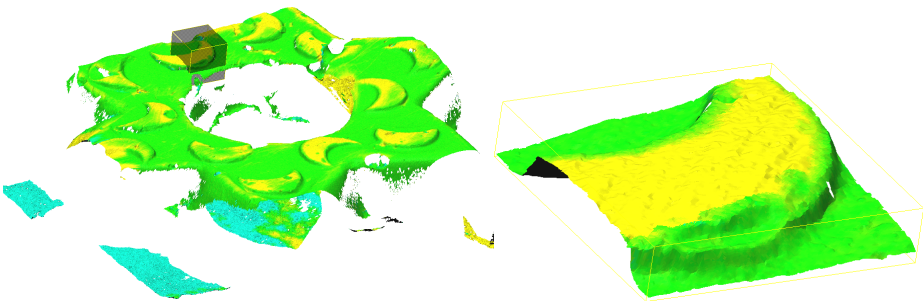
3.7.1 Extraction Tool

Large datasets are impossible to contain within the main memory of the computer, consequently visualizing the dataset becomes a challenging task. However visualizing all points in high-resolution datasets might not be desirable as there simply are not enough pixels on the screen to exploit all data. The full resolution is desirable when we want to inspect a certain area in the dataset. To enable such an inspection, we have implemented an extraction tool that writes the points within an enclosed area, defined by a box, to a separate file during distance computation.Figure 3.9 shows the dataset of a sanded Christmas star and a box covering an area we would like to inspect further.

3.7.2 Colorization

The purpose of the coloring scheme is to visualize accurately how the object has changed between the scans. Both direction and size are properties desirable for visualization. The coloring scheme has been modularized to support custom coloring schemes. We have implemented a coloring scheme based on cylindrical color coordinates(Joblove and Green-

berg, 1978), which enables for a scheme where three different colors represent outward movement, inward movement and little to no movement. The color represent movement along the z-axis, which in the case of Figure 3.10 is the direction of the incised face.



(a) A scan of a plastic star that has been given a sand treatment. The shaded area represents the area covered by the extraction tool

(b) The extracted part of the star

Figure 3.9: The images shows how the extraction tool works

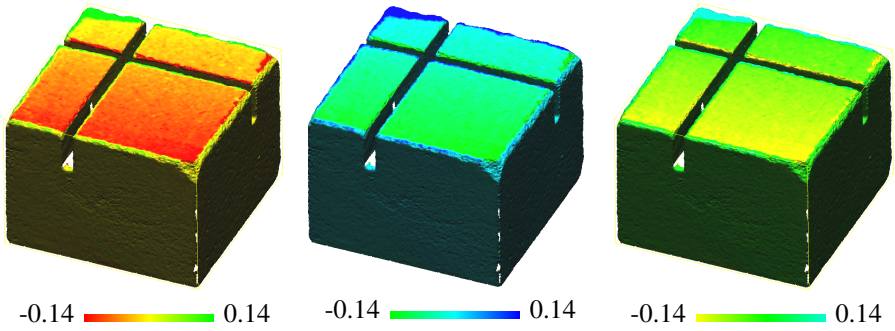


Figure 3.10: Three different configurations with the cylindrical color coordinate scheme. Throughout this thesis, the configuration to the right will be used.

Chapter 4

Tests and results

4.1 Goals of Testing

We intend to benchmark the improvements in the accuracy of ICP caused by the different modifications of the algorithm that we developed during this project. As the goal is to detect erosion, we are only concerned with the applicability to datasets with minor differences like real-world scans of the same object, i.e., reconstructing overlapping datasets like the Stanford Bunny (Turk and Levoy, 1994) is outside of the scope. We would also like to see how sensitive the mechanisms are to different parameters as the difficulty of configuring DGM for a certain case is proportional to how sensitive it is to its parameters.

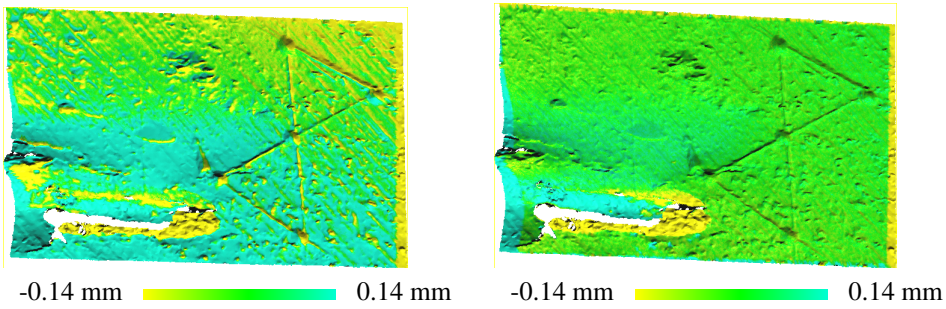
4.2 Setup

As mentioned, the test data consists of 3d scans from cultural heritage objects, e.g. the Nidaros Cathedral and a column in Elefsis. Pomerleau et al. (2013) suggest that we always compare variants of the ICP algorithm to a commonly accepted ICP baseline to obtain an unbiased comparison. We have chosen the original ICP algorithm (Besl and McKay, 1992), which uses point to point distances, as our baseline.

4.2.1 Evaluation Metrics

Measuring accuracy of registration is difficult for real world data because the optimal alignment of the datasets is unknown. Surfaces with color codes indicating the distance and direction from T may be a valuable indicator of the registrations accuracy (see Figure 4.1. When referring to the distance between two points, one often think of the Euclidean distance between them introduced at the beginning of this thesis. There are however several ways to describe the distance between two points. According to Gonzalez and Woods (2011) given the points p, q and z, distance metrics need to fulfill the following criterias:

$d(p, q) \geq 0$ The distance can never be less than zero



(a) The surface fades to blue as we get lower on the surface, and there is also huge variations in the color throughout the whole surface. These are indicators of poor registration

(b) The vast amount of the color green, representing little to no movement, is an indicator of proper registration.

Figure 4.1: The difference between a good and poor registration

$d(p,q) = 0 \quad$ **iff** $p = q \quad$ The distance is 0 if and only if the points are equal

$d(p,q) = d(q,p)$ The metric must be symmetric

$d(p,z) \leq d(p,q) + d(q,z)$ The length of the path between two points can never be decreased by taking a detour to another point.

Common metrics used in Computer Vision apart from the *Euclidean Distance* and L_0 *Distance* mentioned earlier, are the *City-block distance* and the *The chess board distance*. Given p and q with the coordinates (x, y, z) and (s, t, u) respectively, the city block distance is given by

$$d_4(p,q) = |x - s| + |y - t| + |z - u| \tag{4.1}$$

and the chess-board distance is given by

$$d_8(p,q) = max(|x - s|, |y - t|, |z - u|) \tag{4.2}$$

We may adapt the distance function d for surfaces by summing the distances between points We adapt the distance function d for surfaces by introducing a new distance function d_{avg} that computes the mean of point distances.

$$d_{avg}(S,T) = \frac{1}{n}\sum_{i=1}^{n} d(S_i, T_i) \tag{4.3}$$

	L_0	L_1	L_2	L_∞
sum	2.0	1.4	1.08	1.0
avg	0.5	0.35	0.27	0.25

Table 4.1: The different distance measures computed on source and target in Figure 4.2

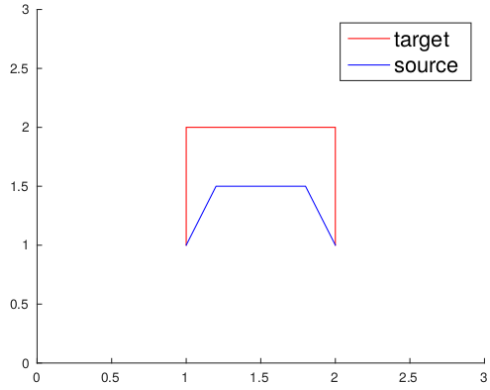


Figure 4.2: A plot showing two datasets in the same geometric space. Each dataset consists of four points.

where n is the number of points in S, and T_i is the point in T that is closest to S_i. Alternatively the sum of point distances may be used

$$d_{sum}(S,T) = \sum_{i=1}^{n} d(S_i, T_i) \tag{4.4}$$

However comparing the registration of one dataset with the registration of another may be difficult when using the sum of point distances because there may be huge variations in the number of points for different datasets. Note that the last two distance metrics are asymmetric, and thus are no longer a real distance metric (Gonzalez and Woods, 2011), but they may still be used as an evaluation metric.

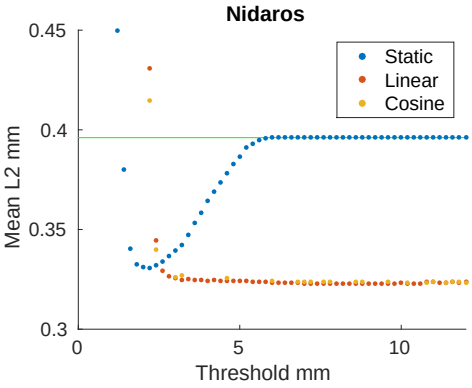
4.3 Observations

For the results of each experiment, see Appendix A. Important observations made during testing is summarized in this section.

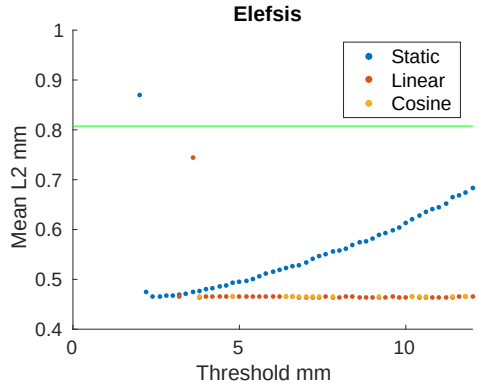
4.3.1 Compatibility Based on Distance

Common for the depreciations is the importance of not setting the starting threshold parameter too low in the beginning. The hard threshold introduced by Godin et al. (1994) yields a good improvement of the ICP algorithm with approximately 50% decrease in the mean L_2 distance compared to the original ICP algorithm even without any depreciation A drawback with using no depreciation is that to achieve high accuracy; we need to pinpoint a particular sweet spot for the threshold (see Experiments A.1.2 and A.3.2).

Based on the experiments ran, a threshold starting at 3mm seems to be comprehensive regardless of dataset and depreciation. However, further investigation is needed for this to be a safe assumption. Using distance as compatibility, the plots in Figure 4.3 tells us how the different depreciations influenced the resulting mean L_2 distances (see Equation 4.3

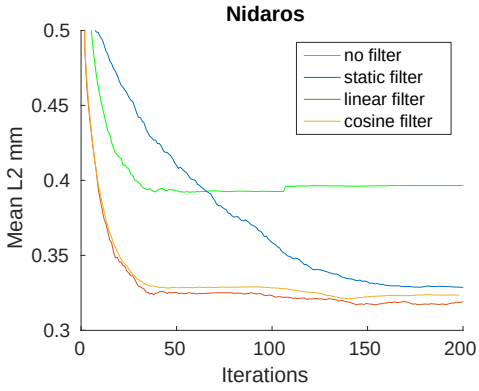


(a) The resulting mean L_2 distances after registration of the Nidaros Cathedral dataset

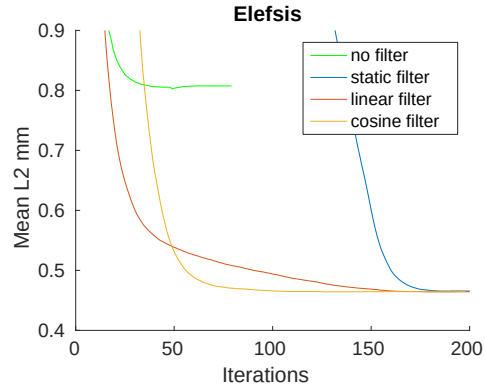


(b) The resulting mean L_2 distances after registration of the Elefsis Column dataset

Figure 4.3: The plots show how the different thresholds impacted the result when we used distance as compatibility measure. Each dot represents the result of one execution. The X-axis represent the predefined starting threshold. The Y-axis represent the resulting mean L_2 distances. The green bar represents the result of registration using the classic ICP algorithm.

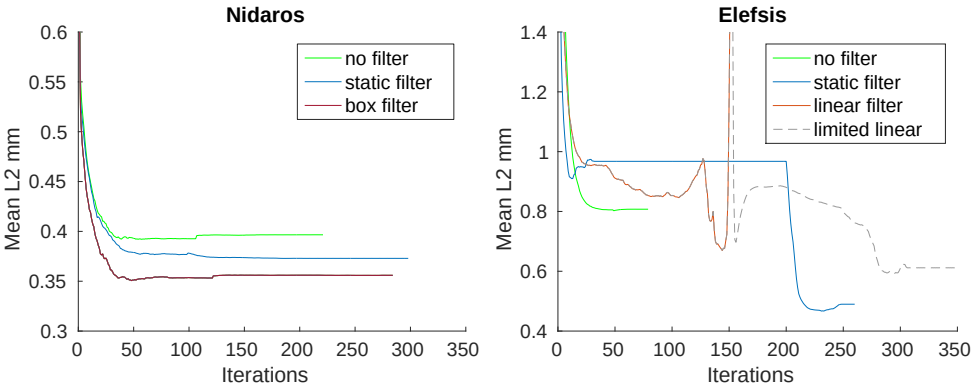


(a) L_2 distance of the Nidaros Cathedral dataset as the computation progresses.



(b) L_2 distance of the Elefsis Column dataset as the computation progresses.

Figure 4.4: These curves show how the mean L_2 distance decreased during execution when the filter was configured with normal vector compatibility and predefined area compatibility. The box filter curve corresponds to the predefined area compatibility. The green curve shows the progression of ordinary ICP.



(a) Mean L_2 distance of the Nidaros Cathedral dataset as the computation progress.

(b) Mean L_2 distance of the Elefsis Column dataset as the computation progress.

Figure 4.5: These curves show how the mean L_2 distance decreased during execution when the filter was configured with normal vector compatibility and predefined area compatibility. The *box filter* curve corresponds to the predefined area compatibility. The green curve shows progression of ordinary ICP

when configured with different parameters. The decreasing thresholds outperforms the static threshold both in robustness towards input parameter and in registration accuracy. An accurate registration is performed as long as the starting parameter is set high enough. Some fine tuning is needed to get the lowest possible mean L_2 distance, a decrease is usually possible to get from fine tuning. Note that the maximum number of iterations will often be reached when using a decreasing threshold.

Linear depreciation is preferred over cosine, as the cosine did not outperform the linear and because it produces many outliers (see Experiments A.1.4, A.1.6, A.3.4 and A.3.6).

The problem with outliers disappears if we restrain the error minimization step from continuation when the number of matched pairs is too small. Still, cosine depreciation was not able to outperform the simpler linear depreciation though they perform quite similar (see Experiments A.1.7).

4.3.2 Compatibility Based on Predefined Areas

Excluding areas with an abnormal difference between the two datasets influence the accuracy of registration positively (see Experiment A.1.8). The flake in the Nidaros dataset is an example of such an abnormal difference. Whether the exclusion takes place in S, T or both makes little difference. We detected a decrease in the mean L_2 distance at 14% by excluding a troubled area before registration of the Nidaros dataset.

Configuring the filter to match points inside a predefined area in S only with points inside a predefined area in T does not cause a better registration (see Experiment A.1.9). We ran experiments on datasets where the initial alignment was in a bad local minima to see if this filter would produce a correct registration (see Experiment A.4). The filter was unable to escape that local minima see Section A.1.9.

4.3.3 Compatibility Based on the Angle Between Normal Vectors

Using the angle between the normal vectors of the corresponding points for computing the compatibility seems to increase accuracy for curvy datasets (see Experiments A.3.3 and A.1.3). We were able to detect a decrease in the mean L_2 distance at about 5% for the Nidaros experiment and 60% for the Elefsis experiment. When using a compatibility measure other than distance, it makes sense to consider more than one candidate because the set of candidate points consist of the n nearest points in T. However, when using the normal vector compatibility, a better accuracy is not achieved when considering more than one candidates.

Configuring a depreciation for the threshold did not yield accurate registration (see Experiment A.1.5). In most cases, the source dataset would be oriented more than 10mm away from target.

If we stop decreasing the threshold when it has reached a certain threshold, the iterative threshold performs evenly with the static threshold (see Experiment A.3.5). In Figure 4.5, the iterative threshold decreases the mean L_2 distance evenly until suddenly it erects, and the computation converges. Running with the same parameters, but restricting the threshold from decreasing any further when it hits 0.2mm gives a more accurate registration.

4.3.4 Combining Filters

Combining multiple filters did not cause a more accurate registration. None of the multiple possible combinations of the filters with optimal parameters acquired from experimentation outperformed the best performing filter.

4.3.5 Combining Fitting Procedures

In almost all experiments a combination of fitting procedures were used, mainly to attain the initial orientation which is shown at the beginning of each section describing the experiments in A. In those cases, the *Predefined Transformation Module* was used. One of the drawbacks with using a threshold based on distance, is that a good initial alignment is required. Registration with the accuracy gained from using a filter is still possible by combining two ICP modules, one with a threshold and one without. If the first ICP procedure fails to align, e.g. by orienting source in a bad local minima, SA can be used instead (see Experiment A.2). By combining two ICP modules, we were able to decrease the mean L_2 distance between the datasets to 0.3228 when the Nidaros datasets were far apart from each other. For the same case, combining SA with ICP resulted in a mean L_2 distane at 0.3229.

4.4 Potential Sources of Error

Though Arun et al. (1987)'s SVD based technique for error minimization is deemed the most accurate by Eggert et al. (1997), a quick, non-scientific, test using Matlab shows us that an optimal alignment of the dataset is not to be taken for granted. The only difference between the plot in figure 4.6a and 4.6b is the initial orientation of S. Though we certainly

speculate about the error minimization being the cause for a faulty registration, we did not identify it as being the case. Noise is a common source of error in registration. The original ICP algorithm requires a match for all points in S. Thus; it is especially sensitive to noise. Because our algorithm makes use of thresholds when matching a point, outliers are not likely to be influenced by the registration unless they are so close, that it becomes impossible to distinguish it from the actual surface.

4.5 Discussion

In general, all modifications both novel and from the literature study enabled us to increase the accuracy of the registration procedure. From the experiments run, we see that the distance between the points makes the best compatibility measure.

4.5.1 Compatibility Based on Distance

The iterative threshold outperforms the static threshold both in robustness against the threshold input parameter and in the accuracy of the resulting transformation when using the L_2 as compatibility measure. However applying the cosine depreciation, depending on which dataset was being used, caused a poor registration in 50-70% of the test cases. Some results, oriented S more than 10mm from T, which is far off. Figure 4.7 shows how the mean L_2 distance decreases during an execution that fails to accurately register S. When we investigate this occurrence, we see that the cosine threshold continued for eight iterations after hitting 0.48mm, the last threshold when using the linear, thus terminating execution when the threshold reached 0.0006mm. When the spike in mean L_2 distance occurred, only one pair of corresponding points was found. We can think of two plausible reasons for the sudden jump to a seemingly arbitrary orientation. One reason might be that when using a tiny number of pairs, an poor registration like the case in Figure 4.6b is more likely. Another, more likely reason, is that the freedom gained from having to minimize the error between only two points caused the algorithm to set the query point to the same coordinates as the matched candidate without considering anything else. Then, the algorithm converged because the distance between those two points was decreased to zero. We were able to fix this problem by restricting ICP from continuation if less than ten corresponding points were identified.

4.5.2 Compatibility Based on Predefined Areas

The compatibility filter was best utilized for excluding areas vastly divergent from the target dataset. For such datasets, we were able to improve the registration by excluding the divergent area.

We were unable to perform a correct registration of a dataset pre-oriented in a local minimum by using only ICP with the predefined area compatibility. The reason for this might be in how we select the candidate points. By choosing the n candidates closest to the query point before filtering out the incompatible ones, we risk leaving a vertex without a match because the compatible points are too far away from the query point. Godin et al. (1994) used a different approach by computing the compatibility for all points in T and

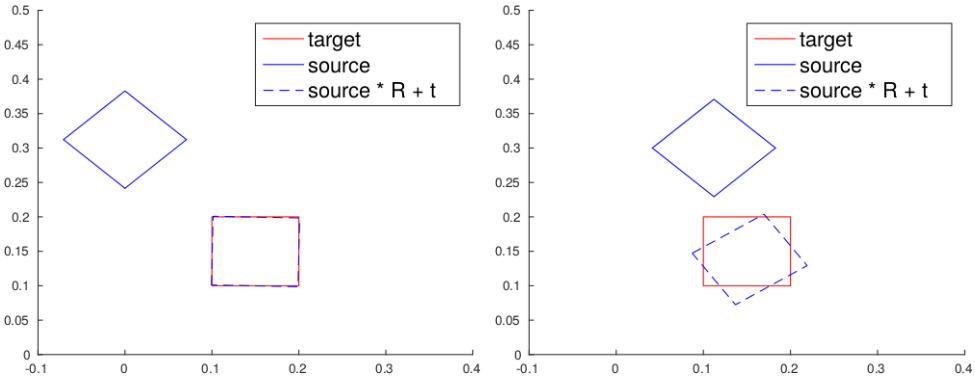
filtering out all incompatible points before selecting the nearest point in that set. This approach might have been better because it allows all points to be considered, not just the n-nearest.

4.5.3 Compatibility Based on Angle Between Normal Vectors

Using the angle between corresponding points as compatibility gave a slight performance boost for a mostly flat surface and a good boost for a curvier surface. The normal vector filter is quite sensitive to the value of the threshold. For both the flat surface and the curvy surface, the threshold had to be pinpointed.

The fact that considering more than one candidate point has a negative effect on the resulting mean L_2 distance is a little surprising. The reason for this might be that the distance becomes an indirect compatibility when the nearest points in T are selected as candidates. Consequently, the first element among the candidates will be better than the n'th point despite the n'th point having a smaller angle between the normal vectors because distance makes a better compatibility measure than the normal vector.

Decreasing the iterative threshold all the way to 0.0 radians proved to be an undesirable option for registration as the result would usually orient S far away from T. We can think of several reasons for this behavior, one of them being that the filter simply caused erroneous pairings. Another reason might be that few samples caused by a strict matching policy made the matched data set susceptible to erroneous error minimization like the case in Figure 4.6b. If we take a look at the progression of the mean L_2 norm for a case that ended in an incorrect registration, (see Figure 4.5b) we see a sudden spike in the mean L_2 norm before the computation converged. When we limited the threshold from decreasing any further after reaching 0.2 radians, a similar curve was produced but instead of converging at the top, computation continued and correct registration was acquired. However, restricting further computation when less than ten samples were found did not result in a better registration.



(a) A *good* alignment of source using the SVD based least squares approach for error minimization

(b) A *poor* alignment of source using the SVD based least squares approach for error minimization

Figure 4.6: The accuracy of the error minimization

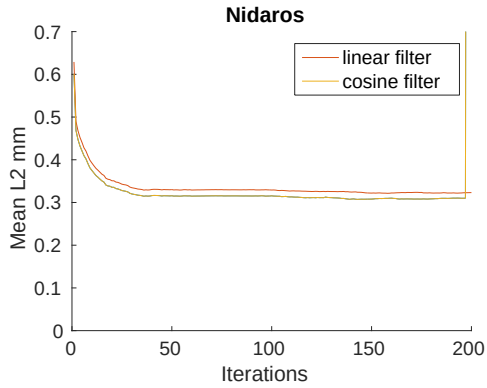


Figure 4.7: Progression of the mean L_2 distance during execution on the Nidaros dataset. Both thresholds have been given the same parameters.

Chapter 5

Conclusion and future work

5.1 Conclusion

We have experimentally tested and compared different techniques for increasing accuracy of ICP for cases where the intention is to measure erosion on objects. Also, we presented a novel formulation of ICP which separates threshold and compatibility computation. Our proposal is to use the iterative threshold with linear depreciation and per point distance as the compatibility measure. We were able to outperform not only the original ICP algorithm, but also modifications to the algorithm that has later been developed.

5.2 Future Work

As the implemented procedure is quite accurate, our suggestion for future work concerns the registration of large datasets.

5.2.1 Registration of Large Datasets

We recommend dividing the surface into multiple patches as in Kazakauskas (2014), and acquire a selection of the most compatible pairs from each patch followed by minimizing the error between those pairs. Some modifications have been proposed that may help us further limit the *minimization set* not primarily to increase accuracy, but to make sure that few enough points are acquired from each subselection such that all patches are represented. One of them being the dynamic threshold (Masuda et al., 1996) that removes the n% worst matches after having generated the selection of pairs from one subsurface. Another modification that may help our cause is the per-point reliability introduced by Weik (1997). The reliability of all points within S and T are computed, and the n% worst points are filtered out before making matches. For reliability measures, we may use curvature and/or chemical properties of the meshes.

If forcing a selection from each subsurface causes inaccurate registration, penalizing weigths (see Section 2.4.1) may be implemented. This also requires the weighted error minimization (see Section 2.5.2) to be implemented. Given a slight modification, compatibility measures may be used as weights. Because $0 \leq C \leq 1$, multiple compatibilities may be combined as weights using the following formula:

$$C(p_1, p_2) = \frac{1}{n} \sum_{i=1}^{n} C_i(p_1, p_2) \tag{5.1}$$

5.2.2 Utilizing skeletonization

Inspired by the skeletonization process in computer vision (Gonzalez and Woods, 2011) we would like to suggest a study of the skeletonization of meshes (Cornea et al., 2007) for usage in the registration procedure. There are several ways a skeleton may help DGM:

1. Registration of just the skeletons may provide efficient rough registration.
2. Candidate selection may be performed using a ray originating at the nearest point in the skeleton that intersects with the query point and continues until it intersects with T
3. Distance computation might be more accurate when using the intersection ray.

Skeleton ICP (SKICP) is an algorithm for registration of 2-D shapes proposed by Li et al. (2012). The two mentioned papers might be good leads when developing a ICP procedure boosted by skeletonization.

Appendix A

Experiments

A presentation of each experiment will be given here. Three different datasets have been used for experimentation, each of which has been given their separate section. These sections start with an explanation of the setup including: a picture of target, a picture of source, an image of the initial orientation of the data sets, and lastly an image of the result after performing a registration using ordinary ICP. Because the subsampling procedure adds an element of randomness, a table showing the results of execution using ten different seeds will also be presented to show how much the results vary when using the same parameters. For evaluation metric, the mean L_2 distance is used as the evaluation metric.

Most of the experiments consist of multiple executions of DGM with different parameters. The oldest dataset is used as the target in each execution. The results are presented in two scatterplots, one with all results and one zoomed to the area at which they clustered. The purpose of the zoomed plot is to show how the results varied depending on the starting threshold, whereas the zoomed out plot is given to show the presence of outliers. The mean L_2 distance resulting from registration using ordinary ICP is displayed by a green bar in each of the plots, as long as it is within reach. The execution that provided the registration with the lowest mean L_2 distance between source and target is presented as a picture of the source dataset with colors representing the distance from target. Additionally, the results are presented in tables containing the following columns:

points The number of points used for registration

iterations How many iterations done before convergence

n The number of candidates selected for the filtering step in the ICP module

range y The starting threshold configured for the filter

l0_avg The mean l_0 distance between source and target, that is the number of points that did not move (more than a small threshold) divided by the total amount of points in source

l2_avg The mean euclidean distance between source and target

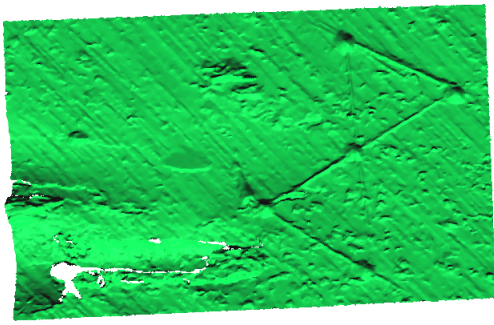
l0_sum The summed l_0 distance between source and target, that is the number of points that did not move (more than a small threshold)

l2_sum The sum euclidean distances between each point in source and its matching point in target

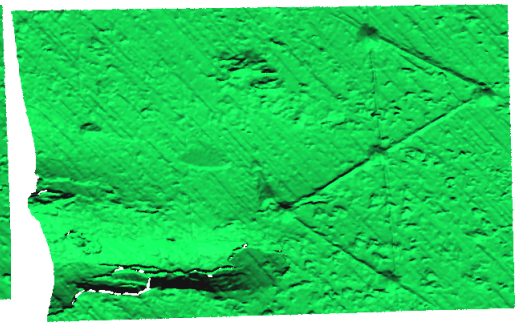
A.1 Nidaros 1

A.1.1 Setup

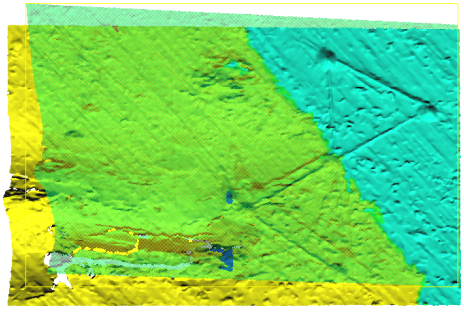
The first dataset is from a scan of the Nidaros Cathedral. The datasets include a troubled area, namely the flake in the lower left corner of the datasets that has fallen off during the passing year. We believe this vastly divergent area is causing the color of the bottom part of the registered dataset to fade to blue indicating that the area has moved outwards since the previous scan was taken. Other areas of interest are the mason mark in the shape of an hourglass.



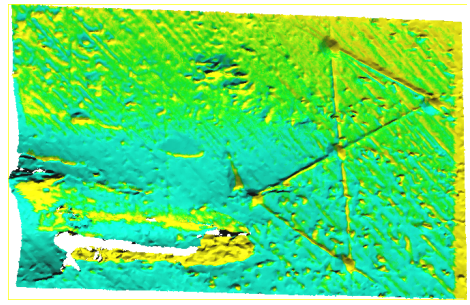
(a) Target scan of the Nidaros Cathedral taken 10.04 2013



(b) Source scan of the Nidaros Cathedral taken 24.09 2014



(c) Initial alignment of the datasets



(d) Registration of Source using ordinary ICP

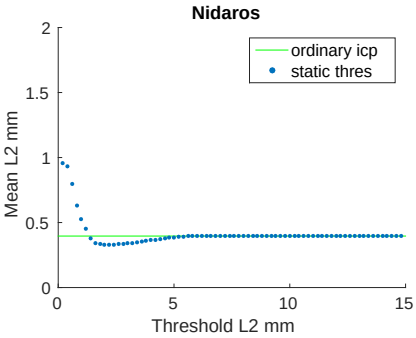
Figure A.1: The setup for the following experiment. Ordinary ICP was run ten times with ten different seeds. Between ten different executions using ordinary ICP, the mean L_2 distance varied with approximately 0.002 mm.

seed	iterations	n	l0_avg	l2_avg	l0_sum	l2_sum
638492957	67	1	0	0.394623	0	625469
390847561	169	1	0	0.398402	0	631458
830521483	39	1	0	0.399645	0	633429
490874508	57	1	0	0.395949	0	627571
192384734	172	1	0	0.393516	0	623715
345008754	63	1	0	0.396769	0	628871
654678432	45	1	0	0.401674	0	636645
234567678	51	1	0	0.397246	0	629626
123456789	95	1	0	0.397551	0	630110
987654321	115	1	0	0.396079	0	627777

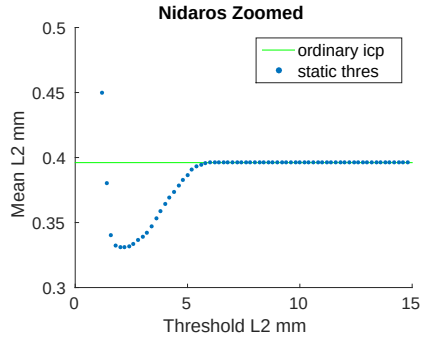
Table A.1: Results using ten different seeds for uniform subsampling

A.1.2 Compatibility: Distance, Depreciation: Static

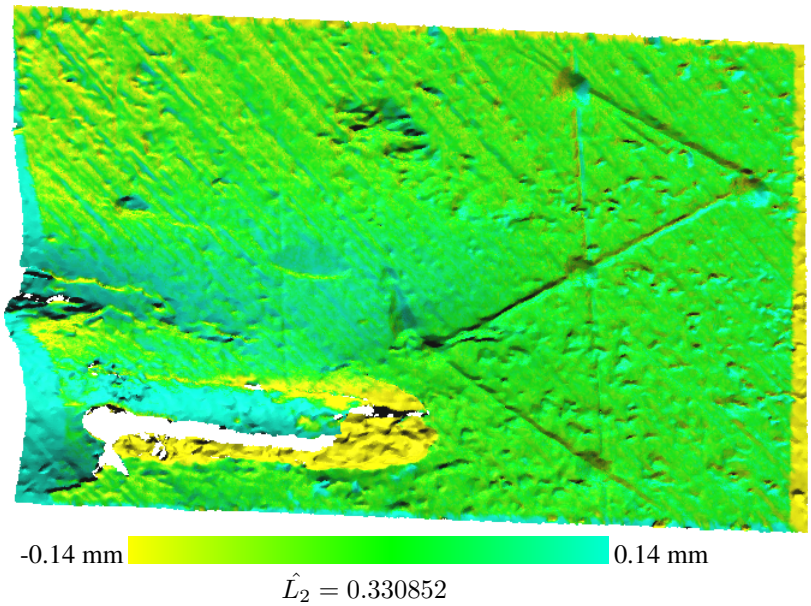
In this experiment, we ran multiple executions of DGM using the distance between the query point and the candidate as compatibility. No depreciation for the threshold was used. We used a different threshold for each execution, ranging from 15mm to 0.1mm. The green color indicating little to no movement tells us that the registration is quite accurate (see Figure A.2). The result may still be improved, the stripes of blue and yellow in the upper left corner and the blue and yellow colors in the mason mark might indicate that the registered mesh is too far to the right.



(a) A plot showing all $\hat{L_2}$ distances between S and T after execution with the static (non-changing) threshold. The X-axis represents the threshold for how close the nearest point has to be for acceptance. The Y-axis represents the mean distance to T per point



(b) A plot showing the same data as in *a)* but zoomed onto the cluster for showing differences between them. The points seem to cluster between 0.3 and 0.4 mm



(c) The registered S dataset closest to T with colors indicating the pointwise distance to T . Setting the threshold to $2.2mm$ yield the best registration. Note that the color only represent movement along the Z-axis, which is inwards/outwards from the image

Figure A.2: The results after registration using no depreciation for the threshold when matching points

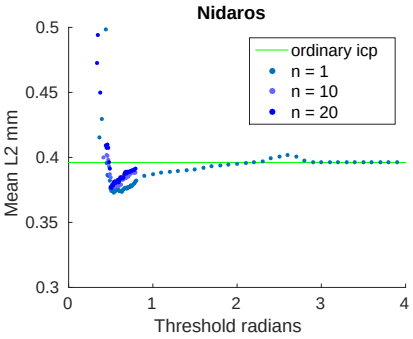
Raw Data

points	iterations	n	range y	l0_avg	l2_avg	l0_sum	l2_sum
1584978	200	1	0.2	0.668524	0.95797	1059600	1518360
1584978	200	1	0.4	0.445616	0.930281	706291	1474470
1584978	200	1	0.6	0.305079	0.795016	483543	1260080
1584978	200	1	0.8	0.175078	0.627818	277494	995078
1584978	200	1	1	0.105531	0.523172	167265	829216
1584978	200	1	1.2	0.0788383	0.44976	124957	712860
1584978	200	1	1.4	0.0672091	0.380214	106525	602631
1584978	200	1	1.6	0.0596734	0.340197	94581	539205
1584978	200	1	1.8	0.0549616	0.332602	87113	527166
1584978	200	1	2	0.0505982	0.331037	80197	524687
1584978	200	1	2.2	0.0461533	0.330852	73152	524393
1584978	200	1	2.4	0.0415078	0.331849	65789	525973
1584978	200	1	2.6	0.0369488	0.333726	58563	528949
1584978	200	1	2.8	0.0321866	0.33658	51015	533472
1584978	200	1	3	0.0280906	0.339308	44523	537796
1584978	200	1	3.2	0.0241568	0.342409	38288	542710
1584978	200	1	3.4	0.0201504	0.34736	31938	550558
1584978	200	1	3.6	0.0164198	0.353439	26025	560193
1584978	200	1	3.8	0.0134185	0.358545	21268	568287
1584978	200	1	4	0.0110614	0.36442	17532	577598
1584978	200	1	4.2	0.00903798	0.369067	14325	584963
1584978	148	1	4.4	0.00687139	0.373417	10891	591857
1584978	179	1	4.6	0.00505307	0.378387	8009	599735
1584978	189	1	4.8	0.00337796	0.382733	5354	606623
1584978	190	1	5	0.00230476	0.38651	3653	612610
1584978	146	1	5.2	0.0013874	0.391052	2199	619809
1584978	89	1	5.4	0.000746383	0.393014	1183	622918
1584978	163	1	5.6	0.000295903	0.394625	469	625472
1584978	200	1	5.8	0.000074449	0.395614	118	627039
1584978	159	1	6	1.82968E-005	0.396013	29	627672
1584978	115	1	6.2	2.52369E-006	0.396079	4	627777
1584978	188	1	6.4	0	0.396079	0	627777
1584978	187	1	6.6	0	0.396079	0	627777
1584978	187	1	6.8	0	0.396079	0	627777
1584978	187	1	7	0	0.396079	0	627777
1584978	186	1	7.2	0	0.396079	0	627777
1584978	115	1	7.4	0	0.396079	0	627777
1584978	186	1	7.6	0	0.396079	0	627777
1584978	187	1	7.8	0	0.396079	0	627777
1584978	115	1	8	0	0.396079	0	627777
1584978	185	1	8.2	0	0.396079	0	627777
1584978	185	1	8.4	0	0.396079	0	627777

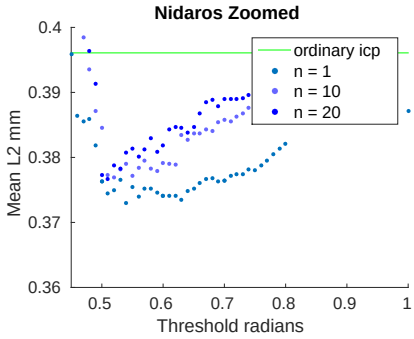
1584978	185	1	8.6	0	0.396079	0	627777
1584978	115	1	8.8	0	0.396079	0	627777
1584978	115	1	9	0	0.396079	0	627777
1584978	185	1	9.2	0	0.396079	0	627777
1584978	185	1	9.4	0	0.396079	0	627777
1584978	185	1	9.6	0	0.396079	0	627777
1584978	185	1	9.8	0	0.396079	0	627777
1584978	115	1	10	0	0.396079	0	627777
1584978	185	1	10.2	0	0.396079	0	627777
1584978	185	1	10.4	0	0.396079	0	627777
1584978	185	1	10.6	0	0.396079	0	627777
1584978	185	1	10.8	0	0.396079	0	627777
1584978	185	1	11	0	0.396079	0	627777
1584978	185	1	11.2	0	0.396079	0	627777
1584978	185	1	11.4	0	0.396079	0	627777
1584978	185	1	11.6	0	0.396079	0	627777
1584978	115	1	11.8	0	0.396079	0	627777
1584978	185	1	12	0	0.396079	0	627777
1584978	115	1	12.2	0	0.396079	0	627777
1584978	185	1	12.4	0	0.396079	0	627777
1584978	115	1	12.6	0	0.396079	0	627777
1584978	115	1	12.8	0	0.396079	0	627777
1584978	115	1	13	0	0.396079	0	627777
1584978	115	1	13.2	0	0.396079	0	627777
1584978	115	1	13.4	0	0.396079	0	627777
1584978	185	1	13.6	0	0.396079	0	627777
1584978	185	1	13.8	0	0.396079	0	627777
1584978	185	1	14	0	0.396079	0	627777
1584978	185	1	14.2	0	0.396079	0	627777
1584978	185	1	14.4	0	0.396079	0	627777
1584978	185	1	14.6	0	0.396079	0	627777
1584978	185	1	14.8	0	0.396079	0	627777

A.1.3 Compatibility: Normal Vector, Depreciation: Static

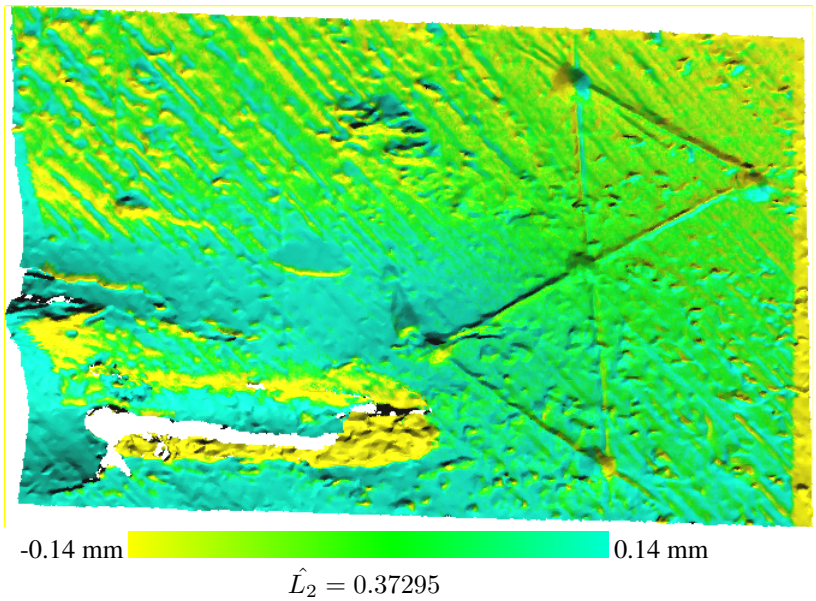
In this experiment, we ran multiple executions of DGM using the normal vector between the query point and the candidate as compatibility. No depreciation for the threshold was used. We used a different threshold for each execution ranging from 4 to 0 radians. We experimented with using the 1, 10 and 20 nearest points as candidates. Though the registration is better than using ordinary ICP, the blue and yellow diagonal stripes indicate that the registered mesh is slightly off. The results can be seen in Figure A.3.



(a) A plot showing all $\hat{L_2}$ distances between S and T after execution with the static (non-changing) threshold. The compatibility was computed from the angle between the normal vectors of corresponding points. The X-axis represents the threshold for how small the angle has to be to be accepted. The Y-axis represents the mean distance to T per point



(b) A plot showing the same data as in Figure A.3a but zoomed onto the cluster. The points seem to cluster between 0.37mm and 0.39mm.



(c) The registered S dataset closest to T with colors indicating the pointwise distance to T . The best result was generated by accepting only points with an angle less than 0.54 radians. Note that the color only represent movement along the Z-axis, which is inwards/outwards from the image

Figure A.3: The results after registration using a static threshold when matching points

Raw Data

points	Iterations	n	range y	l0_avg	l2_avg	l0_sum	l2_sum
1584978	1	1	0.1	0.946727	1.201	1500540	1903550
1584978	1	1	0.2	0.848787	1.17233	1345310	1858110
1584978	1	1	0.3	0.785346	1.0437	1244760	1654240
1584978	1	1	0.31	0.629667	0.845118	998009	1339490
1584978	1	1	0.32	0.533718	0.692914	845932	1098250
1584978	1	1	0.33	0.558677	0.661655	885490	1048710
1369842	5	1	0.34	0.990214	33.373	1356440	45715800
1584978	1	1	0.35	0.576929	0.714376	914419	1132270
988416	1	1	0.36	0.948747	4.8312	937757	4775240
1584978	8	1	0.37	0.183063	0.415173	290151	658039
1584978	29	1	0.38	0.653846	0.848695	1036330	1345160
1584978	27	1	0.39	0.193428	0.429314	306579	680454
1584978	160	1	0.4	0.653915	1.11356	1036440	1764970
1584978	1	1	0.41	0.590301	0.74966	935614	1188190
1584978	152	1	0.42	0.440558	0.611139	698274	968643
1584978	1	1	0.43	0.546068	0.712978	865505	1130050
1584978	1	1	0.44	0.246877	0.498432	391294	790004
1584978	200	1	0.45	0.161398	0.39591	255812	627509
1584978	200	1	0.46	0.150847	0.386389	239089	612418
1584978	81	1	0.47	0.148965	0.385582	236107	611138
1584978	61	1	0.48	0.14648	0.385931	232167	611692
1584978	103	1	0.49	0.141707	0.381895	224602	605295
1584978	108	1	0.5	0.137444	0.376248	217845	596345
1584978	108	1	0.5	0.137444	0.376248	217845	596345
1584978	99	1	0.51	0.135382	0.37445	214578	593495
1584978	70	1	0.52	0.133473	0.374942	211552	594275
1584978	66	1	0.53	0.132555	0.376504	210097	596751
1584978	200	1	0.54	0.12927	0.37295	204890	591118
1584978	77	1	0.55	0.129426	0.375478	205138	595124
1584978	200	1	0.56	0.126817	0.37398	201002	592750
1584978	96	1	0.57	0.126729	0.375154	200862	594610
1584978	200	1	0.58	0.125303	0.375197	198603	594679
1584978	89	1	0.59	0.123426	0.374619	195627	593763
1584978	65	1	0.6	0.121754	0.374114	192977	592962
1584978	65	1	0.6	0.121754	0.374114	192977	592962
1584978	53	1	0.61	0.120373	0.374051	190788	592863
1584978	49	1	0.62	0.119357	0.374083	189179	592914
1584978	53	1	0.63	0.117846	0.373481	186784	591959
1584978	51	1	0.64	0.117501	0.374891	186237	594194
1584978	200	1	0.65	0.116457	0.375155	184581	594612
1584978	200	1	0.66	0.115655	0.376112	183311	596129
1584978	200	1	0.67	0.114487	0.376623	181460	596939

1584978	74	1	0.68	0.113253	0.376751	179504	597143
1584978	42	1	0.69	0.111916	0.376282	177384	596398
1584978	58	1	0.7	0.110457	0.376371	175072	596540
1584978	58	1	0.7	0.110457	0.376371	175072	596540
1584978	73	1	0.71	0.109788	0.377193	174012	597842
1584978	69	1	0.72	0.108754	0.377398	172372	598167
1584978	59	1	0.73	0.10761	0.377418	170559	598200
1584978	53	1	0.74	0.106889	0.378201	169417	599440
1584978	200	1	0.75	0.105492	0.378012	167203	599140
1584978	43	1	0.76	0.104755	0.378749	166035	600308
1584978	98	1	0.77	0.104061	0.379498	164934	601497
1584978	108	1	0.78	0.103493	0.380441	164034	602991
1584978	65	1	0.79	0.102711	0.381363	162794	604452
1584978	78	1	0.8	0.101842	0.382089	161417	605603
1584978	85	1	0.9	0.0942013	0.38568	149307	611294
1584978	80	1	1	0.0869552	0.387086	137822	613522
1584978	53	1	1.1	0.0805191	0.388212	127621	615307
1584978	50	1	1.2	0.0747512	0.388966	118479	616502
1584978	70	1	1.3	0.0696988	0.389497	110471	617344
1584978	69	1	1.4	0.0654022	0.390047	103661	618215
1584978	62	1	1.5	0.061486	0.391038	97454	619786
1584978	110	1	1.6	0.0583415	0.391961	92470	621249
1584978	133	1	1.7	0.0551326	0.392988	87384	622877
1584978	81	1	1.8	0.0524007	0.393715	83054	624030
1584978	89	1	1.9	0.0496575	0.394385	78706	625091
1584978	117	1	2	0.0470738	0.395258	74611	626475
1584978	75	1	2.1	0.0444725	0.395827	70488	627377
1584978	63	1	2.2	0.0417924	0.396384	66240	628259
1584978	44	1	2.3	0.0393078	0.397188	62302	629534
1584978	54	1	2.4	0.0366844	0.399403	58144	633045
1584978	74	1	2.5	0.034338	0.400507	54425	634795
1584978	47	1	2.6	0.0320465	0.402087	50793	637299
1584978	78	1	2.7	0.0298887	0.400467	47373	634731
1584978	79	1	2.8	0.0277373	0.397691	43963	630331
1584978	119	1	2.9	0.0259101	0.396497	41067	628440
1584978	90	1	3	0.0238174	0.396073	37750	627767
1584978	97	1	3.1	0.0220041	0.396177	34876	627931
1584978	83	1	3.2	0.020211	0.396075	32034	627769
1584978	83	1	3.3	0.0186413	0.396075	29546	627769
1584978	83	1	3.4	0.0169592	0.396075	26880	627769
1584978	83	1	3.5	0.0153775	0.396075	24373	627769
1584978	83	1	3.6	0.013963	0.396075	22131	627769
1584978	83	1	3.7	0.0126835	0.396075	20103	627769
1584978	83	1	3.8	0.0115276	0.396075	18271	627769
1584978	83	1	3.9	0.0104866	0.396075	16621	627769

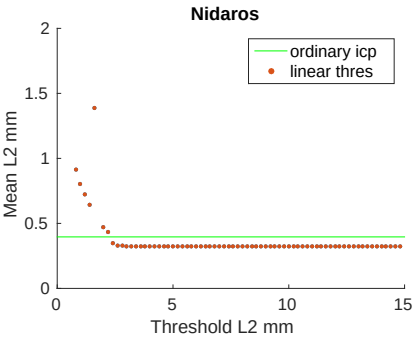
1584978	62	10	0.3	0.324276	0.551576	513970	874236
1584978	1	10	0.31	0.354966	0.550526	562613	872572
1584978	1	10	0.32	0.297463	0.505646	471472	801438
1584978	54	10	0.33	0.293831	0.556156	465716	881495
1584978	1	10	0.34	0.358902	0.562513	568852	891571
295593	2	10	0.35	0.966653	8.88296	285736	2625740
1584978	1	10	0.36	0.916905	3.44121	1453280	5454250
1262163	1	10	0.37	0.980618	12.7477	1237700	16089700
1584978	71	10	0.38	0.534957	0.648224	847895	1027420
1584978	2	10	0.39	0.520586	0.648861	825118	1028430
1339386	1	10	0.4	0.9836	24.4564	1317420	32756500
1584978	1	10	0.41	0.172741	0.400282	273791	634438
1584978	1	10	0.42	0.938059	7.59995	1486800	12045700
1584978	51	10	0.43	0.307568	0.584714	487489	926759
1584978	1	10	0.44	0.565461	0.709254	896244	1124150
1584978	200	10	0.45	0.165354	0.401826	262082	636885
1584978	200	10	0.46	0.162353	0.401267	257326	635999
1584978	200	10	0.47	0.15768	0.39843	249919	631502
1584978	200	10	0.48	0.153689	0.39352	243593	623720
1584978	200	10	0.49	0.146337	0.387187	231941	613683
1584978	200	10	0.5	0.143245	0.384605	227040	609590
1584978	200	10	0.51	0.13702	0.377284	217173	597987
1584978	200	10	0.52	0.13562	0.376951	214955	597459
1584978	200	10	0.53	0.134261	0.378105	212801	599288
1584978	200	10	0.54	0.133056	0.379012	210891	600726
1584978	200	10	0.55	0.130777	0.377181	207278	597823
1584978	200	10	0.56	0.129662	0.378339	205512	599660
1584978	200	10	0.57	0.129107	0.379491	204631	601485
1584978	200	10	0.58	0.126912	0.378233	201153	599491
1584978	200	10	0.59	0.124838	0.377887	197866	598943
1584978	200	10	0.6	0.124208	0.379164	196867	600967
1584978	200	10	0.61	0.122662	0.378995	194417	600699
1584978	200	10	0.62	0.121017	0.378924	191809	600586
1584978	200	10	0.63	0.121044	0.383499	191852	607838
1584978	200	10	0.64	0.119171	0.382735	188883	606627
1584978	200	10	0.65	0.118373	0.383658	187618	608090
1584978	200	10	0.66	0.117176	0.383727	185721	608199
1584978	200	10	0.67	0.116171	0.384346	184129	609180
1584978	200	10	0.68	0.11467	0.384114	181749	608813
1584978	200	10	0.69	0.114233	0.385435	181056	610906
1584978	200	10	0.7	0.113345	0.385826	179649	611525
1584978	200	10	0.71	0.112114	0.385529	177698	611055
1584978	200	10	0.72	0.111244	0.386268	176319	612226
1584978	200	10	0.73	0.110429	0.386773	175027	613026
1584978	200	10	0.74	0.109873	0.387603	174146	614342

1584978	127	10	0.75	0.109098	0.388008	172918	614984
1584978	200	10	0.76	0.108105	0.388138	171344	615191
1584978	200	10	0.77	0.106868	0.388229	169384	615334
1584978	200	10	0.78	0.10561	0.388375	167389	615566
1584978	200	10	0.79	0.104649	0.388258	165867	615381
1584978	42	20	0.3	0.47267	0.60739	749172	962699
1584978	28	20	0.31	0.311081	0.534677	493056	847452
1584978	1	20	0.32	0.460375	0.737171	729685	1168400
1584978	18	20	0.33	0.437698	0.639767	693741	1014020
1584978	30	20	0.34	0.243201	0.472405	385468	748751
1584978	1	20	0.35	0.323893	0.493931	513363	782869
1306182	1	20	0.36	0.984757	15.1271	1286270	19758800
1584978	40	20	0.37	0.480322	0.600806	761300	952264
1584978	49	20	0.38	0.194978	0.449898	309036	713079
1584978	1	20	0.39	0.893527	5.67338	1416220	8992180
1584978	1	20	0.4	0.265985	0.507945	421580	805082
1523766	2	20	0.41	0.985227	24.4513	1501260	37258100
1584978	1	20	0.42	0.895146	5.72433	1418790	9072940
1584978	1	20	0.43	0.29084	0.574619	460975	910758
1584978	200	20	0.44	0.169186	0.409258	268156	648665
1584978	200	20	0.45	0.165327	0.407176	262040	645365
1584978	200	20	0.46	0.164167	0.409826	260201	649566
1584978	200	20	0.47	0.16113	0.407104	255387	645252
1584978	200	20	0.48	0.152319	0.396413	241423	628307
1584978	200	20	0.49	0.145068	0.391273	229930	620159
1584978	200	20	0.5	0.138647	0.377303	219752	598017
1584978	200	20	0.51	0.135992	0.376669	215544	597012
1584978	200	20	0.52	0.136543	0.378792	216417	600377
1584978	200	20	0.53	0.135941	0.378313	215463	599618
1584978	200	20	0.54	0.135203	0.380737	214293	603460
1584978	200	20	0.55	0.133869	0.381368	212179	604459
1584978	200	20	0.56	0.131292	0.380158	208095	602543
1584978	200	20	0.57	0.129715	0.381249	205595	604271
1584978	200	20	0.58	0.128568	0.382944	203777	606957
1584978	200	20	0.59	0.12686	0.380863	201071	603659
1584978	200	20	0.6	0.125387	0.381896	198735	605297
1584978	200	20	0.61	0.12425	0.384334	196933	609160
1584978	200	20	0.62	0.123037	0.384666	195011	609687
1584978	200	20	0.63	0.121756	0.384511	192981	609441
1584978	200	20	0.64	0.119573	0.383755	189521	608243
1584978	200	20	0.65	0.119248	0.384686	189006	609718
1584978	200	20	0.66	0.118231	0.386788	187393	613050
1584978	200	20	0.67	0.117858	0.388496	186802	615757
1584978	200	20	0.68	0.116854	0.388817	185211	616266
1584978	200	20	0.69	0.115069	0.387935	182382	614868

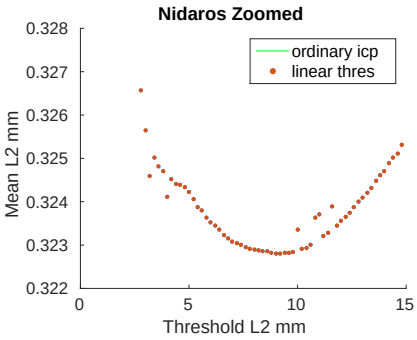
1584978	200	20	0.7	0.114287	0.388973	181143	616514
1584978	200	20	0.71	0.113053	0.388993	179186	616545
1584978	200	20	0.72	0.112307	0.38897	178004	616508
1584978	200	20	0.73	0.111405	0.389082	176575	616687
1584978	200	20	0.74	0.110648	0.389614	175374	617529
1584978	200	20	0.75	0.109506	0.389755	173565	617753
1584978	200	20	0.76	0.108506	0.390019	171980	618171
1584978	200	20	0.77	0.10746	0.390217	170321	618485
1584978	200	20	0.78	0.106433	0.390827	168694	619452
1584978	200	20	0.79	0.105598	0.391225	167371	620083

A.1.4 Compatibility: Distance, Depreciation: Linear

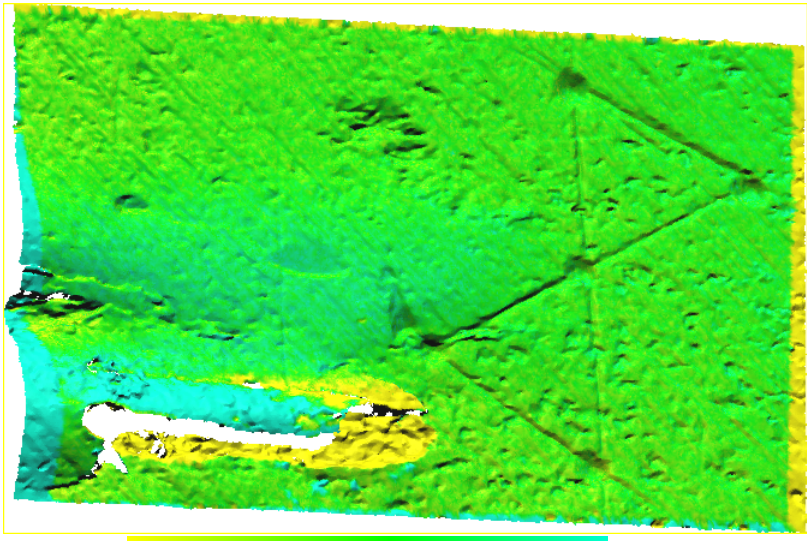
In this experiment, we ran multiple executions of DGM using the distance between the query point and the candidate as compatibility. Linear depreciation was used for the threshold. We used a different starting thresholds for each execution, ranging from 15mm to 0.1mm. The green color indicating little to no movement tells us that the registration is quite accurate (see Figure A.4). The yellow and blue stripes we saw in the previous experiments are not present in this registration.



(a) A plot showing all $\hat{L_2}$ distances between S and T after execution with the iterative linear threshold. The X-axis represents the distance threshold at the start of execution. The Y-axis represents the mean Euclidean distance to T per point



(b) A plot showing the $\hat{L_2}$ distances as in a) but limited to the distances below 0.325. The points seem to cluster between 0.324 and 0.323.



(c) The registered S dataset closest to T with colors indicating the pointwise distance to T . Note that the color only represent movement along the Z-axis, which is inwards/outwards from the image. Setting the starting threshold to 9mm provided the best result with a mean Euclidean distance per point at $\hat{L_2} = 0.322802$

Figure A.4: The results after registration using a linearly decreasing threshold when matching points

Raw Data

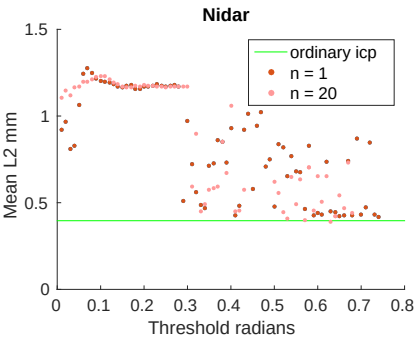
points	iterations	n	range y	l0_avg	l2_avg	l0_sum	l2_sum
1281963	196	1	0.2	0.994235	24.5115	1274570	31422800
1267512	200	1	0.4	0.985072	14.6864	1248590	18615200
1374303	200	1	0.6	0.956995	13.0539	1315200	17940000
1584978	200	1	0.8	0.283177	0.91651	448829	1452650
1584978	200	1	1	0.214363	0.804081	339761	1274450
1584978	199	1	1.2	0.164122	0.723663	260129	1146990
1584978	200	1	1.4	0.114837	0.640968	182014	1015920
1584978	200	1	1.6	0.244767	1.38839	387951	2200570
1584978	200	1	1.8	0.374162	2.02906	593038	3216020
1584978	200	1	2	0.0603573	0.473444	95665	750398
1584978	200	1	2.2	0.0514455	0.430798	81540	682806
1584978	200	1	2.4	0.0430038	0.344648	68160	546260
1584978	200	1	2.6	0.0385198	0.329274	61053	521892
1584978	200	1	2.8	0.03435	0.326566	54444	517600
1584978	200	1	3	0.0304029	0.325645	48188	516140
1584978	1	1	3.2	0.0269404	0.324595	42700	514476
1584978	200	1	3.4	0.0241852	0.325016	38333	515143
1584978	200	1	3.6	0.0214773	0.324811	34041	514818
1584978	200	1	3.8	0.0184356	0.324702	29220	514645
1584978	1	1	4	0.0156715	0.324109	24839	513706
1584978	200	1	4.2	0.0134033	0.324518	21244	514353
1584978	200	1	4.4	0.0116046	0.324407	18393	514178
1584978	200	1	4.6	0.00973831	0.324383	15435	514140
1584978	200	1	4.8	0.00789916	0.32434	12520	514071
1584978	200	1	5	0.00631239	0.324231	10005	513900
1584978	200	1	5.2	0.00486	0.324065	7703	513636
1584978	200	1	5.4	0.00368964	0.323882	5848	513347
1584978	200	1	5.6	0.00284042	0.323792	4502	513204
1584978	200	1	5.8	0.00206312	0.323631	3270	512948
1584978	200	1	6	0.00149403	0.323528	2368	512785
1584978	200	1	6.2	0.000988657	0.323442	1567	512649
1584978	200	1	6.4	0.0005672	0.323348	899	512500
1584978	200	1	6.6	0.00029401	0.323224	466	512303
1584978	200	1	6.8	0.000109781	0.323154	174	512192
1584978	200	1	7	2.08205E-005	0.323075	33	512067
1584978	200	1	7.2	1.26185E-006	0.323051	2	512028
1584978	200	1	7.4	0	0.323009	0	511961
1584978	200	1	7.6	0	0.322959	0	511883
1584978	200	1	7.8	0	0.322911	0	511807
1584978	200	1	8	0	0.322891	0	511775
1584978	1	1	8.2	0	0.322876	0	511751
1584978	200	1	8.4	0	0.322859	0	511725

1584978	200	1	8.6	0	0.322858	0	511723
1584978	200	1	8.8	0	0.322815	0	511655
1584978	200	1	9	0	0.322802	0	511634
1584978	200	1	9.2	0	0.322808	0	511644
1584978	200	1	9.4	0	0.322816	0	511656
1584978	200	1	9.6	0	0.322819	0	511661
1584978	200	1	9.8	0	0.322838	0	511691
1584978	1	1	10	0	0.323355	0	512511
1584978	200	1	10.2	0	0.322906	0	511799
1584978	200	1	10.4	0	0.322938	0	511850
1584978	200	1	10.6	0	0.323	0	511948
1584978	1	1	10.8	0	0.323625	0	512939
1584978	1	1	11	0	0.323704	0	513064
1584978	200	1	11.2	0	0.323206	0	512274
1584978	200	1	11.4	0	0.323281	0	512394
1584978	1	1	11.6	0	0.323887	0	513354
1584978	200	1	11.8	0	0.323441	0	512647
1584978	200	1	12	0	0.323554	0	512826
1584978	200	1	12.2	0	0.323656	0	512988
1584978	200	1	12.4	0	0.323751	0	513138
1584978	200	1	12.6	0	0.323866	0	513320
1584978	200	1	12.8	0	0.324006	0	513543
1584978	200	1	13	0	0.324099	0	513691
1584978	200	1	13.2	0	0.324213	0	513870
1584978	200	1	13.4	0	0.324319	0	514039
1584978	200	1	13.6	0	0.324485	0	514301
1584978	200	1	13.8	0	0.324614	0	514506
1584978	200	1	14	0	0.324705	0	514650
1584978	200	1	14.2	0	0.324884	0	514934
1584978	200	1	14.4	0	0.325019	0	515149
1584978	95	1	14.6	0	0.325114	0	515299
1584978	200	1	14.8	0	0.325308	0	515606

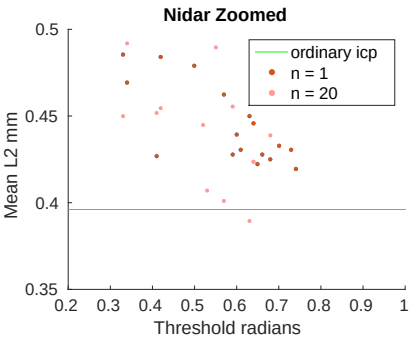
A.1.5 Compatibility: Normal Vector, Depreciation: Linear, Restricted

In this experiment, we ran multiple executions of DGM using the normal vector between the query point and the candidate as compatibility. The threshold was configured with linear depreciation. We used a different threshold for each execution ranging from 4 to 0 radians. We experimented with using the 1 and 20 nearest points as candidates. This configuration outperformed the ordinary ICP algorithm in only one case and, in that case, the increase in accuracy was too small to be considered caused by the compatibility measure. In the start of this chapter, we explained how the result using ordinary ICP could vary with +/- 0.01mm. The best configuration was with a starting threshold at 0.63 radians and 20 candidates. As with all cases, we assume that a lower mean L_2 distance is equivalent to a better registration, this, however, is debatable. In this case, one may argue that the ordinary ICP algorithm performed better despite the resulting mean L_2 distance being higher.

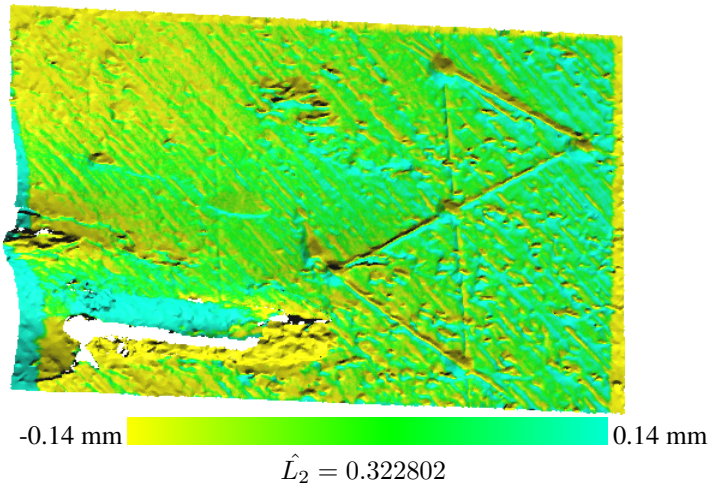
The reason we speculate about it here is because, in the best registrations, the birth of a flake was revealed above the troubled area. When ordinary ICP was used, this area was blue. Whether that was because of erroneous registration caused by the flake or because of actual outward movement, is unknown. The results can be seen in Figure A.5c



(a) A plot showing all $\hat{L_2}$ distances between S and T after execution with the iterative linear threshold. The X-axis represents the distance threshold at the start of execution. The Y-axis represents the mean Euclidean distance to T per point



(b) A plot showing the $\hat{L_2}$ distances as in a) but limited to the distances below 0.325. The points seem to cluster between 0.324 and 0.323.



(c) The registered S dataset closest to T with colors indicating the pointwise distance to T . Note that the color only represent movement along the Z-axis, which is inwards/outwards from the image. Setting the starting threshold to 9mm provided the best result with a mean Euclidean distance per point at $\hat{L_2} = 0.322802$

Figure A.5: The results after registration using a linearly decreasing threshold when matching points

Raw Data

points	iterations	n	range y	l0_avg	l2_avg	l0_sum	l2_sum
1578522	1	1	0	1	7.06812	1578520	11157200
1584978	2	1	0.01	0.999906	0.921503	1584830	1460560
1584978	2	1	0.02	0.999155	0.968686	1583640	1535350
1584978	1	1	0.03	0.995778	0.808018	1578290	1280690
1584978	1	1	0.04	0.989608	0.83013	1568510	1315740
1584978	1	1	0.05	0.988097	1.0624	1566110	1683880
1584978	1	1	0.06	0.984905	1.24451	1561050	1972530
1584978	1	1	0.07	0.978966	1.27711	1551640	2024190
1584978	1	1	0.08	0.968107	1.24727	1534430	1976890
1584978	1	1	0.09	0.957522	1.21722	1517650	1929270
1584978	1	1	0.1	0.946611	1.2008	1500360	1903240
1584978	1	1	0.11	0.935686	1.19683	1483040	1896950
1584978	1	1	0.12	0.924845	1.19434	1465860	1893010
1584978	1	1	0.13	0.916797	1.18596	1453100	1879720
1584978	1	1	0.14	0.902924	1.16828	1431110	1851690
1584978	1	1	0.15	0.89075	1.16562	1411820	1847480
1584978	1	1	0.16	0.88832	1.17635	1407970	1864490
1584978	1	1	0.17	0.878513	1.17937	1392420	1869270
1584978	1	1	0.18	0.866041	1.15749	1372660	1834590
1584978	1	1	0.19	0.855106	1.15757	1355320	1834720
1584978	1	1	0.2	0.848354	1.16909	1344620	1852980
1584978	1	1	0.21	0.840226	1.17215	1331740	1857830
1584978	1	1	0.22	0.831005	1.17602	1317120	1863970
1584978	1	1	0.23	0.823479	1.18405	1305200	1876690
1584978	1	1	0.24	0.813466	1.17385	1289330	1860520
1584978	1	1	0.25	0.803673	1.16995	1273800	1854340
1584978	1	1	0.26	0.795118	1.173	1260240	1859180
1584978	1	1	0.27	0.788891	1.18016	1250380	1870520
1584978	1	1	0.28	0.777516	1.17134	1232340	1856550
1584978	36	1	0.29	0.315631	0.510204	500268	808662
1584978	2	1	0.3	0.674035	0.971195	1068330	1539320
1584978	1	1	0.31	0.620705	0.721168	983804	1143040
1584978	12	1	0.32	0.384624	0.561923	609621	890635
1584978	14	1	0.33	0.30258	0.485301	479583	769191
1584978	17	1	0.34	0.285786	0.469497	452965	744143
1584978	18	1	0.35	0.603048	0.715302	955818	1133740
1584978	18	1	0.36	0.61324	0.728311	971972	1154360
1584978	19	1	0.37	0.671329	0.85917	1064040	1361760
1584978	1	1	0.38	0.658407	0.853339	1043560	1352520
1584978	1	1	0.39	0.577575	0.733339	915444	1162330
1584978	1	1	0.4	0.673966	0.930559	1068220	1474920
1584978	19	1	0.41	0.189527	0.427044	300396	676856

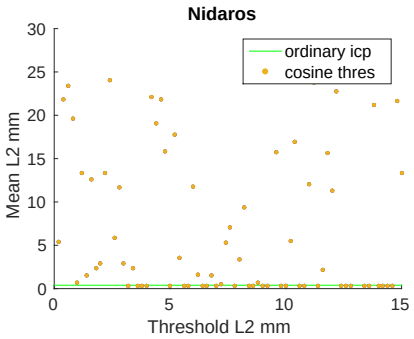
1584978	32	1	0.42	0.221881	0.484258	351676	767539
1584978	24	1	0.43	0.662433	0.920238	1049940	1458560
1584978	28	1	0.44	0.679859	1.01202	1077560	1604020
1584978	23	1	0.45	0.39565	0.579579	627097	918620
1584978	35	1	0.46	0.640098	0.942909	1014540	1494490
1584978	37	1	0.47	0.663726	1.02325	1051990	1621830
1584978	1	1	0.48	0.493977	0.709051	782943	1123830
1584978	1	1	0.49	0.526394	0.74973	834323	1188300
1584978	1	1	0.5	0.205839	0.479199	326250	759519
1584978	1	1	0.51	0.575395	0.836816	911989	1326340
1584978	50	1	0.52	0.542366	0.818711	859638	1297640
1584978	1	1	0.53	0.396876	0.65176	629039	1033020
1584978	52	1	0.54	0.505908	0.767694	801853	1216780
1584978	1	1	0.55	0.421639	0.681885	668289	1080770
1584978	1	1	0.56	0.415924	0.678145	659231	1074850
1584978	1	1	0.57	0.179998	0.462356	285293	732824
1584978	65	1	0.58	0.505075	0.828565	800533	1313260
1584978	1	1	0.59	0.152183	0.427983	241206	678344
1584978	1	1	0.6	0.158391	0.439256	251047	696212
1584978	1	1	0.61	0.149749	0.430438	237349	682235
1584978	72	1	0.62	0.439585	0.736876	696733	1167930
1584978	1	1	0.63	0.158285	0.449941	250878	713147
1584978	1	1	0.64	0.154987	0.445839	245651	706646
1584978	1	1	0.65	0.140253	0.422086	222298	668997
1584978	1	1	0.66	0.142157	0.427625	225315	677776
1584978	83	1	0.67	0.409261	0.739162	648670	1171550
1584978	1	1	0.68	0.138115	0.424857	218910	673389
1584978	84	1	0.69	0.482308	0.868337	764447	1376290
1584978	1	1	0.7	0.137676	0.432931	218214	686186
1584978	93	1	0.71	0.128869	0.471262	204254	746941
1584978	90	1	0.72	0.457637	0.849151	725344	1345890
1584978	1	1	0.73	0.132609	0.430442	210183	682242
1584978	1	1	0.74	0.1243	0.41937	197013	664692
1578522	1	20	0	1	7.06812	1578520	11157200
1584978	1	20	0.01	0.999903	1.10368	1584820	1749300
1584978	1	20	0.02	0.999392	1.1482	1584020	1819880
1584978	1	20	0.03	0.997718	1.1176	1581360	1771380
1584978	1	20	0.04	0.994997	1.16608	1577050	1848220
1584978	1	20	0.05	0.990546	1.17083	1569990	1855740
1584978	1	20	0.06	0.984469	1.19946	1560360	1901120
1584978	1	20	0.07	0.976625	1.19736	1547930	1897780
1584978	1	20	0.08	0.966984	1.21101	1532650	1919420
1584978	1	20	0.09	0.95719	1.22483	1517120	1941330
1584978	1	20	0.1	0.946439	1.23046	1500080	1950250
1584978	1	20	0.11	0.936231	1.23029	1483910	1949990

1584978	1	20	0.12	0.924048	1.21302	1464600	1922600
1584978	1	20	0.13	0.910955	1.19188	1443840	1889110
1584978	1	20	0.14	0.899303	1.18468	1425380	1877700
1584978	1	20	0.15	0.886587	1.17151	1405220	1856820
1584978	1	20	0.16	0.87542	1.16654	1387520	1848940
1584978	1	20	0.17	0.865254	1.16428	1371410	1845350
1584978	1	20	0.18	0.857165	1.17304	1358590	1859240
1584978	1	20	0.19	0.847887	1.17166	1343880	1857050
1584978	1	20	0.2	0.840356	1.17777	1331940	1866750
1584978	1	20	0.21	0.831221	1.17686	1317470	1865290
1584978	1	20	0.22	0.821127	1.17281	1301470	1858880
1584978	1	20	0.23	0.811627	1.17107	1286410	1856120
1584978	1	20	0.24	0.801852	1.16838	1270920	1851860
1584978	1	20	0.25	0.791707	1.16411	1254840	1845100
1584978	1	20	0.26	0.784688	1.16834	1243710	1851790
1584978	1	20	0.27	0.775594	1.1679	1229300	1851100
1584978	1	20	0.28	0.767242	1.16841	1216060	1851900
1584978	1	20	0.29	0.759387	1.16931	1203610	1853340
1584978	1	20	0.3	0.751385	1.17017	1190930	1854700
1584978	27	20	0.31	0.51622	0.594121	818198	941669
1584978	1	20	0.32	0.561711	0.896159	890300	1420390
1584978	12	20	0.33	0.232551	0.449766	368589	712869
1584978	17	20	0.34	0.255015	0.491798	404193	779489
1584978	21	20	0.35	0.422836	0.573764	670186	909404
1584978	24	20	0.36	0.436147	0.584638	691284	926638
1584978	1	20	0.37	0.403233	0.594716	639116	942611
1584978	45	20	0.38	0.63355	0.851668	1004160	1349870
1584978	1	20	0.39	0.496657	0.671373	787191	1064110
1584978	16	20	0.4	0.723703	1.05921	1147050	1678820
1584978	29	20	0.41	0.176888	0.451709	280364	715949
1584978	34	20	0.42	0.173464	0.454577	274936	720494
1584978	38	20	0.43	0.367693	0.575358	582786	911930
1584978	67	20	0.5	0.388055	0.621432	615059	984956
1584978	60	20	0.51	0.26764	0.558344	424204	884963
1584978	73	20	0.52	0.137148	0.444763	217377	704939
1584978	1	20	0.53	0.147446	0.406807	233698	644781
1584978	62	20	0.54	0.388839	0.64934	616302	1029190
1584978	1	20	0.55	0.194037	0.489622	307544	776040
1584978	88	20	0.56	0.370539	0.633626	587296	1004280
1584978	1	20	0.57	0.137275	0.400802	217578	635262
1584978	1	20	0.58	0.437997	0.705233	694216	1117780
1584978	1	20	0.58	0.437997	0.705233	694216	1117780
1584978	76	20	0.59	0.127306	0.455298	201777	721638
1584978	72	20	0.6	0.347431	0.651887	550671	1033230
1584978	1	20	0.61	0.178575	0.507191	283038	803887

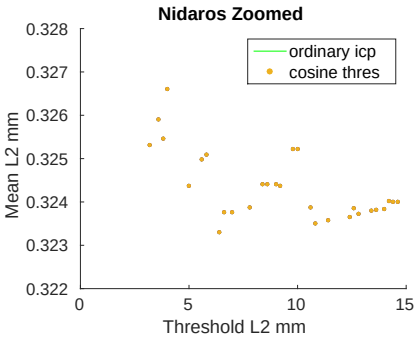
1584978	74	20	0.62	0.324986	0.654857	515095	1037930
1584978	1	20	0.63	0.102321	0.389355	162177	617119
1584978	94	20	0.64	0.107184	0.423519	169885	671268
1584978	94	20	0.64	0.107184	0.423519	169885	671268
1584978	91	20	0.65	0.171359	0.543539	271600	861497
1584978	99	20	0.66	0.115332	0.468681	182798	742849
1584978	1	20	0.67	0.403163	0.732926	639005	1161670
1584978	89	20	0.68	0.110085	0.438941	174483	695712

A.1.6 Compatibility: Distance, Depreciation: Cosine

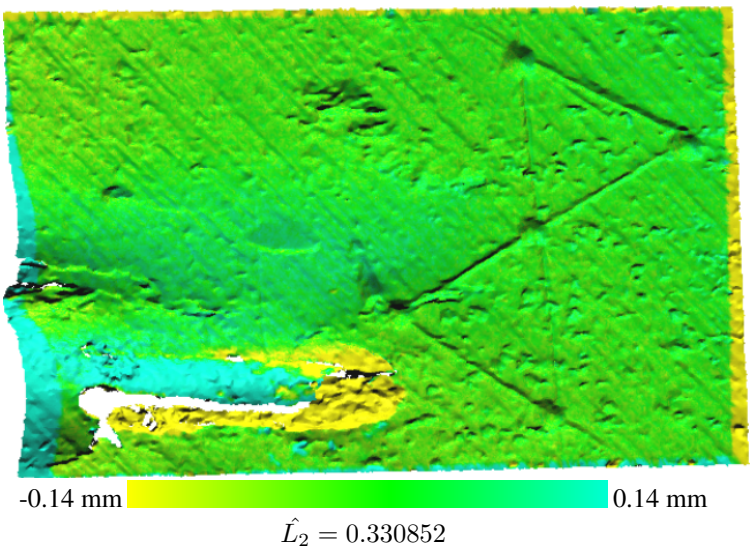
In this experiment, we ran multiple executions of DGM using the distance between the query point and the candidate as compatibility. Cosine depreciation was used for the threshold. We used different starting thresholds for each execution, ranging from 15mm to 0.1mm. The green color indicating little to no movement tells us that the registration is quite accurate (see Figure A.6). Most of the executions, however, resulted in registrations far from optimal.



(a) A plot showing all $\hat{L_2}$ distances between *S* and *T* after execution with the iterative linear threshold. The X-axis represents the distance threshold at the start of execution. The Y-axis represents the mean Euclidean distance to *T* per point



(b) A plot showing the $\hat{L_2}$ distances as in a) but limited to the distances below 0.325. The points seem to cluster between 0.324 and 0.323.



(c) The registered *S* dataset closest to *T* with colors indicating the pointwise distance to *T* . Note that the color only represent movement along the Z-axis, which is inwards/outwards from the image.

Figure A.6: The results after registration using a linearly decreasing threshold when matching points

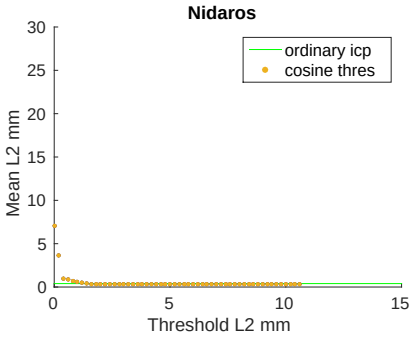
Raw Data

points	iterations	n	range y	l0_avg	l2_avg	l0_sum	l2_sum
1578522	1	1	0	0.994255	30.9596	909132	28308900
914385	1	1	0.2	0.953468	5.43134	1511230	8608560
1584978	1	1	0.4	0.980218	21.8254	1206700	26868200
1231053	1	1	0.6	0.968295	23.4129	1087570	26296900
1123182	1	1	0.8	0.963238	19.6003	803274	16345300
833931	1	1	1	0.156697	0.701966	248361	1112600
1584978	1	1	1.2	0.936482	13.3174	866029	12315500
924768	1	1	1.4	0.304383	1.49438	482441	2368560
1584978	1	1	1.6	0.909843	12.6392	1050490	14592900
1154580	1	1	1.8	0.499787	2.38963	792151	3787510
1584978	1	1	2	0.562186	2.93397	891052	4650290
1584978	1	1	2.2	0.896461	13.31	873256	12965500
974115	1	1	2.4	0.925785	24.0281	777387	20176600
839706	1	1	2.6	0.730092	5.88659	1157180	9330110
1584978	1	1	2.8	0.845841	11.7017	408706	5654210
483195	1	1	3	0.389287	2.92527	616999	4636400
1584945	1	1	3.2	0.0238609	0.325308	37819	515606
1584978	1	1	3.4	0.230179	2.36886	364828	3754580
1584978	1	1	3.6	0.0179876	0.325905	28510	516552
1584978	1	1	3.8	0.0154273	0.32547	24452	515862
1584978	1	1	4	0.0130721	0.326607	20719	517665
1584978	1	1	4.2	0.819641	22.1034	1062520	28653100
1296321	1	1	4.4	0.809886	19.0466	1083740	25487000
1338138	1	1	4.6	0.797646	21.8583	1044050	28610600
1308912	1	1	4.8	0.736104	15.8634	849917	18316100
1154616	1	1	5	0.0047048	0.324366	7457	514113
1584978	1	1	5.2	0.790184	17.7847	968792	21804600
1226034	1	1	5.4	0.235517	3.53589	373289	5604310
1584978	1	1	5.6	0.00195208	0.324988	3094	515099
1584978	1	1	5.8	0.00141327	0.325094	2240	515267
1584978	1	1	6	0.641395	11.7359	594812	10883500
927372	1	1	6.2	0.0345462	1.57871	54755	2502220
1584978	1	1	6.4	0.000230287	0.323293	365	512413
1584978	1	1	6.6	7.94964E-005	0.323762	126	513155
1584978	1	1	6.8	0.0277367	1.52869	43962	2422930
1584978	1	1	7	0	0.323762	0	513155
1584978	1	1	7.2	0	0.524642	0	831546
1584978	1	1	7.4	0.261001	5.28443	364541	7380780
1396704	1	1	7.6	0.367398	7.04318	565997	10850400
1540554	1	1	7.8	0	0.323872	0	513330
1584978	1	1	8	0.0516031	3.39224	81723	5372230
1583685	1	1	8.2	0.444339	9.33076	569393	11956800

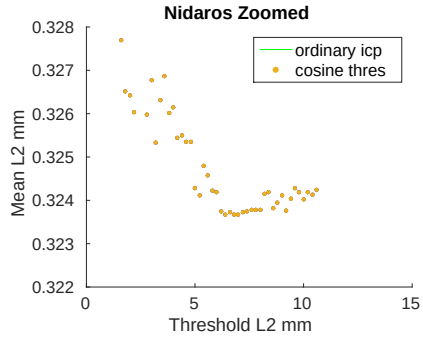
1281438	1	1	8.4	0	0.324417	0	514193
1584978	1	1	8.6	0	0.324416	0	514193
1584978	1	1	8.8	0	0.708963	0	1123690
1584978	1	1	9	0	0.324417	0	514193
1584978	1	1	9.2	0	0.324376	0	514129
1584978	1	1	9.4	0.873403	60.6567	1384320	96139500
1584978	1	1	9.6	0.576998	15.74	580061	15823500
1005309	1	1	9.8	0	0.325216	0	515460
1584978	1	1	10	0	0.325216	0	515460
1584978	1	1	10.2	0.135891	5.4927	212639	8594840
1564776	1	1	10.4	0.63435	16.9605	457474	12231400
721170	1	1	10.6	0	0.323881	0	513344
1584978	1	1	10.8	0	0.323502	0	512744
1584978	1	1	11	0.438006	12.0535	554898	15270200
1266873	1	1	11.2	0.639285	23.8124	856854	31916500
1340331	1	1	11.4	0	0.32358	0	512868
1584978	200	1	11.6	0	2.18146	0	3457560
1584978	1	1	11.8	0.566061	15.6442	423749	11711100
748593	1	1	12	0.392187	11.3239	364188	10515500
928608	1	1	12.2	0.605849	22.7819	777596	29240100
1283481	1	1	12.4	0	0.323649	0	512977
1584978	1	1	12.6	0	0.323854	0	513302
1584978	1	1	12.8	0	0.323735	0	513114
1584978	1	1	13	0.547859	26.7355	468548	22865100
855234	200	1	13.2	0.58353	32.29	457555	25319100
784116	200	1	13.4	0	0.3238	0	513216
1584978	200	1	13.6	0	0.323821	0	513249
1584978	200	1	13.8	0.516217	21.1928	676864	27788000
1311201	200	1	14	0	0.323839	0	513278
1584978	199	1	14.2	0	0.324026	0	513574
1584978	200	1	14.4	0	0.324011	0	513550
1584978	200	1	14.6	0	0.324005	0	513540
1584978	200	1	14.8	0.550962	21.6442	727047	28561500
1319595	200	1	15	0.896461	13.31	873256	12965500

A.1.7 Compatibility: Distance, Depreciation: Cosine, Restricted

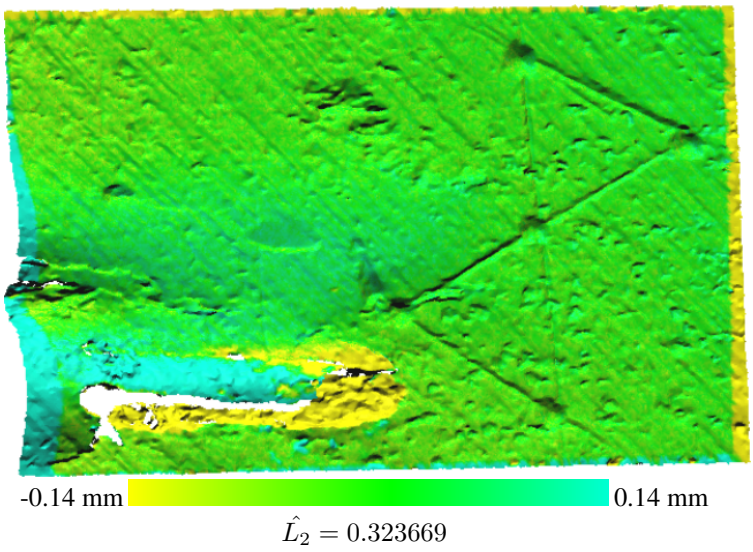
In this experiment, we ran multiple executions of DGM using the distance between the query point and the candidate as compatibility. Cosine depreciation was used for the threshold. We used different starting thresholds for each execution, ranging from 15mm to 0.1mm. The ICP algorithm was configured to terminate if less than ten matches are made during the matching step. The problem with DGM producing mostly outliers were eliminated when a minimum number of matches were introduced. The results can be seen in Figure A.7.



(a) A plot showing all $\hat{L_2}$ distances between S and T after execution with the iterative cosine threshold. The X-axis represents the distance threshold at the start of execution. The Y-axis represents the mean Euclidean distance to T per point



(b) A plot showing the $\hat{L_2}$ distances as in a) but limited to the distances below 0.33. The points seem to cluster between 0.324 and 0.323.



(c) The registered S dataset closest to T with colors indicating the pointwise distance to T . Note that the color only represent movement along the Z-axis, which is inwards/outwards from the image. The best result was at $\hat{L_2} = 0.323669$ when we set the starting threshold to 6.4 mm.

Figure A.7: The results after registration using a linearly decreasing threshold when matching points

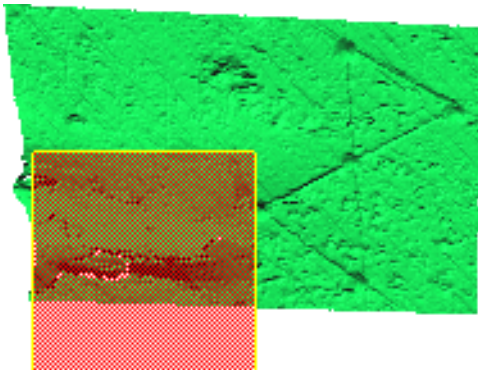
Raw Data

points	iterations	n	range y	l0_avg	l2_avg	l0_sum	l2_sum
1578522	1	1	0	1	7.06812	1578520	11157200
1584978	170	1	0.2	0.960338	3.66643	1522110	5811200
1584978	186	1	0.4	0.459059	0.977632	727598	1549530
1584978	187	1	0.6	0.337272	0.905001	534569	1434410
1584978	191	1	0.8	0.214794	0.703369	340443	1114820
1584978	192	1	1	0.11797	0.559696	186980	887106
1584978	191	1	1.2	0.0886814	0.50407	140558	798939
1584978	194	1	1.4	0.0705821	0.404547	111871	641198
1584978	195	1	1.6	0.0605977	0.327688	96046	519378
1584978	194	1	1.8	0.0558184	0.326513	88471	517516
1584978	196	1	2	0.0514297	0.326421	81515	517369
1584978	196	1	2.2	0.0472019	0.326035	74814	516758
1584978	1	1	2.4	0.042429	0.340999	67249	540476
1584978	1	1	2.6	0.0385747	0.328635	61140	520879
1584978	1	1	2.8	0.0340737	0.325987	54006	516682
1584978	196	1	3	0.0302793	0.326775	47992	517931
1584978	1	1	3.2	0.0267808	0.325338	42447	515654
1584978	197	1	3.4	0.0239745	0.326313	37999	517198
1584978	197	1	3.6	0.0213227	0.326869	33796	518080
1584978	197	1	3.8	0.0182936	0.326009	28995	516717
1584978	197	1	4	0.0156223	0.326151	24761	516942
1584978	197	1	4.2	0.0132178	0.325436	20950	515808
1584978	197	1	4.4	0.0114292	0.325496	18115	515904
1584978	197	1	4.6	0.00956039	0.325356	15153	515682
1584978	197	1	4.8	0.0077837	0.325345	12337	515665
1584978	1	1	5	0.00611428	0.324289	9691	513991
1584978	1	1	5.2	0.00470164	0.324115	7452	513715
1584978	1	1	5.4	0.00359374	0.324803	5696	514805
1584978	197	1	5.6	0.00279058	0.324572	4423	514440
1584978	197	1	5.8	0.00203788	0.324232	3230	513901
1584978	197	1	6	0.00145112	0.324195	2300	513842
1584978	1	1	6.2	0.000903483	0.323736	1432	513114
1584978	1	1	6.4	0.000523036	0.323669	829	513009
1584978	1	1	6.6	0.000230287	0.323733	365	513109
1584978	199	1	6.8	0.000102841	0.323674	163	513016
1584978	199	1	7	1.70349E-005	0.323675	27	513018
1584978	1	1	7.2	0	0.323731	0	513106
1584978	1	1	7.4	0	0.323752	0	513140
1584978	1	1	7.6	0	0.323778	0	513181
1584978	1	1	7.8	0	0.323778	0	513182
1584978	1	1	8	0	0.323778	0	513181
1584978	1	1	8.2	0	0.324153	0	513775

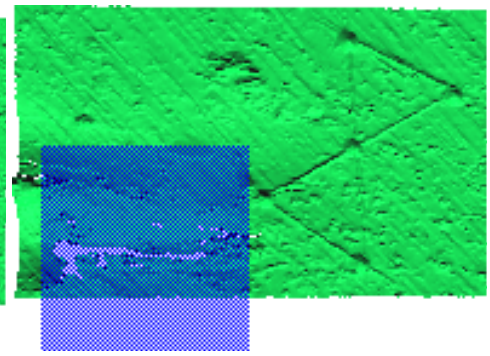
1584978	1	1	8.4	0	0.324181	0	513819
1584978	198	1	8.6	0	0.323824	0	513254
1584978	198	1	8.8	0	0.32395	0	513453
1584978	1	1	9	0	0.324119	0	513721
1584978	198	1	9.2	0	0.323762	0	513155
1584978	1	1	9.4	0	0.324035	0	513589
1584978	1	1	9.6	0	0.324284	0	513983
1584978	1	1	9.8	0	0.324189	0	513832
1584978	1	1	10	0	0.324027	0	513576
1584978	1	1	10.2	0	0.324179	0	513817
1584978	1	1	10.4	0	0.324124	0	513730
1584978	1	1	10.6	0	0.324245	0	513921

A.1.8 Compatibility: Predefined Area, Area: Troubled Area

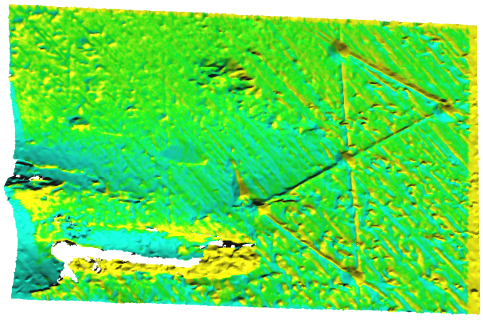
In this experiment, we tested the predefined compatibility threshold. The predefined areas were configured to enclose the troubled area in source and target. Using the areas for excluding the flake yielded the best result. The results can be seen in Figure A.8f.



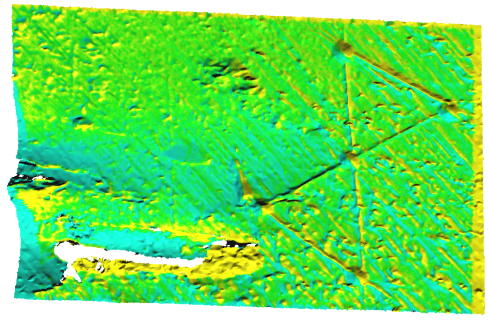
(a) Initial orientation of the target datasets and its exclusion box



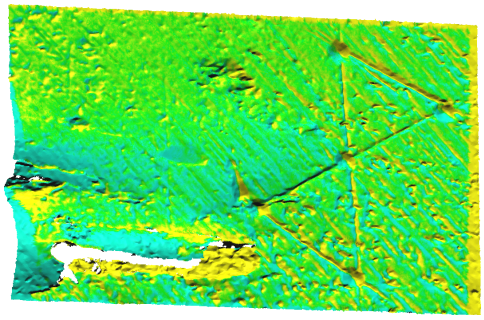
(b) Initial orientation of the source dataset and its exclusion box



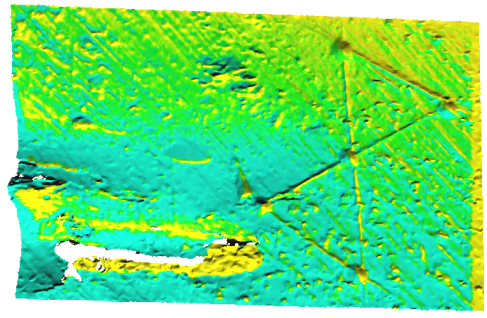
(c) Registration after excluding parts in both datasets. $\hat{L_2} = 0.353767$. Iterations: 85



(d) Registration after excluding a part in only source. $\hat{L_2} = 0.353646$. Iterations: 105



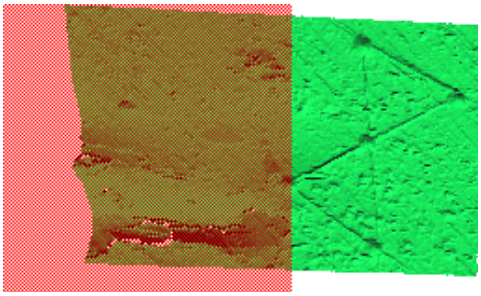
(e) Registration after excluding a part in only target. $\hat{L_2} = 0.355954$. Iterations: 85



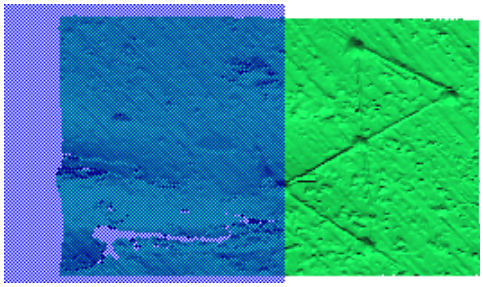
(f) Registration after matching points in unmarked areas with points in unmarked areas and matching points in the blue area only with points in the red area. . $\hat{L_2} = 0.395553$. Iterations: 87

A.1.9 Compatibility: Predefined Area, Area: a Half of the Surface

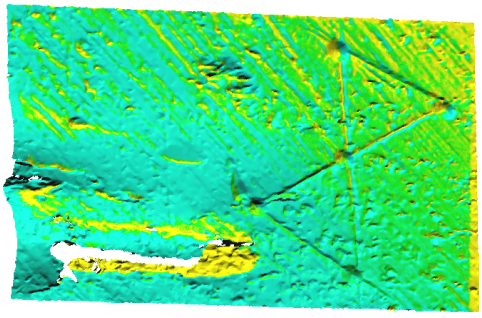
In this experiment, we tested the predefined compatibility threshold. The predefined areas were configured to enclose the left area in source and target. None of the executions yielded a notably good result. The results can be seen in Figure A.9f.



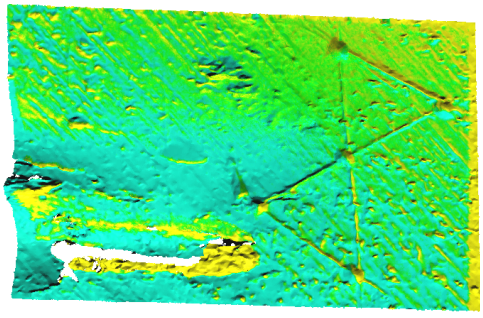
(a) Initial orientation of the data sets and the exclusion boxes



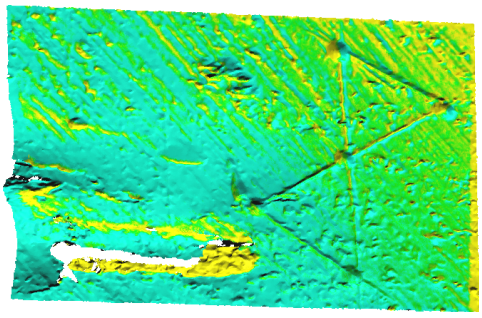
(b) Initial orientation of the data sets and the exclusion boxes



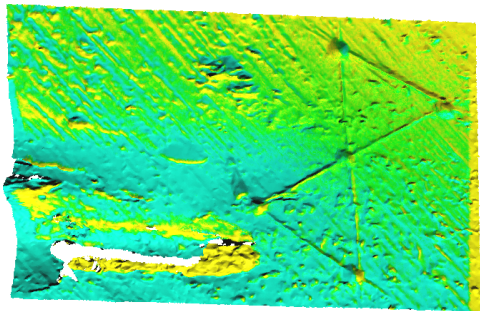
(c) Registration after excludig parts in both datasets. $\hat{L_2} = 0.418919$. Iterations: 67



(d) Registration after excluding a part in only source. $\hat{L_2} = 0.386721$. Iterations: 55



(e) Registration after excluding a part in only target. $\hat{L_2} = 0.418919$. Iterations: 67

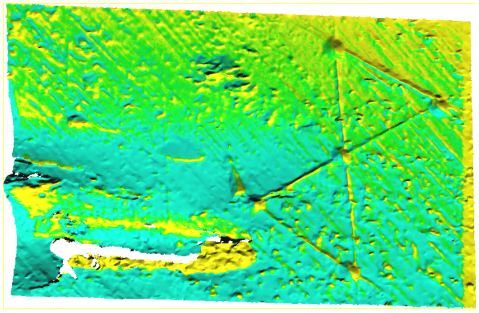


(f) Registration after belonging. $\hat{L_2} = 0.390013$. Iterations: 85

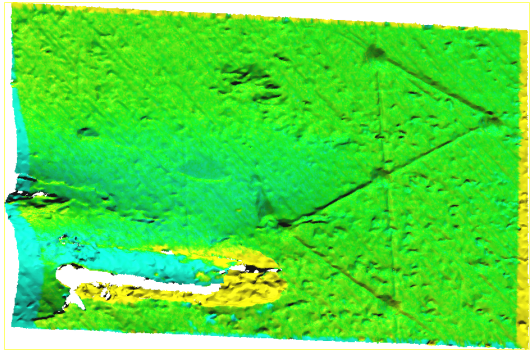
A.2 Nidaros 2

In this experiment, we successfully attempted to align datasets without a good initial alignment. We did so by chaining the ordinary ICP procedure, which works without a proper initial alignment, with an ICP fitting procedure with distance-compatibility and a linearly decreasing threshold. The results can be seen in Figure A.10e.

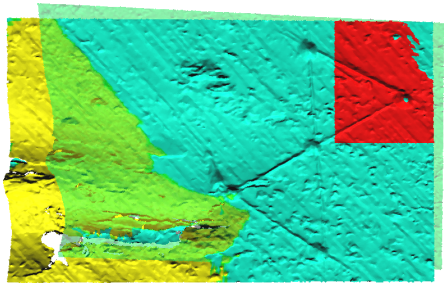
(a) Source and target at a big distance apart from each other



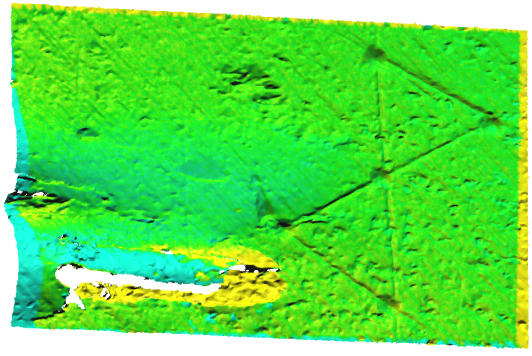
(b) Registration with ordinary ICP. Distance: 0.396079



(c) Registration with ordinary ICP then registered using ICP with the linear threshold. The starting threshold was set to 9 mm. Distance: 0.322805 mm



(d) Placeholder: Registration with SA. Distance: 6.43258

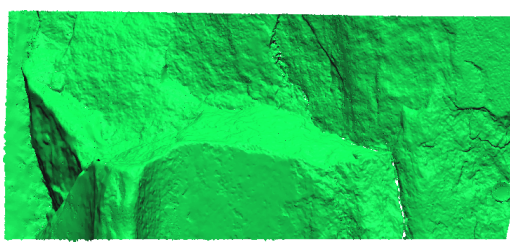


(e) Registration with SA then registered using ICP with the linear threshold. The starting threshold was set to 9 mm. Distance: 0.322946 mm

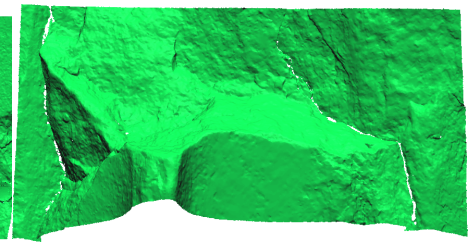
A.3 Elefsis 1

A.3.1 Setup

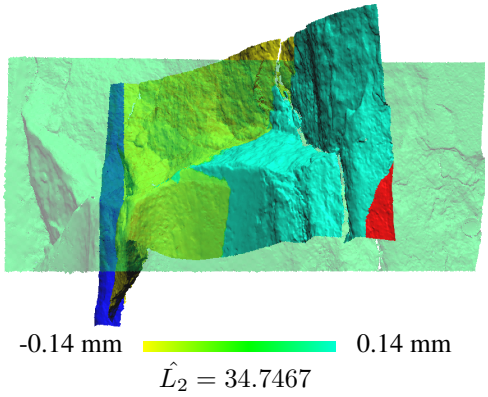
The second dataset we experimented on is from a scan of a column in Elefsis. ICP also performs poorly on this dataset as the green color seems absent after registration. There do not appear to be any large differences between the scans. The results can be seen in Figure A.11.



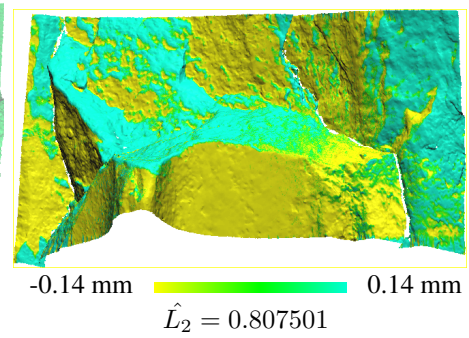
(a) Target scan of the Elefsis column taken 20.03 2013



(b) Source scan of the Elefsis column taken 24.09 2014



(c) Initial alignment of the datasets

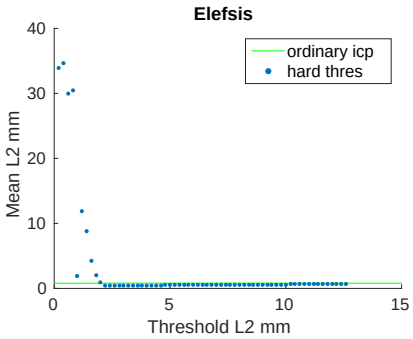


(d) Registration of Source using ordinary ICP

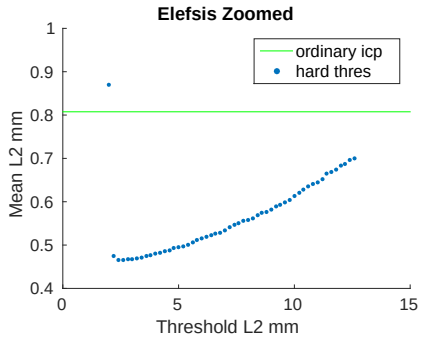
Figure A.11: The setup for the following experiment

A.3.2 Compatibility: Distance, Depreciation Static

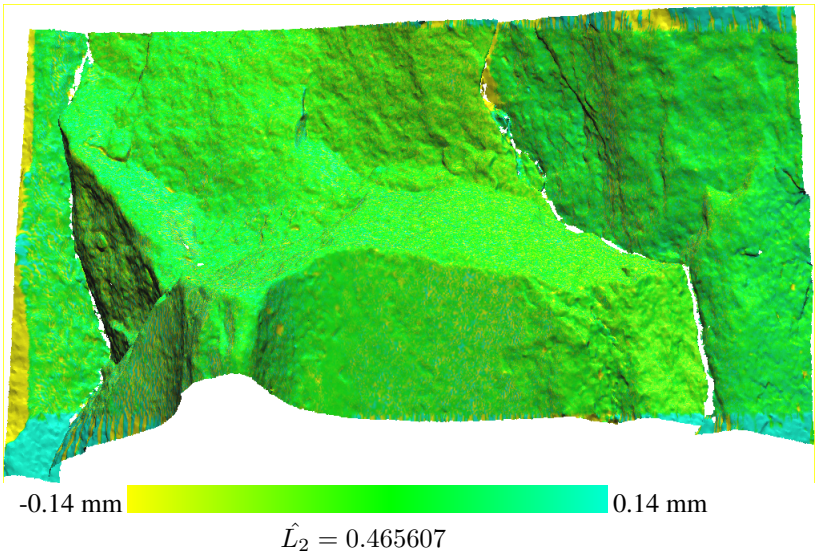
In this experiment, we ran multiple executions of DGM using the distance between the query point and the candidate as compatibility. The threshold was configured with no depreciation. We used different thresholds for each execution, ranging from 15mm to 0.1mm. The green color indicating little to no movement during the year tells us that the registration is quite accurate (see Figure A.11). The results can be seen in Figure A.12.



(a) A plot showing all $\hat{L_2}$ distances between S and T after execution with the static (non-changing) threshold. The X-axis represents the threshold for how close the nearest point has to be for acceptance. The Y-axis represents the mean distance to T per point



(b) A plot showing the same data as in a) but zoomed onto the cluster.



(c) The registered S dataset closest to T with colors indicating the pointwise distance to T .In the best result, only points within $2.4mm$ were considered. Note that the color only represent movement along the Z-axis, which is inwards/outwards from the image

Figure A.12: The results after registration using a static threshold when matching points

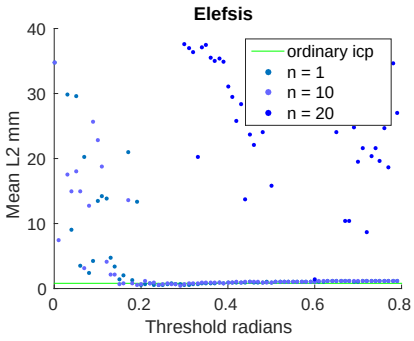
Raw Data

points	iterations	n	range y	l0_avg	l2_avg	l0_sum	l2_sum
4339632	200	1	0.2	0.996434	33.9252	4324160	147223000
4362207	200	1	0.4	0.991635	34.702	4325720	151377000
4303722	200	1	0.6	0.986319	29.9473	4244840	128885000
4342221	50	1	0.8	0.981617	30.4097	4262400	132045000
4705956	200	1	1	0.568732	1.95806	2676430	9214570
4653300	200	1	1.2	0.832665	11.9375	3874640	55548600
4680504	200	1	1.4	0.801811	8.80764	3752880	41224200
4682841	200	1	1.6	0.665189	4.28572	3114970	20069300
4705956	200	1	1.8	0.402856	2.03799	1895820	9590700
4705956	200	1	2	0.068412	0.869267	321944	4090730
4705956	200	1	2.2	0.0435034	0.474098	204725	2231090
4705956	200	1	2.4	0.0417165	0.465607	196316	2191130
4705956	30	1	2.6	0.0397284	0.465964	186960	2192810
4705956	61	1	2.8	0.0378616	0.466758	178175	2196540
4705956	199	1	3	0.0360877	0.467761	169827	2201260
4705956	38	1	3.2	0.0344349	0.469449	162049	2209210
4705956	167	1	3.4	0.0328139	0.471665	154421	2219630
4705956	161	1	3.6	0.0312844	0.474594	147223	2233420
4705956	155	1	3.8	0.0299654	0.477052	141016	2244990
4705956	31	1	4	0.0287066	0.479559	135092	2256780
4705956	91	1	4.2	0.0275362	0.481285	129584	2264900
4705956	41	1	4.4	0.0263651	0.485458	124073	2284540
4705956	144	1	4.6	0.0251796	0.488509	118494	2298900
4705956	127	1	4.8	0.0240693	0.492993	113269	2320000
4705956	114	1	5	0.0229962	0.495688	108219	2332690
4705956	39	1	5.2	0.02203	0.497715	103672	2342220
4705956	139	1	5.4	0.0210331	0.500746	98981	2356490
4705956	137	1	5.6	0.0199664	0.505533	93961	2379020
4705956	124	1	5.8	0.0189133	0.511017	89005	2404820
4705956	55	1	6	0.0179035	0.514815	84253	2422700
4705956	58	1	6.2	0.0169519	0.518265	79775	2438930
4705956	69	1	6.4	0.0161036	0.522504	75783	2458880
4705956	110	1	6.6	0.0153236	0.526407	72112	2477250
4705956	122	1	6.8	0.0146151	0.528512	68778	2487150
4705956	139	1	7	0.0139428	0.534579	65614	2515710
4705956	142	1	7.2	0.0133216	0.541343	62691	2547540
4705956	112	1	7.4	0.0128244	0.546606	60351	2572300
4705956	50	1	7.6	0.0123344	0.551033	58045	2593140
4705956	107	1	7.8	0.0118675	0.555721	55848	2615200
4705956	58	1	8	0.0113953	0.558001	53626	2625930
4705956	110	1	8.2	0.0109568	0.561927	51562	2644400
4705956	131	1	8.4	0.0104905	0.568138	49368	2673630

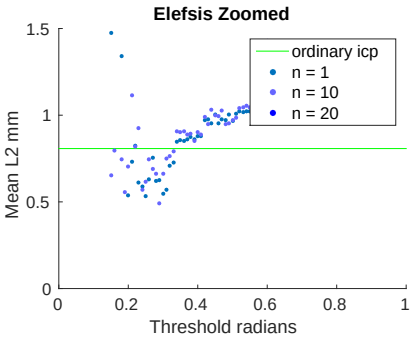
4705956	40	1	8.6	0.010043	0.573792	47262	2700240
4705956	106	1	8.8	0.00962865	0.576987	45312	2715280
4705956	114	1	9	0.00922108	0.581966	43394	2738710
4705956	116	1	9.2	0.0088388	0.589215	41595	2772820
4705956	40	1	9.4	0.00847564	0.593013	39886	2790690
4705956	52	1	9.6	0.0080838	0.598601	38042	2816990
4705956	122	1	9.8	0.00770874	0.603578	36277	2840410
4705956	109	1	10	0.00728927	0.612455	34303	2882190
4705956	118	1	10.2	0.0069225	0.62001	32577	2917740
4705956	30	1	10.4	0.00655403	0.628418	30843	2957310
4705956	105	1	10.6	0.00622828	0.635151	29310	2988990
4705956	121	1	10.8	0.00592356	0.640151	27876	3012520
4705956	110	1	11	0.00561905	0.644705	26443	3033950
4705956	101	1	11.2	0.00531964	0.652492	25034	3070600
4705956	43	1	11.4	0.00498496	0.664367	23459	3126480
4705956	200	1	11.6	0.00471743	0.669074	22200	3148630
4705956	125	1	11.8	0.00445074	0.674271	20945	3173090
4705956	93	1	12	0.00416111	0.683099	19582	3214630
4705956	94	1	12.2	0.00389379	0.687701	18324	3236290
4705956	94	1	12.4	0.00362668	0.695642	17067	3273660
4705956	92	1	12.6	0.00340058	0.700092	16003	3294600

A.3.3 Compatibility: Normal Vector, Depreciation: Static

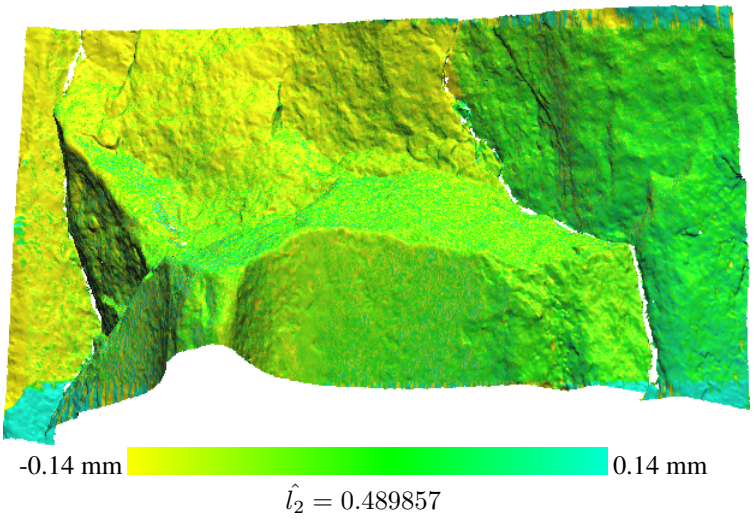
In this experiment, we ran multiple executions of DGM using the normal vector between the query point and the candidate as compatibility. No depreciation for the threshold was used. We used a different threshold for each execution ranging from 4 to 0 radians. We experimented with using the 1, 10 and 20 nearest points as candidates. Though the registration is better than using ordinary ICP, the normal vector compatibility did not outperform the distance compatibility. The results can be seen in Figure A.13.



(a) a plot showing all $\hat{l_2}$ distances between S and T after execution with the static (non-changing) threshold. the compatibility was computed from the angle between the normal vectors of corresponding points. the x-axis represents the threshold for how small the angle has to be to be accepted. the y-axis represents the mean distance to T per point



(b) a plot showing the same data as in figure A.3a but zoomed onto the cluster. the points seem to cluster between 0.5mm and 1.0mm.



(c) the registered S dataset closest to T with colors indicating pointwise distance to T. the best result was generated by accepting only points with an angle less than 0.2 radians. note that the color only represent movement along the z-axis, which is inwards/outwards from the image

Figure A.13: the results after registration using a static threshold when matching points

Raw Data

points	iterations	n	range y	l0_avg	l2_avg	l0_sum	l2_sum
4364064	1	1	0	1	34.7467	4364060	151637000
4262349	1	1	0.01	0.999999	49.3135	4262340	210191000
4262349	1	1	0.02	0.999987	49.3135	4262290	210191000
4264608	1	1	0.03	0.999983	29.8153	4264540	127150000
4680489	1	1	0.04	0.999863	9.06613	4679850	42433900
1452654	1	1	0.05	0.999904	29.643	1452520	43061000
4705956	1	1	0.06	0.99905	3.51508	4701480	16541800
3886182	6	1	0.07	0.998785	20.2407	3881460	78658900
4705956	200	1	0.08	0.990025	2.40319	4659020	11309300
4705956	1	1	0.09	0.992442	4.27781	4670390	20131200
4188870	1	1	0.1	0.997947	13.4856	4180270	56489500
4294314	1	1	0.11	0.997077	14.27	4281760	61280000
4312185	1	1	0.12	0.9966	13.9001	4297520	59939900
4692321	200	1	0.13	0.987198	4.78938	4632250	22473300
4705956	200	1	0.14	0.980787	3.35464	4615540	15786800
4705956	200	1	0.15	0.920846	1.47674	4333460	6949490
4705956	200	1	0.16	0.945635	2.01585	4450120	9486510
3762507	1	1	0.17	0.992004	21.0278	3732420	79117300
4705956	200	1	0.18	0.901201	1.34245	4241010	6317520
1175796	3	1	0.19	0.968165	13.3773	1138360	15728900
4705956	200	1	0.2	0.47291	0.539198	2225500	2537440
4705956	42	1	0.21	0.649332	0.731064	3055730	3440350
4705956	30	1	0.22	0.685135	0.82464	3224210	3880720
4705956	200	1	0.23	0.591188	0.612977	2782110	2884650
4705956	200	1	0.24	0.59515	0.590175	2800750	2777340
4705956	33	1	0.25	0.3264	0.534227	1536020	2514050
4705956	200	1	0.26	0.47769	0.630809	2247990	2968560
4705956	200	1	0.27	0.580526	0.756422	2731930	3559690
4705956	200	1	0.28	0.427988	0.62009	2014090	2918120
4705956	60	1	0.29	0.427705	0.626715	2012760	2949290
4705956	200	1	0.3	0.34373	0.54716	1617580	2574910
4705956	200	1	0.31	0.385035	0.569037	1811960	2677860
4705956	56	1	0.32	0.541731	0.707298	2549360	3328510
4705956	65	1	0.33	0.540658	0.725634	2544320	3414800
4705956	48	1	0.34	0.606454	0.845084	2853940	3976930
4705956	42	1	0.35	0.608011	0.857521	2861270	4035460
4705956	46	1	0.36	0.593044	0.850976	2790840	4004660
4705956	43	1	0.37	0.587494	0.858824	2764720	4041590
4705956	42	1	0.38	0.588883	0.875004	2771260	4117730
4705956	36	1	0.39	0.569922	0.85849	2682030	4040020
4705956	56	1	0.4	0.571125	0.881181	2687690	4146800
4705956	36	1	0.41	0.55903	0.87907	2630770	4136870

4705956	21	1	0.42	0.611612	0.970729	2878220	4568210
4705956	29	1	0.43	0.605556	0.975202	2849720	4589260
4705956	31	1	0.44	0.586015	0.951049	2757760	4475590
4705956	19	1	0.45	0.601843	1.00524	2832250	4730610
4705956	34	1	0.46	0.570572	0.951962	2685090	4479890
4705956	68	1	0.47	0.574662	0.973938	2704330	4583310
4705956	27	1	0.48	0.565479	0.970756	2661120	4568340
4705956	18	1	0.49	0.573531	1.00489	2699010	4728960
4705956	30	1	0.5	0.551682	0.969201	2596190	4561020
4705956	15	1	0.51	0.564999	1.01062	2658860	4755910
4705956	22	1	0.52	0.566412	1.02182	2665510	4808620
4705956	52	1	0.53	0.559258	1.01843	2631840	4792690
4705956	20	1	0.54	0.557115	1.02435	2621760	4820560
4705956	24	1	0.55	0.551843	1.02441	2596950	4820820
4705956	62	1	0.56	0.55875	1.05038	2629450	4943030
4705956	51	1	0.57	0.54679	1.03698	2573170	4879960
4705956	57	1	0.58	0.541336	1.03998	2547500	4894100
4705956	200	1	0.59	0.540339	1.04877	2542810	4935450
4705956	64	1	0.6	0.534037	1.04658	2513150	4925170
4705956	56	1	0.61	0.55184	1.09333	2596940	5145190
4705956	44	1	0.62	0.552716	1.10731	2601060	5210960
4705956	44	1	0.63	0.550946	1.11419	2592730	5243310
4705956	43	1	0.64	0.551092	1.12515	2593410	5294920
4705956	43	1	0.65	0.543481	1.12157	2557600	5278070
4705956	33	1	0.66	0.539715	1.12468	2539870	5292700
4705956	36	1	0.67	0.529683	1.1155	2492670	5249480
4705956	40	1	0.68	0.523366	1.11467	2462940	5245600
4705956	49	1	0.69	0.514985	1.10703	2423500	5209650
4705956	37	1	0.7	0.509867	1.10715	2399410	5210190
4705956	51	1	0.71	0.510101	1.11867	2400510	5264420
4705956	45	1	0.72	0.505434	1.11926	2378550	5267180
4705956	39	1	0.73	0.497752	1.11423	2342400	5243540
4705956	50	1	0.74	0.496097	1.1209	2334610	5274890
4705956	32	1	0.75	0.499828	1.1385	2352170	5357740
4705956	40	1	0.76	0.490285	1.12988	2307260	5317170
4705956	32	1	0.77	0.485827	1.12789	2286280	5307820
4705956	36	1	0.78	0.477406	1.11929	2246650	5267330
4705956	34	1	0.79	0.471401	1.11753	2218390	5259070
4364064	1	10	0	1	34.7467	4364060	151637000
4291326	1	10	0.01	0.999996	7.45929	4291310	32010200
3755226	4	10	0.02	0.999997	45.2698	3755220	169998000
1580469	1	10	0.03	0.999975	17.5843	1580430	27791500
3813435	2	10	0.04	0.999856	14.9313	3812890	56939600
1665198	1	10	0.05	0.999727	17.9906	1664740	29957900
1581075	2	10	0.06	0.998788	14.8959	1579160	23551500

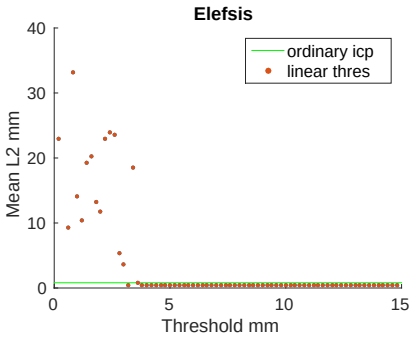
4705956	1	10	0.07	0.996427	3.14131	4689140	14782900
2592432	1	10	0.08	0.999398	12.7449	2590870	33040200
1659540	1	10	0.09	0.99918	25.6743	1658180	42607600
2455821	1	10	0.1	0.996642	22.7699	2447570	55918700
1817994	1	10	0.11	0.996677	18.8232	1811950	34220400
4703541	1	10	0.12	0.987765	4.13396	4645990	19444200
4705956	1	10	0.13	0.966728	2.15814	4549380	10156100
4705956	1	10	0.14	0.961867	2.13998	4526500	10070600
4705956	200	10	0.15	0.844531	0.653862	3974320	3077050
4705956	200	10	0.16	0.840901	0.798432	3957240	3757390
3139818	1	10	0.17	0.990449	13.6071	3109830	42723800
4705956	200	10	0.18	0.728322	0.743145	3427450	3497210
4705956	200	10	0.19	0.545937	0.55757	2569160	2623900
4705956	200	10	0.2	0.665	0.704884	3129460	3317150
4705956	200	10	0.21	0.801488	1.11286	3771770	5237070
4705956	200	10	0.22	0.680338	0.817989	3201640	3849420
4705956	200	10	0.23	0.713207	0.924037	3356320	4348480
4705956	200	10	0.24	0.441728	0.567903	2078750	2672530
4705956	200	10	0.25	0.490862	0.617584	2309980	2906320
4705956	200	10	0.26	0.597016	0.747278	2809530	3516660
4705956	200	10	0.27	0.529331	0.687929	2491010	3237370
4705956	200	10	0.28	0.489074	0.661775	2301560	3114280
4705956	200	10	0.29	0.162805	0.489857	766154	2305250
4705956	200	10	0.3	0.454499	0.66	2138850	3105930
4705956	200	10	0.31	0.578545	0.75092	2722610	3533800
4705956	200	10	0.32	0.575442	0.761742	2708000	3584730
4705956	200	10	0.33	0.58577	0.791611	2756610	3725290
4705956	200	10	0.34	0.637091	0.906041	2998120	4263790
4705956	200	10	0.35	0.625168	0.903009	2942010	4249520
4705956	200	10	0.36	0.619643	0.908825	2916010	4276890
4705956	200	10	0.37	0.599307	0.887914	2820310	4178480
4705956	200	10	0.38	0.594467	0.891039	2797540	4193190
4705956	200	10	0.39	0.566682	0.851967	2666780	4009320
4705956	200	10	0.4	0.583075	0.902443	2743920	4246860
4705956	200	10	0.41	0.570235	0.8897	2683500	4186890
4705956	200	10	0.42	0.618817	0.992112	2912130	4668830
4705956	200	10	0.43	0.593777	0.947844	2794290	4460510
4705956	200	10	0.44	0.626981	1.02947	2950540	4844650
4705956	200	10	0.45	0.603608	0.997505	2840550	4694210
4705956	200	10	0.46	0.594527	0.992528	2797820	4670790
4705956	200	10	0.47	0.605654	1.02884	2850180	4841660
4705956	200	10	0.48	0.561725	0.947208	2643450	4457520
4705956	200	10	0.49	0.55574	0.952296	2615290	4481460
4705956	200	10	0.5	0.561933	0.97152	2644430	4571930
4705956	200	10	0.51	0.563046	0.985833	2649670	4639290

4705956	200	10	0.52	0.582547	1.04107	2741440	4899210
4705956	200	10	0.53	0.577889	1.04428	2719520	4914330
4705956	200	10	0.54	0.577761	1.0551	2718920	4965260
4705956	200	10	0.55	0.567109	1.04574	2668790	4921210
4705956	200	10	0.56	0.569754	1.06535	2681240	5013470
4705956	200	10	0.57	0.562321	1.06011	2646260	4988820
4705956	200	10	0.58	0.559357	1.06596	2632310	5016370
4705956	200	10	0.59	0.549583	1.06148	2586310	4995270
4705956	200	10	0.6	0.545295	1.06413	2566140	5007760
4705956	200	10	0.61	0.570879	1.13486	2686530	5340620
4705956	200	10	0.62	0.570209	1.1436	2683380	5381730
4705956	200	10	0.63	0.567456	1.14952	2670420	5409580
4705956	200	10	0.64	0.565088	1.1585	2659280	5451840
4705956	200	10	0.65	0.558793	1.15408	2629660	5431030
4705956	200	10	0.66	0.553144	1.15323	2603070	5427050
4705956	200	10	0.67	0.545725	1.15054	2568160	5414400
4705956	200	10	0.68	0.539647	1.14912	2539560	5407700
4705956	200	10	0.69	0.534544	1.15002	2515540	5411940
4705956	200	10	0.7	0.528895	1.14855	2488960	5405020
4705956	200	10	0.71	0.526234	1.15341	2476440	5427900
4705956	200	10	0.72	0.519939	1.15161	2446810	5419450
4705956	200	10	0.73	0.517071	1.15546	2433320	5437560
4705956	200	10	0.74	0.512938	1.15652	2413870	5442520
4705956	200	10	0.75	0.513517	1.16884	2416590	5500520
4705956	200	10	0.76	0.50621	1.16316	2382200	5473790
4705956	200	10	0.77	0.502443	1.1617	2364480	5466920
4705956	200	10	0.78	0.49922	1.16502	2349310	5482520
4705956	200	10	0.79	0.488969	1.15237	2301060	5422980
2152854	1	20	0.3	0.99969	37.6429	2152190	81039600
2182377	1	20	0.31	0.999578	36.9231	2181460	80580100
2214159	1	20	0.32	0.999253	36.4218	2212500	80643700
2411181	1	20	0.33	0.994789	20.1953	2398620	48694600
2319192	1	20	0.34	0.99963	37.1528	2318330	86164400
2281503	1	20	0.35	0.999605	37.4184	2280600	85370100
2259711	1	20	0.36	0.999082	35.5225	2257640	80270700
2174229	1	20	0.37	0.998533	35.0301	2171040	76163500
2095848	1	20	0.38	0.999213	35.3718	2094200	74133800
2021478	1	20	0.39	0.999102	34.9185	2019660	70587100
4314645	1	20	0.4	0.987059	31.0509	4258810	133974000
3605028	1	20	0.41	0.987436	29.4614	3559740	106209000
3996624	1	20	0.42	0.983354	25.7948	3930100	103092000
3867978	1	20	0.43	0.983397	28.4121	3803760	109898000
2508615	1	20	0.44	0.968289	13.6728	2429060	34299800
3081333	1	20	0.45	0.979424	23.6492	3017930	72871000
2910090	1	20	0.46	0.957085	22.1395	2785200	64427900

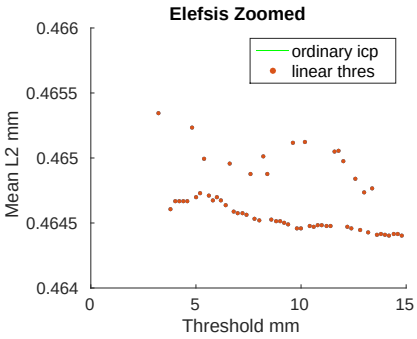
4023849	60	20	0.47	0.972353	28.3446	3912600	114054000
2911302	64	20	0.48	0.969347	24.0245	2822060	69942500
3083409	1	20	0.49	0.980116	26.16	3022100	80662000
3514323	200	20	0.5	0.969458	15.8586	3406990	55732100
4074963	2	20	0.51	0.997257	75.8666	4063780	309153000
3999291	200	20	0.52	0.959327	27.1172	3836630	108450000
4008732	200	20	0.53	0.955124	26.777	3828840	107342000
4023870	200	20	0.54	0.958323	27.1867	3856170	109396000
4007202	200	20	0.55	0.954727	26.7602	3825780	107234000
4001778	200	20	0.56	0.951838	26.6796	3809040	106766000
3995181	200	20	0.57	0.950526	27.0779	3797520	108181000
3982335	200	20	0.58	0.949056	26.2414	3779460	104502000
4281069	200	20	0.59	0.950169	30.9638	4067740	132558000
4705956	200	20	0.6	0.712057	1.42752	3350910	6717840
2570562	1	20	0.61	0.982461	34.3769	2525480	88367900
4189593	200	20	0.62	0.944559	29.0729	3957320	121804000
4192059	200	20	0.63	0.9431	29.1663	3953530	122267000
4209195	200	20	0.64	0.942371	29.2722	3966620	123212000
3258363	1	20	0.65	0.963731	24.0367	3140190	78320200
4196973	200	20	0.66	0.944138	27.8444	3962520	116862000
4131822	200	20	0.67	0.959197	10.414	3963230	43029000
4130292	200	20	0.68	0.957288	10.3882	3953880	42906300
3517965	1	20	0.69	0.983374	24.7628	3459480	87114500
3755607	1	20	0.7	0.982055	19.5021	3688210	73242200
4010814	1	20	0.71	0.981708	21.5998	3937450	86632800
4505913	200	20	0.72	0.949678	8.66185	4279170	39029600
3933405	1	20	0.73	0.984097	20.353	3870850	80056600
4141653	1	20	0.74	0.982296	21.6169	4068330	89529600
3877290	1	20	0.75	0.981835	19.6639	3806860	76242600
3173067	1	20	0.76	0.9884	24.6726	3136260	78287800
3704346	1	20	0.77	0.978321	18.6457	3624040	69070100
3550482	1	20	0.78	0.997763	34.6484	3542540	123019000
4295688	2	20	0.79	0.984131	27.0369	4227520	116142000

A.3.4 Compatibility: Distance, Depreciation: Linear

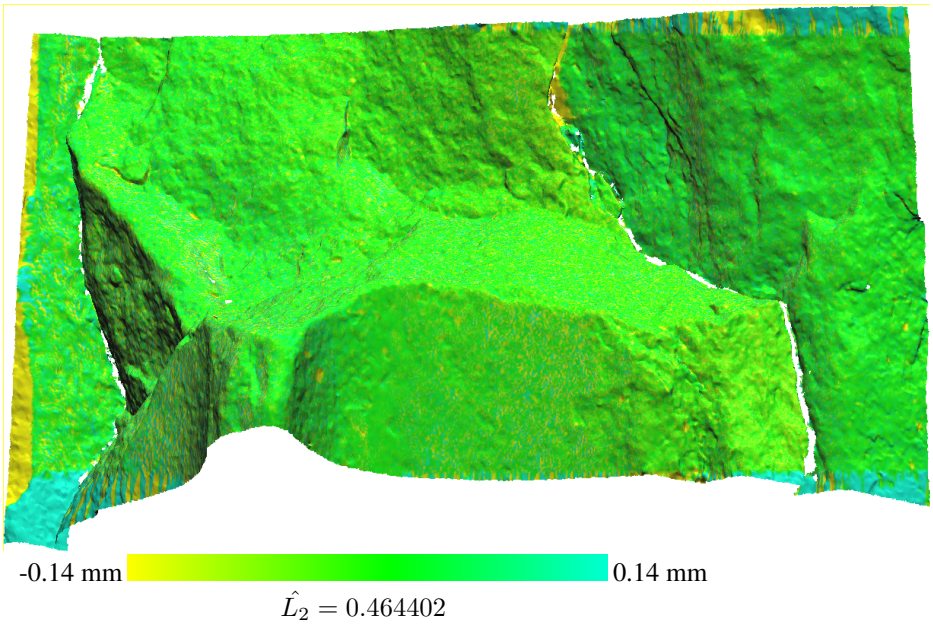
In this experiment, we ran multiple executions of DGM using the distance between the query point and the candidate as compatibility. The threshold was configured with linear depreciation. We used a different starting threshold for each execution ranging from 15.0mm to 0.0mm. The results can be seen in Figure A.14.



(a) A plot showing all $\hat{L_2}$ distances between S and T after execution with the linearly decreasing threshold. The X-axis represents the threshold for how close the nearest point has to be for acceptance. The Y-axis represents the mean distance to T per point



(b) A plot showing the same data as in a) but zoomed onto the cluster.



(c) The registered S dataset closest to T with colors indicating the pointwise distance to T . Setting the starting threshold to 14.8 provided the best result. Note that the color only represent movement along the Z-axis, which is inwards/outwards from the image

Figure A.14: The results after registration using a static threshold when matching points

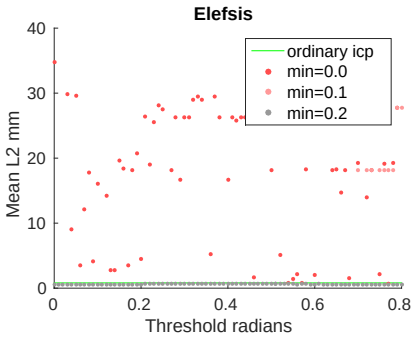
Raw Data

points	iterations	n	range y	l0_avg	l2_avg	l0_sum	l2_sum
3890835	195	1	0.2	0.995062	23.0124	3871620	89537400
4276326	200	1	0.4	0.996297	59.4475	4260490	254217000
3837612	200	1	0.6	0.9519	9.31349	3653020	35741600
3588078	199	1	0.8	0.977085	33.1468	3505860	118933000
3348810	199	1	1	0.92309	14.1463	3091250	47373200
2572899	200	1	1.2	0.894565	10.3551	2301630	26642700
3885621	200	1	1.4	0.95491	19.2463	3710420	74783900
3670224	200	1	1.6	0.934614	20.1939	3430240	74116000
4587558	200	1	1.8	0.837344	13.2417	3841360	60747000
4665141	200	1	2	0.781738	11.7977	3646920	55038000
4011696	200	1	2.2	0.917922	23.0127	3682430	92319900
3335790	200	1	2.4	0.911778	23.9121	3041500	79765600
2169438	200	1	2.6	0.948894	23.6301	2058570	51263900
4573920	199	1	2.8	0.606723	5.38804	2775100	24644500
4692075	199	1	3	0.483037	3.66642	2266450	17203100
4705956	1	1	3.2	0.0350762	0.465345	165067	2189890
3753441	200	1	3.4	0.868333	18.4622	3259240	69296800
4705956	200	1	3.6	0.0278124	0.74491	130884	3505510
4705956	200	1	3.8	0.0307243	0.464608	144587	2186420
4705956	200	1	4	0.0295336	0.464668	138984	2186710
4705956	200	1	4.2	0.0284293	0.46467	133787	2186720
4705956	200	1	4.4	0.027281	0.46467	128383	2186720
4705956	200	1	4.6	0.0261834	0.464667	123218	2186700
4705956	1	1	4.8	0.0250755	0.465235	118004	2189380
4705956	200	1	5	0.0240706	0.464696	113275	2186840
4705956	200	1	5.2	0.0230202	0.464727	108332	2186980
4705956	1	1	5.4	0.0221147	0.464994	104071	2188240
4705956	200	1	5.6	0.0211419	0.464709	99493	2186900
4705956	200	1	5.8	0.0202477	0.464676	95285	2186740
4705956	200	1	6	0.019348	0.464697	91051	2186840
4705956	200	1	6.2	0.0185208	0.464673	87158	2186730
4705956	200	1	6.4	0.017708	0.464635	83333	2186550
4705956	1	1	6.6	0.0169538	0.464957	79784	2188070
4705956	200	1	6.8	0.0162139	0.464588	76302	2186330
4705956	200	1	7	0.0155771	0.464574	73305	2186260
4705956	200	1	7.2	0.0149602	0.464578	70402	2186280
4705956	200	1	7.4	0.01442	0.464561	67860	2186210
4705956	1	1	7.6	0.0138686	0.464875	65265	2187680
4705956	200	1	7.8	0.0133818	0.464535	62974	2186080
4705956	200	1	8	0.0129583	0.464522	60981	2186020
4705956	1	1	8.2	0.0126013	0.46501	59301	2188320
4705956	1	1	8.4	0.0122311	0.46488	57559	2187700

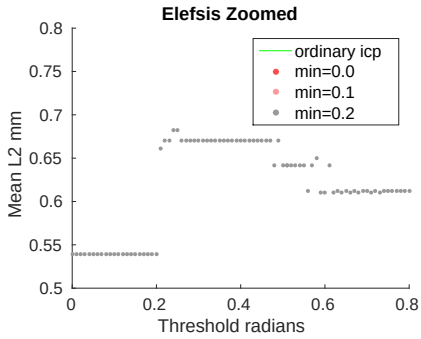
4705956	200	1	8.6	0.0118031	0.464525	55545	2186030
4705956	200	1	8.8	0.0114298	0.464512	53788	2185970
4705956	200	1	9	0.0110632	0.464512	52063	2185970
4705956	200	1	9.2	0.0107147	0.4645	50423	2185920
4705956	200	1	9.4	0.0103573	0.464491	48741	2185880
4705956	1	1	9.6	0.0100432	0.465116	47263	2188820
4705956	200	1	9.8	0.00967285	0.464461	45520	2185730
4705956	200	1	10	0.00934093	0.464461	43958	2185730
4705956	1	1	10.2	0.00906086	0.465124	42640	2188850
4705956	200	1	10.4	0.00876464	0.464477	41246	2185810
4705956	200	1	10.6	0.00847692	0.464472	39892	2185780
4705956	200	1	10.8	0.00819345	0.464486	38558	2185850
4705956	200	1	11	0.00790488	0.464486	37200	2185850
4705956	200	1	11.2	0.00761461	0.464478	35834	2185810
4705956	200	1	11.4	0.00730181	0.464478	34362	2185810
4705956	1	1	11.6	0.0070415	0.465047	33137	2188490
4705956	1	1	11.8	0.00676462	0.465053	31834	2188520
4705956	1	1	12	0.00652152	0.464978	30690	2188160
4705956	200	1	12.2	0.00623168	0.46447	29326	2185780
4705956	200	1	12.4	0.00597519	0.464459	28119	2185720
4705956	1	1	12.6	0.00573507	0.464841	26989	2187520
4705956	200	1	12.8	0.00547668	0.464447	25773	2185670
4705956	1	1	13	0.00522806	0.464738	24603	2187040
4705956	200	1	13.2	0.00500982	0.46443	23576	2185590
4705956	1	1	13.4	0.00479201	0.464767	22551	2187170
4705956	200	1	13.6	0.00457973	0.464411	21552	2185500
4705956	200	1	13.8	0.00436341	0.464414	20534	2185510
4705956	200	1	14	0.00412839	0.464409	19428	2185490
4705956	200	1	14.2	0.00391206	0.464405	18410	2185470
4705956	200	1	14.4	0.00370169	0.464413	17420	2185510
4705956	200	1	14.6	0.00350662	0.464413	16502	2185510
4705956	200	1	14.8	0.00332196	0.464402	15633	2185450

A.3.5 Compatibility: Normal Vector, Depreciation: Linear

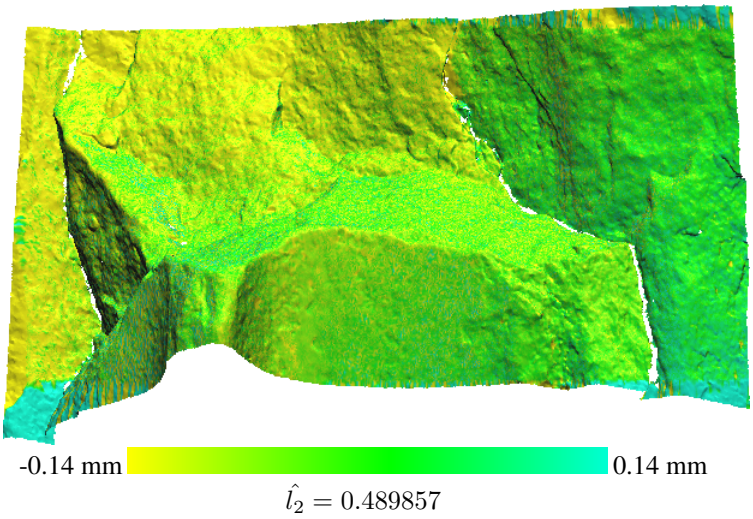
In this experiment, we ran multiple executions of DGM using the angle between the normal vector of the query point and the candidate as compatibility. The threshold was configured with linear depreciation. We used a different starting threshold for each execution ranging from 4 to 0 radians. One candidate point were selected per query point. We experimented using a lower threshold for the depreciation set to 0.0 radians, 0.1 radians, and 0.2 radians. Configuring no depreciation for the threshold outperformed configuring a threshold when using the normal vector compatibility. The results can be seen in Figure A.15.



(a) The plot shows all $\hat{l_2}$ distances between S and T after execution with the iterative linear threshold. The compatibility was computed from the angle between the normal vectors of corresponding points. The x-axis represents the threshold for how small the angle has to be to be accepted. The y-axis represents the mean distance to T per point



(b) a plot showing the same data as in figure A.3a but zoomed onto the cluster. the points seem to cluster between 0.7mm and 0.5mm.



(c) the registered S dataset closest to T with colors indicating pointwise distance to T . the best result was generated by accepting only points with an angle less than 0.54 radians. note that the color only represent movement along the z-axis, which is inwards/outwards from the image

Figure A.15: the results after registration using a static threshold when matching points

Raw Data

points	iterations	n	range y	l0_avg	l2_avg	l0_sum	l2_sum
4364064	1	1	0	1	34.7467	4364060	151637000
4262349	1	1	0.01	0.999999	49.3135	4262340	210191000
4262349	1	1	0.02	0.999987	49.3135	4262290	210191000
4264608	1	1	0.03	0.999983	29.8153	4264540	127150000
4680489	1	1	0.04	0.999863	9.06613	4679850	42433900
1452654	1	1	0.05	0.999904	29.643	1452520	43061000
4705956	1	1	0.06	0.99905	3.51508	4701480	16541800
4203309	1	1	0.07	0.999136	12.0959	4199680	50842800
1099983	1	1	0.08	0.999085	17.7649	1098980	19541100
4705956	1	1	0.09	0.993454	4.13204	4675150	19445200
4355793	1	1	0.1	0.997939	16.0041	4346820	69710600
4414173	1	1	0.11	0.999395	55.5556	4411500	245232000
4295685	1	1	0.12	0.996284	14.1984	4279720	60991700
4705956	1	1	0.13	0.974909	2.76575	4587880	13015500
4705956	1	1	0.14	0.971491	2.74373	4571800	12911900
3622107	2	1	0.15	0.996202	19.65	3608350	71174300
4042236	75	1	0.16	0.993421	18.439	4015640	74534900
4705956	54	1	0.17	0.961891	3.45145	4526620	16242400
4064196	70	1	0.18	0.992006	18.1369	4031710	73712100
3771036	1	1	0.19	0.990983	20.7674	3737030	78314800
4690746	1	1	0.2	0.966359	4.48503	4532940	21038100
3291564	1	1	0.21	0.993074	26.3479	3268770	86725800
3624966	1	1	0.22	0.994229	18.9936	3604050	68851000
1215396	1	1	0.23	0.994134	25.5865	1208270	31097700
3872163	125	1	0.24	0.98642	28.0989	3819580	108804000
3433689	1	1	0.25	0.992036	27.4892	3406340	94389300
3102798	1	1	0.26	0.992545	42.911	3079670	133144000
4059561	73	1	0.27	0.985116	18.211	3999140	73928700
3282603	1	1	0.28	0.989066	26.3115	3246710	86370100
2198811	3	1	0.29	0.990858	16.6302	2178710	36566600
3282609	1	1	0.3	0.988015	26.3114	3243270	86370100
3282609	1	1	0.31	0.987481	26.3114	3241510	86370100
3461718	1	1	0.32	0.988577	28.9688	3422180	100282000
3475158	1	1	0.33	0.98831	29.4985	3434530	102512000
3461718	1	1	0.34	0.987622	28.9689	3418870	100282000
2354508	2	1	0.35	0.97167	53.5655	2287800	126120000
4680618	1	1	0.36	0.946217	5.18284	4428880	24258900
3475158	1	1	0.37	0.986475	29.4987	3428160	102513000
3282603	1	1	0.38	0.984045	26.3115	3230230	86370100
2354508	2	1	0.39	0.968444	53.5655	2280210	126120000
2193624	3	1	0.4	0.98687	16.659	2164820	36543500
3282603	1	1	0.41	0.982546	26.3115	3225310	86370100

3450276	3	1	0.42	0.991016	25.7247	3419280	88757400
3282603	1	1	0.43	0.981548	26.3114	3222030	86370000
3282603	1	1	0.44	0.981046	26.3115	3220390	86370100
3282609	1	1	0.45	0.980517	26.3114	3218650	86370000
4705956	30	1	0.46	0.807489	1.67867	3800010	7899730
3282603	1	1	0.47	0.979569	26.3114	3215540	86369900
3282603	1	1	0.48	0.979057	26.3114	3213860	86369900
3282603	1	1	0.49	0.978577	26.3114	3212280	86369900
4064838	42	1	0.5	0.969332	18.0924	3940180	73542700
3439020	1	1	0.51	0.981151	27.7931	3374200	95581200
4681146	2	1	0.52	0.924378	5.13669	4327150	24045600
4119285	25	1	0.53	0.980374	29.3005	4038440	120697000
4705956	39	1	0.54	0.400472	0.764536	1884600	3597870
4705956	26	1	0.55	0.755594	1.42536	3555790	6707690
4705956	25	1	0.56	0.840159	2.18904	3953750	10301500
4705956	37	1	0.57	0.382341	0.764533	1799280	3597860
4057365	34	1	0.58	0.96219	18.2981	3903960	74242100
3439020	1	1	0.59	0.977877	27.7931	3362940	95581200
4705956	23	1	0.6	0.759918	1.98336	3576140	9333610
3439020	1	1	0.61	0.977089	27.7931	3360230	95581200
3439020	1	1	0.62	0.976708	27.7931	3358920	95581200
3439020	1	1	0.63	0.976303	27.7931	3357530	95581100
4064838	32	1	0.64	0.959061	18.0924	3898430	73542700
4057365	30	1	0.65	0.956806	18.2981	3882110	74242100
2548830	23	1	0.66	0.981892	14.7421	2502680	37575000
4059561	29	1	0.67	0.956495	18.211	3882950	73928700
4705956	22	1	0.68	0.736537	1.57276	3466110	7401340
3439098	1	1	0.69	0.97384	27.7682	3349130	95497500
4007817	24	1	0.7	0.953136	19.2764	3820000	77256100
3439008	1	1	0.71	0.973148	27.793	3346660	95580300
3546549	23	1	0.72	0.968672	14.0214	3435440	49727700
4064202	27	1	0.73	0.952011	18.137	3869160	73712300
3439098	1	1	0.74	0.971843	27.7682	3342260	95497500
4705956	23	1	0.75	0.712689	2.10493	3353880	9905710
4008108	24	1	0.76	0.947845	19.1945	3799070	76933700
4705956	26	1	0.77	0.234388	0.703337	1103020	3309870
4009485	19	1	0.78	0.947712	19.2223	3799840	77071500
3439020	1	1	0.79	0.969931	27.7932	3335610	95581200

min range	iterations	n	range y	l0_avg	l2_avg	l0_sum	l2_sum
0.2	200	1	0	1	0.539198	4705960	2537440
0.2	200	1	0.01	0.99982	0.539198	4705110	2537440
0.2	200	1	0.02	0.998498	0.539198	4698890	2537440
0.2	200	1	0.03	0.995014	0.539198	4682490	2537440
0.2	200	1	0.04	0.988209	0.539198	4650470	2537440

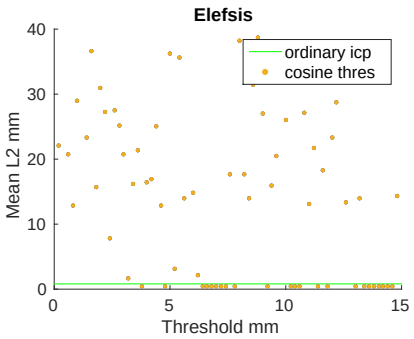
0.2	200	1	0.05	0.977091	0.539198	4598150	2537440
0.2	200	1	0.06	0.960801	0.539198	4521490	2537440
0.2	200	1	0.07	0.938616	0.539198	4417080	2537440
0.2	200	1	0.08	0.910781	0.539198	4286100	2537440
0.2	200	1	0.09	0.877657	0.539198	4130220	2537440
0.2	200	1	0.1	0.841053	0.539198	3957960	2537440
0.2	200	1	0.11	0.801889	0.539198	3773650	2537440
0.2	200	1	0.12	0.761517	0.539198	3583670	2537440
0.2	200	1	0.13	0.721546	0.539198	3395560	2537440
0.2	200	1	0.14	0.681651	0.539198	3207820	2537440
0.2	200	1	0.15	0.642749	0.539198	3024750	2537440
0.2	200	1	0.16	0.605261	0.539198	2848330	2537440
0.2	200	1	0.17	0.569242	0.539198	2678830	2537440
0.2	200	1	0.18	0.535249	0.539198	2518860	2537440
0.2	200	1	0.19	0.503074	0.539198	2367440	2537440
0.2	200	1	0.2	0.47291	0.539198	2225500	2537440
0.2	200	1	0.21	0.610001	0.660664	2870640	3109060
0.2	200	1	0.22	0.594537	0.670751	2797860	3156530
0.2	200	1	0.23	0.577997	0.670751	2720030	3156520
0.2	200	1	0.24	0.576549	0.682421	2713210	3211440
0.2	200	1	0.25	0.561443	0.682421	2642120	3211440
0.2	200	1	0.26	0.532331	0.670751	2505120	3156530
0.2	200	1	0.27	0.518504	0.670751	2440060	3156530
0.2	200	1	0.28	0.504684	0.670751	2375020	3156530
0.2	200	1	0.29	0.491348	0.670753	2312260	3156530
0.2	200	1	0.3	0.478333	0.670753	2251010	3156530
0.2	200	1	0.31	0.465709	0.670751	2191610	3156520
0.2	200	1	0.32	0.453545	0.670751	2134360	3156530
0.2	200	1	0.33	0.441439	0.670752	2077390	3156530
0.2	200	1	0.34	0.429746	0.670753	2022370	3156530
0.2	200	1	0.35	0.418432	0.670752	1969120	3156530
0.2	200	1	0.36	0.407238	0.670751	1916440	3156530
0.2	200	1	0.37	0.396531	0.670751	1866060	3156530
0.2	200	1	0.38	0.386105	0.670753	1816990	3156530
0.2	200	1	0.39	0.375997	0.670751	1769420	3156530
0.2	200	1	0.4	0.366207	0.670753	1723350	3156530
0.2	200	1	0.41	0.356745	0.670753	1678830	3156530
0.2	200	1	0.42	0.347532	0.670751	1635470	3156530
0.2	200	1	0.43	0.338625	0.670753	1593560	3156530
0.2	200	1	0.44	0.329794	0.670753	1552000	3156530
0.2	200	1	0.45	0.321207	0.670752	1511590	3156530
0.2	200	1	0.46	0.312891	0.670751	1472450	3156520
0.2	200	1	0.47	0.304788	0.670753	1434320	3156530
0.2	200	1	0.48	0.257145	0.641974	1210110	3021100
0.2	200	1	0.49	0.288844	0.670753	1359290	3156530

0.2	200	1	0.5	0.241308	0.641976	1135580	3021110
0.2	200	1	0.51	0.23382	0.641974	1100340	3021100
0.2	200	1	0.51	0.23382	0.641974	1100340	3021100
0.2	200	1	0.52	0.22649	0.641975	1065850	3021110
0.2	200	1	0.53	0.219183	0.641974	1031470	3021100
0.2	200	1	0.54	0.212119	0.641973	998223	3021100
0.2	200	1	0.55	0.205187	0.641974	965603	3021100
0.2	200	1	0.56	0.159155	0.61173	748975	2878770
0.2	200	1	0.57	0.192276	0.641976	904841	3021110
0.2	200	1	0.58	0.196872	0.650094	926471	3059310
0.2	200	1	0.59	0.141294	0.610618	664922	2873540
0.2	200	1	0.6	0.136426	0.610619	642013	2873540
0.2	200	1	0.61	0.169781	0.641974	798980	3021100
0.2	200	1	0.62	0.127112	0.610619	598184	2873540
0.2	200	1	0.63	0.124198	0.61173	584472	2878770
0.2	200	1	0.64	0.118927	0.610617	559667	2873540
0.2	200	1	0.65	0.116095	0.61173	546340	2878770
0.2	200	1	0.66	0.111494	0.610618	524684	2873540
0.2	200	1	0.67	0.108973	0.61173	512823	2878770
0.2	200	1	0.68	0.104943	0.610619	493858	2873540
0.2	200	1	0.69	0.102795	0.61173	483750	2878770
0.2	200	1	0.7	0.0999805	0.61173	470504	2878770
0.2	200	1	0.71	0.0967661	0.610617	455377	2873540
0.2	200	1	0.72	0.0948538	0.61173	446378	2878770
0.2	200	1	0.73	0.0920125	0.610618	433007	2873540
0.2	200	1	0.74	0.0902964	0.61173	424931	2878770
0.2	200	1	0.75	0.0882562	0.61173	415330	2878770
0.2	200	1	0.76	0.0863368	0.61173	406297	2878770
0.2	200	1	0.77	0.0844772	0.61173	397546	2878770
0.2	200	1	0.78	0.0827364	0.61173	389354	2878770
0.2	200	1	0.79	0.0811334	0.61173	381810	2878770
0.2	200	1	0.8	0.0795537	0.61173	374376	2878770

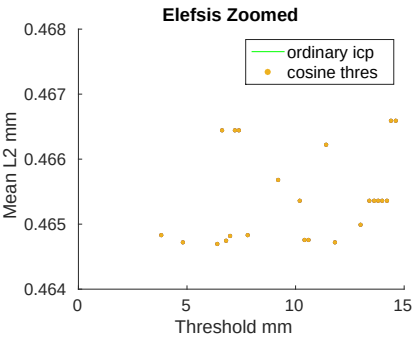
min range	iterations	n	range y	l0_avg	l2_avg	l0_sum	l2_sum
0.1	176	1	0.7	0.954049	18.137	3877450	73712300
0.1	1	1	0.71	0.973148	27.793	3346660	95580300
0.1	180	1	0.72	0.953358	18.0924	3875240	73542700
0.1	176	1	0.73	0.952011	18.137	3869160	73712300
0.1	1	1	0.74	0.971843	27.7682	3342260	95497500
0.1	178	1	0.75	0.950608	18.137	3863460	73712300
0.1	179	1	0.76	0.950604	18.0924	3864050	73542700
0.1	176	1	0.77	0.949388	18.2112	3854090	73929200
0.1	179	1	0.78	0.948547	18.137	3855090	73712300
0.1	1	1	0.79	0.969931	27.7932	3335610	95581200
0.1	1	1	0.8	0.969572	27.7932	3334380	95581200

A.3.6 Compatibility: Distance, Depreciation: Cosine

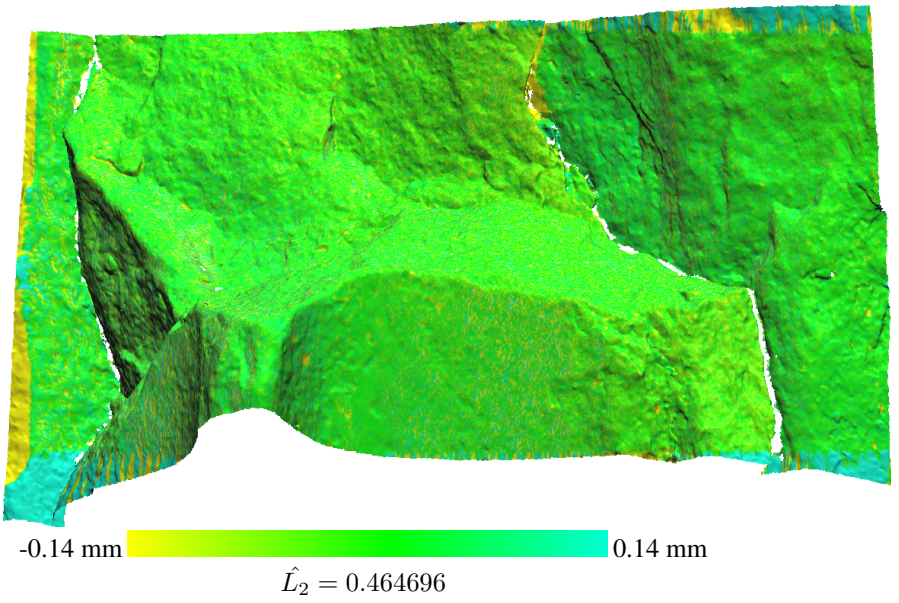
In this experiment, we ran multiple executions of DGM using the distance between the query point and the candidate as compatibility. The threshold was configured with cosine depreciation. We used a different starting threshold for each execution ranging from 15.0mm to 0.0mm. A vast majority of the executions produced outliers; this problem vanished when we restricted further computation if less than ten matches were made. The results can be seen in Figure A.16.



(a) A plot showing all $\hat{L_2}$ distances between S and T after execution with the cosine depreciation. The X-axis represents the starting threshold for how close the nearest point has to be for acceptance. The Y-axis represents the mean distance to T per point



(b) A plot showing the same data as in a) but zoomed onto the cluster.



(c) The registered S dataset closest to T with colors indicating the pointwise distance to T . Setting the starting threshold to 6.4 provided the best result. Note that the color only represent movement along the Z-axis, which is inwards/outwards from the image

Figure A.16: The results after registration using a static threshold when matching points

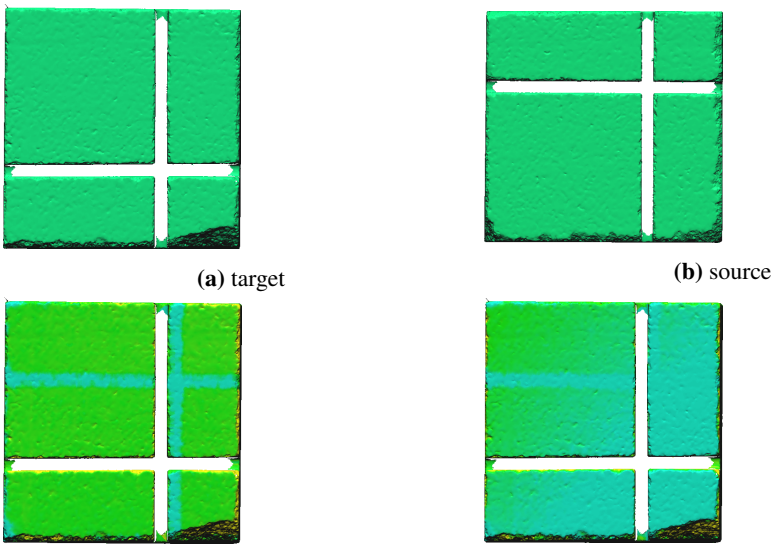
Raw Data

points	iterations	n	range y	l0_avg	l2_avg	l0_sum	l2_sum
3937740	194	1	0.2	0.997282	22.0611	3927040	86871000
4171515	198	1	0.4	0.996312	53.6341	4156130	223735000
1203735	185	1	0.6	0.980791	20.6941	1180610	24910200
2688450	193	1	0.8	0.969644	12.8542	2606840	34558000
3537309	194	1	1	0.97908	28.9252	3463310	102317000
4411803	194	1	1.2	0.977639	51.7149	4313150	228156000
3367884	1	1	1.4	0.95311	23.3268	3209960	78562100
4349016	196	1	1.6	0.972562	36.5862	4229690	159114000
3442746	1	1	1.8	0.900103	15.6615	3098830	53918700
4019295	195	1	2	0.961741	30.9264	3865520	124302000
3936366	200	1	2.2	0.939719	27.2727	3699080	107355000
4705956	195	1	2.4	0.70371	7.8263	3311630	36830200
3656826	197	1	2.6	0.93414	27.5003	3415990	100564000
3121806	199	1	2.8	0.925419	25.1622	2888980	78551600
3546870	195	1	3	0.824186	20.7917	2923280	73745300
4705956	193	1	3.2	0.153281	1.61348	721333	7592970
4179399	1	1	3.4	0.872537	16.2225	3646680	67800200
3492999	200	1	3.6	0.882717	21.3332	3083330	74516700
4705956	197	1	3.8	0.0307449	0.464836	144684	2187500
4339902	198	1	4	0.786013	16.4062	3411220	71201500
3368580	199	1	4.2	0.726445	16.8876	2447090	56887100
3674574	199	1	4.4	0.861965	25.0127	3167350	91910900
2080494	199	1	4.6	0.745936	12.8852	1551920	26807600
4705956	198	1	4.8	0.0250563	0.464726	117914	2186980
2887149	198	1	5	0.820774	36.2077	2369700	104537000
4704573	198	1	5.2	0.14017	3.08797	659438	14527600
3788694	1	1	5.4	0.872655	35.6199	3306220	134953000
3406185	1	1	5.6	0.710223	13.9874	2419150	47643600
2890095	199	1	5.8	0.851113	45.1831	2459800	130583000
2728920	199	1	6	0.647748	14.8109	1767650	40417700
4705956	199	1	6.2	0.0377666	2.15661	177728	10148900
4705956	199	1	6.4	0.0177158	0.464696	83370	2186840
4705956	1	1	6.6	0.0169685	0.466439	79853	2195040
4705956	199	1	6.8	0.0162254	0.464739	76356	2187040
4705956	198	1	7	0.0155926	0.464822	73378	2187430
4705956	1	1	7.2	0.0149827	0.46644	70508	2195050
4705956	1	1	7.4	0.0144368	0.46644	67939	2195050
4030980	199	1	7.6	0.654014	17.6307	2636320	71069100
4705956	199	1	7.8	0.0133966	0.464836	63044	2187500
3709497	199	1	8	0.686352	38.1888	2546020	141661000
4030983	199	1	8.2	0.633031	17.6307	2551740	71069100
4514541	200	1	8.4	0.504377	14.0296	2277030	63337200

2330973	199	1	8.6	0.651315	31.4384	1518200	73282100
3656802	198	1	8.8	0.769046	38.6492	2812250	141333000
3929247	200	1	9	0.653707	27.0279	2568580	106199000
4705956	199	1	9.2	0.0107343	0.465678	50515	2191460
4348755	199	1	9.4	0.566805	15.9769	2464900	69479400
3611040	199	1	9.6	0.642839	20.553	2321320	74217900
3218550	199	1	9.8	0.764785	37.7744	2461500	121579000
4193751	199	1	10	0.691	26.0604	2897880	109291000
4705956	200	1	10.2	0.00905023	0.465363	42590	2189980
4705956	198	1	10.4	0.0087776	0.464755	41307	2187120
4705956	198	1	10.6	0.00849094	0.464758	39958	2187130
3867099	199	1	10.8	0.692542	27.0879	2678130	104752000
2150397	199	1	11	0.4069	13.1361	874997	28247700
3351099	199	1	11.2	0.579194	21.7619	1940940	72926400
4705956	1	1	11.4	0.00733284	0.466218	34508	2194000
2604987	199	1	11.6	0.519233	18.3157	1352600	47712000
4705956	198	1	11.8	0.00675994	0.464716	31812	2186930
3216273	199	1	12	0.697063	23.3747	2241940	75179400
2774415	200	1	12.2	0.698486	28.6954	1937890	79613100
3993036	200	1	12.4	0.761364	35.6248	3040160	142251000
4211514	199	1	12.6	0.378507	13.3934	1594090	56406400
3885021	199	1	12.8	0.711083	32.0851	2762570	124651000
4705956	200	1	13	0.00524378	0.464989	24677	2188220
4323915	200	1	13.2	0.409922	13.9856	1772470	60472700
4705956	200	1	13.4	0.00479987	0.465361	22588	2189970
4705956	200	1	13.6	0.00459992	0.465361	21647	2189970
4705956	200	1	13.8	0.00437977	0.465361	20611	2189970
4705956	200	1	14	0.00414624	0.46536	19512	2189960
4705956	200	1	14.2	0.00393799	0.465362	18532	2189970
4705956	200	1	14.4	0.00371657	0.466589	17490	2195750
4705956	200	1	14.6	0.00352064	0.466592	16568	2195760
4352247	200	1	14.8	0.313507	14.29	1364460	62193800

A.4 Estone 1

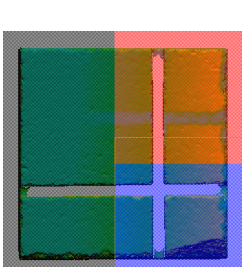
The dataset used for this experiment consisted of two marble stones that have artificially been eroded in an erosion chamber between the scans. The source dataset is oriented in a local minima 90 degrees from target, which is problematic for registration. We experimented with using different configurations of the predefined area compatibility for registration but were unable to escape the local minima. The setup can be seen in Figure A.17, the results can be seen in Figure A.18.

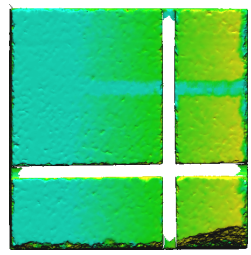


(a) target **(b)** source

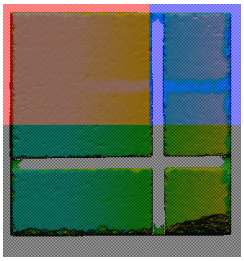
(c) mean L_2: 0.254395 **(d)** Iterations: 44, mean L_2: 0.262986

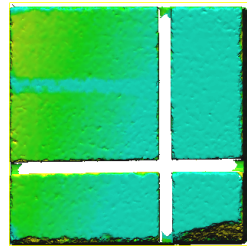
Figure A.17: The setup for this experiment.



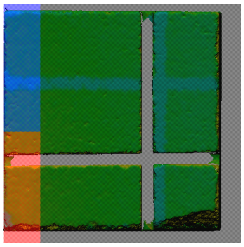


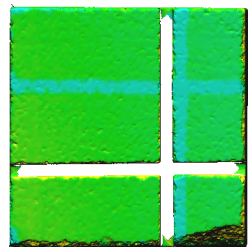
(a) Iterations: 35, mean L_2: 0.291887





(b) Iterations: 75, mean L_2: 0.253448





(c) Iterations: 44, mean L_2: 0.226075

Figure A.18: The results after trying to escape a bad local minima using predefined areas.

Appendix B

Manual

The configuration of DGM has to be done from *main()*, in main.cpp. The software then has to be compiled before the program runs with set configuration. This document will describe how to configure the program.

B.1 Getting started

DGM is configured through a configuration object. The object can be created through calling the static function *create_standard_config()*. As the name implies, the standard configuration will be created. It will look for *source.stl* and *target.stl* in data/example. Then, the procedure for registration and the procedure for displaying distances has to be set. Currently, only one of each exists, *ViliusRegistration* and *ViliusDistance*. The registration procedure is set calling *set_registration_procedure()*, and the distance procedure is set calling *set_distance_procedure*. Then we will decide how the two meshes are going to be aligned. Because the program aligns the meshes twice (, once before dividing the meshes, and once after), one may use different alignment procedures for each stage. For now we will use the same procedure at both stages. *Libicp_adapter* is a class that can be used for the alignment procedure. It uses ordinary ICP for aligning *target* and *source*. To start the registration, call the *run_registration* procedure in the Configuration object. After this procedure, the meshes will be aligned and divided into chunks on disk. Then, to create a model of *source* with the distance between *source* and *target* displayed as color, call *calculate_distance*.

```
int main(int, char**) {
  Configurations * conf;
  conf = Configurations::create_standard_config();
  conf->set_registration_procedure(new ViliusRegistration);
  conf->set_distance_procedure(new ViliusDistance);

  /* Implementation of Iterative Closest Point */
```

```
  ICP_impl::Libicp_adapter * libicp = new ICP_impl::
      Libicp_adapter(NULL);

  /* Alignment before the dividing stage */
  conf->icp_pre = libicp;
  /* Alignment after the dividing stage */
  conf->icp_post = libicp;

  conf->run_registration();
  conf->calculate_distance_octrees();
  delete conf;
}
```

B.2 Configuring Alignment Procedure

B.2.1 Configuring ICP

The ICP implementation we used in B.1 has several parameters we may decide.

1. sub step: Amount of points to ignore during alignment
2. max iterations: Maximum allowed iterations ICP can do before termination
3. min delta: A procedure is used for evaluating the quality of the alignment. The stopping criteria of ICP is when this evaluation is below min delta.
4. inlier distance: When the distance between the corresponding points is bigger than the inlier distance, one of the points may be an outlier and the points are no longer corresponding. When the inlier distance is negative, no points are considered outliers

A procedure we refer to as *delta computation* is the procedure that evaluates the alignment between *source* and *target*. This procedure, together with the *min delta* parameter represent our stopping criteria. The standard *delta computation* returns the L_2 norm of which ever of the newly computed transformation matrices that are the biggest. We may change this to instead return the L_0 distance between *target* and *source* with the transformation applied. We do this by creating a *Surface_DeltaWrapper* and setting the *norm* field in it to a *L0_Norm*. To create a custom delta computation, we create a class and implement the *DeltaWrapper* interface. The input parameters represent the next alignment, and not the whole transformation. If your delta computation depends on the whole transformation, the class has to keep track of previous transformations.

```
class Surface_DeltaWrapper : public DeltaWrapper {
  Matrix R_;
  Matrix t_;
  public:
  Surface_DeltaWrapper(){
    R_ = Matrix::eye(3);
```

```
    t_ = Matrix(3, 1);
  }
  double delta(Matrix &R, Matrix &t) {
    R_ = R_ * R;
    t_ = t_ + t;

    Matrix source(M_num, 3, M);
    source =  R_ * ~source;
    source = ~source;
    source = ICP_impl::translate(source, t_);
    Matrix target(T_num, 3, T);

    Matrix delta = source - target;
    delta.norm_wrapper = norm;

    double norm = delta.custom_norm();
    std::cout << "norm: " << norm << std::endl;
    return norm;
  }
};
int main(int, char**) {
  Configurations * conf;
  conf = Configurations::create_standard_config();
  conf->set_registration_procedure(new ViliusRegistration);
  conf->set_distance_procedure(new ViliusDistance);

  /* Iterative Closest Point */
  ICP_impl::Libicp_adapter * libicp;
  libicp = new ICP_impl::Libicp_adapter(NULL);
  libicp->min_delta = 5000;
  libicp->max_iter = 200;

  /* delta computation on surface and target */
  Surface_DeltaWrapper * delta = new Surface_DeltaWrapper;
  delta->norm = new L0_Norm();
  libicp->delta_wrapper = delta;

  conf->icp_pre = libicp;
  conf->icp_post = libicp;

  conf->run_registration();
  conf->calculate_distance_octrees();
  delete conf;
}
```

B.2.2 Configuring Simulated Annealing

Simulated Annealing may be used for aligning the meshes. Evaluation of alignment is the same as for ICP, thus one may use the same *delta computation*. Random numbers

need to be provided by the user. The depreciation field decides how much temperature is to be lowered each step. The *rotation_translation* fields decides whether use simulated annealing for translation or rotation. Set it to *true* for rotation and *false* for translation. If both are needed, there is an ability to chain alignment objects. Inserting an alignment object into the constructor of another alignment object, results in the first alignment object to do its work before the one that received it through its constructor.

```
int main(int, char**) {
  Configurations * conf;
  conf = Configurations::create_standard_config();
  conf->set_registration_procedure(new ViliusRegistration);
  conf->set_distance_procedure(new ViliusDistance);

  /* Simulated Anneiling */
  ICP_impl::SimulatedAnnealingDecorator * sa_decorator;
  sa_decorator = new ICP_impl::SimulatedAnnealingDecorator(
      NULL);
  int n_rand = 1000;
  std::vector<double> rand;

  for(int i = 0; i < n_rand; ++i) {
    double val = 0.01*i;
    rand.push_back(val);
  }
  std::random_shuffle(rand.begin(), rand.end());
  sa_decorator->setRandomNumbers(rand.data(), n_rand);
  sa_decorator->delta_wrapper = new CustomDeltaWrapper();
  sa_decorator->delta_wrapper->norm = new L2_Norm();
  sa_decorator->temperature = 10;
  sa_decorator->deprecitation = 1;
  sa_decorator->rotation_translation = false;

  /* ICP chained with Simulated Annealing */
  ICP_impl::Libicp_adapter * sa_libicp;
  sa_libicp = new ICP_impl::Libicp_adapter(sa_decorator);
  conf->icp_pre = sa_libicp;

  /* plain ICP */
  ICP_impl::Libicp_adapter * libicp;
  libicp = new ICP_impl::Libicp_adapter(NULL);

  conf->icp_post = libicp;

  conf->run_registration();
  conf->calculate_distance_octrees();
  delete conf;
}
```

B.3 Configuring Mesh Coloring

Two options for displaying the difference between the meshes are available. One with color and one as gray scale. The one with color is set as standard. Both maps the value that is to be represented into a scale that is suitable for the coloring procedure. The following fields are available:

1. *min_distance*: The smallest distance between the matching points that are expected.
2. *max_distance*: The biggest distance between the matching points that are expected.
3. *min_range*: The highest value that the distance can be mapped to.
4. *max_range*: The lowest value that the distance can be mapped to.

In the following example, we will map to the range of 0.3 to 1.0. We start at 0.3 so that the mesh will not appear as dark as it would if we started from 0.0.

```
int main(int, char**) {
  Configurations * conf;
  conf = Configurations::create_standard_config();
  conf->set_registration_procedure(new ViliusRegistration);
  conf->set_distance_procedure(new ViliusDistance);

  /* plain ICP */
  ICP_impl::Libicp_adapter * libicp;
  libicp = new ICP_impl::Libicp_adapter(NULL);

  conf->icp_pre = libicp;
  conf->icp_post = libicp;

  GrayScale_DistancePainter * painter = new
     GrayScale_DistancePainter();
  painter->min_distance = 0.0;
  painter->max_distance = 10.0;
  painter->min_range = 0.3;
  painter->max_range = 1.0;

  conf->color = painter;

  conf->run_registration();
  conf->calculate_distance_octrees();
  delete conf;
  delete libicp;
  delete painter;
}
```

B.4 Configuring Extraction Box

For extracting a part of the mesh, call the configuration method *set_extraction()* using a fixed 6 element array as argument. The 6 elements respectively represent the x, y, z

coordinates and the width, depth and height properties of the box.

```
Configurations * conf;
conf = Configurations::create_standard_config();

double  extraction_box[6];
extraction_box[0] =-14.0; /* x */
extraction_box[1] = 26.0; /* y */
extraction_box[2] = 16.0; /* z */
extraction_box[3] = 10.0; /* width  */
extraction_box[4] = 10.0; /* depth  */
extraction_box[5] = 10.0; /* height */

conf->set_extraction(extraction_box);
```

B.5 Configuring Filters

Filters are configured calling either *setBlackListFilter* or *setWhiteListFilter*. The difference between them are that the white list filter accepts the points that it deem compatible whereas the black list filter rejects the points that the filter deem incompatible e.g., if the blacklist filter deem 1 out of ten points in a set incompatible, nine points are passed through. If the whitelist filter deem 1 out of ten points compatible, the nine points that were not compatible are rejected. The available compatibilities are

Distance Compatibility ThresholdFilter

Normal Vector Compatibility ClosestCompatiblePointFilter

Predefined Area Compatibility DefinedBoxExclusionFilter

Wether a point is compatible or not is determined by the relationship between the compatibility and the threshold. The threshold can be set with or without a depreciation. Cosine depreciation and linear depreciation is implemented. Configuring a threshold with no depreciation is equal to adding the *HardThreshold*. In the following example, we configure icp to reject points within a predefined area based on normal vector and distance.

```
ClosestCompatiblePointFilter * normal;
normal = new ClosestCompatiblePointFilter(
              /*starting threshold, ,iterations*/
      new CosineThreshold(0.63, 0, 200), NULL);
ThresholdFilter * distance1 = new ThresholdFilter(
              /*starting threshold, ,iterations*/
      new LinearThreshold(0.4, 0, 200), normal);

ThresholdFilter * distance2 = new ThresholdFilter(
              /*starting threshold, ,iterations*/
      new HardThreshold(0.4), distance1);

ICP_impl::Libicp_adapter * icp;
```

```
icp = new ICP_impl::Libicp_adapter(1, NULL);
icp->setBlackListFilter(distance2);
/* target */    /* source */
double t_x =  5.0; double s_x = 20.0;
double t_y = 15.0; double s_y = 25.0;
double t_z = 20.0; double s_z = 30.0;
double t_w = 10.0; double s_w = 10.0; /* width */
double t_d = 10.0; double s_d = 10.0; /* depth */
double t_h = 10.0; double s_h = 10.0; /* height*/

std::string path = "marker.ply";
DefinedBoxExclusionFilter * predefined = new
   DefinedBoxExclusionFilter(
                       s_x, s_y, s_z, s_w, s_d, s_h,
                       t_x, t_y, t_z, t_w, t_d, t_h,
                       //0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
                       path);
icp->setWhiteListFilter(predefined);
```

B.6 General Configuration

The following is an example of a general configuration that can be applied to any dataset.

```
int main(int, char**) {

  /* Create standard config */
  Configurations * conf;
  conf = Configurations::create_standard_config();

  /* configure color presentation */
  HSV_DistancePainter* color = new HSV_DistancePainter;
  color->min_distance = -.14;
  color->max_distance = .14;
  int start = 180;
  color->min_range = 60;
  color->max_range = 170;
  color->writeScale(); /* writes legend */
  conf->set_color(color);

  /* create instance of ordinary icp */
  ICP_impl::Libicp_adapter * icp;
  icp = new ICP_impl::Libicp_adapter(1, NULL);

  /* create instance with decreasing filter
   * chained with the ordinary icp */
  ICP_impl::Libicp_adapter * icp_with_filter;
  icp_with_filter = new ICP_impl::Libicp_adapter(1, icp);

  /* set distance compatibility */
```

```
  ThresholdFilter * c_d;
  c_d = new ThresholdFilter(
      new LinearThreshold(0.4, 0, 200));
  icp_with_filter->setBlackListFilter(c_d);

  /* configure the pre registration module
   * with the chained icp procedure */
  conf->icp_pre = icp_with_filter;

  /* create empty data fitter */
  ICP_impl::Empty_data_fitter * empty;
  empty = new ICP_impl::Empty_data_fitter;
  conf->icp_post = empty;
  conf->set_registration_procedure(new ViliusRegistration);
  conf->set_distance_procedure(new ViliusDistance);

  /* log is needed before registration is run */
  LogUnit log = Configurations::createStdLogUnit();
  conf->logs.push_back(log);
  std::ostringstream target_path;
  target_path << "result.ply";
  conf->ply_file_name = target_path.str();

  /* Perform the registration */
  conf->run_registration();
  conf->calculate_distance_octrees();

  /* Print params */
  std::string log_file_name = "logfile.txt";
  std::ofstream log_item(log_file_name.c_str(),
      std::ios::out | std::ofstream::app);
  conf->print_log_item(log_item);
  log_item.close();

  delete empty;
  delete icp_with_filter;
  delete conf;
}
```

Bibliography

Arun, K. S., Huang, T. S., and Blostein, S. D. (1987). Least-squares fitting of two 3-d point sets. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (5):698–700.

Besl, P. J. and McKay, N. D. (1992). A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence - Special issue on interpretation of 3-D scenes–part II*, 14(2):239–256.

Bouaziz, S., Taliasacchi, A., and Pauly, M. (2013). Sparse iterative closest point. *Eurographics Symposium on Geometry Processing*, 32(5):113–123.

Chen, Y. and Medioni, G. (1991). Object modeling by registration of multiple range images. In *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*, pages 2724–2729. IEEE.

Cornea, N. D., Silver, D., and Min, P. (2007). Curve-skeleton properties, applications, and algorithms. *IEEE Transactions on Visualization & Computer Graphics*, (3):530–548.

Eggert, D. W., Lorusso, A., and Fisher, R. B. (1997). Estimating 3-d rigid body transformations: a comparison of four major algorithms. *Machine Vision and Applications*, 9(5-6):272–290.

Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395.

Flöry, S. and Hofer, M. (2010). Surface fitting and registration of point clouds using approximations of the unsigned distance function. *Computer Aided Geometric Design*, 27(1):60–77.

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

Godin, G., Rioux, M., and Baribeau, R. (1994). Three-dimensional registration using range and intensity information. In *Photonics for Industrial Applications*, pages 279–290. International Society for Optics and Photonics.

Gonzalez, R. and Woods, R. (2011). *Digital Image Processing*. Pearson Education.

Haralick, R. M., Joo, H., Lee, D., Zhuang, S., Vaidya, V. G., and Kim, M. B. (1989). Pose estimation from corresponding point data. *Systems, Man and Cybernetics, IEEE Transactions on*, 19(6):1426–1446.

Horn, B. K. (1987). Closed-form solution of absolute orientation using unit quaternions. *JOSA A*, 4(4):629–642.

Horn, B. K., Hilden, H. M., and Negahdaripour, S. (1988). Closed-form solution of absolute orientation using orthonormal matrices. *JOSA A*, 5(7):1127–1135.

Joblove, G. H. and Greenberg, D. (1978). Color spaces for computer graphics. In *ACM siggraph computer graphics*, volume 12, pages 20–25. ACM.

Kazakauskas, V. (2014). Comparing big datasets.

Li, C., Luo, X., Du, S., and Xiao, L. (2012). A method of registration based on skeleton for 2-d shapes. In *Image and Signal Processing (CISP), 2012 5th International Congress on*, pages 810–813. IEEE.

Lin, T., Shih, W., Chen, W., and Ho, W. (2006). 3d face authentication by mutual coupled 3d and 2d feature extraction. In *Proceedings of the 44th annual Southeast regional conference*, pages 423–427. ACM.

Marinov, A. and Zlateva, N. (2010). Icp algorithm for alignment of stars from astronomical photographic images. In *Proceedings of the 11th International Conference on Computer Systems and Technologies and Workshop for PhD Students in Computing on International Conference on Computer Systems and Technologies*, pages 485–489. ACM.

Marjanovic, G. and Solo, V. (2012). On optimization and matrix completion. *Signal Processing, IEEE Transactions on*, 60(11):5714–5724.

Masuda, T., Sakaue, K., and Yokoya, N. (1996). Registration and integration of multiple range images for 3-d model construction. In *Pattern Recognition, 1996., Proceedings of the 13th International Conference on*, volume 1, pages 879–883. IEEE.

Mavridis, P., Andreadis, A., and Papaioannou, G. (2015). Efficient sparse icp. *To appear in Computer Aided Geometric Design (Proc. Geometric Modeling and Processing)*.

Meer, P., Mintz, D., Rosenfeld, A., and Kim, D. Y. (1991). Robust regression methods for computer vision: A review. *International journal of computer vision*, 6(1):59–70.

Papaioannou, G., Karabassi, E.-A., and Theoharis, T. (2001). Virtual archaeologist: Assembling the past. *Computer Graphics and Applications, IEEE*, 21(2):53–59.

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

Rousseeuw, P. J. (1984). Least median of squares regression. *Journal of the American statistical association*, 79(388):871–880.

Rusinkiewicz, S. and Levoy, M. (2001). Efficient variants of the icp algorithm. In *3-D Digital Imaging and Modeling, 2001. Proceedings. Third International Conference on*, pages 145–152. IEEE.

S. Kirkpatrick, C. D. Gelatt, M. P. V. (1983). Optimization by simulated annealing. *Science*, 220(4598):671–680.

Simon, D. A. (1996). *Fast and accurate shape-based registration*. PhD thesis, Carnegie Mellon University Pittsburgh.

Theoharis, T. (2008). *Graphics and Visualization: Principles & Algorithms*. Ak Peters Series. Taylor & Francis.

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

Walker, M. W., Shao, L., and Volz, R. A. (1991). Estimating 3-d location parameters using dual number quaternions. *CVGIP: image understanding*, 54(3):358–367.

Weik, S. (1997). Registration of 3-d partial surface models using luminance and depth information. In *3-D Digital Imaging and Modeling, 1997. Proceedings., International Conference on Recent Advances in*, pages 93–100. IEEE.

Zhang, Z. (1994). Iterative point matching for registration of free-form curves and surfaces. *International journal of computer vision*, 13(2):119–152.