Geometric Hybrid Registration

Dissertation zur Erlangung des Doktorgrades

vorgelegt am

Fachbereich Mathematik und Informatik
der Freien Universität Berlin

2010

von

Fabian Stehn

Institut für Informatik
Freie Universität Berlin
Takustraße 9
14195 Berlin

gefördert durch die Deutsche Forschungsgemeinschaft (DFG)
im Rahmen der Projekte Kn 591/2-1, Kn 591/2-2 und Al 253/6-2

Author: Fabian Stehn
EMail: fabian.stehn@me.com

Supervisor: Prof. Dr. Christian Knauer
Universität Bayreuth
Institut für Angewandte Informatik
Universitätsstrasse 30
95447 Bayreuth, GERMANY
EMail: christian.knauer@uni-bayreuth.de
Phone: +49 921-5577-40
Fax: +49 921-5577-42

Co-Supervisor: PD Dr. Klaus Kriegel
Freie Universität Berlin
Institut für Informatik
Takustrasse 9
14195 Berlin, GERMANY
EMail: kriegel@inf.fu-berlin.de
Phone: +49 30-838-75156
Fax: +49 30-838-75192

Referees: Prof. Dr. Christian Knauer
Universität Bayreuth
Institut für Angewandte Informatik
Universitätsstrasse 30
95447 Bayreuth, GERMANY
EMail: christian.knauer@uni-bayreuth.de
Phone: +49 921-5577-40
Fax: +49 921-5577-42

Prof. Dr. Alexander Wolff
Universität Würzburg
Lehrstuhl für Informatik I
Am Hubland
97074 Würzburg, GERMANY
EMail: alexander.wolff@uni-wuerzburg.de
Phone: +49 931-31-85055
Fax: +49 931-31-84600

defended: February 14, 2011
version: March 2, 2011

This thesis is dedicated to all who have read it;
especially to those who had to read it.

Abstract

Geometric matching problems are among the most intensely studied fields in Computational Geometry. A geometric matching problem can be formulated as follows: given are two geometric objects P and Q. These objects are taken from a class of geometric objects $\mathcal{G}$ and P is called the *pattern* and Q is called the *model*. A geometric matching instance is defined for a distance measure $\mathrm{dist}_{\mathcal{G}} : \mathcal{G} \times \mathcal{G} \to \mathbb{R}^+$ and a transformation class $\mathcal{T}$. The task is to find the transformations $t \in \mathcal{T}$ that minimize $\mathrm{dist}_{\mathcal{G}}(t(P), Q)$.

In this thesis, *geometric hybrid registration problems* are studied. Registration problems are closely related to geometric matching problems. The term geometric registration problem describes the task of mapping points from one space (*pattern space*) to their corresponding points in a deformed copy of that space called *model space*.

This research is motivated by a real world application: navigated surgery. Here, the goal is to register an operation theatre space (pattern space) to the internal coordinate system (model space) of a medical navigation system. The purpose of a medical navigation system is to support surgeons by visualizing the used surgical instruments at their correct position in a 3D-model of a patient. The models are generated beforehand based on CT or MRT scans.

Hybrid registration is a novel strategy to compute solutions for this alignment problem. Geometric hybrid registrations reduce the spatial synchronization problem to a series of (at least two) geometric matching problems that are solved interdependently. Usually, a computationally involved point-to-surface matching is combined with a comparably simpler but underdefined point-to-point matching. The point-to-surface matching is computed for a sufficiently large and suitably distributed set of points (called surface points) measured in the pattern space to a geometric surface in the model space. For the point-to-point matching, a small set of (one to three) characteristic points are measured in the pattern space and are defined in the model space. In the context of the intended application, these points are called *anatomic landmarks* – anatomically exposed spots within the field of interest.

In **Chapter 2**, surprisingly simple constant-factor approximations are presented for a special subproblem of registrations under rigid motions: computing a rotation around a fixed axis that minimizes the Hausdorff distance of a point set to a set of lines, line segments or triangles in the plane or in $\mathbb{R}^3$.

In **Chapter 3**, we present two hybrid rigid motion registration strategies . These two algorithms compute fast and robust absolute-error approximations for settings in which two anatomic landmarks are available. Two different discretization techniques are used, based on whether or not the defined anatomic landmarks are available already during the preprocessing phase.

In **Chapter 4**, approximation schemes and heuristics are introduced for the computationally difficult task of computing hybrid registrations with a single given anatomic landmark. Here, rigid motion registrations are considered that minimize the Hausdorff distances of the involved geometric features to surfaces that are given as point clouds or as triangulated surfaces.

In **Chapter 5**, the *weighted directed Hausdorff distance* is introduced along with matching algorithms that minimize this measure. Translations and rigid motions of point sets in arbitrary dimensions are considered. The weighted directed Hausdorff distance allows to take the different precision levels into account with which the different feature types can be measured.

A novel matching concept named *non-uniform matchings* is presented in **Chapter 6**. This approach allows to compute a *set* T of transformations instead of a single transformation to perform the mapping from the pattern space to the model space. The pattern space is partitioned into areas of interest and a computed transformation $t \in T$ is only valid within one cell of this partition. The transformation set has to satisfy two competing targets: the transformations have to match the geometric features of a cell close to their counterparts in the model. On the other hand, transformations for neighboring cells have to be *similar* to guarantee a smooth mapping.

Computing a set of transformations reduces the effect of local deformations. Dealing with local deformations and several areas of interest is an essential step towards developing registration algorithms for the clinically highly relevant application of navigated *soft tissue* surgeries.

In the practical part of this thesis, in **Chapter 7**, a study that has been done in cooperation with the *Klinik und Hochschulambulanz für Neurochirurgie, Charité-Universitätsmedizin Berlin* and the industry partner *Prosurgics Ltd.* is presented. In this study, the degree of precision of measuring surface points and anatomic landmarks are theoretically and empirically analyzed. From this study, adjustment parameters are deduced that are essential for all hybrid registration methods that are based on these feature types.

A framework is presented in **Chapter 8** in which most of the hybrid registration algorithms have been implemented.

Zusammenfassung

Geometrische Musteranpassungsprobleme gehören zu den am intensivsten studierten Felder der Algorithmischen Geometrie. Ein geometrisches Musteranpassungsproblem kann wie folgt formuliert werden: Gegeben sind zwei geometrische Objekte P und Q. Diese Objekte gehören zu einer Klasse $\mathcal{G}$ von geometrischen Objekten, wobei P *Muster* und Q *Modell* genannt wird. Ein geometrisches Musteranpassungsproblem ist bestimmt durch ein Abstandsmaß $\mathrm{dist}_{\mathcal{G}} : \mathcal{G} \times \mathcal{G} \to \mathbb{R}^+$ sowie durch eine Transformationsklasse $\mathcal{T}$. Gesucht sind diejenigen Abbildungen $t \in \mathcal{T}$, welche die Zielfunktion $\mathrm{dist}_{\mathcal{G}}\,(t(P), Q)$ minimieren.

In dieser Arbeit werden hybride Registrierungsprobleme untersucht, welche in einem engen Zusammenhang zu geometrischen Musteranpassungsproblemen stehen. Bei einem Registrierungsproblem besteht die Aufgabe darin, eine Abbildung zu finden, die jeden Punkt eines *Musterraums* auf seinen entsprechenden Punkt in einem *Modellraum* abbildet. Die Forschung an dieser Problemstellung ist motiviert durch eine Anwendung aus in der Neurochirurgie – durch bildgeführte Operationsverfahren. Hierbei wird während einer Operation mit Hilfe eines medizinischen Navigationssystems das verwendete Operationsbesteck in der korrekten relativen Lage und Position in einem 3D-Modell des Patienten visualisiert. Das Modell wird zuvor aus einer CT- oder MRT-Aufnahme generiert. Hierfür ist es notwendig, das Operationsfeld und das interne Koordinatensystem des Navigationssystems aneinander auszurichten; also eine Abbildung zu finden, welche jedem Punkt des Operationsfeldes seinen entsprechenden Punkt im Modellraum zuordnet.

Der Schwerpunkt dieser Arbeit liegt auf Registrierungsstrategien, welche das beschriebene Ausrichtungsproblem mit Hilfe eines neuartigen *hybriden* Ansatzes lösen. Hierbei wird das Problem auf eine Menge von (zumindest zwei) geometrischen Musteranpassungsproblemen reduziert, welche in wechselseitiger Abhängigkeit gelöst werden. Die Grundidee ist, ein rechnerisch anspruchsvolles Punkt-zu-Oberflächen Anpassungsproblem mit einem vergleichsweise einfacherem aber unterdefiniertem Punkt-zu-Punkt-Anpassungsproblem zu kombinieren. Die Punkt-zu-Oberflächen-Anpassung wird für eine Menge von im Operationsfeld eingemessenen Punkten und einer im Modellraum definierten Oberfläche berechnet. Die Punkt-zu-Punkt Anpassung hingegen basiert auf einer kleinen Menge von (ein bis drei) so genannten *anatomischen Landmarken*. Anatomische Landmarken sind anatomisch exponierte Stellen, welche auf dem Patienten leicht eingemessen und in dem Modell leicht definiert werden können.

In **Kapitel 2** wird eine überraschend einfache Approximation mit konstantem Approximationsfaktor für folgendes Problem vorgestellt: Gesucht sind diejenigen Rotationen um eine gegebene Achse, welche

den gerichteten Hausdorff-Abstand einer Punktmenge zu einer Menge von Geraden, Strecken oder Dreiecken in der Ebene oder im $\mathbb{R}^3$ minimieren. Die Bestimmung einer minimierenden Rotation ist ein wesentliches Teilproblem bei der Berechnung einer Registrierung für die Transformationsklasse der starren Bewegungen.

Zwei hybride Registrierungsverfahren für starre Bewegungen werden in **Kapitel 3** vorgestellt. Diese Verfahren berechnen schnelle und robuste Approximationen mit absolutem additiven Fehler für den Fall, dass genau zwei anatomische Landmarken zur Registrierung zu Verfügung stehen. Diese Verfahren verwenden unterschiedliche Diskretisierungsstrategien in Abhängigkeit davon, ob die im Modellraum definierten Landmarken bereits zum Zeitpunkt der Vorverarbeitung zur Verfügung stehen.

In **Kapitel 4** werden Approximationsschemata sowie eine Heuristik vorgestellt für die rechnerisch anspruchsvolle Aufgabe der Bestimmung einer Registrierung bei nur einer gegebenen anatomischen Landmarke.

Die Genauigkeit mit welcher anatomische Landmarken eingemessen werden können, unterscheidet sich von der Genauigkeit für die Einmessung von Oberflächenpunkten. Um diesem Unterschied Rechnung zu tragen, wird in **Kapitel 5** ein neues Abstandsmaß, der *gerichtete gewichtete Hausdorff Abstand* eingeführt. Die in diesem Kapitel vorgestellten Registrierungsverfahren berechnen Abbildungen bei denen diese Genauigkeitsunterschiede berücksichtigt werden können.

Eine vielversprechende Verallgemeinerung des geometrischen Musteranpassungsproblems wird in **Kapitel 6** vorgestellt: so genannte *nicht-uniforme geometrische Musteranpassungsprobleme*. Bei diesem Ansatz wird eine *Menge* von Transformationen berechnet, welche den Musterraum in den Modellraum abbilden anstatt einer einzelnen Transformation. Hierfür wird der Musterraum in Zellen partitioniert und eine berechnete Transformation ist nur gültig innerhalb der ihr zugewiesenen Zelle. Die Transformationen haben zwei potentiell konkurrierende Kriterien zu erfüllen: Zum Einen sollen die sich in den Zellen befindlichen Objekte nah an ihre Entsprechungen im Modellraum abgebildet werden. Gleichzeitig sollen Transformationen aus benachbarten Zellen *ähnlich* sein um eine gleichmäßige Gesamtabbildung zu gewährleisten.

Die Berechnung einer Menge von Transformationen ermöglicht es, den Einflussbereich von lokalen Deformationen zu beschränken. Dies ist ein wichtiger Schritt hin zur Entwicklung von Algorithmen für die in der klinischen Praxis sehr interessante Anwendung der *Weichteilnavigation*.

Im praktischen Teil dieser Arbeit wurde in **Kapitel 7** eine Studie vorgestellt, welche in Zusammenarbeit mit Partnern der *Klinik und Hochschulambulanz für Neurochirurgie, Charité-Universitätsmedizin Berlin* sowie der Firma *Prosurgics Ltd.* durchgeführt wurde. In dieser Studie werden die unterschiedlichen Messungenauigkeiten untersucht, welche beim Einmessen von anatomischen Landmarken und Oberflächenpunkten auftreten. Diese theoretische wie empirische Untersuchung ermöglicht es, Gewichtungsparameter abzuleiten, welche für alle hybriden Registrierungsmethoden essentiell sind, die auf diesen Eingabetypen beruhen.

Schließlich wird in **Kapitel 8** ein Rahmenwerk vorgestellt, in welchem die meisten der hier vorgestellten hybriden Registrierungsalgorithmen implementiert wurden.

Acknowledgement

The work presented in this thesis was carried out during my employment at the Insitute of Computer Science at the Freie Universität Berlin. My position was funded by the Deutsche Forschungsgemeinschaft (DFG) by the grants Kn 591/2-1, Kn 591/2-2 and Al 253/6-2.

I am indebted to a large number of people who supported and accompanied me on this exciting journey. My last ten years in Berlin were without doubt the most instructive, joyful and challenging time in my life so far.

First and foremost, I would like to express my gratitude to my supervisors Christian Knauer and Klaus Kriegel for giving me the opportunity to work on this thesis, for their advice, for their support and for their encouragements. And I thank Alexander Wolff for agreeing to review this thesis.

I thank my friends and colleagues from the work group Theoretical Computer Science at the Freie Universität Berlin and I am glad about the redundancy in this sentence. I consider myself very lucky that I had the chance to do my PhD in such an inspiring and friendly environment. I especially like to thank Daria Schymura from whom I learned a lot about math and life and Marc Scherfenberg for being a close friend and a perfect office mate. Again by being redundant, I like to thank all contributors to the “Kaffeerunde” for great discussions about “life, the universe and everything”.

I owe to Hee-Kap Ahn , Esther Arkin , Otfried Cheong , Rolf Klein and Joseph Mitchell for being great hosts and for giving me the opportunity to work together with them on a variety of most interesting problems and I thank Darko Dimitrov for our joined research during my first years in the workgroup.

During my time at the Institute of Computer Science I had the luck to win really great friends. First and foremost I want to thank Georg Wittenburg without whom this time would have definitely been a lot less enjoyable. I also thank Katharina Hahn for always having an open ear and door.

I also want to express my deep gratitude to my family, especially to my mother Ulrike “Riki” Stehn for more than could be expressed here (including answering even the most difficult questions regarding my thesis) and to my friend and sister Rena Stehn .

Mein aufrichtiger Dank gilt ebenso meinen Großeltern Gertrud und Hans Cicurs für ihren festen Glauben an mich und für ihre Unterstützung in den vergangenen Jahren.

Last but my no means least I thank Daniela “Fuchskind” Schreiter for being the great and adorable person she is.

Contents

Chapter 1

Introduction

A geometric registration problem, or registration problem for short, is the task of computing a mapping from a given *pattern space* into a given *model space*. Such a mapping is also called *registration* in this context. The model space can be seen as a potentially distorted copy of the pattern space with its own coordinate system. A registration maps each point of the pattern space to its corresponding point in the model space, see Figure 1.1. A registration can be seen as an alignment of the two spaces, converting the coordinate system of the pattern space into the coordinate system of the model space.

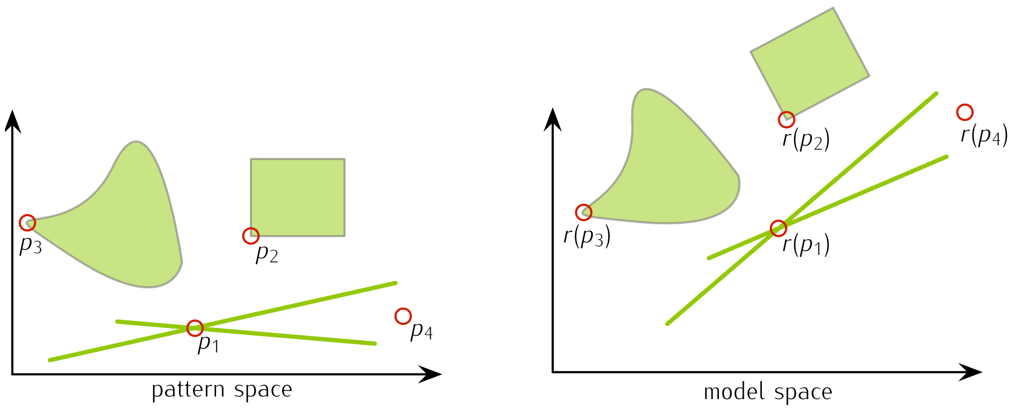


Figure 1.1: Illustration of a registration r that maps points from the pattern space to their corresponding points in the model space.

The registration problems considered here are motivated by a real life application: navigated surgery in rigid tissue. Nowadays most neurosurgical interventions are supported by so called medical navigation systems. A medical navigation system is a device that visualizes the surgical instruments that are used during an operation in a 3D-model of the anatomically relevant area of the patient. The model itself is constructed beforehand based on the result of an imaging process such as an X-ray computed tomography (CT) or magnetic resonance imaging (MRT) scans.

It is of utmost importance that the instruments are displayed at the correct relative position and in the correct alignment in the model with respect to the patient in the operation theatre. The surgeon depends on these systems as typically the spot (tumor) on which the operation is performed is occluded by other tissue and hence can not be seen directly. In this thesis, we primarily concentrate on *rigid tissue registrations* as required for surgeries on brain tumors that are surrounded by rigid (generally not deformable) tissue – the skull.

A medical navigation system has to solve a registration problem. Namely, to compute an alignment of the operation theatre space (pattern space) to the space containing the anatomical model of the patient (model space).

1.1 Problem Definition

Intuitively, a registration is the combination of dislocation, noise, distortion and deformation that if applied to the pattern space gives the model space. In most real life applications, there is neither a precise model of these influencing factors nor an explicit description of them, which makes it impossible to determine registrations exactly.

Formally stated, a registration $r\colon \mathbb{R}^n \to \mathbb{R}^m$ is a mapping from the pattern space $\mathbb{P}$ into the model space $\mathbb{Q}$. Applications where the two spaces are of different dimensionality are for example registrations of slice images (2D) to MRT models (3D).

For our context however, the spaces $\mathbb{P}$ and $\mathbb{Q}$ can be seen as representations of the same space where $\mathbb{Q}$ is a copy of $\mathbb{P}$ *deformed* by r. Hence, we can write $\mathbb{Q} = r(\mathbb{P})$.

Definition 1 (ϵ-Registration of $\mathbb{P}$ to $\mathbb{Q}$)**.**
For $\mathbb{P}$, $\mathbb{Q}$ and r as before and a distance measure dist$\colon \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}^+$*, a* ϵ-registration *for an $\epsilon \geqslant 0$ of $\mathbb{P}$ to $\mathbb{Q}$ is a function $r'\colon \mathbb{P} \to \mathbb{Q}$ with the property that*

$$\forall p \in \mathbb{P} \;\; \operatorname{dist}\big(r(p), r'(p)\big) \leqslant \epsilon. \tag{1.1}$$

Note, that in general the actual deformation r itself as well as the transformation class from which r is drawn are forever unknown. Thus, it is not possible to compute an ϵ-registration r' for some $\epsilon \geqslant 0$ that is guaranteed to satisfy Equation 1.1 for *all* points of the pattern space $\mathbb{P}$.

As it is hard to compute a registration directly, a closely related problem is solved instead. In a first step, geometric features such as point sets, lines, triangles or surfaces are extracted from both spaces. Then, a transformation from a given transformation set is computed that maps the features from the pattern space as closely as possible to their corresponding features in the model space with respect to a suitable distance measure.

In the context of the presented application, the pattern space $\mathbb{P}$ is the operation theatre space. The coordinate system that is used for the pattern space is the internal coordinate system of a tracking device with which the position of objects in that space is determined. The space $\mathbb{Q}$ on the other hand is the space of the visualization component of the navigation system that visualizes the model of the patient. The features that are extracted from and defined in both spaces are described in detail in Section 1.3.

1.2 Reducing Registrations to Geometric Matching Problems

The features that are measured in the pattern space are called the *pattern* and the measured or defined features in the model space are called the *model*. Registration problems are typically reduced to *geometric matching problems*. A geometric matching problem can be formulated as:

Definition 2 (Geometric Matching Problem).
Given:

$\mathcal{G}$	*a class of geometric objects*
$\mathrm{dist}_{\mathcal{G}} : \mathcal{G} \times \mathcal{G} \to \mathbb{R}^+$	*a distance measure on* $\mathcal{G}$
$P \in \mathcal{G}$	*a pattern object*
$Q \in \mathcal{G}$	*a model object*
$\mathcal{T}$	*a set of admissible transformations of* $\mathcal{G}$

Task: Compute a transformation $t \in \mathcal{T}$ minimizing

$$\mathrm{dist}_{\mathcal{G}}(t(P), Q). \tag{1.2}$$

The transformation t that minimizes Equation 1.2 is then taken as the registration to map all points of $\mathbb{P}$ into $\mathbb{Q}$.

Geometric matching problems are among the most intensely studied problems in Computational Geometry and a lot of research has been done in this field. A survey by Alt and Guibas [1] gives an overview of 30 years of research in this area.

Matching problems have a vast set of applications in a variety of fields. Among these are drug design [17], company logo detection [2], Egyptian hieroglyph matching [47], automatic telescope guiding [8], optical character recognition [9], geometric graph matching [7] and satellite image registration [19] to name just a few.

1.2.1 Current Approaches to Registration Problems

Fiducial Registrations

A standard strategy to solve the registration problem of the presented clinical application is to perform a *fiducial-to-fiducial* matching [24] based on geometric hashing methods. A *fiducial* is an artificial marker that is attached to the head of the patient either by glueing it onto the skin or by screwing it into the skull, see also Section 1.4.1.

From a Computational Geometry point of view a *fiducial-to-fiducial* matching is a point-to-point matching either with or without known correspondence. Point-to-point matchings are matchings where both, the pattern and the model are point sets. If the correspondence is given, it is additionally known which point of P is mapped onto which point of Q.

The solution to the geometric matching problem is a transformation, typically a rigid motion or an almost rigid affine map, that minimizes the root mean square distance of the transformed pattern to the model. With *almost rigid* affine maps we understand mappings whose transformation matrix have a determinant that is close to 1.

Definition 3 (Root Mean Square Distance).
The root mean square *distance* (RMS) *of finite point sequences* $A = \{a_1, \ldots, a_n\}, B = \{b_1, \ldots, b_n\} \subset \mathbb{R}^d$ *is defined as*

$$\mathrm{RMS}(A, B) := \min_{\pi \in \Pi(n)} \sqrt{\frac{1}{n} \sum_{i \in [n]} \|a_i - b_{\pi(i)}\|^2},$$

where $\|\cdot\|$ *is the Euclidean Norm and* $\Pi(n)$ *is the set of all permutations of* $\{1, \ldots, n\}$*.*

Other Approaches

A common and general technique to solve point-to-surface matching problems is the *iterative closest point* (ICP) method [5, 42, 38]. The basic idea is that from an unspecified initial position of a point set A the nearest Euclidean neighbor for each $a \in A$ in a geometric surface B is computed. Then, a transformation is applied to A that minimizes the mean square distance of the point pairs $(a, n_B(a))$, where $n_X(z)$ denotes the nearest neighbor of z in X. This process is iterated until either

- a certain threshold is reached,
- a certain number of iterations is reached,
- the nearest neighbor assignment does not change after applying a motion (i.e., a local minimum is reached),
- the decrease of distance between two successive iteration steps is below a certain threshold.

This approach, however, computes only local minima and does not guarantee to compute a solution whose quality is in any relation to the global optimum.

Other surface matching techniques are based on *label relaxation* [32], *spin images* [26], *surface point images* [49], *geometric fingerprints* [44] or geodesics in combination with local geometry [48].

1.3 A New Approach: Hybrid Registrations

The novel approach that is studied in this thesis is called *hybrid registration*.

For a hybrid registration problem, the alignment problem is reduced to a *series* of (typically two) geometric matching problems that are solved interdependently. Usually, a computationally difficult point-to-surface matching is combined with a comparably simpler but underdefined point-to-point matching. A system is called *underdefined* if it does not determine all degrees of freedom. In this case, knowing the image of one or of two points does not fix all degrees of freedom of a rigid motion or of an affine map in $\mathbb{R}^3$.

The point-to-surface matching is computed for a sufficiently large and suitably distributed set of points P (called surface points) measured in the pattern space to a geometric surface $\mathcal{S}$ (the surface of the 3D-model) in the model space. For the point-to-point matching, a small set (one to three) characteristic points are measured in the pattern space P_c and their counterparts Q_c are manually defined in the model space. In the context of the intended application, these points are called *anatomic landmarks* – anatomically exposed spots within the field of interest, like the outer corner of an eye or the root of the nasal bone, see Figure 1.2. Throughout this thesis we will use the terms characteristic point and anatomic landmark synonymously.

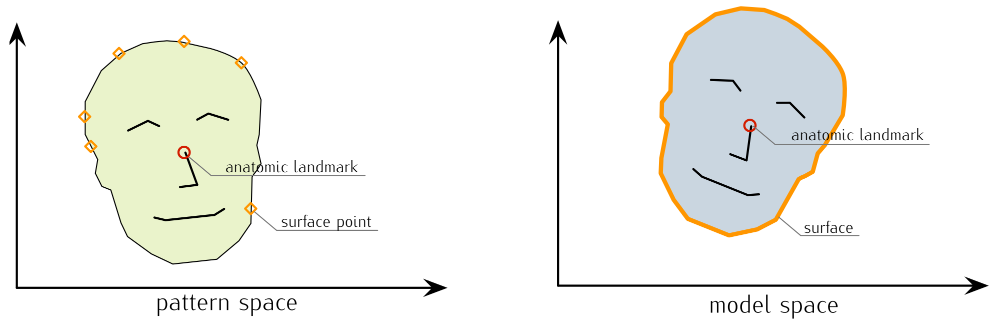


Figure 1.2: Two dimensional example of the input of a hybrid registration problem. Six surface points (P_2) and one anatomic landmark (P_1) are measured in the pattern space. One anatomic landmark (Q_1) as well as the surface (Q_2) is defined in the model space.

The features are manually measured in the operation theatre space with a traceable device after the patient is affixed on the operation table. In a pre-processing step, the surface $\mathcal{S}$ is extracted by a *skin-segmentation process* that determines all voxels that can be classified as skin. The anatomical landmarks on the model are again defined manually during the operation planning phase.

A hybrid registration problem can be formulated as follows:

Definition 4 (Hybrid Registration Problem).
Given:

$\mathcal{G}$	*a class of geometric objects*
$\mathrm{dist}_{\mathcal{G}} : \mathcal{G} \times \mathcal{G} \to \mathbb{R}^+$	*a distance measure on* $\mathcal{G}$
$\mathcal{P} = P_1, \ldots, P_m$	*a sequence of geometric features with* $P_i \in \mathcal{G}$ *for* $i \in [m]$
$\mathcal{Q} = Q_1, \ldots, Q_m$	*a sequence of geometric features with* $Q_i \in \mathcal{G}$ *for* $i \in [m]$
$\mathcal{T}$	*a set of admissible transformations of* $\mathcal{G}$
$f : \mathbb{R}^m \to \mathbb{R}$	*an aggregate function*

Task: *Compute a transformation* $t \in \mathcal{T}$ *minimizing*

$$f\left(\left(\mathrm{dist}_{\mathcal{G}}\left(t(P_i), Q_i\right)\right)_{i \in [m]}\right) \tag{1.3}$$

The objective function that is minimized in almost all registration algorithms in this thesis is the *directed Hausdorff distance* or variants of it. The directed Hausdorff distance is an established distance measure (but *not* a metric) that is used in many geometric matching applications.

Definition 5 (Directed Hausdorff Distance).
The directed Hausdorff distance $\vec{h}(A, B)$ *for two sets* $A, B \subset \mathbb{R}^d$ *and a distance measure* $\mathrm{dist} \colon \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^+$ *is defined as:*

$$\vec{h}(A, B) := \sup_{a \in A} \inf_{b \in B} \mathrm{dist}(a, b).$$

Throughout this thesis we consider closed and bounded sets and set the embedded distance measure to the Euclidean distance $\|\cdot\|$, which allows us to define the directed Hausdorff distance for two sets

$A, B \subset \mathbb{R}^d$ as

$$\vec{h}(A, B) := \max_{a \in A} \min_{b \in B} \|a - b\|,$$

see Figure 1.3.

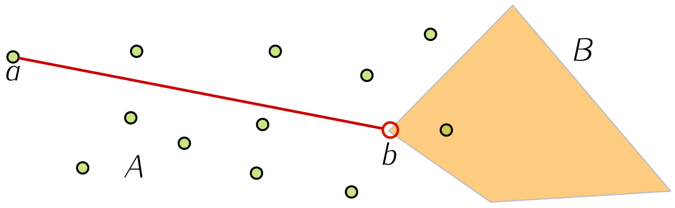


Figure 1.3: The directed Hausdorff distance of the point set A and the polygon B is realized by the Euclidean distance $\|a - b\|$.

For the typical case of solving a point-to-surface problem (surface points P to surface $\mathcal{S}$) and a point-to-point problem (anatomic landmarks P_c to anatomic landmarks Q_c) with respect to the directed Hausdorff distance, we can rewrite the objective function of Equation 1.3 as

$$\max\left(\vec{h}(t(P), \mathcal{S}), \vec{h}(t(P_c), Q_c)\right),$$

where the maximum function is used as the aggregate function f.

Remarks

The embedded distance measure $\text{dist}_\mathcal{G}$ has been fixed to the directed Hausdorff distance. The Hausdorff distance has two advantageous properties with respect to this application:

1. it is a *conservative* distance measure, as it is defined by the *worst* matched feature,
2. as it is the maximum over individual point-to-feature distances, it often allows to decompose the problem into smaller subproblems that can efficiently be combined.

As mentioned in Chapter 9.2 one could also consider other distance measures or aggregate functions such as the sum of all individual $\text{dist}_\mathcal{G}$ distances.

1.3.1 Previous Results for Hybrid Registrations

The research in this thesis was preceded by the work of Dimitrov et al. [13]. The authors present rigid motion registrations for settings in which *exactly two* characteristic points are given and where the surface $\mathcal{S}$ is given as a triangulated domain. They present an $O(mn\, \varphi(mn) \log mn)$-time algorithm to compute a *semioptimal* solution, where n is the number of points in P, m is the number of triangles in $\mathcal{S}$ and $\varphi(\cdot)$ is the inverse Ackermann function. Let $\mathcal{T}_{sem} \subset \mathcal{T}$ be the subset of transformations (here: rigid motions) that minimize the directed Hausdorff distance of P_c to Q_c. Semioptimal registrations are registrations $t \in \mathcal{T}_{sem}$ that minimize the directed Hausdorff distance of P to $\mathcal{S}$.

The authors also present for all $\epsilon > 0$ a $(1+\epsilon)$-approximation scheme of an optimal solution that has a runtime of

$$\mathrm{O}\left(\frac{\rho_P^5}{\epsilon^5}\, mn\ \varphi(mn) \log mn\right),$$

where ρ_P is the relative diameter of the set P with respect to the center of $P_c = \{p_1, p_2\}$, i.e.,

$$\rho_P = \frac{2\ \max_{p \in P} \|(p_1 - p_2)/2 - p\|}{\|p_1 - p_2\|}.$$

1.4 Medical Image Registration

The considered application of registrating a patient to his or her model is also studied by biomedical engineers in the context of *medical image registrations*. The term medical image registration actually covers a vast set of registration applications where the introduced motivation forms one subfield.

An overview over the remarkable amount of work in this field is given in survey articles by Maurer et al. [35], Elsen et. al. [46], Maintz et al. [33] as well as Dawant [10].

Maintz and Viergever propose a classification system based on nine criteria to classify registration approaches. A classification of the approaches of this thesis can be found in Table 1.1.

criterion	**classification**	**description**
dimensionality	2D-2D as well as	
	3D-3D	in general kD-kD
nature of registration basis	intrinsic	no artificial objects (such as artificial markers) are used to perform the registration
nature of transformation	translations	
	rigid motions	translations plus rotations
domain of transformation	global	one transformation for the entire space
	local	subregions have their own transformation
interaction	automatic	the algorithms are not interactive and do not require manual initial alignments
optimization procedure	parameters computed	the parameters of a registration are deduced or manually defined and not computed by finding an optimum of some function defined in the parameter space
modularities involved	patient to modality	usually to a CT or MRT model
subject	intrasubject	all data are acquired from the same patient, i.e., the patent is registrated to *his* or *her* model
object	head	more specific: brain or skull

Table 1.1: Classification of the approaches that are studied in this thesis with respect to the classification system proposed by Maintz and Viergever [33].

1.4.1 Fiducial Based Registrations

As described in Section 1.2.1, a standard strategy that is currently used to compute patient-to-model (head-to-model-of-head) registrations are fiducial-to-fiducial registrations [36, 16, 24, 20]. In a common variant of this technique, a fiducial is an artificial marker in the form of a plastic cylinder. A fiducial is affixed to the patient by either screwing it into the skull with tiny bolts or by locking it onto a plateau that is glued to the skin of the patient, see Figure 1.4 left.

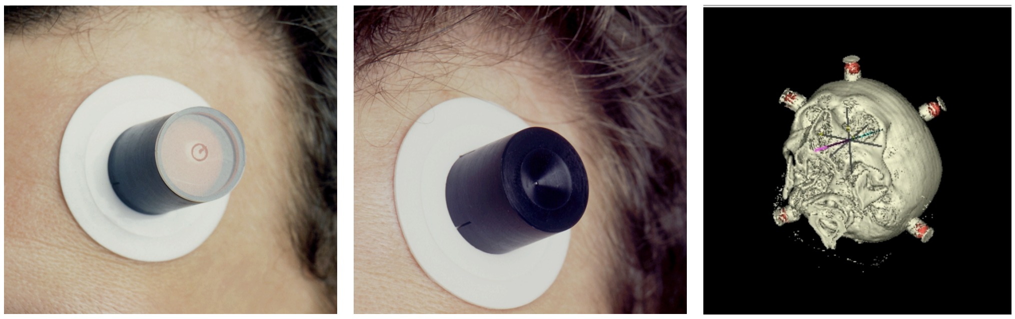


Figure 1.4: **left:** fiducial attached to glued plateau for imaging phase, **middle:** a measurable fiducial on the same plateau, **right:** visualization of MRT scan with detected centers of the fiducials. [Illustration by permission of Prosurgics Ltd.]

Typically, five to eight fiducials are affixed to the head of the patients *before* the imaging process is performed. After the image is gained, the cylinder is removed but the plateau remains. The cylinder of a fiducial contains a small ball of a material that is easily and uniquely detectable in the image. Before the operation is performed, the head of the patient is affixed in a special holding device on the operation table so that its position remains invariant throughout the surgery. After being affixed, new cylinders are locked onto the plateaus. These cylinders have a notch in the form of a cone whose apex is exactly where previously the center of the enclosed ball was. In a final step, these apices are measured by a traceable device to get a point set P in the operation theatre space. The corresponding set Q are the centers of the balls in the image which are automatically detected by the navigation device, see Figure 1.4.

The registration problem is then reduced to a point-to-point matching of P to Q with respect to the root mean square distance.

Fitzpatrick et al. [18] studied the effect of noise and error on the registration quality for fiducial based registrations. They distinguish the following error terms:

FLE fiducial location error, the error introduced by dislocation of fiducials,
FRE fiducial registration error, the error *at the fiducials* after applying the registration,
TRE target registration error, the error *at the point of interest* after applying the registration.

Their analysis shows that the crucial term, the TRE, the error at the point of interest (usually the tumor) to a large part depends on the distribution of the fiducials around the spot of interest.

1.4.2 Real World Restrictions

The input data for the considered geometric matching problems result from real world measuring processes. Geometric data that is acquired by imaging processes or tracking devices are subject to noise and imprecision. One source of imprecision is the resolution of the imaging process. The voxels that result from imaging devices that are currently used in clinical praxis have a resolution of typically

- $1\text{mm} \times 1\text{mm} \times 1\text{mm}$ for *magnetic resonance imaging* (MRT) and
- $0.625\text{mm} \times 0.625\text{mm} \times 0.625\text{mm}$ for *X-ray computed tomography* (CT).

The second source of imprecision is the measuring device that is used to extract features from the operation theatre space. These are optical or electromagnetic devices that capture the position and orientation of a device within a specific spacial frame. A typical device (e.g., the one used for the empirical studies presented in Chapter 7) is the miniBIRD tracker (Ascension Technology Corp., Burlington, Vermont, USA), which has an average accuracy of 1.8mm.

Besides these technical errors, there are other influences that alter the representation of the input. Electromagnetic fields for example significantly influence the imaging process (causing fractals or deformations) as well as the precision of electromagnetic tracking devices.

The fixation of the patient in the operation theatre space exposes the patients head to pressure that slightly deforms the skull. As this deformation is present in the operation theatre but not during the imaging process, the surface $\mathcal{S}$ and the patients head cannot be brought in complete congruence by a rigid motion or an affine map.

1.5 Goals and Classification of This Thesis

In this thesis, we investigate the following hypothesis:

> Are hybrid registration methods appropriate techniques to solve real life registration problems as they occur in the context of navigated surgery?

Hybrid registration strategies follow a purely geometric approach to solve the problem and differ in several ways from the known and currently applied techniques.

Guarantee Bounds

Most point-to-surface registration techniques, such as ICP based strategies [42, 5], compute a solution by descending into a local minimum of the objective function. Even if several initial positions are tested, it is neither guaranteed that the actual global minimum is found nor that there is any relation (additive or multiplicative ratio) between the computed result and the best possible result.

The hybrid registration algorithms presented here allow either to compute the optimum or they provide guarantee bounds for the computed results. Some registration methods also allow to compute *all* registrations that are below a certain quality threshold.

The quality guarantees for the computed results together with the ability to compute a set of equally good registrations, increase the confidence in a navigation system.

Less Effort for Clinical Personnel

The hybrid approach does not require that artificial markers are affixed to the patient. Fixing fiducials to the patients head is a time consuming process that exposes the patient to additional stress.

As shown by Fitzpatrick et al. [18], the quality of a fiducial based registration depends on the distribution of the fiducials around the area of interest. The imaging process, however, does not allow to place fiducials freely on the patients head, as the patient has to lie as motionless as possible on a bench while the scan is performed.

New Applications Fields

Fixing plateaus for locking fiducials to the patients head limits the time between the imaging process and the actual operation. Due to perspiration or by touching the plateaus, their position may shift or even get lost between the imaging process and the operation which might significantly change the computed result.

For hybrid registrations, it is not necessary the attach anything to the patient or measure anything from the patient before he or she is affixed in the operation field. This allows to apply these methods also for emergency operations where there is typically not enough time to prepare the patient before the imaging process.

The developed strategies allow the use of medical navigation systems also for ambulant otolaryngologic interventions. A patient can be scanned in a clinic where an imaging device is available and can later on be operated ambulatory, even days later if the diagnosis allows that.

Intuitive and Conservative Objective

The objective function of Equation 1.3 that is minimized for hybrid registrations is based on the directed Hausdorff distance. In contrast to the root mean square distance, the value of the Hausdorff distance has a one-to-one correspondence to actual features in the operation space: it is the (Euclidean) distance of the worst matched feature either to the surface (if the feature is a surface point) or to a landmark (if the feature itself is a landmark).

1.6 Organization of This Thesis

In **Chapter 2**, surprisingly simple constant-factor approximations are presented for a special subproblem of rigid motion registrations: computing a rotation around a fixed axis that minimizes the Hausdorff distance of a point set to a set of lines, line segments or triangles in the plane or in $\mathbb{R}^3$.

In **Chapter 3** two hybrid rigid motion registration strategies are presented. These two algorithms compute fast and robust absolute-error approximations for settings in which two anatomic landmarks are available. Two different discretization techniques are used, based on whether or not the defined anatomic landmarks are available already at the pre-processing phase.

This research has partially been published in [14].

In **Chapter 4** approximation schemes and heuristics are introduced for the computationally difficult task of computing hybrid registrations with just a single given anatomic landmark. Here, rigid motion registrations are considered that minimize the Hausdorff distances of the involved geometric features to surfaces that are given as point clouds or as triangulated surfaces.

An early version of this research has partially been published in [15].

In **Chapter 5** the *weighted directed Hausdorff distance* is introduced along with matching algorithms that minimize this measure. Here, translations and rigid motions of point sets in arbitrary dimensions are considered. The weighted directed Hausdorff distance allows to take the different precision levels into account with witch the different feature types can be measured.

This research has been published in [27, 28, 29].

A novel matching concept named *non-uniform matchings* is presented in **Chapter 6**. This approach allows to compute a *set* of transformations instead of just a single transformation to perform the mapping from the pattern space to the model space. Here, the pattern space is partitioned into areas of interest. A computed transformation is only valid within a cell of this partition. The transformations have to satisfy two competing targets: they have to match the geometric features of a cell close to their counterparts in the model. Simultaneously, transformations for neighboring cells have to be *similar* to guarantee a smooth mapping.

Computing a set of transformations reduces the effect of local deformations. This is an essential step towards developing registration algorithms for the clinically highly relevant application of performing navigated soft tissue surgeries.

A subset of an early version of this research has been published in [30].

In the practical part of this thesis, in **Chapter 7**, a study in cooperation with the *Klinik und Hochschulambulanz für Neurochirurgie, Charité-Universitätsmedizin Berlin* and the industry partner *Prosurgics Ltd.* is presented in which the different imprecision levels of measuring surface points and anatomic landmarks are theoretically and empirically analyzed. From this study, adjustment parameters are deduced that are essential for all hybrid registration methods that are based in these feature types.

A first draft of this study was published as [12].

Finally, a framework is presented in **Chapter 8** in which most of the hybrid registration algorithms have been implemented.

Part I

Theory

Chapter 2

Constant-Factor Approximations for Point-to-Line and Point-to-Plane Registrations

In this chapter, a simple matching strategy for the following settings is presented and analyzed

1. matching a point set P to a set L of lines or line segments in the plane,
2. matching a point set P to a set Q of planes in $\mathbb{R}^3$.

The transformation class that is considered here are rotations around a fixed center p_r for matchings in the plane and rotations around a fixed axis r for matchings in $\mathbb{R}^3$.

Motivation

Most strategies and algorithms that are presented in the following chapters deal with rigid motion registrations. A rigid motion transformation – the result of a rigid motion registration – can be seen as the joined application of a translation and a rotation. Sometimes it is possible to compute a matching in two phases where the two components of a rigid motion are handled separately, see Chapter 3.

The results presented here lead towards rigid motion registrations by giving insights in how to approximate the rotational part of a rigid motion registration. The discussed algorithms compute matchings whose cost APP is guaranteed to be at most four times the cost OPT of an optimal solution.

2.1 Problem Definition

In the following we will denote with t_α a rotation around the respective rotation center by an angle of α. First, a planar point-to-line(-segment) registration is studied:

Problem 1. *Given a set P of k points, a rotation center p_r and a set L of n lines (line segments) in the Euclidean plane, compute the set A_{OPT} of rotation angles so that the rotation of P around p_r by an angle of $\alpha \in A_{\text{OPT}}$ minimizes its directed Hausdorff distance to L:*

$$A_{\text{OPT}} := \arg \min_{\beta \in [0,2\pi)} \vec{h}\,(t_\beta(P), L)\,.$$

The problem is then extended to point-to-plane registrations in $\mathbb{R}^3$:

Problem 2. *Given a set P of k points, a rotation axis r and a set Q of n planes in Euclidean space $\mathbb{R}^3$. Compute the set A_{OPT} of rotation angels so that the rotation of P around r by an angle of $\alpha \in A_{\text{OPT}}$ minimizes its directed Hausdorff distance to Q:*

$$A_{\text{OPT}} := \arg \min_{\beta \in [0,2\pi)} \vec{h}\,(t_\beta(P), Q)\,.$$

2.2 Computing Exact Solutions

The set A_{OPT} can be computed exactly by applying the theory of Davenport-Schinzel sequences [43]. For each point $p \in P$ and each line $l \in L$ let

$$f_{p,l}(\alpha) := \vec{h}\,(t_\alpha(p), l)$$

denote the distance of $t_\alpha(p)$ to its closest point on l for $\alpha \in [0, 2\pi)$. The lower envelope $f_p(\alpha)$ of the functions is defined as

$$f_p(\alpha) := \min_{l \in L} \left(f_{p,l}(\alpha)\right)$$

and gives for each point p and angle α the Euclidean distance of $t_\alpha(p)$ to its closest point on its closest line. The angle set A_{OPT} consists of all global minima of the upper envelope over all functions f_p for $p \in P$:

$$A_{\text{OPT}} := \arg \min_{\alpha \in [0,2\pi]} \max_{p \in P} f_p(\alpha).$$

Using the strategy presented in [43], these minima can be computed in $\mathrm{O}\left(kn\, 2^{\varphi(kn)} \log kn\right)$ time, where $\varphi(\cdot)$ is the inverse Ackermann function.

The algorithms that are presented here compute 4-approximations and have a runtime of $\mathrm{O}\left(k^2\, n^2\right)$, see also Section 2.6. The advantage of these strategies over the exact and theoretically faster algorithms is that no complicated operations are involved in the computation and therefore it is very simple to implement these algorithms.

2.3 Rotations in the Plane

The central idea of the approximation algorithms is based on so-called *critical points* and *critical angles*. Rotating a point $p \in P$ around the rotation center p_r induces a trajectory tr_p. For a single point p and a fixed line $l \in L$ *critical points* are defined as those points on tr_p that minimize the directed Hausdorff distance to l, see Figure 2.1.

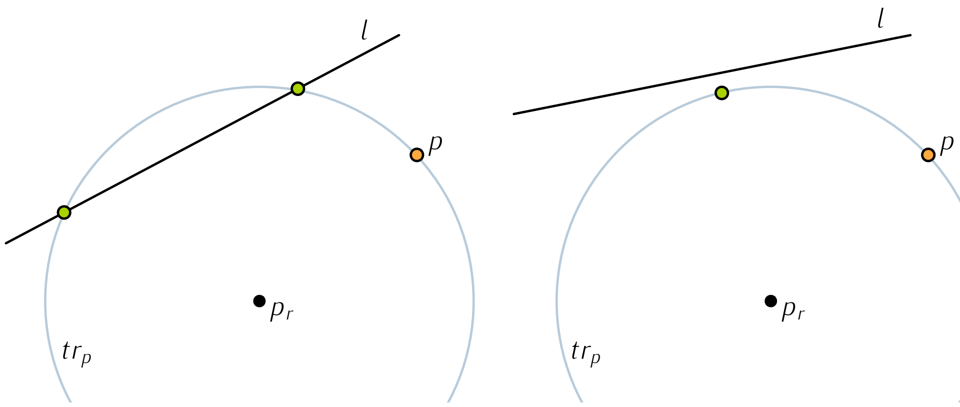


Figure 2.1: All closest points on tr_p to l are critical points.

Let $E_{p,l}$ be the set of *critical angles* for a point p and a line l which are all angles α for which $\vec{h}\,(t_\alpha(p), l)$ is minimal.

$$E_{p,l} \quad := \quad \arg \min_{\alpha \in [0,2\pi)} \vec{h}\,(t_\alpha(p), l)$$

Note that the union $E_{P,L}$ of all critical angles consists of $\mathrm{O}\,(kn)$ angles:

$$E_{P,L} \quad := \quad \bigcup_{p \in P \;\; l \in L} E_{p,l}.$$

2.3.1 The Approximation Algorithm

By construction, rotating the point set P around p_r by a critical angle $\alpha \in E_{P,L}$ moves at least one point in P onto a critical point. Let OPT be the Hausdorff distance of P in optimal position with respect to L.

Theorem 1. *There is an angle $\alpha \in E_{P,L}$ that realizes a 4-approximation of* OPT*:*

$$\exists \alpha \in E_{P,L} : \; \vec{h}\,(t_\alpha(P), L) \leqslant 4\,\mathrm{OPT}\,.$$

Proof. Note that OPT $= 0$ implies APP $= 0$, as every point of P in optimal position lies on a critical point. Suppose otherwise that OPT > 0 and that P is in optimal position, i.e., the distance of each point $p \in P$ to its closest line in L is at most OPT. Let α be the smallest angle by which P has to be rotated in either direction such that an arbitrary point $p \in P$ moves upon one of its critical points $e_p = t_\alpha(p)$ (see Figure 2.2). We now show that such a rotation does not move any other point in P further than $4\,$OPT away from its (initially) closest line. Let l' be the closest line of an arbitrary point $p' \in P \backslash \{p\}$ and let α' be the smallest angle which rotates p' onto one of its critical points $e_{p'}$. By construction, the following inequalities hold:

$$\begin{aligned} \vec{h}\left(p', l'\right) &\leqslant \textsc{opt} \\ \alpha &\leqslant \alpha'. \end{aligned} \tag{2.1}$$

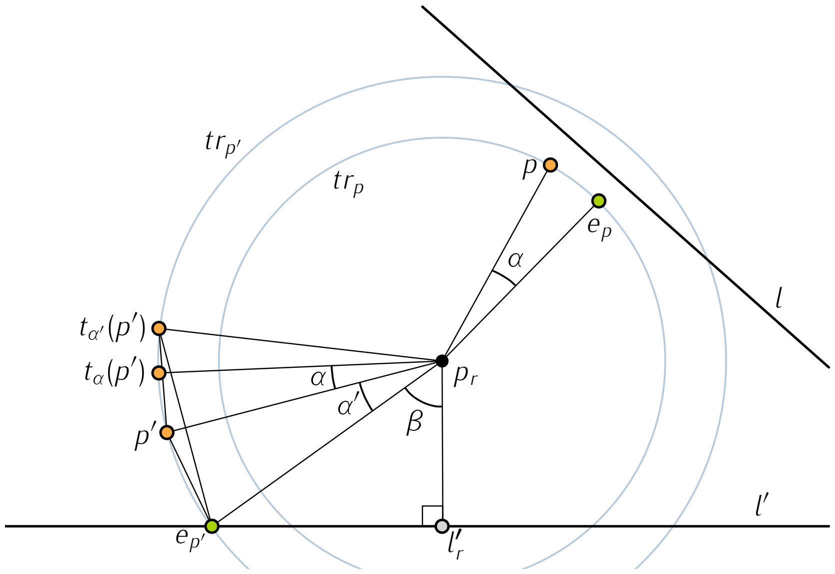


Figure 2.2: Construction for the proof of Theorem 1.

We have that the inner angles of the isosceles triangle $\Delta(p', e_{p'}, p_r)$ that are not incident to p_r are $(\pi - \alpha')/2$, see Figure 2.2. Let $\beta = \angle(e_{p'}, p_r, l'_r)$ be the angle at p_r in the triangle $\Delta(e_{p'}, p_r, l'_r)$ where l'_r is the closest point on l' to p_r. The distance $\vec{h}\,(p', l')$ can be expressed as

$$\begin{aligned} \vec{h}\left(p', l'\right) &= c \sin\left(\pi - \left(\tfrac{\pi}{2} - \beta\right) - \left(\tfrac{\pi - \alpha'}{2}\right)\right) \\ &= c \sin\left(\beta + \tfrac{\alpha'}{2}\right). \end{aligned}$$

Where $c := \|p' - e_{p'}\|$. For small α we can assume by Equation (2.1) that

$$\vec{h}\left(t_\alpha(p'), l'\right) \leqslant \vec{h}\left(t_{\alpha'}(p'), l'\right).$$

By triangle inequality, we get that

$$c_{\alpha'} := \|t_{\alpha'}(p') - e_{p'}\| \leqslant 2c.$$

The distance $\vec{h}\,(t_{\alpha'}(p'), l')$ can be described as

$$\begin{aligned} \vec{h}\left(t_{\alpha'}(p'), l'\right) &= c_{\alpha'} \sin\left(\pi - \left(\tfrac{\pi}{2} - \beta\right) - \left(\tfrac{\pi}{2} - \alpha'\right)\right) \\ &= c_{\alpha'} \sin\left(\beta + \alpha'\right) \end{aligned}$$

With these preliminary observations we are able to compare the distance of p' to l' before and after rotating p' by an angle of α:

$$\begin{aligned}\vec{h}\left(t_\alpha(p'), l'\right) &\leqslant \vec{h}\left(t_{\alpha'}(p'), l'\right)\\ &= c_{\alpha'} \sin\left(\beta+\alpha'\right)\\ &\leqslant 2c\sin\left(\beta+\alpha'\right)\\ &= 2c\sin\left(\left(\beta+\frac{\alpha'}{2}\right)+\frac{\alpha'}{2}\right)\\ &= 2c\Big(\sin\left(\beta+\frac{\alpha'}{2}\right)\cos\left(\frac{\alpha'}{2}\right)+\sin\left(\frac{\alpha'}{2}\right)\cos\left(\beta+\frac{\alpha'}{2}\right)\Big) \qquad (2.2)\end{aligned}$$

To simplify equation (2.2) we make use of the following observations:

1. for small values of α we estimate $\cos(\alpha) \approx 1$,
2. leaving out the factor $\cos\left(\beta+\frac{\alpha'}{2}\right)$ does not decrease the sum,
3. as $\beta \in [0, \pi/2)$ we estimate $\sin(\frac{\alpha'}{2}) \leqslant \sin(\frac{\alpha'}{2}+\beta)$.

Therefore we conclude

$$\begin{aligned}\vec{h}\left(t_\alpha(p'), l'\right) &\leqslant 2\,c\left(\sin\left(\beta+\frac{\alpha'}{2}\right)+\sin\left(\beta+\frac{\alpha'}{2}\right)\right)\\ &= 4\,c\sin\left(\beta+\frac{\alpha'}{2}\right)\\ &= 4\,\vec{h}\left(p', l'\right) \leqslant 4\,\text{OPT}\end{aligned}$$

□

2.4 Point-to-Line-Segment Registration

The result for point-to-line registrations can be extended to point-to-line-segment registrations by introducing additional critical points. The new set of critical points is chosen so that for each point in P the closest point in L remains either an internal point or an endpoint of a segment while P is rotated away from optimal position by the smallest critical angle.

Formally, for a point p and a line segment $l = (l_s, l_e)$, let g_l be the line containing l and let $\phi_p\colon [0, 2\pi) \to g_l$ be the function defined as $\phi_p(\alpha) = q$, so that $\|t_\alpha(p) - q\|$ is minimal. That is, ϕ_p maps each angle α to the closest point of $t_\alpha(p)$ on g_l. The set of critical angles $E'_{p,l}$ for p and a line segment l is defined as

$$E'_{p,l} := E_{p,g_l} \cup \left\{\phi_p^{-1}(l_s), \phi_p^{-1}(l_e)\right\},$$

see Figure 2.3.

Let, as before, α be the smallest angle by which P has to be rotated (in either direction) from optimal position such that a point of P is congruent to one of its critical points.

For all points of P whose closest point on their closest line segment l is neither of both endpoints of l, all arguments of the previous section hold. Note, that no point will be rotated over one of its critical points, as α is chosen to be the smallest angle.

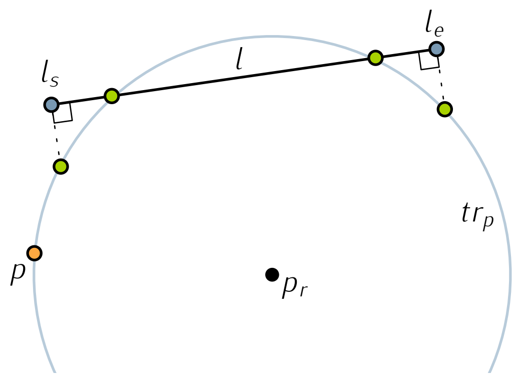


Figure 2.3: For a line segment l, additional critical points for a point p are points on tr_p where an endpoint of the line segment l becomes the closest point on l to p.

It remains to show, that the points $P_e \subseteq P$ whose closest point in optimal position *is* an endpoint of a line segment are not moved away too far when P is rotated by α. For a point $p \in P_e$ let q be the closest endpoint of its closest line segment l and let e_p be the critical point corresponding to $\phi_p^{-1}(q)$ (see Figure 2.4). Two cases have to be considered, depending on the size of the angle $\gamma = \angle(q, e_p, p)$. If $\gamma < \pi/2$ the Hausdorff distance of p to l actually decreases and therefore $\|t_\alpha(p) - l\|$ cannot violate the approximation criterion. Consider the case that $\gamma \geqslant \pi/2$ and let $\alpha' = \angle(p, p_r, e_p)$. By definition of α:

$$\alpha \leqslant \alpha'$$

therefore

$$\|p - t_\alpha(p)\| \leqslant \|p - e_p\|$$

Note that q is the closest point on l for p and $t_\alpha(p)$.

To shorten the next equations we introduce the following abbreviations:

$$x = \|q - e_p\|, \; y = \|p - e_p\|, \; z = \|p - t_\alpha(p)\|.$$

Due to the fact, that $\gamma \geqslant \pi/2$ we can describe $\|p - q\|$ as

$$\begin{aligned} \|p - q\| &= \sqrt{x^2 + y^2 - 2xy\cos(\gamma)} \\ &\geqslant \sqrt{x^2 + y^2} \end{aligned}$$

The distance $\|t_\alpha(p) - q\|$ can be formulated as:

$$\begin{aligned} \|t_\alpha(p) - q\| &\leqslant x + y + z \\ &\leqslant x + y + x + y \\ &= 2(x + y) \end{aligned} \tag{2.3}$$

Here we make use of the following common proposition:

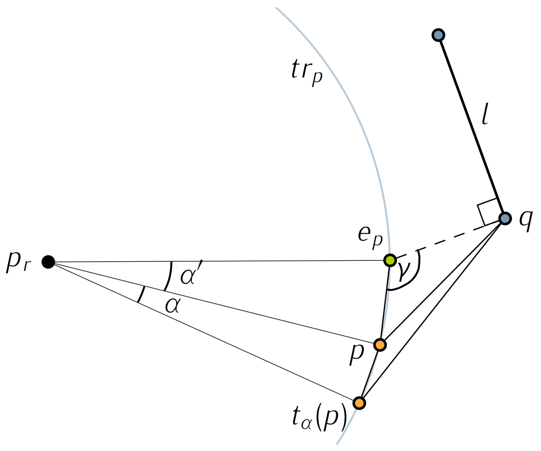


Figure 2.4: Illustration for points p whose closest point is an endpoint of a line segment l.

Proposition 1. *For all $x, y \in \mathbb{R}$:*

$$x + y \leqslant \sqrt{2}\sqrt{x^2 + y^2}$$

Applying Proposition 1 to Equation (2.3) we get:

$$\begin{aligned} \|t_\alpha(p) - q\| \leqslant & \quad 2(x + y) \leqslant 2\sqrt{2}\sqrt{x^2 + y^2} \\ & \leqslant 2\sqrt{2}\|p - q\| < 4 \text{ OPT} . \end{aligned}$$

2.5 Point-to-Plane Registration

In this section, we discuss a 3-dimensional variant of the 2-dimensional point-to-line registration problem.

The general approximation strategy is similar to the approximation algorithm for point-to-line registrations. We first introduce the definition of critical angles for this setting.

For a point $p \in P \subset \mathbb{R}^3$, a rotation axis r and a plane $q \in Q$ the set of critical angels $E_{p,q}$ and the set of all critical angels $E_{P,Q}$ is defined as follows:

$$\begin{aligned} E_{p,q} &:= \arg \min_{\alpha \in [0,2\pi)} \vec{h}\,(t_\alpha(p), q) \\ E_{P,Q} &:= \bigcup_{p \in P \ q \in Q} E_{p,q} . \end{aligned}$$

The critical points of a point $p \in P$ are as before all points $\{t_\alpha(p) \mid \alpha \in E_{p,q}, q \in Q\}$. Let OPT denote the directed Hausdorff distance of P in optimal position to Q.

Theorem 2. *There is an angle $\alpha \in E_{P,Q}$ that realizes a 4-approximation of* OPT*:*

$$\exists \alpha \in E_{P,Q} : \ \vec{h}\,(t_\alpha(P), Q) \ \leqslant \ 4 \text{ OPT} .$$

Proof. Let APP $= \min_{\alpha \in E_{P,Q}} \vec{h}(t_\alpha(P), Q)$ be the cost of the approximation. First we observe that from OPT $= 0$ follows APP $= 0$, as in this case there is at least one angle that moves each point $p \in P$ onto one of its critical points (which lies in one of the planes of Q). As all configurations in which an input point is aligned with its critical point are investigated, the global minima will be tested and found.

If OPT > 0 assume P to be in optimal position, i.e., $\vec{h}(P, Q) =$ OPT. Consider the smallest angle α that rotates (in either direction) a point $p \in P$ onto its critical point. If $\alpha \neq 0$ there must be a point $p' \in P$ whose distance to its closest plane $q' \in Q$ in optimal position increases when being rotated by t_α, i.e., $\vec{h}(p', q') < \vec{h}(t_\alpha(p'), q')$.

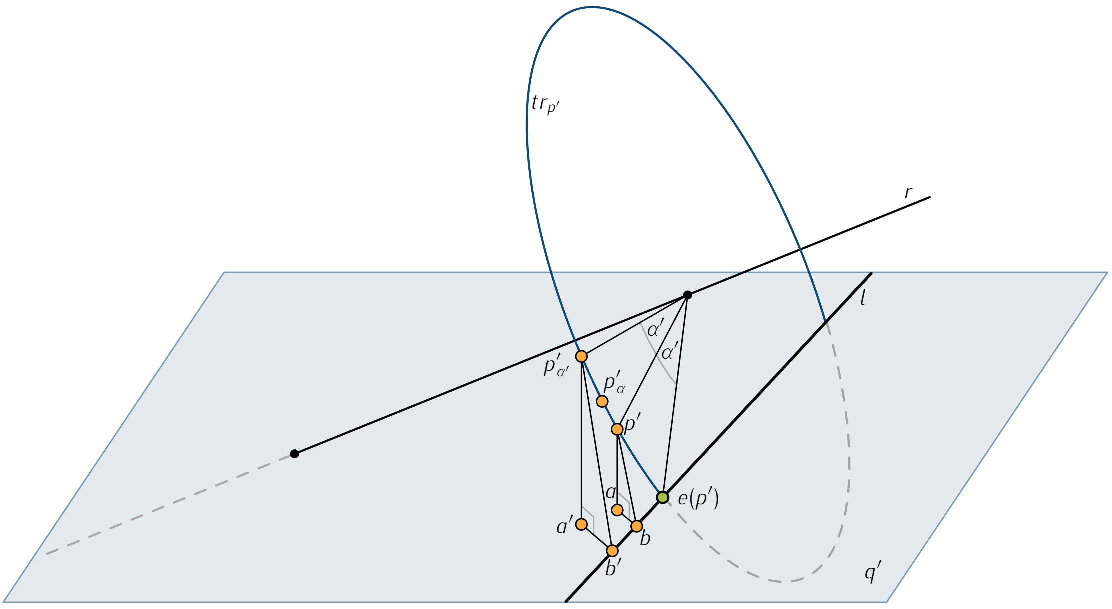


Figure 2.5: Illustration of a point p' whose distance to its initial closest plane q' increases when being rotated by t_α.

Let α' be the angle that rotates p' onto its *closest* critical point $e_{p'}$. As α was chosen to be the smallest angle that rotates a point onto its closest critical point, we have that $\alpha \leqslant \alpha'$ and consequently $\vec{h}(t_{\alpha'}(p'), q') \geqslant \vec{h}(t_\alpha(p'), q')$. We now show that

$$\vec{h}\left(t_{\alpha'}(p'), q'\right) \leqslant 4\,\vec{h}\left(p', q'\right)$$

which implies the theorem, as $\vec{h}(p', q') \leqslant$ OPT.

Let a be the orthogonal projection of p' onto q' and a' the orthogonal projection of $p'_{\alpha'} = t_{\alpha'}(p')$ onto q'. Furthermore, let b be the closest point to a on the line l which is the intersection of q' with the plane that contains the trajectory $tr_{p'}$ (the trajectory of p' when being rotated around r). Let b' be the closest point to a' on l, see Figure 2.5. Note, that that $\vec{h}(p', q') = \|p' - a\|$ and that $\vec{h}(p'_{\alpha'}, q') = \|p'_{\alpha'} - a'\|$

and that the triangles $\Delta(p'_{\alpha'}, a', b')$ and $\Delta(p', a, b)$ have the same angles. Hence we have

$$\frac{\|p'_{\alpha'} - b'\|}{\|p' - b\|} = \frac{\|p'_{\alpha'} - a'\|}{\|p' - a\|}.$$

From the point-to-line registration of the previous chapter we know that

$$\|p'_{\alpha'} - b'\| \leqslant 4\|p' - b\|$$

and therefore $\|p'_{\alpha'} - a'\| \leqslant 4\|p' - a\|$, which completes the proof. $\square$

2.5.1 Remarks

The point-to-plane registration described in Section 2.5 can be extended to point-to-triangle or point-to-rectangle registrations in a similar manner as the point-to-line approximation has been extended to point-to-line-segment registrations.

2.6 Runtime

Each of the k points of P has a constant number of critical points (angels) with each of the n lines, line segments or planes. Each of the $O(k\,n)$ aligning configurations can be tested in $O(k\,n)$ time which results in a total runtime of $O\left(k^2\ n^2\right)$.

In case of non-intersecting line segments L, the Voronoi diagram of L can be computed in $O(n \log n)$ time [3], which reduces the runtime to compute the Hausdorff distance to $O(k \log n)$ time, which results in an overall runtime of $O\left(k^2 n \log n\right)$ for point-to-disjoint-line-segment registrations.

Chapter 3

Absolute-Error Approximations of Semioptimal Point-to-Surface Registrations

Geometric registration problems as they are investigated in this thesis are optimization problems: registrations (in general transformations) have to be computed that minimize a certain objective function. Computing the exact optima for some optimization problems is hard because their corresponding decision problem is NP-hard. Sometimes the best known algorithms are indeed *efficient* from a theoretical point of view but are still too time consuming to be of practical use.

In either case, it is common to develop algorithms that compute solutions that approximate optimal solutions of the original problem. An algorithm $\mathcal{A}$ that minimizes an objective function is called an approximation algorithm if the cost $\textsc{app}$ for a solution computed by $\mathcal{A}$ (the value of the objective function for the computed solution) for any instance of the problem can be bounded from above by a function that depends on the cost $\textsc{opt}$ of an optimal solution and potentially on additional input parameters. From a theoretical point of view, so called *constant-factor approximations* and *(fully) polynomial-time approximation schemes* are of special interest as the computed results have a *relative* dependence on $\textsc{opt}$, i.e., either

$$\textsc{app} \leqslant c \cdot \textsc{opt}$$

for a constant c, or

$$\textsc{app} \leqslant (1 + \epsilon)\ \textsc{opt}$$

for a parameter $\epsilon > 0$. For some real world applications, especially when the input is known to have a finite granularity of a given magnitude (e.g. if the input is based on CT oder MRT images of a known resolution) another type of approximation algorithm is of practical interest: absolute-error approximations. Absolute-error approximation algorithms compute feasible solutions whose cost $\textsc{app}$ satisfies

$$\textsc{app} \leqslant \textsc{opt} + \mu,$$

for a parameter $\mu > 0$.

3.1 Problem Description

In this chapter, we discuss absolute-error approximations for a variant of the hybrid registration problem for exactly two characteristic points: the computation of a semioptimal registration. The objective of a hybrid registration problem consists of two subobjectives:

1. minimizing the distance of the registrated anatomic landmarks to their counterparts in the model,
2. minimizing the distance of the surface points to the model itself.

Formally, for a set of measured characteristic points P_c, their defined counter parts in the model Q_c, a set P of k points arbitrarily measured from the surface and the surface $\mathcal{S}$ itself (given as a set of n triangles or points), the task is to compute a transformation t of a given transformation class $\mathcal{T}$ minimizing

$$\max\left(\vec{h}\left(t(P_c), Q_c\right), \vec{h}\left(t(P), \mathcal{S}\right)\right). \tag{3.1}$$

A transformation $t \in \mathcal{T}$ minimizing Equation 3.1 is called optimal. A transformation t is called *semioptimal* if it minimizes the second term

$$\vec{h}\left(t(P), \mathcal{S}\right) \tag{3.2}$$

given that the first term

$$\vec{h}\left(t(P_c), Q_c\right) \tag{3.3}$$

is minimal. Note that the first subobjective on its own is a bottleneck matching: each measured characteristic point $p \in P_c$ is matched to a *different* defined characteristic point $q \in Q_c$. Let $\mathcal{T}_{sem} \subseteq \mathcal{T}$ be the subset of transformations that minimize the distance of P_c to Q_c (Equation 3.3):

$$\mathcal{T}_{sem} := \arg\min_{t \in \mathcal{T}} \left(\vec{h}\left(t(P_c), Q_c\right)\right).$$

A semioptimal transformation is then any transformation $t \in \mathcal{T}_{sem}$ minimizing Equation 3.2. In case that three or more characteristic points are defined and measured in $\mathbb{R}^3$ that do not lie on a common line, a semioptimal transformation is already defined by the optimal position of P_c.

In this chapter, we discuss two algorithms for computing absolute-error approximations of semioptimal registrations for *exactly two characteristic points*. Let SEM be the cost of a semioptimal registration

$$\text{SEM} = \min_{t \in \mathcal{T}_{sem}} \left(\vec{h}\left(t(P), \mathcal{S}\right)\right),$$

then for any parameter $\mu > 0$, we describe algorithms to compute registrations with a cost APP satisfying

$$\text{APP} \leqslant \text{SEM} + \mu. \tag{3.4}$$

3.1.1 Motivation

Computing the semioptima instead of the optima of a hybrid registration problem with two characteristic points has the advantage that minimizing Equation 3.3 already determines five of the six degrees of freedom of a rigid motion registration in $\mathbb{R}^3$. As described in [13, Lemma 3.6], a semioptimal registration can be used to determine a $(1+\epsilon)$-approximation for any $\epsilon > 0$ by standard discretization techniques.

Semioptimal transformations also qualify as good initial positions for heuristic strategies that descent into local minima, such as simulated annealing [31] or iterative closes point [42] methods.

In settings where the two characteristic points are expected to be defined and measured with a very high precision, semioptimal registrations can be used directly to map points from the pattern space to the model space.

3.1.2 The Searchspace of Semioptimal Registrations

Let $P_c = \{p_1, p_2\}$ and $Q_c = \{q_1, q_2\}$ with $p_1 \neq p_2$ and $q_1 \neq q_2$. As there are just two possible nearest neighbor pairs for the bottleneck matching of P_c to Q_c, we assume that in semioptimal position the closest point in Q_c to p_1 is q_1 and hence the closest point to p_2 is q_2.

By definition, semioptimal registrations have to minimize Equation 3.3, which for rigid motion registrations implies that for all $t \in \mathcal{T}_{sem}$ the points $q_1, q_2, t(p_1)$ and $t(p_2)$ lie on one line and additionally $\|t(p_1) - q_1\| = \|t(p_2) - q_2\|$, see Figure 3.1.

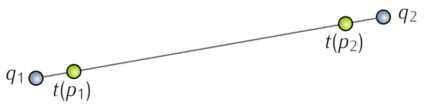


Figure 3.1: Illustration of P_c and Q_c in semioptimal position.

The rigid motions of $\mathcal{T}_{sem}$ can be seen as the successive application of two rigid motions: first a transformation t_{init} that aligns the points $t_{init}(p_1)$ and $t_{init}(p_2)$ with q_1 and q_2 in the way described above and a counterclockwise rotation t_α around the axis defined by $\overline{q_1, q_2}$ by an angle α:

$$\mathcal{T}_{sem} = \{t_\alpha \circ t_{init} \,|\, 0 < \alpha \leqslant 2\pi\}.$$

Computing a semioptimal registration corresponds to the one-dimensional problem of computing the set of angles α such that

$$\vec{h}\left(t_\alpha(\hat{P}), \mathcal{S}\right) \tag{3.5}$$

is minimized for $\hat{P} = \{t_{init}(p) \,|\, p \in P\}$. To compute the exact set $A_{\text{SEM}} \subseteq [0, 2\pi)$ of angles $\alpha \in A_{\text{SEM}}$ so that $t_\alpha \circ t_{init}$ is a semioptimal registration, it is necessary to trace the Hausdorff distance for each $\hat{p} \in \hat{P}$ to $\mathcal{S}$ along its trajectory $t_{\alpha'}(\hat{p})$ for all $\alpha' \in [0, 2\pi)$. Let $f \colon [0, 2\pi) \times \hat{P} \to \mathbb{R}$ be defined as the function $f(\alpha, \hat{p}) := \vec{h}(t_\alpha(\hat{p}), \mathcal{S})$. The semioptimal rotation angles A_{SEM} are the global minima of the upper envelope of all functions $f(\alpha, \hat{p})$ for $\hat{p} \in \hat{P}$:

$$A_{\text{SEM}} := \arg \min_{\alpha \in [0, 2\pi)} \max_{\hat{p} \in \hat{P}} f(\alpha, \hat{p}).$$

The set A_{SEM} can be computed exactly in $O(kn\, \varphi(kn))$ time using the theory of Davenport–Schinzel sequences [43] where $\varphi(\cdot)$ is the inverse Ackermann function. The runtime of this strategy depends on the resolution of the surface as each point-to-surface distance function itself is the lower envelope of $n = |\mathcal{S}|$ point-to-feature functions, where a feature here is either a triangle or a point.

The algorithms that are presented in this chapter compute absolute-error approximations of SEM. The runtime for computing the registrations does *not* depend on the resolution of the surface, except during the (not time-critical) preprocessing phase. The two strategies make use of different discretization techniques whose applicability depends on the information that is available during the preprocessing phase.

In Section 3.2 we present a strategy that needs $\mathrm{O}\left(n/\mu^3\right)$ preprocessing and $\mathrm{O}\left((k/\mu)\log k\right)$ respectively $\mathrm{O}\left((k/\mu)\log k/\mu\right)$ matching time to compute rigid motion registrations, given that only the surface $\mathcal{S}$ is given at preprocessing time.

The algorithm presented in Section 3.3 computes absolute-error approximations in $\mathrm{O}\left(k/\mu\right)$ matching time and needs the same preprocessing time as the previous strategy, given that $\mathcal{S}$ as well as Q_c is given at preprocessing time. Due to the simplicity of the matching process of this method, this algorithm is not only theoretically fast but also fast in practice.

Both strategies allow adding points at a later stage to an already computed solution to increase the accuracy of the result.

3.2 Fast Registration Based on a Cube Discretization

The algorithm that is presented first computes approximative semioptimal registrations and is based on a regular cube decomposition of the model space. In this section, the strategy itself and two implementations of this strategy are discussed. The implementations differ in the way how additional points (measured after the initial matching process is completed) are processed in order to refine an already computed registration.

To ensure that the runtime analysis of this strategy is independent of scaling factors, we assume the model $\mathcal{S}$ to be scaled to fit into the unit cube within the scope of this chapter.

Let G be a subdivision of the unit cube into $O(1/\mu^3)$ subcubes of side length $\mu/\sqrt{3}$, implying that μ is the largest distance between any two points of the same subcube and let $\hat{P} = t_{init}(P)$ for an arbitrary transformation t_{init} minimizing Equation 3.3.

In the preprocessing phase, the distance of each subcube $c \in G$ to $\mathcal{S}$ is set to be the Hausdorff distance of the center x_c of the cube to $\mathcal{S}$. Note, that the distance of any point y that lies within c to $\mathcal{S}$ differs by at most $\|x_c - y\|$ from the Hausdorff distance of x_c to $\mathcal{S}$. To shorten further explanations, the term *the Hausdorff distance of a cell (cube)* will be used instead of *the Hausdorff distance of the center of a cell (cube)*.

The general idea is that during the registration phase, each $\hat{p} \in \hat{P}$ is rotated around the axis $\overline{q_1\, q_2}$ and instead of computing the exact angle-to-surface distance functions for each point, the angles for which $\hat{p}$ enters (or leaves) a subcube of G are collected – together with the Hausdorff distance of the corresponding cells. A set A_{APP} of rotation angles that, together with t_{init}, yield ($\textsc{sem}+\mu$)-approximations is computed by constructing a weighted refinement $\mathcal{A}(\hat{P})$ of the interval $[0, 2\pi)$ that is induced by the collected angles of all $\hat{p} \in \hat{P}$. Each interval of the refined subdivision is weighted by the largest Hausdorff distance of the cube whose corresponding angle range (given by the entering and leaving angles) covers the considered interval of the refinement, see Figure 3.2. The union of the intervals in the refinement with the smallest weight is the sought set A_{APP} of rotation angles .

Formally, for a point $\hat{p} \in \hat{P}$ let $\mathcal{A}(\hat{p})$ be the partition of $[0, 2\pi)$ into subintervals $[\alpha_{in}(\hat{p}, c), \alpha_{out}(\hat{p}, c))$, where $\alpha_{in}(\hat{p}, c)$ is the angle for which $t_{\alpha_{in}(\hat{p},c)}(\hat{p})$ enters the cell $c \in G$ and $\alpha_{out}(\hat{p}, c)$ is the angle for which $t_{\alpha_{out}(\hat{p},c)}(\hat{p})$ leaves c; for all $\alpha \in [\alpha_{in}(\hat{p},c), \alpha_{out}(\hat{p},c))$ it holds that $t_\alpha(\hat{p}) \in c$. Note that there can be more than one angle interval for a point and a cell as the trajectory of $\hat{p}$ can intersect a single subcube more than twice (at most 12 times). The weight $w(i)$ of an interval $i \in \mathcal{A}(\hat{p})$ is the Hausdorff distance of the corresponding cell c to $\mathcal{S}$ as computed in the preprocessing phase. The arrangement $\mathcal{A}(\hat{P})$ is the refinement of $[0, 2\pi)$ induced by all intervals $\mathcal{A}(\hat{p})$ for all $\hat{p} \in \hat{P}$. The weight $w(j)$ for an interval $j \in \mathcal{A}(\hat{P})$ is given as:

$$w(j) = \max\{w(i) \mid i \supseteq j, i \in \mathcal{A}(\hat{p}), \hat{p} \in \hat{P}\}.$$

A set A_{APP} of angles that approximate SEM up to an additive value of μ is then given as

$$A_{\text{APP}} = \bigcup \arg \min_{j \in \mathcal{A}(\hat{P})} w(j).$$

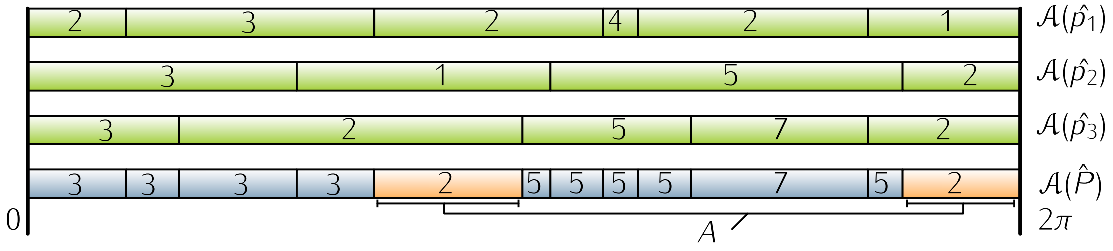


Figure 3.2: The refined subdivision $\mathcal{A}(\hat{P})$ of the interval sets for a set $\hat{P} = \{\hat{p}_1, \hat{p}_2, \hat{p}_3\}$.

Theorem 3. *The angles $\alpha \in A_{\text{APP}}$ imply registrations that are* (SEM $+\mu$)*-approximations of* SEM*:*

$$\forall \alpha \in A: \ \vec{h}(t_\alpha \circ t_{init}(P), \mathcal{S}) \leqslant \text{SEM} + \mu.$$

Proof. For any $\alpha \in A_{\text{APP}}$, let B_α be the set of all subcubes of G that contain at least one point of $\hat{P}$ after being rotated by an angle of α around $\overline{q_1 q_2}$ in counterclockwise direction:

$$B_\alpha = \left\{ c \in G \mid \exists \hat{p} \in \hat{P} : t_\alpha(p) \in c \right\}$$

and let $c_\alpha \in \arg\max_{c \in B_\alpha} \vec{h}(c, \mathcal{S})$ be one of the subcubes of B_α with the largest Hausdorff distance to $\mathcal{S}$.

The Hausdorff distance of a subcube c to $\mathcal{S}$ is the distance of the center x_c of the cube to $\mathcal{S}$. The distance of any point within c to x_c is at most $\mu/2$, hence

$$\forall \alpha \in A_{\text{APP}} \ \forall \hat{p} \in \hat{P}: \ \vec{h}(t_\alpha(\hat{p}), \mathcal{S}) \leqslant \vec{h}(c_\alpha, \mathcal{S}) + \mu/2.$$

In semioptimal position, the point of $\hat{P}$ that is furthest from $\mathcal{S}$ lies within a subcube whose associated distance is at most SEM $+\mu/2$, as do the other points of $\hat{P}$, which implies

$$\forall \alpha \in A_{\text{APP}}: \ \vec{H}\left(t_\alpha(\hat{P}), \mathcal{S}\right) \leqslant \vec{h}(c_\alpha, \mathcal{S}) + \mu/2 \leqslant \text{SEM} + \mu,$$

as A_{APP} is chosen so that $\vec{h}(c_\alpha, \mathcal{S})$ is minimized. $\square$

3.2.1 The Implementation

We present two implementations to compute a set $A_{\text{APP}} \subset \mathcal{A}(\hat{P})$: the first is a standard sweep line approach and the second uses a data structure called *counting segment tree*. Even though the counting segment tree variant is slightly more complex and in theory has a larger matching time, it is shown in Section 3.2.3 that this method outperforms the simple sweep line method in practice. The empirical experiments show a clear speed difference especially when points are added to a computed solution in order to increase its quality.

In the not time critical preprocessing phase, the Hausdorff distance of each of the $O\left(1/\mu^3\right)$ subcubes is naively computed by computing the distance of each cube center to the n features of the surface $\mathcal{S}$.

It is necessary for both implementations to collect the sets of subcubes $C_{\hat{p}}$ that are intersected by the trajectories of all $\hat{p} \in \hat{P}$. Let the cells of G be represented in a three dimensional array. It takes $O(1)$ time to locate the first cell $c_{\hat{p}} \in G$ containing $\hat{p} = t_{init}(p)$ for a point $p \in P$. The succeeding cell of $c_{\hat{p}}$ on the trajectory of $\hat{p}$ around $\overline{q_1\, q_2}$ in counterclockwise direction can be determined in $O(1)$ time by intersecting the six sides of $c_{\hat{p}}$ with the trajectory. The set $C_{\hat{p}}$ is computed by continuing this way until $c_{\hat{p}}$ is reached again. This takes $O(1/\mu)$ time, as $C_{\hat{p}}$ contains at most this many subcubes. While traversing the intersected cells, the extremal angles and also the associated Hausdorff distances of each cell are stored in a linked list for each $\hat{p}$.

The Sweep Line Variant

Here, the subset A_{APP} of the weighted refinement $\mathcal{A}(\hat{P})$ is computed by a sweep over the k interval sets from 0 to 2π. The events that have to be handled are the angles α for which any $t_\alpha(\hat{p})$ lies on the boundary of a subcube of $C_{\hat{p}}$. While processing the individual events, the largest distance value of the corresponding subcubes of the intervals under the sweep line defines the weight of the current cell of the refinement. The Hausdorff distances that are associated with the intervals under the sweep line are organized in a max-heap which is initialized with the distance values of the intervals containing the angle $\alpha = 0$. At start, the event queue consists of the k intervals $[0, \alpha_{out}(\hat{p}, .))$ for $\hat{p} \in \hat{P}$ ordered increasingly by their right limits.

Suppose that $i = [\alpha_{in}(\hat{p}, c), \alpha_{out}(\hat{p}, c))$ is the current element in the event queue. This event is processed by assigning the current top value of the heap to the active cell of the refined arrangement whose right limit becomes $\alpha_{out}(\hat{p}, c)$. The weight $w(i)$ of the event interval is removed from the heap (which is not necessarily the current maximal value of the heap) and the value of the interval that is the successor of i in the corresponding linked list is added to the heap, see Figure 3.3. Also the interval j is inserted into the event queue so that the queue remains ordered increasingly by the right limits of the contained intervals. While constructing the refinement by sweeping over the intervals, the algorithm keeps track of the cells A_{APP} of the arrangement that have the smallest distance value.

The sweep line variant merges all k interval sets in $O(k/\mu \log k)$ time, sweeps through all $O(k/\mu)$ events, and updates the heap in $O(\log k)$ time which results in a total matching time of

$$O\left(\frac{k}{\mu} \log k\right).$$

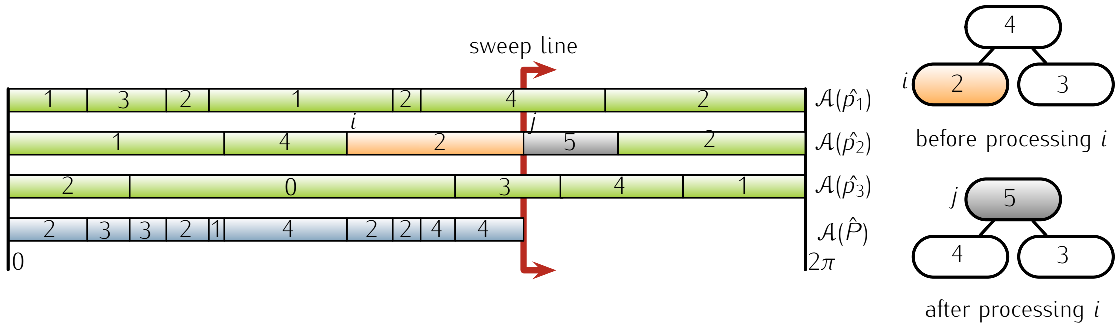


Figure 3.3: **left** current state of the sweep process, **right** heap before and after handling interval i

The Counting Segment Tree Variant

The angle set A can also be computed using a data structure called counting segment tree (cST). A cST is a slightly modified variant of the standard *segment tree* data structure as, e.g., described in the textbook of de Berg et al. [11]. This data structure allows to organize an arrangement of intervals and stores the covering of its cells by intervals that are sequentially added to the tree. The *counting* segment tree additionally keeps track of the number of intervals that cover each cell of the arrangement. The cell t_{max} of the arrangement that is covered by the largest number of intervals can be reported in $O(1)$ time.

To compute the set A_{APP}, all $O(k/\mu)$ angles that move a point $\hat{p} \in \hat{P}$ onto the boundary of a subcube are sorted in increasing order. A counting segment tree T is then initialized based on the refinement induced by this ordered angle set.

In a second step, the union of all collected intervals for all points $\hat{p} \in \hat{P}$ are ordered with respect to the associated Hausdorff distances of the subcubes. Finally, these intervals are added to T in the introduced order.

Let i be the first interval added to the segment tree that causes t_{max} to be covered by k intervals, and let $w(i)$ be the Hausdorff distance of the subcube corresponding to i. The process of inserting intervals is continued as long as the intervals have a distance value equal to $w(i)$. The set of intervals of the counting segment tree that are covered k times is the set A_{APP}. The segment t_{max} after adding i belongs to A_{APP} as it represents the cell of the arrangement that is covered by k segments and that has the smallest largest Hausdorff distance among all sets of k intervals that share a common angle. All intervals that follow i in ordered sequence and that also have an associated Hausdorff distance of $w(i)$ can also cause cells of the arrangement to be covered by k intervals after being added to T. The method that inserts these intervals to T allows to report the cells that are covered k times with no extra cost.

The runtime of the segment tree method is dominated by the time needed to order all intervals with respect to the distance value of their corresponding subcubes, which is also the time needed to build the segment tree. Adding an interval to the tree takes $O(\log(k/\mu))$ time, therefore the run time of this variant is

$$O\left(\frac{k}{\mu}\log\frac{k}{\mu}\right).$$

3.2.2 Increasing the Quality by Adding Points to a Solution

Both implementations have in practice roughly the same matching time, see also Section 3.2.3. In many applications it is possible and desired to refine an already computed solution by measuring additional arbitrary points P'. Both implementation variants allow to add additional points to an existing solution without the need to recompute the entire matching for $P \cup P'$.

Adding Points With the Sweep Line Method

To add an additional point $p' \in P'$ to a computed solution, one performs a new sweep over the weighted arrangement $\mathcal{A}(\hat{P})$ and the intervals encountered by following the trajectory for $\hat{p'} = t_{init}(p')$. The new event queue is initialized with the first cell of $\mathcal{A}(\hat{P})$ and the first interval of $C_{\hat{p'}}$ in the order of their right intervals. As the weights of the new weighted refinement is either defined by a cell of the previous refinement or the intervals defined by $\hat{q}$ it is not necessary to store the Hausdorff distances of the intervals under the sweep line in a heap.

This way, all intervals of the previous arrangement together with the intervals for $\hat{q}$ have to be inspected to find the refined registrations. Apart from the $\log k$ factor needed to maintain the max heap is this method as expensive as restarting the whole sweep line process for $P \cup \{p'\}$.

Formally, all $\mathrm{O}(k/\mu)$ intervals of the refined subdivision and all $\mathrm{O}(1/\mu)$ intervals of the added point are inspected. As both sets are given as linked lists, it takes $\mathrm{O}((k+1)/\mu)$ time to merge them and to update the refinement.

Adding Points With the Segment Tree Variant

To add a point p' to a solution which has been computed with the segment tree method, the intervals corresponding to $C_{p'}$ have to be collected and sorted with respect to their associated Hausdorff distances.

The new registration angles are computed by adding either the first interval of the old ordered interval list whose weight was larger than $w(i)$ or the first interval of the ordered interval list for p' to the tree, depending on which interval has the smaller associated Hausdorff distance. Intervals are added to T based on this criterion as long as after adding an interval i' the first cell of the new arrangement is covered by $k+1$ intervals. As before, this process is continued as long as the succeeding intervals have an associated Hausdorff distance of $w(i')$.

Adding an interval j for the new point p' to the tree might cause that a leaf of T that contains an interval limit of j has to be splitted. This corresponds to refining the underlying arrangement $\mathcal{A}(\hat{P})$. Note, that the segment tree is now not guaranteed to be balanced anymore, but even if all intervals for the new point would cause leaf splits, the height of T increases at most by a factor of two. Using a cST has the advantage, that the time needed to add a point to the solution does not depend linearly but logarithmically on the size of the arrangement of the initially computed arrangement.

Collecting and sorting all intervals for an additional point takes $\mathrm{O}((1/\mu)\log(1/\mu))$ time. Adding a single interval to the segment tree costs $\mathrm{O}(\log(k/\mu))$ time. In the worst case, all intervals that were not added to T during the initial matching process and all new intervals have to be added to find a cell that is covered by $k+1$ intervals. Therefore, the worst case run time for adding a point with this method is $\mathrm{O}((k+1)/\mu\log(k/\mu))$. The experiments, however, show that actually only a small number

	counting segment tree method				sweep line method			
case	MT	1.AP	2.AP	3.AP	MT	1.AP	2.AP	3.AP
1	0.16	0.07	0.07	0.07	0.16	0.3	0.38	0.48
2	0.26	0.09	0.07	0.08	0.25	0.33	0.44	0.55
3	0.05	0.04	0.04	0.04	0.05	0.1	0.16	0.21
4	0.47	0.07	0.03	0.11	0.46	0.53	0.61	0.68
5	0.19	0.04	0.07	0.03	0.2	0.32	0.39	0.45
6	0.47	0.09	0.09	0.07	0.48	0.6	0.7	0.81
7	0.19	0.07	0.08	0.08	0.19	0.33	0.43	0.55
8	0.48	0.07	0.09	0.08	0.48	0.44	0.85	0.99
9	0.35	0.07	0.11	0.09	0.36	0.56	0.69	0.79
10	0.59	0.08	0.09	0.08	0.58	0.71	0.85	0.98
AVG	0.32	0.07	0.07	0.07	0.32	0.42	0.55	0.65

MT=matching time,
i.AP=time for adding the i-th point

Table 3.1: Comparison of of the segment tree and the sweep line method, all values in seconds

of intervals are added to T before a new solution is found.

3.2.3 Evaluation

The two methods have been implemented and compared with respect to their real world performance. The tests were performed on ten test configurations, each consisting of: a model of a skull given as a triangulated surface (consisting of nearly 3000 triangles) scaled to fit into the unit cube, two defined characteristic point pairs and a set P of eight points defined in the pattern space. For all configurations the grid density μ was set to 0.01.

The tests were performed on a 2.33 GHz Intel Core 2 Duo processor computer equipped with 2GB of main memory. The actual running times of the tests are shown in Table 3.1. The matching time (MT) is the time needed to compute the initial solution which is then refined by adding sequentially three points (1.–3. AP) to the solution.

3.3 Fast Registration for a Fixed Rotation Axis

In this section, we discuss an absolute-error approximation algorithm for semioptimal registrations for the case that the model $\mathcal{S}$ and the position of the two characteristic points in model space are given at preprocessing time. Knowing the positions of q_1 and q_2 in model space allows to use a different discretization technique, as the rotation axis around which $\hat{P} = t_{init}(P)$ will be rotated is already known. Instead of computing the Hausdorff distance for a set of subcubes of a grid to the surface $\mathcal{S}$, the distance functions along a set of trajectories around $\overline{q_1\, q_2}$ are computed. To perform the matching, the *closest* trajectories to the points of $\hat{P}$ are determined and their associated distance functions are properly merged to compute rotation angles that result in absolute-error approximations of the semioptimum.

3.3.1 The Approximation Strategy

The central idea to compute a set A_{APP} of angles that approximate A_{SEM} is based on the following simple observation that follows directly from the fact that the Hausdorff distance satisfies the triangle inequality:

Lemma 1. *For all point sets $O \subset \mathbb{R}^3$ and all points $x, x' \in \mathbb{R}^3$ the following inequality holds:*

$$\left| \vec{h}\,(x, O) - \vec{h}\,(x', O) \right| \leqslant \|x - x'\|.$$

The lemma states that if a point x is moved to a position x', the directed Hausdorff distance to a fixed point set O changes by at most $\|x - x'\|$. Or in other words, the directed Hausdorff Distance of a single point x to a fixed object O has a Lipschitz constant of 1 with respect to the Euclidean distance of its first element x.

The Idea

In the preprocessing phase, a set C of trajectories is computed so that the Euclidean distance of $\hat{p} = t_{init}(p)$ for any measurable point p to its closest point

$$\tilde{p} \in \arg\min_{x \in \bigcup C} \|x - \hat{p}\|$$

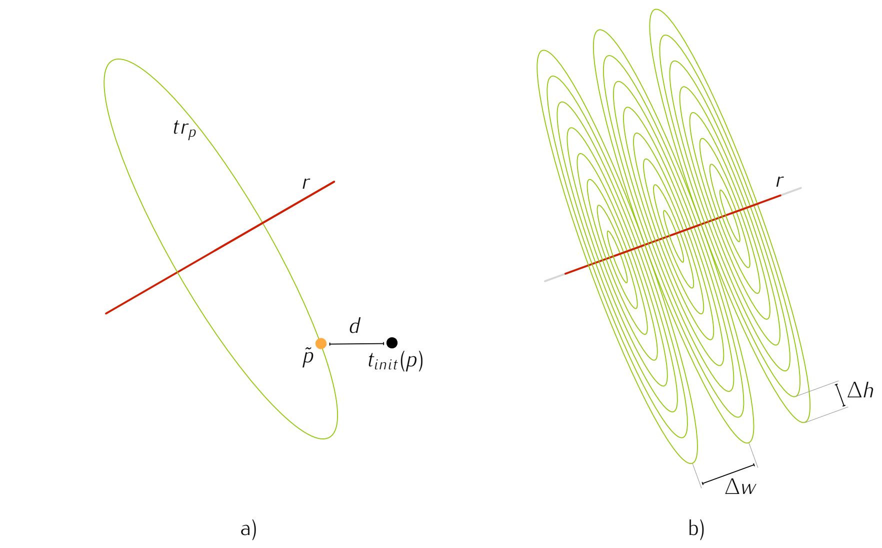
on its closest trajectory $tr_p \in C$ (that is $\tilde{p} \in tr_p$) is bounded from above by a value d, see Figure 3.4 a. The trajectories of C are selected as follows: along the rotation axis $r = \overline{q_1\, q_2}$ trajectory centers c_i are chosen uniformly such that the distance of two consecutive centers is Δw. For each center point c_i trajectories (circles) are generated with radii $j \cdot \Delta h$, $j \in \mathbb{N}$, centered in c_i and with $q_2 - q_1$ as their normal vector, see Figure 3.4 b. How $d, \Delta w$ and Δh have to be chosen will be explained in the following.

To simplify further explanations, we say that all trajectories that have the common center c_i are in the i^{th} *slice* and all trajectories with radius $j \cdot \Delta h$ are in the j^{th} *ring* of C. Let $\tilde{p}$ be the closest point on the trajectory of C that is closest to a measured point $\hat{p} = t_{init}(p)$. Instead of computing the distance function along the trajectory of an actually measured point $\hat{p}$ during the registration phase, the precomputed distance function of the closest trajectory containing $\tilde{p}$ is used to find the proper rotation angles. Due to Lemma 1 the distance function of $f(\alpha, \hat{p})$ stays within a d tube around $f(\alpha, \tilde{p})$, see Figure 3.5.

It follows from this that the function value at the global minima A_{APP} of the upper envelope of the functions $f(\alpha, \tilde{p})$ for all $\hat{p} \in \hat{P}$ differs by at most d from the function value at global minima of the upper envelope of the functions $f(\alpha, \hat{p})$. The transformations that result from first applying the initial transformation t_{init} and then the rotation around r by an angle of A_{APP} satisfy the Equation (3.4) for $d = \mu$.

Simplification by Discretization of the Angle Space

The strategy can significantly be simplified by not storing the continuous distance functions $f(\alpha, \hat{p})$ for each trajectory of C but, instead, replacing it with a piecewise constant distance function $f'(\alpha, \hat{p})$ that sufficiently approximates f. The function f' is defined by the directed Hausdorff distance of sample



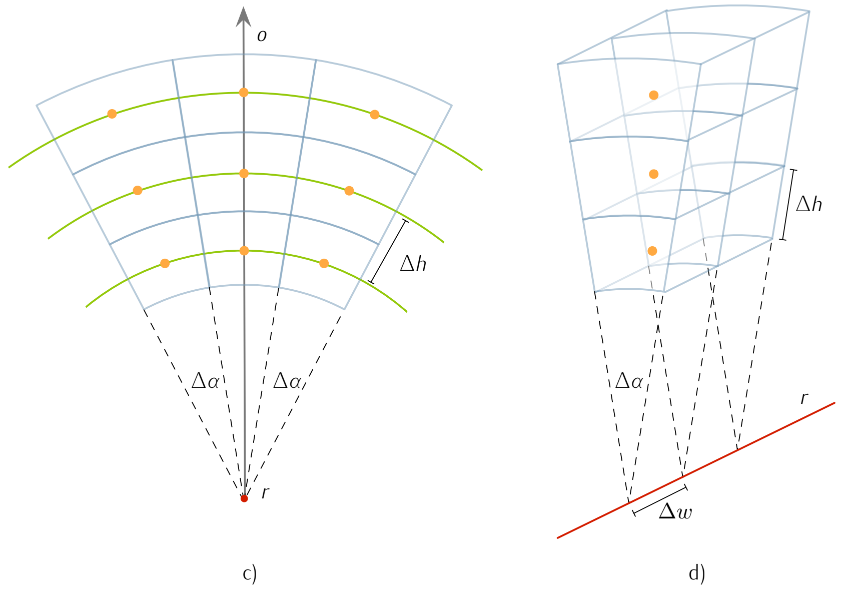


Figure 3.4: **a)** illustration of the closest trajectory, **b)** illustration of the distribution of pre-process trajectories, **c)** front view of a three rings of a single slice, **d)** isometric view of three cells of two consecutive slices.

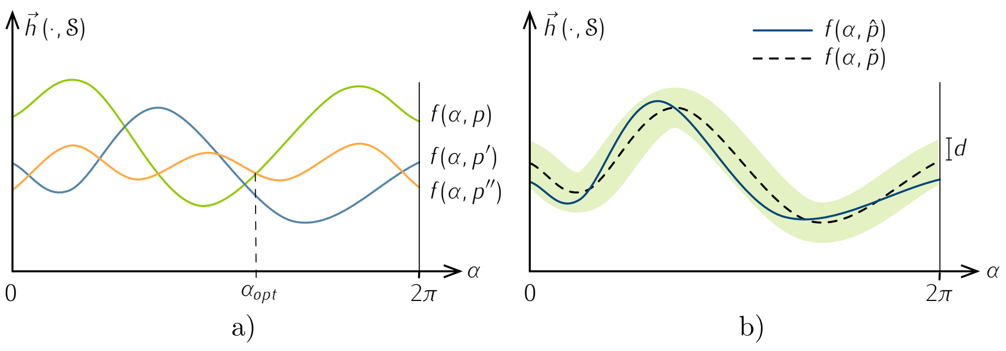


Figure 3.5: **a)** Example of the optimal angle α_{opt} for $t_{init}(P) = \{p, p', p''\}$, **b)** The Hausdorff distance along two trajectories that have a distance of d differs by at most d for all $\alpha \in [0, 2\pi)$.

points that are uniformly distributed along each trajectory. Hence, $f'(\alpha, \hat{p}) = \vec{h}(s, \mathcal{S})$, where $s \in tr_p$ is the sample point closest to $t_\alpha(\hat{p})$.

The sample points are distributed as follows: first a direction o is chosen, such that o is orthogonal to r. On each trajectory $tr \in C$ the extremal point in direction o is chosen to be the first sample point tr_0. The remaining sample points on tr are uniformly distributed with a step length corresponding to an angle difference of $\Delta\alpha$ which gives $tr_i := t_{\Delta\alpha}(tr_{i-1})$ for $1 \leqslant i < \sigma$, where $\sigma := \lceil 2\pi/\Delta\alpha \rceil$ is the number of samples on each trajectory, see Figure 3.4 c. The trajectories and the sample points induce a cell partitioning of the space around the rotation axis r as shown in Figure 3.4 d. All points that fall within one cell of this arrangement have the same closest trajectory and the same closest sample point on this trajectory. Let Δd be the largest possible distance between any point within a cell to the sample point that defines that cell for a specific choice of Δw, Δh and $\Delta\alpha$, see Figure 3.7.

Note, that the described sampling strategy generates the same number of samples on each trajectory – independent of the individual radii of the trajectories. The number of samples that are required to guarantee that largest distance of any point to the closest sample on the closest trajectory is at most Δd is smaller for small radii than for larger radii, given that Δw, Δh are fixed. As shown in the next section, choosing the same number of samples for each trajectory makes the registration process itself fast and almost trivial.

Computing Approximative Semioptimal Rotation Angles

We first describe how a set of angles is computed that imply an absolute-error approximation of a semioptimal registration and then describe how Δd, Δh, Δw and $\Delta\alpha$ have to be chosen so that the resulting registrations are indeed (SEM $+\mu$)-approximations.

Let $P = \{p_1, \ldots, p_k\}$ and let slice$[i]$ be the index of the slice of p_i, which means the closest trajectory to the $\hat{p}_i$ lies within the slice$[i]^{th}$ slice. Analogously, let ring$[i]$ be the index of the ring that contains the trajectory that is closest to $\hat{p}_i$. These indices are computed by projecting $\hat{p}_i$ onto the rotation axis

r to find the closest center (slice) and by computing the Euclidean distance of $\hat{p}$ to its projection on r to find its closest radius (ring).

In the preprocessing phase, the set of piecewise constant distance functions for all sample points of all trajectories were computed. Let dist$[s][r][x]$ be the Hausdorff distance of the x^{th} sample point of the trajectory that is in the s^{th} slice and the r^{th} ring. A set of rotation angles A_{APP} is computed by reporting all global minima of the upper envelope of the distance functions dist[slice$[i]$][ring$[i]$], for all $1 \leqslant i \leqslant k$.

Note, that the measured points $\hat{p}_i$ initially fall into cells that potentially cover different angle ranges, see Figure 3.6.

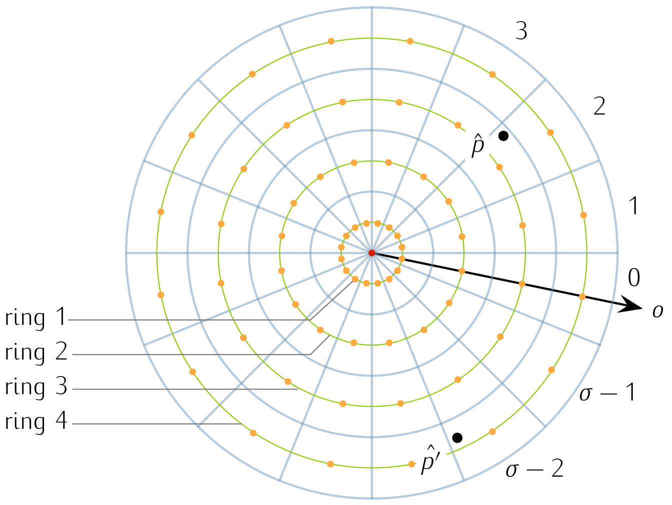


Figure 3.6: Illustration of a single slice of the discretization. Point p falls in the cell belonging to ring 3, sample 3, point p' fall belongs to ring 4 sample $\sigma - 1$.

To compare the proper angle ranges for each point while computing the minima of the upper envelope, another index array offset$[i]$ is needed. The value offset$[i]$ denotes the index of the cell that contains $t_0(\hat{p}_i)$ and is computed by measuring the angle of the vector $\hat{p}_i - z_i$ to the orientation vector o that was chosen in the preprocessing step, where z_i is the orthogonal projection of $\hat{p}$ onto r.

Altogether, the point $\hat{p}_i$ initially lies in slice$[i]$, in ring$[i]$ and in the cell of the offset$[i]^{th}$ sample point on this trajectory.

The upper envelope E is computed for all angle ranges $0 \leqslant x < \sigma$ for $\sigma = \lceil 2\pi/\Delta\alpha \rceil$:

$$E[x] := \max_{1 \leqslant i \leqslant k} \text{dist}[\text{slice}[i]][\text{ring}[i]][(x + \text{offset}[i]) \bmod \sigma]. \tag{3.6}$$

While computing E also its minima A_{APP} are computed and reported as

$$A_{\text{APP}} := \bigcup \{ i\,\Delta\alpha \mid i \in \arg\min_{0 \leqslant x < \sigma} E[x] \}.$$

The corresponding rigid motion registrations are $\{t_\alpha \circ t_{init} | \alpha \in A_{\text{APP}}\}$.

Choosing Proper Constants

By construction we have that a measured point $\hat{p} = t_{init}(p)$ stays within the cells of the sample points $S(tr_p)$ of the closest precomputed trajectory tr_p while being rotated around r. As the distance of any point within a cell to the sample point that defines the cell is at most Δd, we have that the distance function $f(\alpha, \hat{p})$ lies within the Δd-neighborhood of the piecewise constant function $f'(\alpha, \hat{p})$, see Figure 3.7. Consequently, the function value at the global minima of the upper envelope of the functions $f(\alpha, \hat{p})$ for all $\hat{p} \in \hat{P}$ differs by at most Δd from the function value at the global minima of the functions f'.

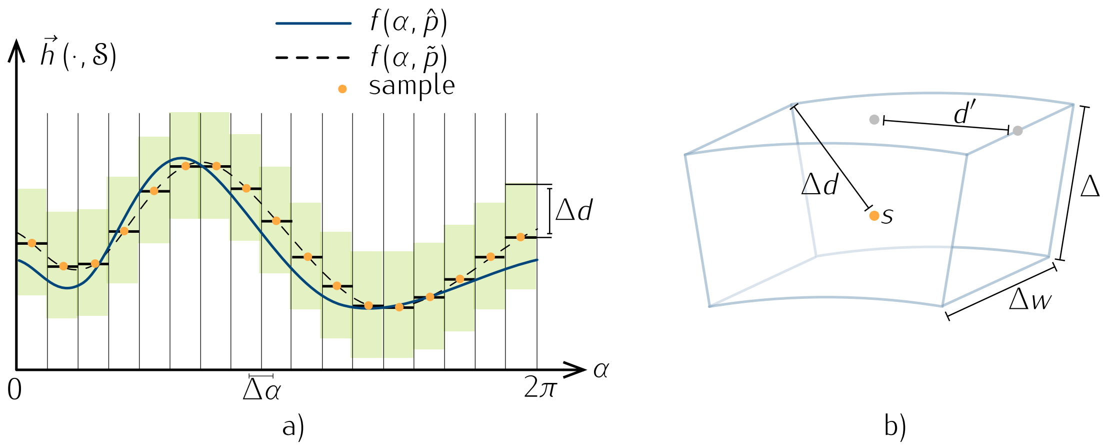


Figure 3.7: **a)** the function $f(\alpha, \hat{p})$ stays within the Δd neighborhood of the piecewise constant function $f(\alpha, \tilde{p})$, **b)** illustration of the furthest distance Δd of a point to a sample and the furthest distance d' a point can be moved by a rotation of an angle of $\Delta\alpha/2$.

The upper envelope E of the piecewise constant functions can be computed by a simple sweep over the individual functions f' (see Equation 3.6) because all distance functions f' are assumed to be aligned with respect to their underlying angle ranges, i.e., all points $t_\alpha(\hat{p})$ for $\hat{p} \in \hat{P}$ are assumed to leave/enter their individual cells for the same angle α. In general this is not the case as can be seen in Figure 3.6, the angle for which $\hat{p}$ leaves its initial cell is smaller than the angle at which $\hat{p'}$ leaves its initial cell.

Alining the functions f' so that the underlying $\Delta\alpha$ ranges are aligned corresponds to a shift of the functions f by an angle of at most $\Delta\alpha/2$. Due to Lemma 1, we have that for all $\alpha \in [0, 2\pi)$ and all $\hat{p} \in \hat{P}$

$$f(\alpha \pm \beta, \hat{p}) \in [f(\alpha, \hat{p}) - d', f(\alpha, \hat{p}) + d'],$$

where $\beta = {}^{\Delta\alpha}/_{2}$ and d' is the largest distance a point $\hat{p}$ can travel when being rotated by an angle of β, hence $d' \leqslant \|t_\beta(\hat{p}) - \hat{p}\|$ for any $\hat{p}$, see Figure 3.7 b. Choosing the values of Δw, Δh, and $\Delta\alpha$ so that $\Delta d + d' \leqslant \mu$ guarantees that the value at the global minima of the upper envelope of the distance functions f differs by at most μ from the value at the global minima of the upper envelope of the functions f'. As $\Delta d > d'$ this can be realized by setting Δd to $\mu/2$. Recall, that the model $\mathcal{S}$ is assumed to be scaled to fit into the unit cube and all points P to be measured within the unit cube. Choosing $\Delta w = \Delta h = \mu/3$ and $\Delta\alpha = \sin^{-1}(\mu/3)$ results in $\Delta d = \mu/2$ and consequently in $d' \leqslant \|\hat{p} - t_\beta(\hat{p})\| \leqslant \mu/2$ for $\beta = \Delta\alpha/2$.

Runtime Analysis

The preprocessing time depends on the number of slices, rings, and sample points σ which in turn is determined by the chosen values for Δw, Δh, and $\Delta\alpha$. The cell partition induced by the set of all samples has to cover the unit cube, as we restricted $\hat{P} \subset [0,1]^3$. In the worst case, the trajectory centers have to be distributed along a line segment of length $\sqrt{3}$ on r and the maximal radius is also bounded by $\sqrt{3}$.

Consequently, $\mathrm{O}(1/\mu)$ slices and $\mathrm{O}(1/\mu)$ rings have to be introduced and each trajectory has to be subdivided into $\mathrm{O}(1/\mu)$ sample points which yields a preprocessing time of $\mathrm{O}\left(n/\mu^3\right)$ if the Hausdorff distance for each sample point to $\mathcal{S}$ is computed by taking the minimal distance to any of the n features of the surface.

The time needed to compute a registration depends on the number of measured points $k = |P|$ and the number of samples $\sigma = \mathrm{O}(1/\mu)$ on each trajectory. A sweep over the k piecewise constant functions f' and hence computing the minima of their upper envelope can be performed in time $\mathrm{O}(k/\mu)$.

Adding Points to a Solution

As for the previously presented strategies, the algorithm presented here allows to increase the precision of the registration by considering additionally measured points. To include an additionally measured point $p' = t_{init}(p)$, the sample of the cell containing p' has to be computed in constant time and the upper envelope as to be updated by sweeping over the previous upper envelope and over the function $f(\alpha, p')$ in $\mathrm{O}(1/\mu)$ time.

Removing Points from a Solution

In some settings it is desirable to be able to remove the influence of certain points of P from a computed registration, e.g., if a point is known to be influenced by significant measurement inaccuracies.

The following modifications of the algorithm allows to remove a single measured point $p \in P$ from the solution in $\mathrm{O}(1/\mu)$ time by slightly increasing the matching time to $\mathrm{O}((k \log k)/\mu)$ and the time needed to insert an additional point to $\mathrm{O}((\log k)/\mu)$:

The upper envelope E consists of $\mathrm{O}(1/\mu)$ constant pieces, the value of each constant piece is defined by the maximum of all k samples at the corresponding angle range. When removing p from a solution, the Hausdorff distances of the sample points on the trajectory closest to p must no longer constitute to the upper envelope.

The following data structures are introduced to support a fast deletion of points: for each of the $\mathrm{O}(1/\mu)$ angle intervals of the upper envelope a balanced search tree U_i is generated. Each search tree stores in its leaves the k distance values of the sample points that correspond to the angle range of that section of the upper envelope. Additionally, the leaves of the search trees are organized in a doubly connected edge list, as illustrated in Figure 3.8. Also, each leaf of U_i stores a pointer to the leaf in U_{i+1} that stores the distance value of the next sample on the same trajectory.

A point is removed by following the pointers through the list of samples of that point. At each sample index i the corresponding pointer to the leaf in the search tree U_i is used to remove this sample from the tree. The new global minima are found, as before, by finding the indices of E where the largest distance

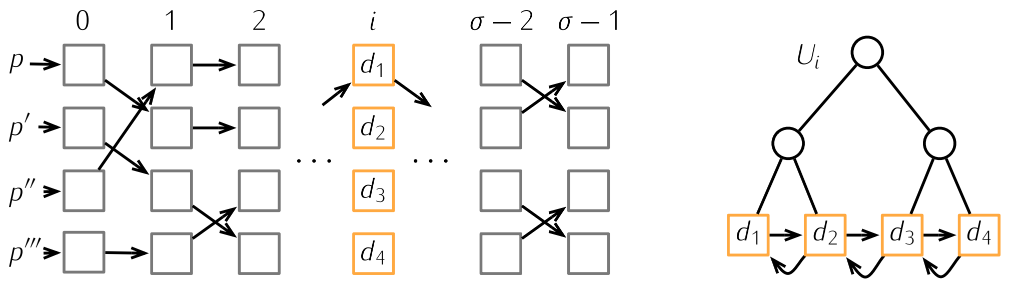


Figure 3.8: Illustration of pointer structure that allows fast removal of points. **left:** the sample sequences for four points, **right:** the search tree U_i for the four samples at index i.

is minimal. As the leaves of the individual search trees are organized in a doubly connected list, this value can be updated in constant time for each index. Search trees are used to store the distance values for the individual angel ranges of the upper envelope to allow the insertion of additionally measured points in $O(\log k)$ (instead of $O(k)$) time for each sample index.

Reducing the Number of Samples in Practice

In the runtime analysis of this algorithm, the trajectory centers are assumed to be distributed along a line segment of length $\sqrt{3}$ on the rotation axis r and he trajectories in the outermost ring are assumed to have a radius of size $\sqrt{3}$ to ensure that $\mathcal{S}$ is completely Δ*d-covered* by the set of all trajectories/samples. To reduce the number of samples that are actually used to compute the registration, the following adjustments can be made.

The radii of the trajectories have to be chosen in a way, such that for any point s of the model $\mathcal{S}$ there is a trajectory that contains a sample point that lies *close* to s. The choice of Δw and Δh only depends on the approximation parameter μ whereas the size of an angle interval $\Delta\alpha$ depends on μ and the radius h_{max} of the trajectories of the outermost ring. This radius can be chosen to be $h_{max} = \min\left(\sqrt{3}, c \cdot \vec{h}(\mathcal{S}, r)\right)$, where $c \geqslant 1$ is a small adjustment parameter (in our experiments chosen to be 1.1). As implied in Figure 3.9, this often reduces the actual number of samples that are considered in the precomputation phase.

The length w_{max} of the line segment along which the centers of the trajectories have to be distributed can be restrained by projecting the corner vertices of the triangles of $\mathcal{S}$ onto r. If w_{max} is chosen to be c times the furthest distance of two vertices projected onto r, it is ensured, that $\mathcal{S}$ is completely enclosed by the trajectories of the first and last slice, see Figure 3.9.

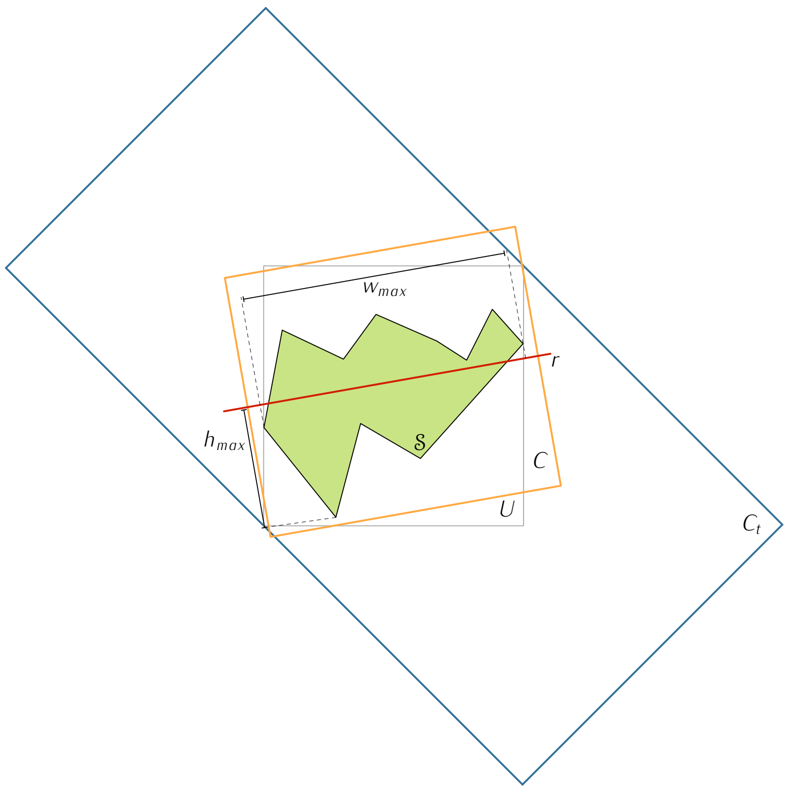


Figure 3.9: Illustration of the difference of the sample range C resulting from w_{max} and h_{max} for a small adjustment parameter c in contrast to the theoretically needed sample range C_t of length and radius $\sqrt{3}$ for a model $\mathcal{S}$ scaled to fit into the unit cube U.

3.4 Evaluation

The presented algorithm has been implemented in C++ and evaluated on a computer equipped with an Intel Core2 Duo processor and 2GB main memory. It took $282\mu s$ ($5.02ms$) on average to compute the approximated registration for a model of about 3000 triangles, an approximation parameter $\mu = 0.01$ ($\mu = 0.004$) and $|P| = 8$ measured points, see Figure 3.10. An approximation parameter of $\mu = 0.01$ ($\mu = 0.004$) corresponds to a physical accuracy 2.5mm (1mm) in the operation field space.

Compared to the hybrid registration algorithm of Section 3.2, this algorithm is about 560 times faster if applied on the same input data for the same parameter set, at the same preprocessing time. The registration is computed only once, in the beginning of the operation and the solution is eventually improved by adding measured points in a later operation phase. Due to the fast registration time, this algorithm meets the requirements for a realtime registration system and could be used in applications where a registration has to be computed for each frame of the visualization system.

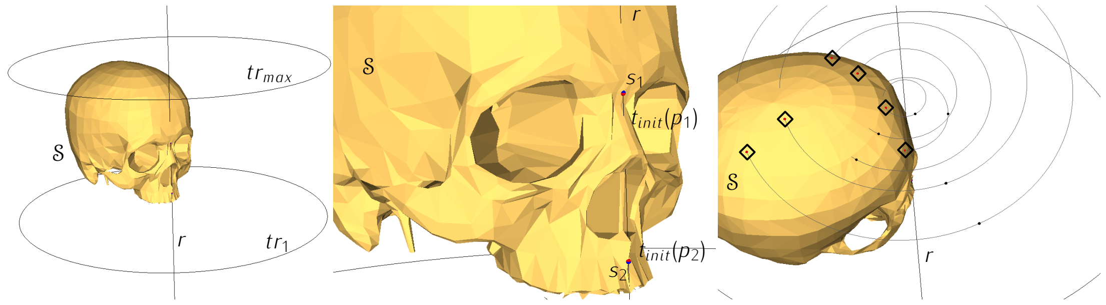


Figure 3.10: **left:** the extremal trajectories in the outermost ring, **center:** the points s_1, s_2 and p_1, p_2 after initial alignment, **right:** the closest trajectories to the measured points and the registrated points (in diamonds).

Chapter 4

Approximate Point-to-Surface Registration with a Single Characteristic Point

The previous chapter, as well as the algorithms presented by Darko et al. [13] and the Diploma thesis of Hartmann [22] deal with hybrid registration methods for settings in which two or more characteristic points are available for computing a registration. In this chapter, we address the problem of computing rigid motion registrations with just a single given characteristic point, i.e., $P_c = \{p\}$ and $Q_c = \{q\}$. More precisely, we present an approximation scheme for the case that the surface is represented by a point set and a $((1+\epsilon)\,\textsc{opt} + \tau)$-approximation (for any $\epsilon, \tau > 0$), where $\textsc{opt}$ is the value of an optimal solution for the case that the surface is represented by trinagles.

Furthermore, we present a general heuristic framework that allows to apply algorithms that are designed for two characteristic points also on input that contains just one measured characteristic point. The general idea of the heuristic is to guess a rotation axis through the proximity of q by picking a second 'virtual' characteristic point $\hat{p}$ in the pattern space and to *guess* its location $\hat{q}$ in the model space. The underlying algorithm for two characteristic points is then called with the original characteristic point and with the virtual characteristic point as input along with the set of surface points. The value of the objective function for this guess of the virtual characteristic point is then used to exclude areas in the model space where no 'better' virtual characteristic points can be found – given that the objective function of the underlying algorithm fulfills certain conditions. This strategy computes registrations that either have a quality of at most a pre-specified absolute parameter or differ at most by a specified value from the best possible choice of a virtual characteristic point. The computed registrations are absolute-error approximations of the solutions of the underlying two-point algorithm for any choice of the position of the virtual point $\hat{q}$ in the model space.

4.1 Problem Definition

The computational complexity of computing a registration with exactly one characteristic point depends on the representation of the surface $\mathcal{S}$. We consider surfaces that are represented as point sets or as triangulated surfaces.

Let $\mathcal{S} \subset \mathbb{R}^3$ be a surface consisting of m points or triangles, representing the anatomic model of the patient in the model space, let $Q_c = \{q\}$ be the characteristic point defined in the model and let $P_c = \{p\}$ be its measured representative in the pattern space. Furthermore, let $P \subset \mathbb{R}^3$ be a point set of n surface points measured from the patient in the pattern space.

Problem 3. *Given $\mathcal{S}, P, p, q$ as above, compute a rigid motion t_{OPT} minimizing*

$$\begin{aligned} \operatorname{dist}(P_c, Q_c, P, \mathcal{S}) &= \max\left(\vec{h}\left(t_{\text{OPT}}(P), \mathcal{S}\right), \vec{h}\left(t_{\text{OPT}}(P_c), Q_c\right)\right) \\ &= \max\left(\vec{h}\left(t_{\text{OPT}}(P), \mathcal{S}\right), \|t_{\text{OPT}}(p) - q\|\right). \end{aligned} \tag{4.1}$$

4.2 FPTAS & Quasi-Approximation Scheme

In this section, we first present an FPTAS for surfaces represented as point sets and then extend this result to a quasi-FPTAS for triangulated surfaces.

Definition 6 (quasi-FPTAS). *A quasi-FPTAS is an algorithm that computes for arbitrary $\epsilon, \tau > 0$ a solution whose cost* APP *fullfills*

$$\text{APP} \leqslant (1+\epsilon)\ \text{OPT} + \tau$$

and that has a runtime that is polynomial in the input size, in ϵ and in τ, where OPT *is the cost of an optimal solution.*

A quasi-FPTAS approximation combines relative and absolute-error approximation aspects.

4.2.1 Approximation Scheme for Point Sets

We first assume the surface to be given as a point set $\mathcal{S} = \{q_1, \dots, q_m\}$. The approximation scheme is based on a constant-factor approximation which, combined with a discretization technique, yields an FPTAS.

Let $\hat{p} \in P$ be a point of the set P of surface points that has the largest distance to p, i.e., $\hat{p} \in \arg\max_{p' \in P} \|p - p'\|$. Assume p and P to be in optimal position, i.e., the transformation t_{opt} that is to be computed for Problem 3 is the identity, i.e., for all $x \in \mathbb{R}^d$ it holds that $t_{opt}(x) = x$. Let OPT be the cost of an optimal solution, and let $\hat{q} \in \mathcal{S}$ be the closest point of $\mathcal{S}$ to $\hat{p}$. Consider the translation $t_1 = q - p$ that moves p onto q and the rotation t_2 around q (by the smallest angle) so that the points $q = t_1(p) = t_2 \circ t_1(p)$, $\hat{q}$ and $t_2 \circ t_1(\hat{p})$ are aligned, see Figure 4.1.

Lemma 2. *The registration $t = t_2 \circ t_1 \circ t_{opt}$ is a $2(1+\sqrt{2}) \approx 4.828$-approximation to* OPT, *that is,*

$$\operatorname{dist}(\{t(p)\}, \{q\}, t(P), \mathcal{S}) \leqslant 2(1+\sqrt{2})\ \text{OPT}.$$

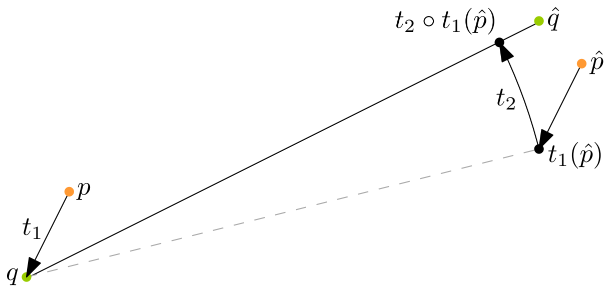


Figure 4.1: Transformation $t_2 \circ t_1$ aligns the points q, $\hat{q}$, and $\hat{p}$ and places p upon q.

Proof. We assumed p and P to be in optimal position, therefore $\|p - q\| \leqslant \text{OPT}$. Applying t_1 to P moves every point at most OPT away from its nearest neighbor in $\mathcal{S}$. The rotation t_2 around q rotates $t_1(\hat{p})$ by the smallest angle $\alpha \leqslant \pi/2$ onto the line through q and $\hat{q}$. As $\|t_1(\hat{p}) - \hat{q}\| \leqslant 2\ \text{OPT}$ and $\alpha \leqslant \pi/2$ we have that $\|t(\hat{p}) - t_1(\hat{p})\| \leqslant 2\sqrt{2}\ \text{OPT}$, see Figure 4.2. As $\hat{p}$ is the furthest point in P to p any other point, $p' \in P$ is moved away by at most $2\sqrt{2}\ \text{OPT}$, hence:

$$\forall p' \in P : \|t(p') - p'\| \leqslant \|t_1(p) - p\| + \|t(\hat{p}) - t_1(\hat{p})\| \leqslant (1 + 2\sqrt{2})\ \text{OPT},$$

which, as the distance of p' to its nearest neighbor $q' \in \mathcal{S}$ is bounded by OPT, results in a $2(1 + \sqrt{2})$-approximation.

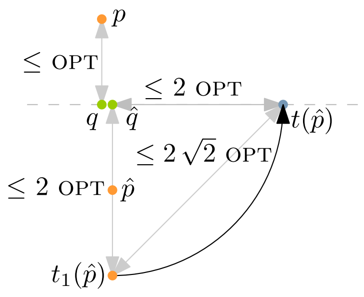


Figure 4.2: Applying t_2 to $t_1(\hat{p})$ moves any $p' \in t_1(P)$ by at most $2\sqrt{2}\,\text{OPT}$ to $t_2(p')$.

□

Corollary 1. *For a given $\hat{q}$, let $\mathcal{T}'$ be the set of rigid motions t so that $t(p) = q$ and $t(\hat{p}), q$ and $\hat{q}$ are aligned. Any Transformation $t_{sem} \in \mathcal{T}'$ that minimizes*

$$\text{dist}\left(\{t_{sem}(p)\}, \{q\}, t_{sem}(P), \mathcal{S}\right) \tag{4.2}$$

is called a semioptimal *registration and is a $2(1 + \sqrt{2})$-approximation to* OPT.

For a given $\hat{q}$, the set of transformations that minimize Equation 4.2 can be computed by applying the theory of Davenport-Schinzel sequences [43] as described by Darko et al. [13, Lemma 3.2] in a runtime of $O\left(mn\, 2^{\varphi(mn)} \log mn\right)$ where $\varphi(\cdot)$ is the inverse Ackerman function. As the point $\hat{q}$ closest to $\hat{p}$ in optimal position is unknown, all m points of $\mathcal{S}$ have to be tested, which increases the runtime to $O\left(m^2 n\, 2^{\varphi(mn)} \log mn\right)$ for computing the constant-factor approximation.

The Approximation Scheme

Lemma 2 states that applying the transformation $t = t_2 \circ t_1$ to p and P in optimal position results in a registration that has a cost APP of at most $2(1+\sqrt{2})$ OPT. The key idea of the approximation scheme is to build a spherical grid G_2 around the axis $\overline{q\,\hat{q}}$ centered in q with radius $\|p-\hat{p}\|$ whose width and granularity depends on APP, see Figure 4.3. By properly choosing the granularity of the grid, it is ensured that there is one transformation $t_2'^{-1}$ among all rotations that move $t(\hat{p})$ onto a vertex of the grid that approximates t_2^{-1} in the sense that $\|t_2'^{-1}(t(\hat{p})) - t_2^{-1}(t(\hat{p}))\| \leqslant \mu$ for a given parameter $\mu > 0$ whose size is to be specified.

The translational component t_1^{-1} of t^{-1} is approximated by testing all translations that move $t(p) = q$ onto a vertex of a cubic grid G_1 centered in q with radius $\|p-\hat{p}\|$ whose width and granularity again depend on APP. A proper choice of the cell width of the second grid guarantees that there is a translation $t_1'^{-1}$ that approximates t_1^{-1} so that

$$\begin{aligned}
\|t_1'^{-1}(t_2'^{-1} \circ t(p)) - t_1^{-1}(t_2^{-1} \circ t(p))\| &= \|t_1'^{-1}(t(p)) - t_1^{-1}(t_1(p))\| \\
&= \|t_1'^{-1}(t_1(p)) - p\| \\
&\leqslant \mu.
\end{aligned}$$

This in turn implies that there is a combination $t'^{-1} = t_1'^{-1} \circ t_2'^{-1}$ of transformations so that

$$\max\left(\|t'^{-1}(t(p)) - p\|, \vec{h}\left(t'^{-1}(t(P)), P\right)\right) \leqslant 2\mu.$$

To achieve a $(1+\epsilon)$-approximation, we choose

$$\mu = \frac{\epsilon \text{ APP}}{4(1+\sqrt{2})}.$$

The grid G_2 has to fulfill the property that the distance of the closest grid point $g \in G_2$ to $t_1(\hat{p})$ is bounded by $\|g - t(\hat{p})\| \leqslant \mu$. Choosing an opening angle of

$$\alpha = \min\left(\pi, 4\sin^{-1}\left(\frac{\text{APP}}{2\|p-\hat{p}\|}\right)\right)$$

and an angel step width of

$$\beta = 2\sin^{-1}\left(\frac{\sqrt{2}\,\mu}{2\,(1+\sqrt{2})\,\|p-\hat{p}\|}\right)$$

ensures this property and results in $O\left({}^1\!/\!_{\epsilon^2}\right)$ grid points for G_2, see Figure 4.3.

The grid G_1 has to provide a grid point g at distance at most μ to p. Choosing the width of the cubical grid as $2\,$APP and the edge length of each cell of G_1 as $w = {}^2\!/\!_{\sqrt{3}}\mu$ results in a set of grid points that fulfills this property, see Figure 4.3. The grid G_1 has $O\left({}^1\!/\!_{\epsilon^3}\right)$ grid points.

Theorem 4. *For any $\epsilon > 0$, a $(1+\epsilon)$-approximation of* OPT *for Problem 3 can be computed in*

$$O\left(\frac{m^2 n}{\epsilon^5}\; 2^{\varphi(mn)} \log mn\right)$$

time, provided that $\mathcal{S}$ is given as a set of m points.

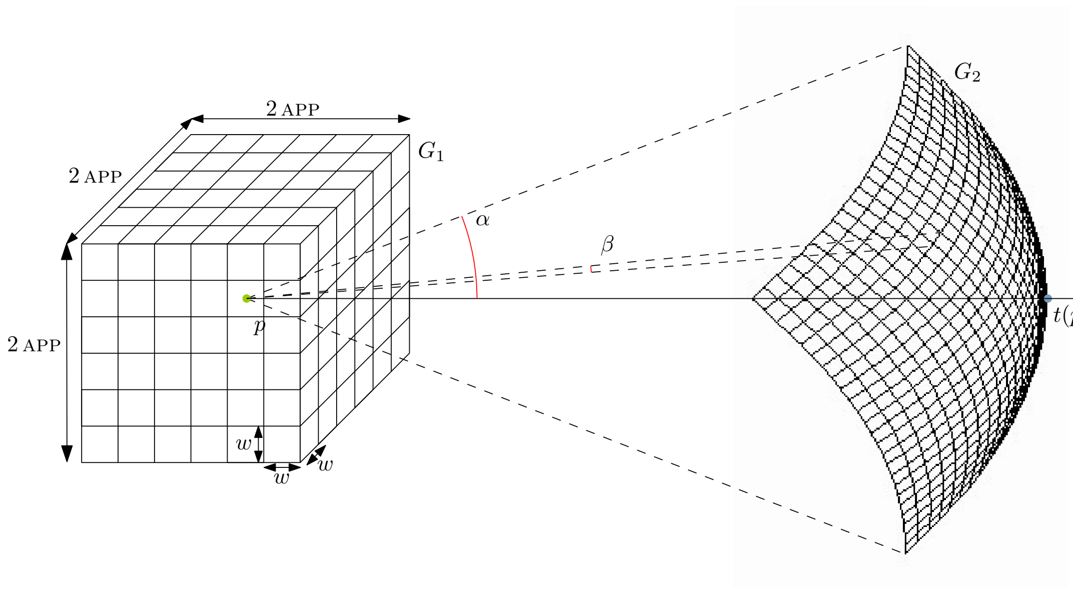


Figure 4.3: Illustration of the discretization technique of the approximation scheme.

Proof. Let t_{sem} be a rigid motion that moves p upon q and $\hat{p}$ on the line through q and $\hat{q}$. Among the two possible positions of $t_{sem}(\hat{p})$ on that line we choose the one with the smaller distance to $\hat{q}$. For each choice of $\hat{q}$ we first compute the constant-factor approximation described in Lemma 2 which provides an approximation value of $\text{APP}_{\hat{q}}$.

Based on $\text{APP}_{\hat{q}}$ the grids G_1 and G_2 are generated in the manner described before. There are $O(1/\epsilon^5)$ combinations of a rotation t_a around q that moves $t_{sem}(\hat{p})$ onto a grid point $a \in G_2$ and of a translation t_b that moves $t_{sem}(p)$ onto a grid point $b \in G_1$. For each combination t_a, t_b the exact algorithm (see [9, Lemma 3.2]) that computes a rotation $t_{r,a,b}$ around the axis through $t_b \circ t_a \circ t_{sem}(p)$ and $t_b \circ t_a \circ t_{sem}(\hat{p})$ that minimizes $\vec{h}(t_{r,a,b} \circ t_b \circ t_a \circ t_{sem}(P), \mathcal{S})$ is called. □

4.2.2 Approximation for Triangulated Surfaces

By a further discretization step we extend the FPTAS for surfaces that are represented by point sets to a quasi-FPTAS for triangulated surfaces.

Theorem 5. *For any $\epsilon > 0$ and $\tau > 0$, a $((1+\epsilon)\,\text{OPT} + \tau)$-approximation for Problem 3 can be computed in*

$$O\left(\frac{mn}{\epsilon^5}\, 2^{\varphi(mn)} \log mn \left(\frac{\|p-\hat{p}\|}{\tau}\right)^2\right)$$

time, provided that $\mathcal{S}$ is given as a set of m triangles and $\hat{p} \in \arg\max_{p' \in P} \|p - p'\|$.

Proof. Assume that the closest point $\hat{q} \in \mathcal{S}$ to $\hat{p}$ in optimal position is known. Applying the transformation $t_2 \circ t_1$ (as defined in the proof of Theorem 4) on p and P moves p onto q and $\hat{p}$ on the line through q

and $\hat{q}$. Consider the sphere s centered in q with radius $\|p - \hat{p}\|$ and a set $Q_\tau \subset s$ of sample points on s with the property

$$\forall x \in s\ \exists y \in Q_\tau : \|x - y\| \leqslant \tau. \tag{4.3}$$

This property ensures that the distance of the intersection point of the ray starting in q passing through $\hat{q}$ with s is in distance of at most τ to its closest sample $y \in Q_\tau$. Trying transformations $t_2 \circ t_1$ that move p onto q and $\hat{p}$ onto y results in an additional absolute additive error term of at most τ to the registration, as $\hat{p}$ is the furthest point in P to p. □

In the case of a surface that is given as a point set, it is possible to try all candidates for $\hat{q}$ the closest point to $\hat{p}$ in optimal position. This is not possible for triangulated surfaces, as the point that realizes the smallest distance of $\hat{p}$ to $\mathcal{S}$ could be a corner, on an edge or in the interior of any triangle, which is the reason why the absolute-error term τ has to be introduced if the same techniques as for the FPTAS shall be applied.

In the proof of Theorem 5, the whole sphere s was sampled "τ-densely". It is actually enough to sample the portion of s that is covered by the perspective projection of all triangles of $\mathcal{S}$ from q onto s.

4.3 Heuristic Framework

The key idea of the heuristic is to convert an instance I of the *one-point case*, i.e., an input that contains exactly one characteristic point (see Problem 3), into a sequence of problem instances $I_1, I_2, \ldots$ for the *two point case*, i.e., inputs that contain exactly two characteristic points (see Chapter 3). Each problem instance I_i is generated based on I by adding an additional *virtual* characteristic point to the instance. The virtual characteristic point that is added to I is chosen by a strategy of the heuristic and does not correspond to defined or measured features of the surface.

The heuristic does not compute an approximation to an optimal registration for Problem 3, it computes a set of registrations that have a cost that is at most a given parameter $\rho > 0$ where ρ is part of the input, given that $\textsc{opt} \leqslant \rho$.

Let $\mathcal{A}_2$ be an algorithm that comutes rigid motion registrations for the two-point case minimizing Equation 4.1. The heuristic strategy presented here proceeds as follows: first, the point in P furthest to p is chosen to be the virtual characteristic point $\hat{p}$ for the pattern space ($P_c = \{p, \hat{p}\}$). This choice of $\hat{p}$ will remain the same for all instances that will be generated in the course of the computation. For the first instance I_1, an initial position q_1 is guessed for $\hat{p}$ from a specific region of the model space. The subsequent instances I_i for $i > 1$ are generated by exploiting the following observation:

Observation 1. *Let q' and q'' be two choices of virtual characteristic points and let $P_c = \{p, \hat{p}\}$, q, P and $\mathcal{S}$ be defined as before. Let t' be an optimal registration for the instance $(P_c, \{q, q'\}, P, \mathcal{S})$ and t'' an optimal registration for the instance $(P_c, \{q, q''\}, P, \mathcal{S})$, then*

$$\| \operatorname{dist}\left(t'(P_c), \{q, q'\}, t'(P), \mathcal{S}\right) - \operatorname{dist}\left(t''(P_c), \{q, q''\}, t''(P), \mathcal{S}\right) \| \leqslant \|q' - q''\|$$

as

$$\| \operatorname{dist}\left(P_c, \{q, q'\}, P, \mathcal{S}\right) - \operatorname{dist}\left(P_c, \{q, q''\}, P, \mathcal{S}\right) \| \leqslant \|q' - q''\|.$$

In other words, the function $\operatorname{dist}(\cdot)$ *is Lipschitz continuous with a constant of* 1 *with respect to the position of the added virtual characteristic point in model space.*

For a virtual characteristic point q', let $t[q']$ be the rigid motion registration computed by $\mathcal{A}_2$ for the instance $(P_c, \{q, q'\}, P, \mathcal{S})$ and let $d[q']$ be its registration value, i.e.,

$$d[q'] := \operatorname{dist}\left(t[q'](P_c), \{q, q'\}, t[q'](P), \mathcal{S}\right).$$

Observation 1 implies that for any virtual characteristic point q'' we have that

$$d[q''] \in \left[d[q'] - \|q' - q''\|, d[q'] + \|q' - q''\|\right].$$

4.3.1 The Heuristic

The heuristic takes a valid instance I for the one point case, a parameter $\rho > 0$ and an algorithm $\mathcal{A}_2$ (solving two point instances) as input. The output is either a (set of) rigid motion registrations for Problem 3 that have a cost of at most ρ or the empty set in case that no such registrations have been found.

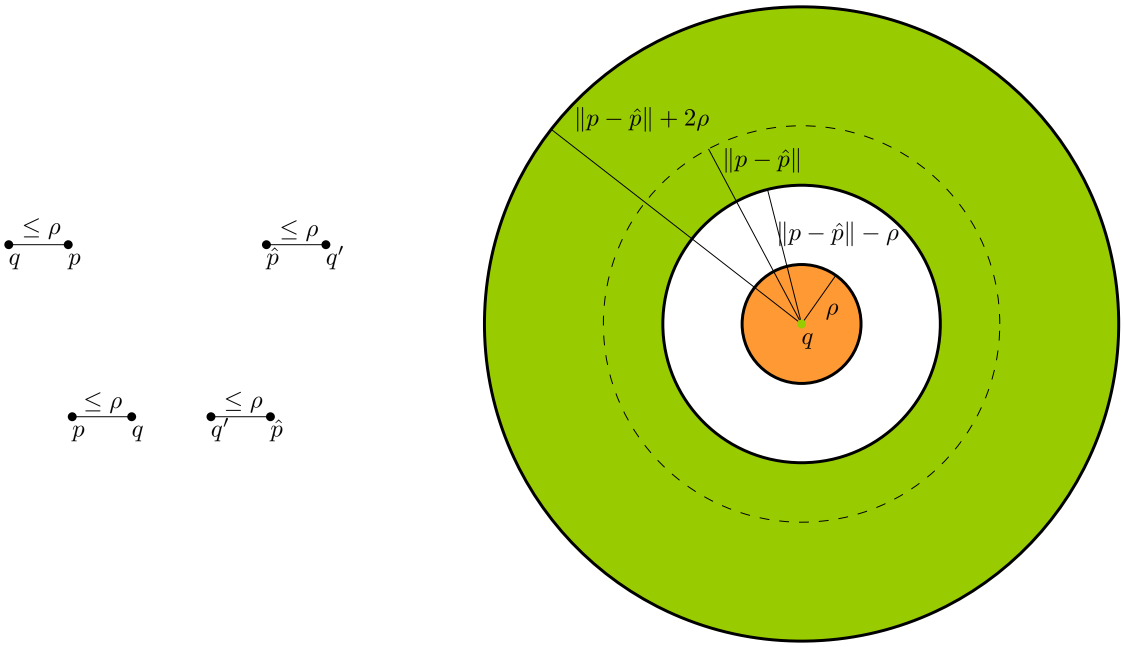


Figure 4.4: **left:** Illustration of the two extreme positions for the measured characteristic point p and the virtual characteristic point $\hat{p}$, given that $\|p - q\| \leqslant \rho$, **right:** Illustration of the annulus $\circledcirc$ from which virtual characteristic points can be drawn.

The objective function of Equation 4.1 is the maximum over the Euclidean distance of the (single) characteristic point and its registrated image and the directed Hausdorff distance of the registrated point set P to the surface $\mathcal{S}$. Any registration with a cost of at most ρ must satisfy that p lies within or on the boundary of the ball centered in q with radius ρ. Any registration for the two point case that fulfills that $\|q - p\| \leqslant \rho$ must place the second characteristic point in the closed annulus $\circledcirc$ with inner radius $\|p - \hat{p}\| - \rho$ and outer radius $\|p - \hat{p}\| + 2\rho$, see Figure 4.4. For $x \in \mathbb{R}^3$ and $r > 0$ let

$Ball(x, r) := \{p \mid \|p - x\| \leqslant r\}$ be a Euclidean ball around x with radius r. In this context, the term *annulus* describes objects of the form $Ball(x, r) \setminus Ball(x, r')$ for $r > r'$.

The heuristic actually computes a set $\mathcal{Q}$ of virtual characteristic points for which it is guaranteed that $\mathcal{A}_2$ will compute a rigid body registration that has a cost of at most ρ if any $q' \in \mathcal{Q}$ is chosen as the virtual characteristic point in the model space, see Algorithm 1.

Algorithm 1: Computing a set $\mathcal{Q}$ of virtual characteristic points providing a cost $\leqslant \rho$.

Data: $p, q, P, \mathcal{S}$ as before, an algorithm $\mathcal{A}_2$ applicable on two point instances, a parameter $\rho > 0$.
Result: The set $\mathcal{Q}$ of virtual characteristic points realizing a distance function value of at most ρ.

```
   // initializing the result set
   $\mathcal{Q} := \varnothing$;
   $\hat{p}$ := any point of $\arg\max_{p' \in P} \|p - p'\|$;
   $M := \circledcirc$;
   while $M \neq \varnothing$ do
       // select a random point of $M$
       $q'$ := takeRandomPoint($M$);
       if $d[q'] > \rho$ then
           // exclude neighborhood
1          $M := M \setminus (S_r \cap Ball(q', |d[q'] - \rho|))$;
       else
           // include neighborhood
           // compute intersection
2          $I := \circledcirc \cap Ball(q', |d[q'] - \rho|)$;
           // remove $I$ from search space
           $M := M \setminus I$;
           // add $I$ to solution
           $\mathcal{Q} := \mathcal{Q} \cup I$;
       end
   end
   return $\mathcal{Q}$ ;
```

Run-time Analysis

The number of samples the heuristic chooses and therefore its runtime partially depends on the difference between the cost $d[q']$ for the chosen samples, the approximation value ρ, the portion of $\circledcirc$ that provides virtual characteristic points that result in registrations with cost at most ρ compared to the volume of $\circledcirc$ and how these *good* virtual characteristic points are distributed in $\circledcirc$. The runtime hence depends on geometric properties of the input that a priori cannot be determined and therefore cannot be expressed as a function in the magnitudes or in the description size of the input.

The heuristic as described so far is not even guaranteed to terminate, as choosing samples q' with $d[q'] = \rho$ does not reduce the search space. The termination of the process can be enforced by ensuring that for each chosen sample an area of at least a pre-described volume is excluded, i.e., by replacing $Ball(q', |d[q'] - \rho|)$ by $Ball(q', \min(|d[q'] - \rho|, \epsilon))$ in lines 1 and 2 of Algorithm 1 for a parameter $\epsilon > 0$ that is part of the input of the heuristic.

In this variant, at least a ball with radius ϵ is excluded from the admissible region M. An upper bound to the number of samples that are drawn in total can be derived by a packing argument as described in [34]. The maximal number of samples that can be drawn equals the largest number of balls B of radius ϵ so that all centers are within $\circledcirc$, all centers are mutually at least ϵ apart and their union covers $\circledcirc$, i.e., $\bigcup_{b \in B} b \supseteq \circledcirc$. This in turn is the largest number N of disjoint balls of radius $\epsilon/2$ that can be packed into $\circledcirc$. By replacing the annulus with a ball whose radius is equal to the outer radius of $\circledcirc$, we get a bound of

$$N\pi \frac{\epsilon^3}{6} \leqslant \frac{4}{3}\pi \left(\|p + \hat{p}\| + 2\rho\right)^3,$$

implying

$$N \in O\left(\left(\frac{\|p - \hat{p}\| + 2\rho}{\epsilon}\right)^3\right).$$

Introducing the parameter ϵ to the strategy of the heuristic guarantees its termination. On the other hand it may also cause that not all *good* virtual characteristic points are found. Also areas of $\circledcirc$ might be marked as *good* even though they provide virtual characteristic points that result in registration with a cost larger than ρ (in fact the cost may become as large as $\rho + \epsilon$). It might even be that no virtual characteristic points that provide good registrations will be found even though they exist, if $\rho \leqslant \text{OPT} + \epsilon$.

4.3.2 Implementation

The heuristic has been implemented twice. An implementation that follows the description given above is described in Appendix A. Here, a simpler variant is described where the candidate set of virtual characteristic points in Algorithm 1 is set to the sphere S centered in q with radius $\|p - \hat{p}\|$. As argued for Lemma 2 the best choice of a sample point $q' \in S$ is guaranteed to provide a $2(1+\sqrt{2})$-approximation to OPT, which implies that for choices $\rho < 2(1 + \sqrt{2})$ OPT the heuristic is not guaranteed to compute any registrations even though OPT $\leqslant \rho$. This variant of the heuristic can be applied when the expected cost of an optimal solution is expected to be small in comparison to ρ or the measured characteristic point is expected to be measured with high precision, e.g., if the computed registrations are used as the initial position of an ICP registration.

As the computation of an arrangement of circles on a sphere is complex and comparably time consuming, the implementation uses six quad-trees [11] to approximate the arrangement. These quad-trees form the six sides of the smallest axis parallel cube C containing S and are initialized with a single square as the root facet. All facets are also stored in a max-heap data structure with respect to the length of their diagonals.

In each round, a facet f with the largest diameter is dequeued from the heap. Then, the intersection q_f of S with the line segment from q to the center of f is computed and selected as a virtual characteristic point. The ball $Ball(q_f, |d[q_f] - \rho|)$ is then projected back (using a perspective projection with focus q) onto the sides of the cube, see Figure 4.5. The faces of the quad-trees that intersect this projection are removed from the heap and refined until they approximate the intersection ϵ-fine (for a suitably small ϵ). This means that facets that have a diagonal smaller or equal to ϵ are not refined if they are intersected by the boundary of the projection. All facets that are generated within a step of the process and that lie outside of the projection are added to the heap. If $d[q_f] < \rho$, all facets that are within the intersection are added to the return set $\mathcal{Q}$. This process is repeated until the heap contains no further facets, see Figure 4.6 for an illustration.

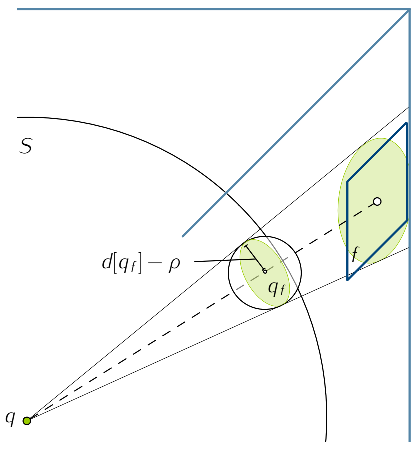


Figure 4.5: Illustration of how the center of a facet f is projected onto the virtual characteristic point $q_f \in S$ and how the ball of radius $\|d[q_f] - \rho\|$ centered in q_f is projected back onto the bounding cube.

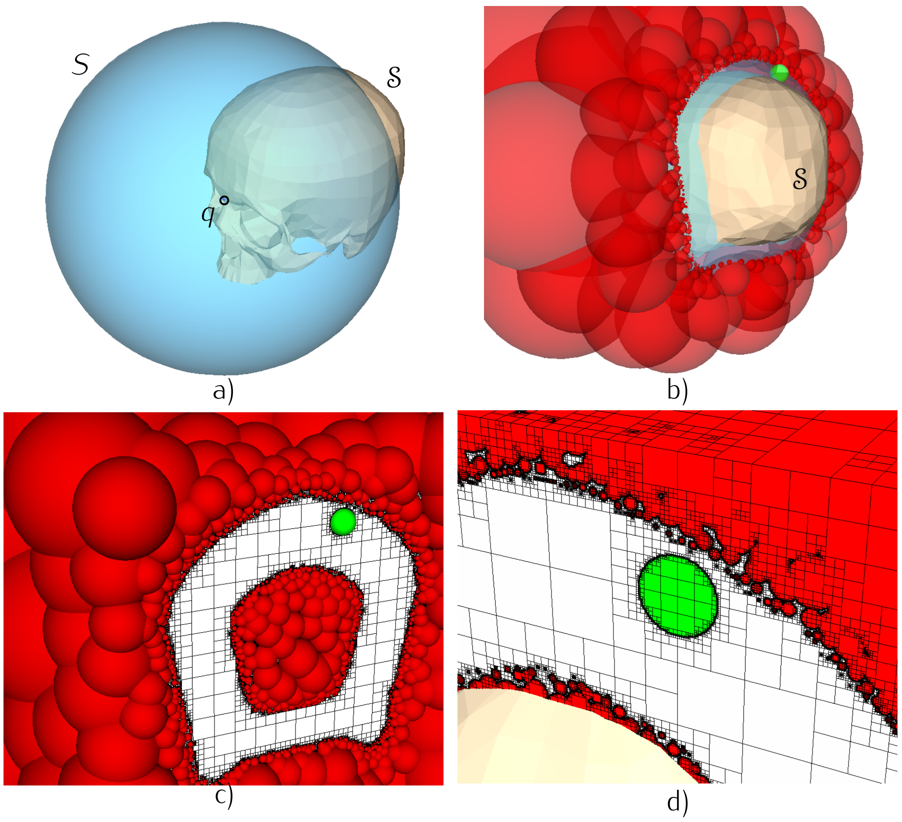


Figure 4.6: **a)** The model $\mathcal{S}$, the measured characteristic point q and the sphere S, **b)** A set of exclusion areas and one inclusion area, **c)+d)** their approximation on the cube surrounding S.

Chapter 5

Minimizing the Weighted Directed Hausdorff Distance

The central ideal of a hybrid registration algorithm is to compute a mapping from the pattern space to the model space by simultaneously matching different geometric features. As mentioned before, these different geometric features are of different types, in the context of the considered application: measured anatomical landmarks and surface points. In this chapter, we address the fact that different feature types can be measured with different precisions and study how to incorporate this information in a registration algorithm.

In Chapter 7 we present a theoretical and empirical study on the measurement precision of anatomic landmarks in comparison to arbitrarily measured points. This study shows that it is reasonable to distinguish the input (precision wise) not only by their 'nature' but also by other properties like the spatial position with respect to the positioning of the patient in the operation theatre. The positioning of a patient in turn depends on the location of the tumor that has to be operated on.

Features that can be measured with a higher accuracy should have a stronger influence on the registration in contrast to features that have a comparable weaker precision. We present exact and approximative algorithms to compute registrations where the different measurement precisions are part of the input and are considered in the objective function.

The weighted directed Hausdorff distance that we are about to introduce is defined for sequences of weighted geometric objects – in this context point sets. The coordinates of a point in such a set is the defined (in model space) or measured (in pattern space) position of a landmark. The weight of a point is proportional to the precision with which the landmark can be measured and is based on the mentioned empirical study presented in Chapter 7. Landmarks that can be measured with the same precision are collected in the same set.

5.1 Problem Definition

Given are two sequences of finite point sets $\mathcal{P} = (P_1, \dots, P_k)$ and $\mathcal{Q} = (Q_1, \dots, Q_k)$ with $P_i, Q_i \subset \mathbb{R}^d$ and a sequence of weights $w = (w_1, \dots, w_k)$ with $w_i \in \mathbb{R}^+$ and $w_1 \geqslant w_2 \geqslant \cdots \geqslant w_k$. For $n_i := |P_i|$ and $m_i := |Q_i|$, let $n = \sum_{i=1}^k n_i$ and $m = \sum_{i=1}^k m_i$. For the purpose of illustration one can think of $\mathcal{P}$ and $\mathcal{Q}$ being colored in k different colors, where all points in P_i and Q_i are colored with the i^{th} color.

Definition 7. *The* weighted directed Hausdorff distance $\vec{h}(\mathcal{P}, \mathcal{Q}, w)$ *for $\mathcal{P}$, $\mathcal{Q}$ and w given as before is defined as:*

$$\vec{h}(\mathcal{P}, \mathcal{Q}, w) = \max_{1 \leqslant i \leqslant k} w_i \cdot \vec{h}(P_i, Q_i).$$

For points in the plane, $\vec{h}(\mathcal{P}, \mathcal{Q}, w)$ can be computed in $O((n + m) \log m)$ time using Voronoi diagrams to compute the directed Hausdorff distance for every color; for higher dimensions it can be computed easily in $O(mn)$ time.

In this chapter, we discuss the problem of matching two sequences of weighted point sets under the weighted directed Hausdorff distance:

Problem 4. *Given $\mathcal{P}$, $\mathcal{Q}$ and w defined as before and a transformation class $\mathcal{T}$, find the transformations $t \in \mathcal{T}$ that minimize the function*

$$f(t) = \vec{h}(t(\mathcal{P}), \mathcal{Q}, w), \tag{5.1}$$

where $t(\mathcal{P}) := (\{t(p) \mid p \in P_i\})_{1 \leqslant i \leqslant k}$.

Let OPT denote the value of the minimum of f.

5.1.1 Generalization of the Problem

In the following sections, we present approximations and exact solutions for Problem 4 for various transformation classes and dimensions. A reasonable generalization of the problem is to maintain the partitioning of the points into k sets but assigning a weight to each point individually. The originally stated problem then becomes a special variant of this formulation. The generalized problem can be solved using the same techniques and with the same runtime as for solving Problem 4. We decided to present the solutions for Problem 4 as it simplifies the notation and meets the specification of the motivating application.

In Section 5.2 we present a 2-approximation for colored point sets in arbitrary dimension under translations and extend it to an FPTAS in Section 5.3. In Section 5.4, a constant-factor approximation for rigid motions is presented. An exact algorithm optimizing the Hausdorff distance under translations for points in the plane is presented in Section 5.5.

5.2 A 2-Approximation for Translations in $\mathbb{R}^d$

Theorem 6. *A 2-approximation for translations of colored points can be computed in $O(m_1(m + n) \log m)$ time in $\mathbb{R}^2$ and in $O(m_1 mn)$ time for point sets in higher dimensions.*

Proof. Choose a point $p \in P_1$ and let T be the set of all translation vectors that move p upon a point of Q_1:

$$T := \{q - p \mid q \in Q_1\}.$$

Then, any translation t_{2app} of T that realizes the smallest Hausdorff distance is a 2-approximations of OPT:

$$f(t_{2app}) = \min_{t \in T} h(t(\mathcal{P}), \mathcal{Q}, w) \leqslant 2\,\text{OPT}\,.$$

To show this, assume $\mathcal{P}$ to be in optimal position, let $\text{nn}\,(p, Q_i) \subseteq Q_i$ be the set of nearest neighbors of $p \in P_i$ in Q_i, and let $n_i(p)$ be one representative of this set. In optimal position, all weighted distances are bounded by OPT:

$$\forall\, 1 \leqslant i \leqslant k\ \forall\, p' \in P_i \quad w_i \|p' - n_i(p')\| \leqslant \text{OPT}\,.$$

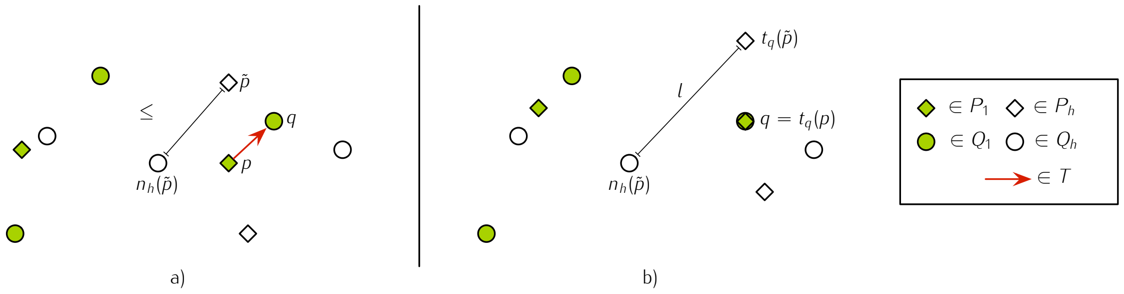


Figure 5.1: Illustration of an aligned point configuration for the weighted directed Hausdorff distance.

Consider the translation $t_q \in T$ that moves the chosen point p upon one of its nearest neighbors $q \in \text{nn}\,(p, Q_1)$. By applying t_q to $\mathcal{P}$, all points in P_i for any $i \in [k]$ are moved away from their optimal positions by $\|p - q\|$, see Figure 5.1. The new weighted distance l of any $\tilde{p} \in P_i$ to one of its closest neighbors after applying t_q is at most:

$$\begin{aligned} l &= w_i \cdot h(t_q(\tilde{p}), Q_i) \\ &\leqslant w_i\, (\|\tilde{p} - n_i(\tilde{p})\| + \|p - n_1(p)\|) \\ &\leqslant \text{OPT} + w_i\, \|p - n_1(p)\| \\ &\leqslant \text{OPT} + w_1\, \|p - n_1(p)\| \\ &\leqslant 2\,\text{OPT}\,. \end{aligned}$$

For a fixed $p \in P_1$, we simply test all translations that move p upon a point $q \in Q_1$, which yields the stated runtime. □

The key argument that guarantees an approximation factor of 2 is that w_1 is at least as large as any other weight. Choosing an arbitrary point $p' \in P_i$ for any color i and testing all translation vectors that move p' onto a point $q' \in Q_i$ yields a $(1 + {}^{w_1}/_{w_i})$-approximation that can be computed in $O\,(m_i(m+n)\log m)$ time in the plane or in $O\,(m_i\, m\, n)$ time for arbitrarily dimensions. By the pigeonhole principle, this implies that there is a $1 + ({}^{w_1}/_{w_k})$-approximation that can be computed in $O\left({}^{m^2}/_k\, n\right)$ time for general d and in $O\,({}^{m}/_k\,(m+n)\log m)$ time for point sets in the plane.

5.3 An FPTAS for Translations in $\mathbb{R}^d$

Theorem 7. *For every $\epsilon > 0$ an $(1+\epsilon)$-approximation for translations can be computed in*

$$O\left(1/\epsilon^2\, m_1 (m+n) \log m\right)$$

time in the plane and in $O\left(\left(\sqrt{d}/\epsilon\right)^d\, m_1\, m\, n\right)$ time in higher dimensions.

Proof. Let $\textsc{app}_2 = f(t_{2app})$ where t_{2app} is a solution of the 2-approximation as described in Theorem 6. Choose any $p \in P_1$ and for every $q \in Q_1$ build a regular cubic grid G_q centered in q with side length $\textsc{app}_2 / w_1$ where each cell has a side length of $(\epsilon\ \textsc{app}_2)/(\sqrt{d}\, w_1)$, see Figure 5.2.

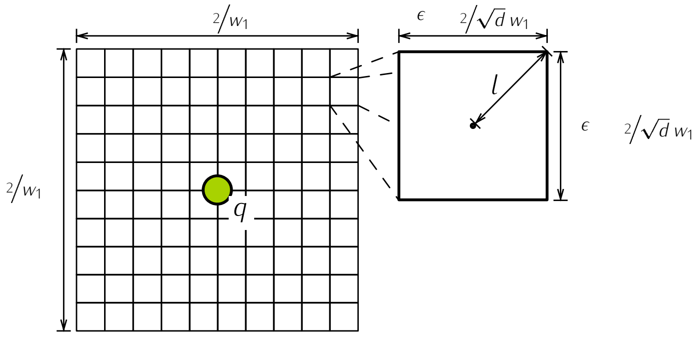

$2/w_1$
$\epsilon\quad 2/\sqrt{d}\, w_1$
l
$\epsilon\quad 2/\sqrt{d}\, w_1$
$2/w_1$
q

Figure 5.2: Illustration of G_q.

Let T be the set of all translations that move a point $p \in P_1$ onto a grid point of a grid G_q for any $q \in Q_1$. Then, the translation t_{app} of T that realizes the smallest weighted Hausdorff distance satisfies

$$f(t_{app}) = \min_{t \in T} \vec{h}\left(t(\mathcal{P}), \mathcal{Q}, w\right) \leqslant (1+\epsilon)\,\textsc{opt}\,.$$

Assume $\mathcal{P}$ to be in optimal position and let $q = n_1(p)$ be one of the nearest neighbors of the chosen $p \in P_1$. By construction, the point p lies within the boundaries of the grid G_q. Due to the choice of the cell length, the furthest distance l of p to one of its closest grid points is at most $l \leqslant (\epsilon\ \textsc{app}_2)/(2\sqrt{d} w_1)$ and as $\textsc{opt} \geqslant \textsc{app}_2/2$ we have that $l \leqslant (\epsilon\ \textsc{opt})/(\sqrt{d} w_1)$. As all w_i for $1 < i \leqslant k$ are at most as large as w_1, the weighted distance of all other points increases by at most $\epsilon \cdot \textsc{opt}$ when translated along the distance l.

There are $\left(\sqrt{d}/\epsilon\right)^d$ grid points for each G_q that need to be tested, which implies the stated runtime. □

Using the same arguments as for the 2-approximation, it is easy to show that an $(1+\epsilon)$-approximation can be computed in $O\left(\left(\sqrt{d}\, c/\epsilon\right)^d\, m^2\, n/k\right)$ time for points sets in dimension $d \geqslant 3$ or in

$$O\left(\left(c/\epsilon\right)^2\, m/k(m+n)\log m\right)$$

time in $\mathbb{R}^2$ with $c = w_1/w_k$.

5.4 A Constant-Factor Approximation for Rigid Motions in the Plane

The constant-factor approximation described in Section 5.2 for a planar setting can be extended to cover rigid motions.

Theorem 8. *A $2(1+\sqrt{2})$-approximation for rigid motions can be computed in*

$$O\left(m_1\, m_i\, (m+n) \log m\right)$$

time for point sets in the plane.

Proof. Choose any $p \in P_1$. Let $P := \bigcup_{1 \leqslant i \leqslant k} P_i$ and let $\omega : P \to \mathbb{R}$ be the function that maps a point $x \in P$ to its weight $\omega(x) = w_i$ for $x \in P_i$. Let $p' \in P$ be a point that maximizes the *weighted distance* to p, that is,

$$p' \in \arg\max_{x \in P} \omega(x) \|x - p\|.$$

Without loss of generality, let $\omega(p') = w_i$. This choice of p' ensures that the change of the weighted Hausdorff distance of any point $\tilde{p} \in P$ when rotated around p is bounded by the change of the weighted Hausdorff distance of p'.

For any $q \in Q_1$ and any $q' \in Q_i$, let t be defined as $t := t_{q,q'} \circ t_q$, where t_q is the translation that moves p upon q and $t_{q,q'}$ is the rotation around q by the smallest absolute angle such that q, q' and $t_{q,q'} \circ t_q(p')$ are aligned, see Figure 5.3.

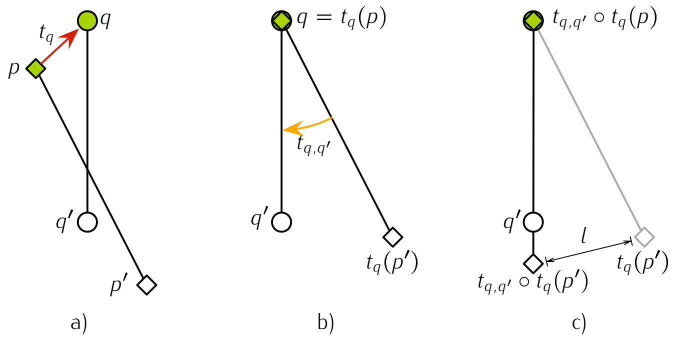


Figure 5.3: **a)** The translational part t_q, **b)** the rotational part $t_{q,q'}$, **c)** the aligned configuration.

Let T be the set of all transformations that align $p \in P_1$ and $p' \in P_i$ with some $q \in Q_1$ and $q' \in Q_i$ in the described manner.

We claim that any transformation t_{rapp} of T that realizes the smallest Hausdorff distance satisfies the stated approximation factor, that is,

$$f(t_{rapp}) = \min_{t \in T} \vec{h}\,(t(\mathcal{P}), \mathcal{Q}, w) \leqslant 2\,\textsc{opt} + 2\sqrt{2}\,\textsc{opt} = 2(1+\sqrt{2})\,\textsc{opt}\,.$$

The first summand is due to the influence of the translational component t_q as discussed in the proof of Theorem 6. The second term, $2\sqrt{2}$ OPT, reflects the furthest weighted distance that a point $\tilde{p}$ can possibly cover while moving from position $t_q(\tilde{p})$ to $t(\tilde{p})$, see Figure 5.4. $\square$

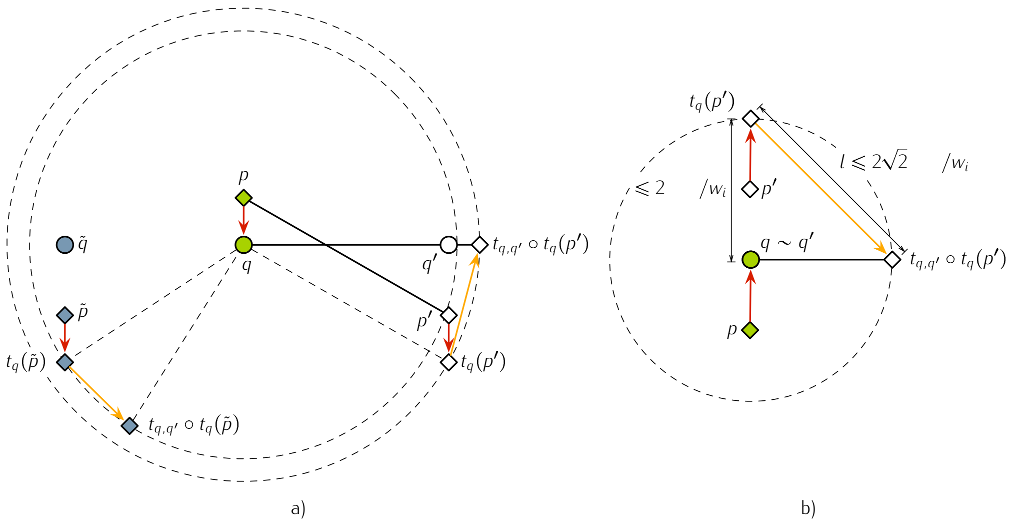


Figure 5.4: **a)** Example of a constant-factor approximation, **b)** Illustration of an aligned configuration that could realize an approximation factor of $2(1+\sqrt{2})$ OPT.

Scenarios where $\overline{q\,q'}$ and $\overline{q\,t_q(p')}$ are orthogonal and additionally $\|q - t_q(p')\| = \text{OPT}$ (see Figure 5.4 b) almost reach the upper bound of $2(1+\sqrt{2})$ OPT.

5.4.1 Extending the Result to $\mathbb{R}^3$

In $\mathbb{R}^3$ one additional degree of freedom has to be fixed. Based on the registration in the previous section, this freedom is a rotation around the axis $\overline{p\,p'}$.

The algorithm can be adjusted by choosing p and p' as before and by choosing a point $p'' \in P_j$ such that

$$p'' \in \arg\max_{x \in P} \omega(x)\,\text{dist}\left(x, \overline{p\,p'}\right),$$

where $\text{dist}\left(a, \overline{b\,c}\right)$ is the distance of the point a to its orthogonal projection onto the line through b and c. For any $q'' \in Q_j$ consider additionally the rotation $t_{q,q',q''}$ around the axis $\overline{q\,q'}$ such that $t_{q,q',q''} \circ t(p'')$, q, q' and q'' are coplanar. It is easy to see that the additional rotation results in a $(6+2\sqrt{2})$-approximation (≈ 11.657) of the optimum and the runtime increases to $O\left(m_1\, m_i\, m_j\, mn\right)$.

Note that similar discretization techniques as those presented for translations can be applied to gain an FPTAS for rigid motions.

5.5 An Exact Algorithm for Computing a Registration Unter Translations in the Plane

The minima of the objective function $f(t)$ (Equation 5.1) under translations in the plane can be computed similar to the approach of Huttenlocher et al. [25] by partitioning the translation space into cells such that for all translations of one cell the weighted Hausdorff distance is realized by the same point pair. As we consider translation throughout this section, we write a translation of a point p by a translation vector t as $p + t$ instead of $t(p)$.

Recall that the objective function f is given as:

$$\begin{aligned} f(t) &= \max_{1\leqslant i\leqslant k} w_i\, h(t(P_i), Q_i) \\ &= \max_{1\leqslant i\leqslant k} w_i \max_{p\in P_i} \min_{q\in Q_i} \|(p+t)-q\| \\ &= \max_{1\leqslant i\leqslant k} w_i \max_{p\in P_i} \min_{q\in Q_i} \|(q-p)-t)\| \end{aligned}$$

Computing the distance of the point $p+t$ to q is equivalent to computing the distance of the translation vector t to the vector $q-p$. By introducing the sets $S_i(p) := \{q-p \mid q \in Q_i\}$ we can reformulate $f(t)$ as

$$f(t) = \max_{1\leqslant i\leqslant k} w_i \max_{p\in P_i} \min_{c\in S_i(p)} \|c-t\|.$$

To compute the minima of f, we compute the decomposition of the translation space into (potentially empty) cells $C(p,q) \subseteq \mathbb{R}^2$ with $p \in P_i$ and $q \in Q_i$, such that

$$C(p,q) := \{t \mid f(t) = w_i \|(p+t)-q\|\}.$$

Theorem 9. *The decomposition of the translation space into cells $C(p,q)$ and with it the transformations t that minimize $f(t)$ can be computed in $O\left(m^2n^2\ \varphi\left(m^2n^2\right)\log mn\right)$ time, where $\varphi(\cdot)$ is the inverse Ackermann function.*

Proof. To characterize $C(p,q)$ with $p \in P_i$ and $q \in Q_i$ we fist observe that $f(t) = w_i \|(p+t)-q\|$ implies $t \in \mathrm{Vor}(q-p, S_i(p))$, where $\mathrm{Vor}(a, A)$ is the Voronoi cell of the site $a \in A$ in the Voronoi diagram of the set A. Otherwise, another point q' of color i would be closer to p than q, implying that $f(t)$ would not by realized by p and q.

There are two possible reasons why for a translation $t \in \mathrm{Vor}(q-p, S_i(p))$ the value of $f(t)$ might not be realized by p and q. This is either because another point $p' \in P_i$ of the same color has a larger distance to its nearest neighbor for this translation. Or the weighted distance of a closest point pair of another color $j \neq i$ exceeds $w_i\|(p+t)-q\|$, that is, $f(t) = w_i\|(p+t)-q\|$ implies

$$\forall_{1\leqslant j\leqslant k}\forall_{p'\in P_j} \min_{q'\in Q_j} w_j\|(p'+t)-q'\| \leqslant w_i\|(p+t)-q\|.$$

We define $B(p,q)$ as the *blocked area* for the pair p and q, i.e., all translations $t \in \mathrm{Vor}(q-p, S_i(p))$ for which the value of $f(t)$ is not realized by p and q due to either of the aforementioned reasons.

The characterization of the shape of $B(p,q)$ for a point pair of the i^{th} color follows directly from the previous observation and is given as

$$B(p,q) := \bigcup_{\tilde{p}\in P_i\setminus\{p\}} \mathrm{Vor}\left(q-p, S_i(\tilde{p}) \cup S_i(p)\right) \cup \bigcup_{\substack{1\leqslant j\leqslant k\\ i\neq j}} \bigcup_{\hat{p}\in P_j} \mathrm{MWVor}\left(q-p, S_i(p), S_j(\hat{p}), w_i, w_j\right),$$

where $\mathrm{MWVor}\left(a, A, B, w, w'\right)$ is the Voronoi cell of a point $a \in A$ in the multiplicatively weighted Voronoi diagram of the set $A \cup B$ where all points in A have weight w and all points in B have weight w'. Note, that the Voronoi cells that are united in the characterization of $B(p,q)$ correspond to the portions of the translation space for which $w_i\|(p+t)-q\|$ is not the maximal distance shortest pair. A cell $C(p,q)$ of a point pair of the i^{th} color is then fully characterized by

$$C(p,q) = \mathrm{Vor}\left(q-p, S_i(p)\right) \setminus B(p,q).$$

For a non-empty cell that has an empty blocking area, the value of $f(t)$ is minimal at the site that defined the cell and increases linearly to the distance from this site. Therefore, the minimal value of $f(t)$ is either realized at a site or at the boundary of a blocking area. This means that the minima of $f(t)$ can be computed while computing the decomposition of the translation space into cells.

Let $b_{p,q}$ denote the number of edges contributing to the boundary of $B(p,q)$.

Lemma 3. *The combinatorial complexity of $B(p,q)$ is $\mathrm{O}\left(b_{p,q}\, 2^{\varphi\left(b_{p,q}\right)}\right)$.*

Proof. By introducing polar coordinates (r,θ) with $q-p$ as the origin, the boundary of $B(p,q)$ can be seen as the upper envelope of the partially-defined, continuous, univariate functions given as the edges of the Voronoi diagrams parametrized by θ. Two Voronoi edges can intersect in at most two points. Applying the theory of Davenport-Schinzel sequences [43], this results in a complexity of $\mathrm{O}\left(b_{p,q}\; 2^{\varphi\left(b_{p,q}\right)}\right)$. □

Let $b = \sum_{1\leqslant i\leqslant k}\sum_{p\in P_i}\sum_{q\in Q_i} b_{p,q}$ be the total number of edges that contribute to the boundary of any blocking region.

Lemma 4. $b = \mathrm{O}\left(m^2 n^2\right)$

Proof. First, fix a color i and a point $p \in P_i$. The edges e that contribute to the boundaries of any blocking area of a facet of $\mathrm{Vor}\left(S_i(p)\right)$ result from the edges of the Voronoi diagrams

$$e \in \bigcup_{\tilde{p}\in P_i\setminus p} \mathrm{Vor}\left(S_i(p) \cup S_i(\tilde{p})\right) \cup \bigcup_{\substack{1\leqslant j\leqslant k\\ i\neq j}} \bigcup_{\hat{p}\in P_j} \mathrm{MWVor}\left(S_i(p), S_j(\hat{p}), w_i, w_j\right).$$

Let b_p^i be the number of edges contributing to the boundaries of the blocking areas for the facets of $\mathrm{Vor}\left(S_i(p)\right)$. The combinatorial complexity of a standard Voronoi diagram of n sites is $\mathrm{O}\left(n\right)$, the complexity of an multiplicatively weighted Voronoi diagram for the same number of sites however is $\Theta(n^2)$ as shown by Aurenhammer et al. [4], even if just two different weights are involved. This is the main reason why the runtime for colored points increases compared to the monochromatic variant discussed by Huttenlocher et al. [25].

This leads to a combinatorial complexity for b_p^i of

$$\begin{aligned}
b_p^i &= \mathrm{O}\left(\sum_{\tilde{p}\in P_i\setminus\{p\}} (m_i + m_i) + \sum_{\substack{1\leqslant j\leqslant k\\ i\neq j}} \sum_{\hat{p}\in P_j} (m_i + m_j)^2\right)\\
&= \mathrm{O}\left(n_i m_i + \sum_{\substack{1\leqslant j\leqslant k\\ i\neq j}} n_j (m_i + m_j)^2\right)\\
&= \mathrm{O}\left(\sum_{1\leqslant j\leqslant k} n_j (m_i + m_j)^2\right)
\end{aligned}$$

Summing over all colors i and all points $p \in P_i$ gives

$$\begin{aligned}
b &= \sum_{1\leqslant i\leqslant k} \sum_{p\in P_i} b_p^i\\
&= \mathrm{O}\left(\sum_{1\leqslant i\leqslant k} \sum_{p\in P_i} \sum_{1\leqslant j\leqslant k} n_j (m_i + m_j)^2\right)\\
&= \mathrm{O}\left(\sum_{1\leqslant i\leqslant k} n_i \sum_{1\leqslant j\leqslant k} n_j (m_i + m_j)^2\right)\\
&= \mathrm{O}\left(\sum_{1\leqslant i\leqslant k} n_i \sum_{i\leqslant j\leqslant k} n_j (m_i + m_j)^2\right)\\
&= \mathrm{O}\left(m^2 n^2\right).
\end{aligned}$$

□

Lemma 3 and Lemma 4 together imply that the combinatorial complexity of the whole decomposition is $\mathrm{O}\left(m^2 n^2\, 2^{\varphi\left(m^2 n^2\right)}\right)$.

The algorithm to compute the translations that minimize the directed Hausdorff distance involves two steps.

First, all (multiplicatively weighted) Voronoi diagrams have to be computed. Aurenhammer et al. [4] presented an algorithm to compute weighted Voronoi diagrams of n sites in $\mathrm{O}\left(n^2\right)$ time. It takes $\mathrm{O}\left(m^2 n^2\right)$ time to compute all diagrams, as argued for Lemma 4.

In the second step, the blocking areas of all facets of all Voronoi diagrams have to be computed. As shown by Hershberger [23], the upper envelope of n partially defined functions that mutually intersect in at most s points can be computed in $\mathrm{O}\left(\lambda_{s+1}(n)\log n\right)$ time, where $\lambda_s(n)$ denotes the maximum possible length of a (n, s) Davenport–Schinzel sequence. This leads to a runtime of $\mathrm{O}\left(m^2 n^2\ \varphi\left(m^2 n^2\right)\log mn\right)$, as stated in Theorem 9. □

As shown by Rucklidge [41], the lower bound of the geometric complexity of the graph of $f(t)$ for a single color, i.e., two point sets in the plane, both of size n is already $\Omega(n^3)$.

Note that an exact solution for the special case where all weights of all points are equal ($w_1 = w_k$), can be computed faster. For this case, all multiplicatively weighted Voronoi diagrams are just regular Voronoi diagrams consisting of line segments only. This reduces the complexity of the upper envelope of the boundary of the blocking areas, as two line segments can mutually intersect at most once. For this case, the algorithm and the analysis of the method presented by Huttenlocher et al. [25] can be applied.

Corollary 2. *The transformations t that minimize $f(t)$ for uniformly weighted point sets in the plane under translations can be computed in $\mathrm{O}\left(n^2 m \log nm\right)$ time.*

Chapter 6

Towards Non-Uniform Geometric Matchings

The strategies and algorithms presented in the previous chapters compute exact or approximate registrations by reducing the registration problem to a geometric matching problem. As stated in Definition 2 (geometric matching, page 3), the task in a geometric matching problem is to compute a single transformation of a given transformation class that matches a pattern to a model so that a certain objective function is optimized. This transformation is then used to map any point of the space that contains the pattern to its "corresponding" point in the model space.

The restriction of using a single transformation, no matter which reasonable transformation class is considered, limits the ability to handle or to consider local deformations. Often one has to decide whether a specific region should be mapped well or whether the registration should give results that are good on average over the entire space. This is especially disadvantageous for *soft tissue registrations*. In that context also tissue deformations (e.g., due to respiration or physical pressure) have to be considered.

Computing a set of transformations We generalize the concept of geometric matchings to so-called *non-uniform geometric matchings*. In a non-uniform matching problem, a *set* of transformations is computed. Each transformation is locally valid within predefined regions of the pattern space. The regions of interest form a partition of the pattern space. To map a point p from the pattern space to the model space one first has to determine the cell that contains p. In a second step, the transformation that is associated to that cell is used to perform the actual mapping. The transformation for a certain cell is computed by solving a geometric matching problem that maps geometric features of that cell to the model, see Figure 6.1.

Non-uniform registrations have to optimize two competing objectives: to match the pattern features close to the model features while simultaneously assuring conformity of the mapping by demanding that transformations of two neighbored cells are "similar" with respect to their effect.

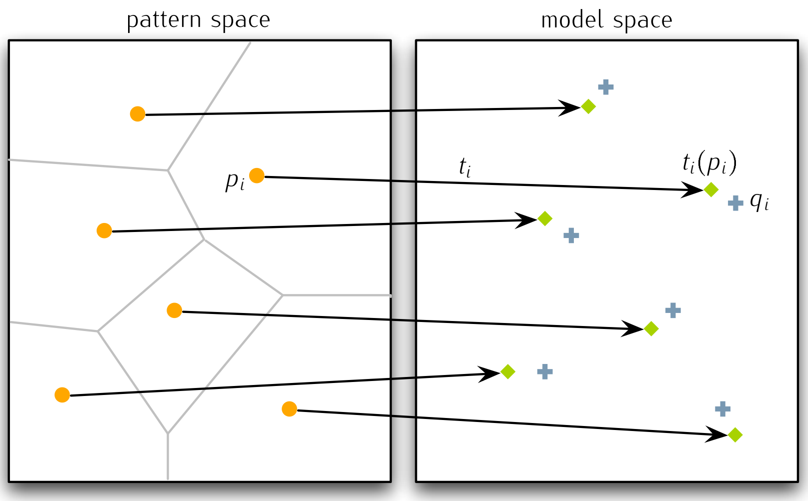


Figure 6.1: Illustration of a non-uniform registration for point sequences under translations. A translation is computed for each cell of the pattern space. The geometric features p_i are matched using the corresponding translations t_i to minimize the distance to the model feature q_i while simultaneously enforcing that transformations of neighboring cells are *similar*.

In its most general form a non-uniform geometric matching problem can be stated as follows:

Definition 8 (Non-Uniform Geometric Matching Problem)**.**
Let $\mathfrak{P}(\mathbb{R}^d, k)$ denote the set of all partitions of $\mathbb{R}^d$ into k cells.
Given:

$\mathcal{G}$	*a class of geometric objects*
$\mathrm{dist}_{\mathcal{G}} : \mathcal{G} \times \mathcal{G} \to \mathbb{R}^+$	*a distance measure in object space*
$P = \{p_1, \dots, p_n\} \in \mathcal{G}$	*a pattern object*
$Q = \{q_1, \dots, q_m\} \in \mathcal{G}$	*a model object*
$\mathcal{T}$	*a transformation class admissible on* $\mathcal{G}$
$C = \{C_1, \dots, C_k\} \in \mathfrak{P}(\mathbb{R}^d, k)$	*a partition of* $\mathbb{R}^d$ *into* k *cells such that* $\forall i \in [n]\ \exists j \in [k]\ p_i \subseteq C_j$
$\mathrm{dist}_{\mathcal{T}} : \mathfrak{P}(\mathbb{R}^d, k) \times \mathcal{T}^k \to \mathbb{R}^+$	*a distance measure in transformation space*
$f : \mathbb{R}^+ \times \mathbb{R}^+ \to \mathbb{R}^+$	*a weight function*

Task: compute a set of transformations $T = (t_1, \dots, t_k) \in \mathcal{T}^k$ *minimizing*

$$f\left(\mathrm{dist}_{\mathcal{G}}\left(\{t_i(C_i \cap P)) \mid 1 \leqslant i \leqslant k\}, Q\right), \mathrm{dist}_{\mathcal{T}}(C, T)\right)$$

.

Note that for $k = 1$ Definition 8 is equal to the definition of the usual geometric matching problem (see Definition 2, page 3).

A pattern feature $p_i \in P$ is mapped by the transformation t_j that corresponds to the cell C_j containing p_i. The matched feature set P' is hence given as

$$P' := \bigcup_{j \in [k]} t_j(P \cap C_j).$$

The objective function that defines the quality of a matching consists of two parts: the function $\mathrm{dist}_{\mathcal{G}}$ that measures the distance of the matched point set P' to Q. The second factor is the function $\mathrm{dist}_{\mathcal{T}}$ that measures the "similarity" of a transformation set T by considering the neighborhood relations of the individual transformations as induced by the partition C of the pattern space.

6.1 Non-Uniform Matchings for Point Sequences

For now, we consider a simple yet not trivial variant of the non-uniform matching problem where we restrict the transformation class $\mathfrak{T}$ to translations. We further assume the geometric features to be point sequences of equal size ($|P| = |Q| = n$), measured in the pattern space and defined in the model space. We also assume that the correspondence between the point sequences is known, that is, point p_i is mapped to q_i for all $i \in [n]$. As the measure in the feature space ($\mathrm{dist}_{\mathcal{G}}$) we consider the maximum Euclidean 1-to-1 distance, that is

$$\mathrm{dist}_{\mathcal{G}}(P', Q) := \max_{i \in [n]} \|t_i(p_i) - q_i\|. \tag{6.1}$$

We consider decompositions of the pattern space into n cells so that each cell contains exactly one point of P (as it is the case for the Voronoi diagram of P). As stated above, the transformations are not computed independently from each other. To control the conformity of the registration around cell boundaries of the decomposition, one has to ensure that two transformations t_a and t_b whose corresponding cells C_a and C_b are neighbors (share parts of their boundary) are *similar* with respect to their effect. As a measure of similarity of two translations t_a and t_b we consider the Euclidean distance $\|t_a(x) - t_b(x)\|$ of their images of a point $x \in \mathbb{R}^d$. From now on, we do not distinguish between a translation and its translation vector and measure the similarity of two translations by the Euclidean norm of the translation vector difference, i.e., $\|t_a - t_b\|$ (as the distance of two images of the same point does not depend on the preimage of the point).

The information about the pairs of translations that have to be similar is encoded in a graph $G = (T, E)$ which we call *neighborhood graph*. The vertex set of G is the set T of translations that are to be computed and $\{t_i, t_j\} \in E$ if the cells corresponding to translations t_i and t_j share parts of their boundary. Note that the edges of the neighborhood graph could also be selected by criteria other than the adjacency of cells and could for instance be manually chosen by the user. The algorithms presented in this chapter do not require that the neighborhood graph resembles the partition of the pattern space. For some algorithms, however, the approximation factors depend on the structure of G.

To simplify notation, we define for any two translations $t_i, t_j \in T$:

$$d_{ij} = \begin{cases} 1 & \text{if } \{t_i, t_j\} \in E \\ 0 & \text{otherwise.} \end{cases}$$

As the measure $\mathrm{dist}_{\mathcal{T}}$ for the similarity of the translation set T we take the maximum of the similarity of any two translations that are adjacent in G:

$$\mathrm{dist}_{\mathcal{T}}(T, G) := \max_{i,j \in [n]} d_{ij} \|t_i - t_j\|.$$

We chose to measure the distances in the pattern space as well as the deviations in the translation space using the Euclidean metric. This problem could just as well be studied with another reasonable underlying measure, such as the Manhattan metric.

Putting all this together and taking the maximum of the distance measured in object space and the similarity measure in translation space, we get the following problem description:

Problem 5. *Given P, Q and G as above, compute a sequence T of translations $(t_1, \ldots, t_n)$ minimizing*

$$\operatorname{dist}(P, Q, G, T) := \max\left(\max_{i\in[n]} \|t_i(p_i) - q_i\|, \max_{i,j\in[n]} d_{ij}\|t_i - t_j\|\right), \tag{6.2}$$

where $\|\cdot\|$ denotes the Euclidean norm. The first term accounts for the distance of the matched point set P to Q by considering the L_∞ norm of the vector $t_i(p_i) - q_i$.

We have chosen to minimize the maximum of the distances in the pattern space and the deviations in the model space. Again, other weight functions e.g. minimizing the sum of both components could be considered as well. For translations however minimizing the maximum of both involved measures seems natural as the minimum will be achieved where both influence variables are equal. As the displacement of a point that caused by choosing either of two neighboring translations (Euclidean distance of the two images) is equal to the deviation of these two transformations (Euclidean norm of the difference vector), taking the maximum of both magnitudes results in a good balance between the two measures by not favoring one over the other.

One advantage of considering translations is that the distance of a matched point p_i to its corresponding point q_i and also the similarity of two translations can be measured in translation space. Consider the translations $s_i = q_i - p_i$ for $1 \leqslant i \leqslant n$ and let $S := (s_1, \ldots, s_n)$. The distance $\|t_i(p_i) - q_i\|$ for a point p_i matched with translation t_i to q_i can be expressed as

$$\|t_i(p_i) - q_i\| = \|t_i + p_i - q_i\| = \|q_i - p_i - t_i\| = \|s_i - t_i\|.$$

The problem of computing a non-uniform matching for point sequences under translations can also be formulated in the following way: Consider a straight line embedding of the graph $G' = (S \cup T, E')$ with $E' = \{\{s_i, t_i\} \mid i \in [n]\} \cup \{\{t_i, t_j\} \mid d_{ij} = 1\}$. The edge set E' consists of two sorts of edges:

1. edges connecting two translations t_i and t_j indicating that they have to be similar,
2. n edges $\{s_i, t_i\}$ whose lengths measure the Euclidean distance of $t_i(p_i)$ to q_i.

Note, that the positions of all $s \in S$ are already determined by the input. The problem of computing a non-uniform registration optimizing Equation 6.2 can be formulated as:

Problem 6. *Find a placement for all $t \in T$ such that the length of the longest edge of the induced straight line embedding of G' is minimal.*

As the vertices of G' represent translations, we also call G' the *translation graph* of S.

6.1.1 Convex Programming Formulation

The problem of computing a non-uniform registration optimizing Equation 6.2 can be phrased as a convex optimization problem (see [6] for an introduction into this field):

$$\begin{array}{rrcll} \text{minimize} & & & \epsilon & \\ \text{subject to} & \|s_i - t_i\| & \leqslant & \epsilon, & i = 1, \ldots, n, \\ & d_{ij}\,\|t_i - t_j\| & \leqslant & \epsilon & 1 \leqslant i < j \leqslant n. \end{array}$$

As any metric norm is convex and the maximum of two convex functions is also convex. Convex optimization problems have the property that they have a unique minimum, i.e., any local minimum is also a global minimum. Furthermore, convex optimization problems (such as Problem 6) can be solved in polynomial time, e.g., by using the interior-point or the ellipsoid method [6].

In Sections 6.2 and 6.3 we present fast approximation algorithms that are based on geometric insights into the problem. There are two reasons for considering geometric approximation algorithms for this problem, even though the machinery of convex programming provides us with exact polynomial solutions to this problem:

1. the approximation factors of the constant-factor approximations are close to 1 and the approximate solutions can be computed in linear time with only small constants hidden in the O-Notation.
2. the geometric insights we gained during the study of the geometric nature of this problem help to develop approximation strategies for non-uniform matching variants that can *not* be formulated as a convex optimization problem, see Section 6.4.

6.2 Constant-Factor Approximations

Let T_{opt} be an optimal solution and let $\textsc{opt} := \operatorname{dist}(P, Q, G, T_{opt})$ be the value of the objective function for T_{opt}.

6.2.1 Arbitrary Graphs and Bounded Diameter Graphs

Theorem 10. *Choosing* $t_i = q_i - p_i = s_i$ *for* $1 \leqslant i \leqslant n$ *results in a 3-approximation of* $\textsc{opt}$.

Proof. Assume T to be in optimal position. For any i and j with $d_{ij} = 1$ we have that $\|t_i - t_j\| \leqslant \textsc{opt}$ as well as $\|t_i - s_i\| \leqslant \textsc{opt}$ and $\|t_j - s_j\| \leqslant \textsc{opt}$. Moving t_i upon s_i and t_j upon s_j increases the distance $\|t_i - t_j\|$ by at most $2 \cdot \textsc{opt}$ while setting the distances $\|t_i - s_i\|$ and $\|t_j - s_j\|$ to zero, hence $\|t_i - t_j\| \leqslant 3 \cdot \textsc{opt}$ for all i, j with $d_{ij} = 1$, see Figure 6.2a. □

Let k be the diameter of the neighborhood graph G, i.e., the largest number of edges on a shortest path between any two vertices of G (short with respect to the number of edges on the path).

Theorem 11. *Choosing* $t_1 = t_2 = \cdots = t_n = q_i - p_i$ *for some* $1 \leqslant i \leqslant n$ *results in a* $(k + 2)$*-approximation of* $\textsc{opt}$.

Proof. Assume T to be in optimal position and let i be the selected index. The distance of any t_j to t_i is at most $k \cdot \textsc{opt}$ as each edge on the path from t_i to t_j has length at most $\textsc{opt}$ and the number of edges on the path is bounded by k. As the distance $\|t_i - s_i\|$ is also bounded by $\textsc{opt}$, we have that $\|t_i(p_j) - q_j\| \leqslant (k + 2)\,\textsc{opt}$, see Figure 6.2b. □

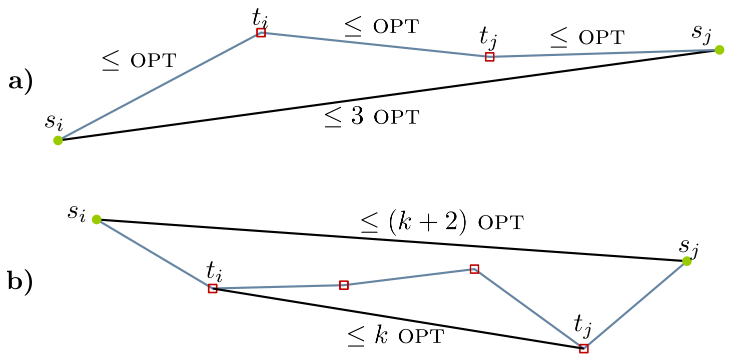


Figure 6.2: **a)** Illustration of the 3-approximation of Theorem 10. **b)** Illustration of the $(k+2)$-approximation of Theorem 11.

6.2.2 Complete Graphs in the Plane

Assume $P = (p_1, \ldots, p_n)$ and $Q = (q_1, \ldots, q_n)$ to be point sequences in the plane and the neighborhood graph G to be complete, that is, any two translations have to be compared.

Lemma 5. *Let T_{opt} be an optimal choice of translations. The center c_{opt} of the smallest disc enclosing T_{opt} provides a $(1 + 1/\sqrt{3})$-approximation for points in the plane and complete neighborhood graphs, if c_{opt} is chosen for all $t \in T$:*

$$\mathrm{dist}(P, Q, G, (t_1 = c_{opt}, t_2 = c_{opt}, \ldots, t_n = c_{opt})) \leqslant (1 + 1/\sqrt{3})\ \text{OPT}\,.$$

Proof. In optimal position, the distance $\|s_i - t_i\|$ for any $1 \leqslant i \leqslant n$ is bounded by OPT. All translations of T_{opt} lie within the smallest disc enclosing T_{opt} whose radius is bounded by $\text{OPT}/\sqrt{3}$, as stated in the following lemma.

Lemma 6. *The radius of the smallest disc enclosing a point set of width μ in the plane is bounded by $\mu/\sqrt{3}$.*

Proof. Any planar point set X of at least two points contains a subset of two or three points $\{x_i\} \subseteq X$ such that the smallest enclosing disc δ_X of the subset is identical to the smallest enclosing disc of X, moreover all x_i lie on the boundary of δ_X and define δ_X. If δ_X is described by two points x_1 and x_2 then $\|x_1 - x_2\|$ is the diameter of δ_X and as $\|x_1 - x_2\|/2 \leqslant \mu/2 < \mu/\sqrt{3}$ the lemma holds.

Assume that δ_X is defined by three points x_1, x_2, x_3 and assume w.l.o.g. that $\|x_1 - x_2\|$ is the longest of the pairwise distances of $\{x_1, x_2, x_3\}$ and assume that x_3 lies to the left of the ray starting in x_1 through x_2.

A Reuleaux triangle of width μ is the intersection of three discs of radius μ centered at the corners of an equilateral triangle with side length μ. The circumcircle δ_Δ of an equilateral triangle of side length μ is identical to the circumcircle of its induced Reuleaux triangle and has a radius of $\mu/\sqrt{3}$, see Figure 6.3 right.

Consider the rigid motion that moves x_1 on the origin and x_2 on the positive x-axis, hence x_3 lies in the intersection of the first quadrant with two discs of radius $\|x_1 - x_2\|$ centered in x_1 and x_2 respectively.

The set $\{x_1, x_2, x_3\}$ is fully contained in the Reuleaux triangle with one corner on the origin, one corner on the positive x-axis and the third corner in the first quadrant and is therefore also covered by δ_Δ, see Figure 6.3 left. This implies, that the radius of δ_X is bounded by the radius of δ_Δ. $\square$

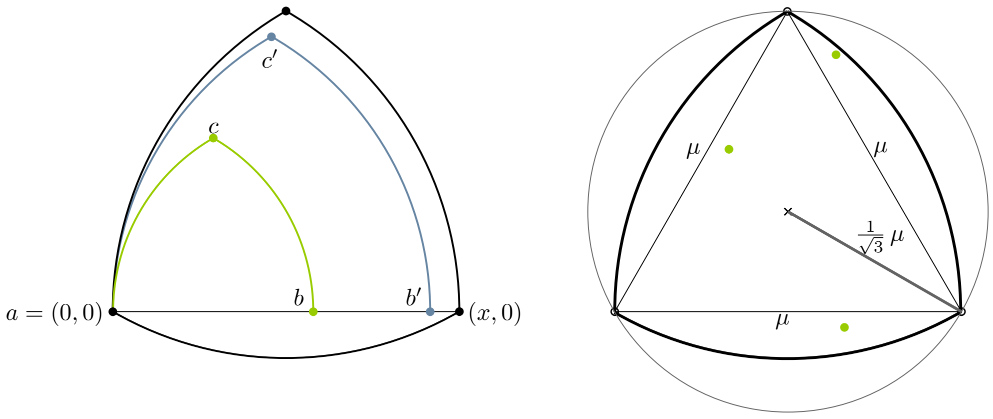


Figure 6.3: **left:** Illustration of the proof of Lemma 6, **right:** the equilateral- and Reuleaux triangle containing all point sets of width x.

The distance of each point $s \in S$ to c_{opt} is bounded by $\text{OPT} + {}^1\!/\!\sqrt{3}\ \text{OPT}$. Therefore, the center c_{opt} implies a $(1 + {}^1\!/\!\sqrt{3})$-approximation as stated in Lemma 5. $\square$

Lemma 5 implies that there exists a single translation that results in a maximal distance of $(1+{}^1\!/\!\sqrt{3})\ \text{OPT}$ to any $s \in S$. But as T_{opt} is unknown, the center c_{opt} of its smallest enclosing disc is unknown as well. On the other hand, the translation that *minimizes* the largest distance to any point of S can be computed in linear time [25, 37].

It is easy to see that the center c of the smallest disc enclosing S is the translation that minimizes the maximal distance to any translation in S. We have determined the approximation factor for choosing $t_i = c_{opt}$ for $i \in [n]$ and know that c is the best possible choice of a single translation. Together with Lemma 5 this implies the following constant-factor approximation:

Theorem 12. *The center c of the smallest disc enclosing the point sequence S results in a $(1 + {}^1\!/\!\sqrt{3})$-approximation:*

$$\text{APP} = \text{dist}(P, Q, G, (t_1 = c, t_2 = c, \ldots, t_n = c)) \leqslant (1 + {}^1\!/\!\sqrt{3})\ \text{OPT}\,.$$

The approximation factor can be improved to ${}^2\!/\!3\,(1+{}^1\!/\!\sqrt{3}) \approx 1.05157$ by choosing n different translations in the following way: Let APP be the value of the approximation as presented in Theorem 12. Choose, for $1 \leqslant i \leqslant n$, t_i to be the intersection o_i of the straight line $\overline{s_i\, c}$ with the circle δ_{app} centered in c with radius $\text{APP}/3$. If δ_{app} does not intersect the line segment $\overline{s_i\, c}$, then t_i is chosen to be s_i. For this choice of t_i, the distance $\|s_i - t_i\|$ is bounded by ${}^2\!/\!3\ (1 + {}^1\!/\!\sqrt{3})\ \text{OPT}$ for each $1 \leqslant i \leqslant n$ which is also the diameter of the circle δ_{app}, implying that the distances $\|t_i - t_j\|$ for $1 \leqslant i < j \leqslant n$ are also bounded by ${}^2\!/\!3\ (1 + {}^1\!/\!\sqrt{3})\ \text{OPT}$, see Figure 6.4.

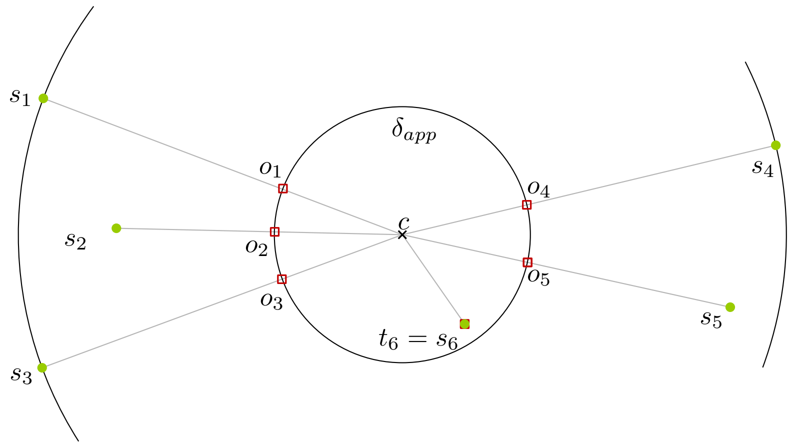


Figure 6.4: The outer circle is the smallest circle enclosing S (radius APP), the inner circle δ_{app} has a radius of APP $/3$ with the same center.

Theorem 13. *Let c be the center of the smallest disc enclosing the sequence S and let* APP *be the approximation value resulting from applying Theorem 12. Choosing t_i as the intersection o_i of the straight line $\overline{s_i\, c}$ with the circle δ_{app} with center c and radius* APP $/3$*, or $t_i = s_i$ if δ_{app} does not intersect $\overline{s_i\, c}$, results in a $^2/_3\ (1 + {}^1/\sqrt{3})$-approximation that can be computed in $\mathrm{O}(n)$ time:*

$$\widetilde{\mathrm{APP}} = \mathrm{dist}(P, Q, G, (t_1, t_2, \dots, t_n)) \leqslant {}^2/_3\ (1 + {}^1/\sqrt{3})\ \mathrm{OPT} \approx\ 1.05157\, \mathrm{OPT}\,.$$

Center of Optimal Translations vs. Center of Approximation

Figure 6.5 shows a picture of an example for which the center c_{opt} of the optimal translations and the center c of the smallest disc enclosing S differ. The translations $s_1 = (0, 0), s_2 = (3, 0), s_3 = (1.5, 1 + \sqrt{3}/2)$ have an unique optimal solution sequence $T_{opt} = (t_1 = (1, 0), t_2 = (2, 0), t_3 = (1.5, \sqrt{3}/2))$ which has its center at $c_{opt} = (1.5, \sqrt{3}/6)$. The center of S is $c = (1.5, \approx 0.33011)$ which results in an approximation quotient of $\widetilde{\mathrm{APP}}/\,\mathrm{OPT} \approx 1.0239$.

6.2.3 Neighborhood Graphs with Diameter k

In the previous section the neighborhood graph has been assumed to be complete, that is, any two translations had to be compared. We extend the idea of Theorem 13 to graphs with diameter $k > 1$.

Lemma 7. *The radius of the smallest disc that encloses a straight line embedding of a graph with diameter k, given that the length of any edge in the embedding is at most μ, is bounded by*

$$f(k) \cdot \mu = \frac{k - (1 + k^2)\lfloor {}^k/_2 \rfloor}{\cos\left(2 \arcsin\left({}^1/_{2k}\right)\right) + \sin\left(2 \arcsin\left({}^1/_{2k}\right)\right)\sqrt{4\lfloor {}^k/_2\rfloor^2 - 1} - 2k \cos\left(2 \arcsin\left({}^1/_{2k}\right)\right)\lfloor {}^k/_2 \rfloor} \cdot \mu$$

The radius of the smallest disc that encloses a straight line embedding of a graph with diameter k (so that the longest edge of the embedding has a length of at most μ) is maximized for embeddings that

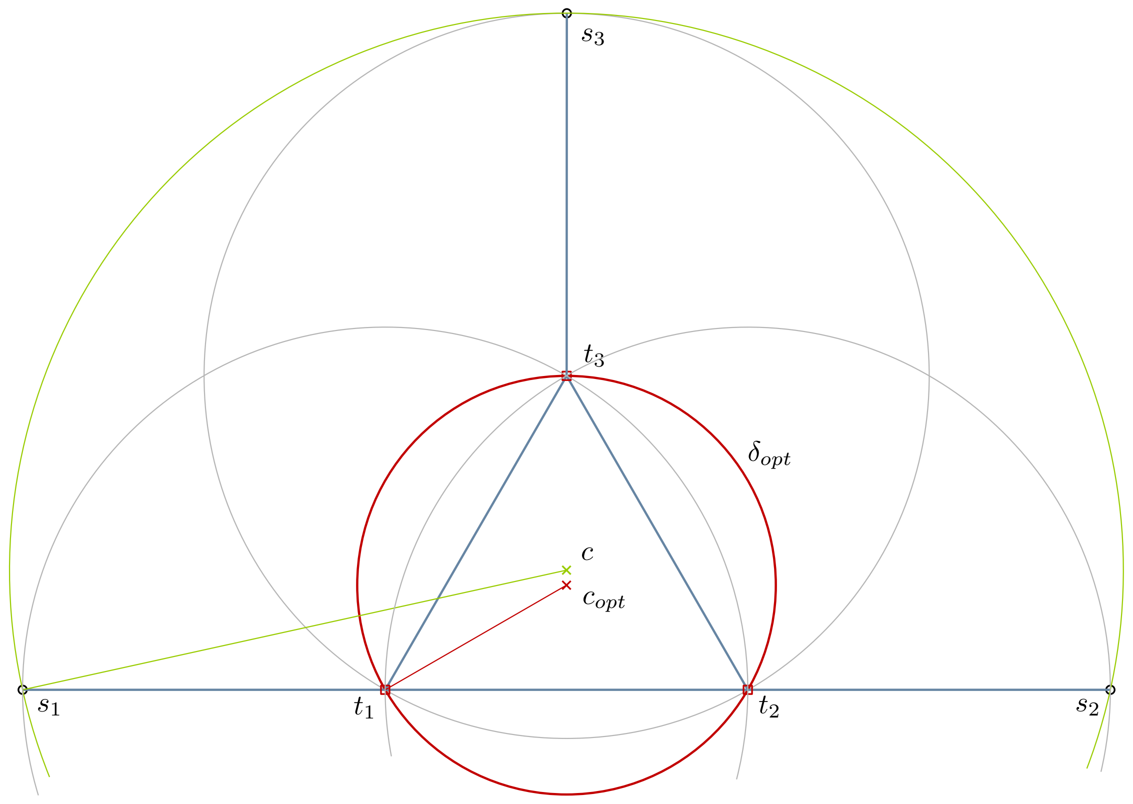


Figure 6.5: Example of an approximative registration in contrast to an optimal solution.

contain an isosceles triangle of side length $k \cdot \mu$ and a base side length of length μ. Another isosceles triangle is attached to the $\lfloor k/2 \rfloor^{th}$ vertex of this embedding that shares a vertex of the base line of the first triangle and also has a base side length of μ, see Figure 6.6.

Replacing this bound for the radius of the smallest circumcircle of the optimal translations in Theorem 13 leads to the following result for graphs with diameter k:

Theorem 14. *Let P, Q be as before, let G be a neighborhood graph with diameter k, let c be the center of the smallest ball enclosing the sequence S and let* APP *be the approximation value resulting from applying Theorem 12. Choosing t_i as the intersection o_i of the straight line $\overline{s_i\,c}$ with the ball δ_{app} centered in c with radius* APP */3, or $t_i = s_i$ if δ_{app} does not intersect $\overline{s_i\,c}$, results in a $2/3\,(1 + f(k))$-approximation, that is,*

$$\operatorname{dist}(P, Q, G, (t_1, t_2, \dots, t_n)) \leqslant 2/3\,(1 + f(k))\,\text{OPT}\,.$$

This approximation – compared to the trivial 3-approximation described in Theorem 10 – gives better approximation factors for neighborhood graphs with a diameter of up to 6, as shown in Table 6.1.

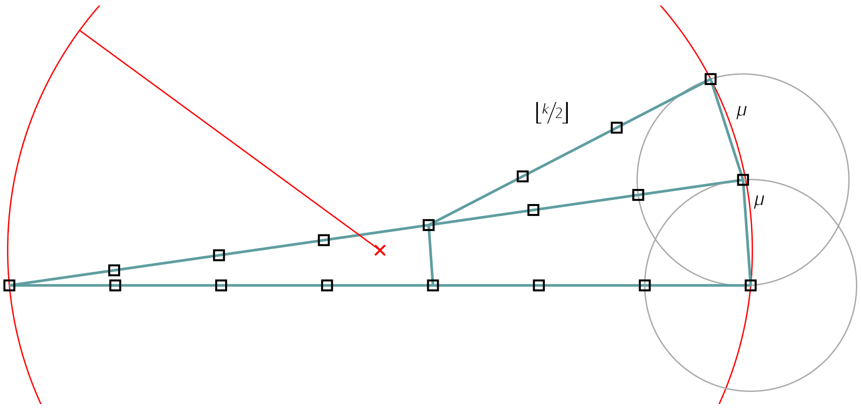


Figure 6.6: Illustration of the proof of Lemma 7.

k	$\approx 2/3\,(1 + f(k))$
2	1.7862
3	1.7904
4	2.1405
5	2.4221
6	2.7541
7	3.0658

Table 6.1: The approximation values of Theorem 14 for $1 < k \leqslant 7$.

6.3 Approximation Schemes for Trees and Cycles

In this section, we present two $(1 + \epsilon)$-approximation schemes for settings in which the neighborhood graph is a tree or a cycle. The first technique uses a dense sampling combined with a dynamic programming approach and works in arbitrary dimensions. The second strategy computes an approximation based on deciding a relaxed decision problem variant for different guesses of the value of OPT and can be applied if additionally P and Q are planar point sequences.

By slightly abusing notation, we impose the information of G on S, that is, we call s_i and s_j adjacent, if $\{t_i, t_j\} \in E(G)$.

6.3.1 An FPTAS for Trees

The basic idea of the approximation strategy is to construct a sufficiently dense sample set around each $s \in S$. Samples are chosen from the sample sets in a bottom-up manner, from the leafs to the root using dynamic programming: starting in the leafs, the *cost* of selecting a sample for a point s is computed based on the optimal choices of samples in the subtree rooted in s.

Sufficiently Dense Sample Sets

In the following we consider samplings $C(\alpha, \beta, p) \subset \mathbb{R}^d$ of points $p \in \mathbb{R}^d$ with $\alpha > 0$, $\beta \geqslant 0$ that have the following properties:

1. $\forall\, x \in C(\alpha, \beta, p) : \ \|x - p\| \leqslant \beta$
2. $\forall\, t \in \{t \mid \|t - p\| \leqslant \beta\} : \ \exists\, x \in C(\alpha, \beta, p) : \ \|x - t\| \leqslant \alpha$

We call such a sampling an α-dense sampling of p with radius β. For the Euclidean norm and a point $p \in \mathbb{R}^d$, an α-dense sampling of radius β with

$$|C(\alpha, \beta, p)| \in \mathrm{O}\left(\left(\sqrt{d}\,\frac{\beta}{\alpha}\right)^d\right)$$

can be constructed, e.g., by intersecting a regular rectangular gird with side length $\alpha\sqrt{4/d}$ with a ball of radius β centered in p.

Let APP' be the value of the 3-approximation as described in Theorem 10. In the following, as α and β will be fixed to $\alpha = 1/6\,\epsilon\,\text{APP}'$ and $\beta = \text{APP}'$, we write $C(x)$ instead of $C(1/6\,\epsilon\,\text{APP}', \text{APP}', x)$, to simplify notation.

Dynamic programming

Let r be an arbitrarily selected root of G. We define the cost $c[x]$ of a sample point $x \in C(s)$ for a point $s \in S$ inductively. If s is a leaf in G, then $c[x] := \|x - s\|$, the distance of the sample x to the translation s. Otherwise, let $c_1, \dots, c_l$ be the children of s in G then

$$c[x] := \max\left(\|x - s\|, \max_{1 \leqslant j \leqslant l} \min_{y \in C(c_j)} \max(c[y], \|y - x\|)\right). \tag{6.3}$$

Additionally, pointers $n[x, j]$ are stored together with the cost $c[x]$ for each sample $x \in C(s)$ pointing to one sample point y that realizes the minimum in $\min_{y \in C(c_j)} \max(c[y], \|y - x\|)$ for x in Equation 6.3 for each child c_j of s.

A sequence of translations $t'_1, \dots, t'_n$ that realizes a $(1 + \epsilon)$-approximation can be found by following the pointers stored at a sample point m of the root r that has the smallest cost value of any sample in $C(r)$, that is, $m \in \arg\min_{x \in C(r)} c[x]$.

Theorem 15. *The set $t'_1, \dots, t'_n$ of translations that result from following the pointers of the sample $m \in C(r)$ with the smallest cost realize a $(1 + \epsilon)$-approximation and can be computed in $\mathrm{O}\left(n/\epsilon^{2d}\right)$ time.*

Proof. Consider an optimal translation t_i, from Theorem 10 we know that t_i lies within a circle with radius APP' centered in s_i. The distance of t_i to its closest sample is at most $\alpha = 1/6\,\epsilon\,\text{APP}'$ and as $\text{APP}' \leqslant 3\,\text{OPT}$ this distance is bounded by $1/2\,\epsilon\,\text{OPT}$. Moving each optimal translation to its closest sample point increases the distance of two neighbored translations by at most $\epsilon\,\text{OPT}$ and the distance of each t_i to s_i by at most $1/2\,\epsilon\,\text{OPT}$.

That the best combination of samples from each sample set is selected can be shown by using an inductive argument. We argue that the algorithm maintains the best cost $c[x]$ and pointers $n[x, i]$ for all samples $x \in C(s_i)$ in the subtree rooted in s_i.

For a leaf s_i, each sample $x \in C(s_i)$ has a cost of $c[x] = \|s_i - x\|$. This is the distance of q_i to $p_i + x$, which is clearly optimal as only p_i and q_i have to be considered (s_i is a leaf). If s_i is an internal node, the cost $c[x]$ of a sample $x \in C(s_i)$ is either the distance $\|x - s_i\|$ to its sample center s_i (invariant to any choice of samples in the subtree), the distance $\|y - x\|$ to a sample $y \in C(c_j)$ of some child c_j of s_i in G or the cost $c[y]$ of a sample of a child. For all children c_j of s_i the sample y realizing the minimum $\min_{y \in C(c_j)} \max(c[y], \|y - x\|)$ is chosen and –by supposition– $c[y]$ contains the best possible value in the subtree rooted in c_j, given that $y \in C(c_j)$ was selected. Therefore, $c[x]$ contains the best possible value of any choices of samples in the sample sets of the children of s_i.

Each sample set $C(s_i)$ consists of $\mathrm{O}\left(\left({}^{\mathrm{APP}'}\!/_{\epsilon\,\mathrm{APP}'}\right)^d\right) = \mathrm{O}\left({}^1\!/_{\epsilon^d}\right)$ sample points for fixed d. Each of the $\mathrm{O}\left({}^n\!/_{\epsilon^d}\right)$ samples (except the ones of the root) is touched $\mathrm{O}\left({}^1\!/_{\epsilon^d}\right)$ times while updating the cost information for the samples of the parent node, which results in a runtime of $\mathrm{O}\left({}^n\!/_{\epsilon^{2d}}\right)$, as stated in the theorem. □

6.3.2 An FPTAS for Cycles

Let S be ordered so that for $1 \leqslant i < n$, s_i and s_{i+1} are neighbored in G, as well as s_n and s_1. Consider the graph $G^* = G \backslash \{s_1\}$ resulting from removing s_1 from the cycle. Fix a sample $x \in C(s_1)$ and perform the $(1 + \epsilon)$-approximation for trees on G^* rooted in s_n, with the difference that the cost $c[y]$ for any $y \in C(s_2)$ is initialized to

$$\forall y \in C(s_2)\ c[y] := \max(\|x - s_1\|, \|y - s_2\|, \|x - y\|),$$

which is the value of the registration in the chain s_1, s_2 given that x is chosen for s_1 and y is chosen for s_2. Then, a best possible choice for samples in the sample sets of $s_2, \ldots, s_n$, given that $x \in C(s_1)$ was selected, can be gained by following the pointers of any sample $z \in C(s_n)$ with

$$z \in \arg\min_{w \in C(s_n)} \max(c[w], \|w - x\|).$$

As all $\mathrm{O}\left({}^1\!/_{\epsilon^d}\right)$ choices for $x \in C(s_1)$ have to be tested, the runtime increases to $\mathrm{O}\left({}^n\!/_{\epsilon^{3d}}\right)$.

6.3.3 A Faster FPTAS for Trees in the Plane

In this subsection, we present an alternative approximation scheme for neighborhood graphs that are trees in the plane. This approximation is based on a relaxed decision problem formulation. The usual (unrelaxed) decision variant to Problem 6 can be stated as follows:

Problem 7. *For a given $\mu \geqslant 0$ and a translation graph $G' = (S \cup T, E')$ that is a tree with $S, T \subset \mathbb{R}^2$, is there a placement of T such that the length of any edge in the induced straight line embedding of G' is at most μ?*

Before presenting an algorithm to decide Problem 7, we need to introduce some notation and mention basic geometric observations. As in the previous subsections, we impose the structure of the neighborhood graph $G(T, E)$ on S and hence call s_i and s_j adjacent if $\{t_i, t_j\} \in E$. Let $\delta(c, r)$ be a disc of radius r centered in c and define $\delta_\mu := \delta((0, 0), \mu)$. The Minkowski sum $X \oplus Y$ of two sets X and Y is defined as $X \oplus Y := \{x + y \mid x \in X,\ y \in Y\}$. For X being a geometric figure, the set $X \oplus \delta_\mu$ is the set of all points z so that there is a point $x \in X$ with $\|z - x\| \leqslant \mu$.

The following simple geometric observations hold for embeddings that meet the edge length constraint, see Figure 6.7:

1. for all $t_i \in T$ we have that $t_i \in \delta(s_i, \mu)$
2. if $\{t_a, t_b\} \in E(G)$ then $t_a \in \delta(s_b, 2\mu)$ and $t_b \in \delta(s_a, 2\mu)$
3. if $c_1, \ldots, c_k$ are the children of s_i then $t_i \in \bigcap_{j\in[k]} \delta(c_j, 2\mu) \cap \delta(s_i, \mu)$

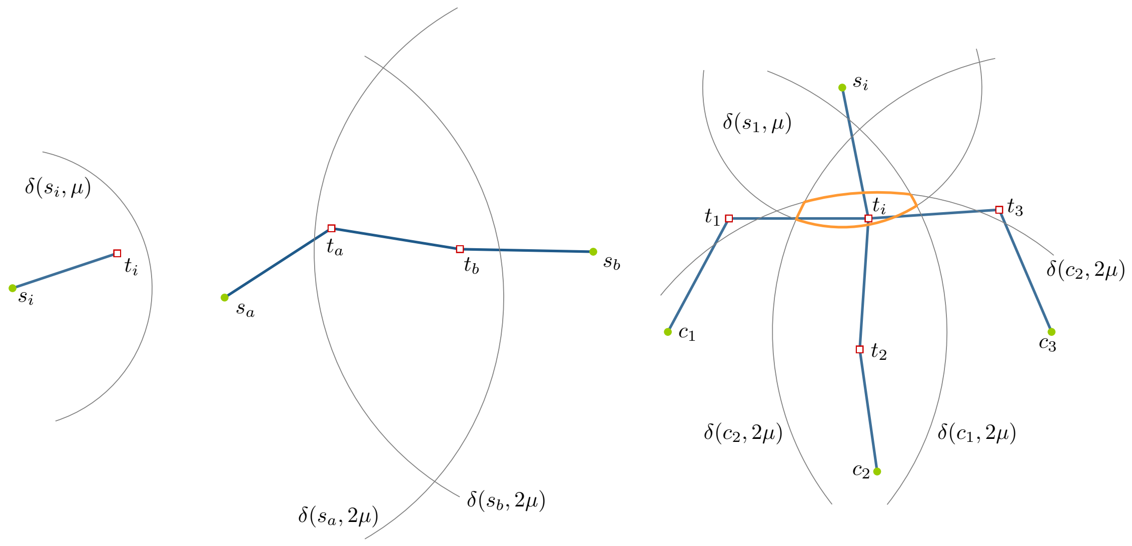


Figure 6.7: Illustration of geometric properties that every registration whose edges have a length of at most μ has to satisfy.

These observations motivate the definition of the *admissible region* $\operatorname{reg}_\mu(s)$ for a point $s \in S$. The admissible region of a point s is defined as the set of all translations t for which a straight line embedding of the subtree rooted in s exists that satisfies the edge length constraint.

Definition 9 (admissible region). *The convex admissible region* $\operatorname{reg}_\mu(s)$ *of a point* s *for a given* μ *is defined inductively:*

- *if s is a leaf in G' then* $\operatorname{reg}_\mu(s) = \delta(s, \mu)$*,*
- *if s is an internal node with children $c_1, \ldots, c_k$, then*

$$\operatorname{reg}_\mu(s) = \bigcap_{i\in[k]} \left(\operatorname{reg}_\mu(c_i) \oplus \delta_\mu\right) \cap \delta(s, \mu). \tag{6.4}$$

Let $r \in S$ be an arbitrarily chosen root of G. The decision problem can be solved by computing the admissible region of r:

Lemma 8. *There exists a straight line embedding of G' so that each edge has a length of at most μ iff* $\operatorname{reg}_\mu(r) \neq \emptyset$.

Solving the decision problem exactly involves computing Minkowski sums of admissible regions and their intersections. The boundary of an admissible region of a point s can in the worst case be defined by all "μ inflated" admissible regions of the children of s. Fortunately, it is not necessary to maintain the exact shape of the admissible regions to compute a $(1+\epsilon)$-approximation. Instead of computing the exact admissible regions $\operatorname{reg}_{\mu}(s)$, we approximate its shape by an convex polygon $\widetilde{\operatorname{reg}}_{\mu}(s)$ so that $\vec{h}\left(\widetilde{\operatorname{reg}}_{\mu}(s), \operatorname{reg}_{\mu}(s)\right) \leqslant \lambda$ for a $\lambda > 0$ that is to specified later and additionally $\operatorname{reg}_{\mu}(s) \subset \widetilde{\operatorname{reg}}_{\mu}(s)$. We also approximate the inflated admissible regions $\operatorname{reg}_{\mu}(s) \oplus \delta_{\mu}$ by convex polygons $\operatorname{infl}_{\mu}(s)$ so that $\operatorname{reg}_{\mu}(s) \oplus \delta_{\mu} \subset \operatorname{infl}_{\mu}(s)$ and $\vec{h}\left(\operatorname{infl}_{\mu}(s), \operatorname{reg}_{\mu}(s) \oplus \delta_{\mu}\right) \leqslant \lambda$.

Relaxing the decision problem

An algorithm $\mathcal{A}$ that uses the described inductive strategy stated in Equation 6.4 to decide Problem 7 for given $\mu \geqslant 0$ and $\lambda > 0$ by maintaining the regions $\widetilde{\operatorname{reg}}_{\mu}(s)$ ($\operatorname{infl}_{\mu}(s)$) instead of $\operatorname{reg}_{\mu}(s)$ ($\operatorname{reg}_{\mu}(s) \oplus \delta_{\mu}$) for all $s \in S$ has the following properties:

- it returns FALSE for any $\mu < \text{OPT} - \lambda$
- it returns TRUE for any $\mu > \text{OPT}$
- it returns either TRUE or FALSE for any $\mu \in [\text{OPT} - \lambda, \text{OPT})$.

Note, that two inflated approximate admissible regions $\operatorname{infl}_{\mu}(s)$ and $\operatorname{infl}_{\mu}(s')$ might intersect, even though $\operatorname{reg}_{\mu}(s) \oplus \delta_{\mu} \cap \operatorname{reg}_{\mu}(s') \oplus \delta_{\mu} = \emptyset$. Let $s \in S$ be an internal node of G and let $c_1, \ldots, c_k$ be the k children of s. For any $t \in \widetilde{\operatorname{reg}}_{\mu}(s)$ we have that $\|t - s\| \leqslant \mu + \lambda$ and

$$\forall i \in [k]\ \exists t' \in \widetilde{\operatorname{reg}}_{\mu}(c_i) :\ \|c_i - t'\| \leqslant \mu + \lambda \wedge \|t - t'\| \leqslant \mu + \lambda.$$

As before, let APP' be the value of the 3-approximation as described in Theorem 10, hence $\text{APP}'/3 \leqslant \text{OPT} \leqslant \text{APP}'$ and set λ to $\epsilon \cdot \text{APP}'/3$. Consider an uniform sampling of the interval $[\text{APP}'/3, \text{APP}']$ with sample width λ (i.e., the distance of two consecutive samples). The smallest sample μ' of the sample set for which the approximated admissible region of the root r of G is not empty satisfies that $|\mu - \text{OPT}| \leqslant \epsilon \cdot \text{APP}'/3 < \epsilon \cdot \text{OPT}$, hence the embedding computed for the value μ' realizes a $(1+\epsilon)$-approximation.

Theorem 16. *For a neighborhood graph G that is a tree and point sequences $P, Q \in \mathbb{R}^2$ and any $\epsilon > 0$ a sequence of translations $T = (t_1, \ldots, t_n)$ can be computed in $O(\log 1/\epsilon \; n/\sqrt{\epsilon})$ time so that*

$$\operatorname{dist}(P, Q, G, T) \leqslant (1+\epsilon)\ \text{OPT}.$$

Proof. Using binary search, it takes $O(\log(2\,\text{APP}'/3 \cdot 3/\epsilon\,\text{APP}')) = O(\log 1/\epsilon)$ time to find the smallest value μ' for which the approximated admissible region of r is not empty. A single relaxed decision problem for a $\mu \in [\text{APP}'/3, \text{APP}']$ can be decided in $O\left(n\sqrt{1/\epsilon}\right)$ time: as shown by Rote [40], any convex planar figure can be approximated by a convex polygon that circumscribe the figure and has $O\left(\sqrt{B/\lambda}\right)$ points on its boundary and is in λ Hausdorff distance to the figure, where B is the length of the boundary of the figure. This strategy is used to approximate admissible regions: any admissible region is defined as – or intersected by – a disc of radius μ and is inflated (by taking the Minkowski sum) by a disc of radius μ. Hence any admissible convex region can be covered by a disc of radius 2μ which bounds the length of the boundary of an admissible region to $4\pi\mu$. By choosing $\lambda = \epsilon \cdot \text{APP}'/3$ we have that any inflated admissible region can be approximated by a convex polygon using $\mathrm{O}\left(\sqrt{4\pi\mu/\epsilon \cdot \text{APP}'/3}\right) = \mathrm{O}\left(1/\sqrt{\epsilon}\right)$ vertices, as $\mu \leqslant \text{APP}'$. Each region $\mathrm{infl}_\mu(s)$ for all nodes $s \in S \setminus \{r\}$ is intersected exactly once to gain the (approximated) admissible region of the parent of s. As shown by Toussaint [45], two convex polygons can be intersected in linear time, which leads to a total runtime of $\mathrm{O}\left(n/\sqrt{\epsilon}\right)$ to compute a single relaxed decision problem instance. □

6.4 Non-Uniform Matchings for Point Sets

In Problem 5 (stated in Section 6.1) we assumed that the correspondence of the points in P and Q is given. That is, point p_i is mapped to q_i and the objective function with respect to p_i is influenced by the Euclidean distance of $t_i + p_i$ to q_i. In this section, we consider a variant of the problem where this correspondence is not given, while still one translation for each $p \in P$ is computed. Instead of the 1-to-1 distance stated Equation 6.1, we consider the directed Hausdorff distance of the set P to the set Q.

The problem considered in this section can be formulated as:

Problem 8. *Given a point set $P = \{p_1, \dots, p_n\}$ and a point set $Q = \{q_1, \dots, q_m\}$ and a neighborhood graph $G = ([n], E)$, compute a set of translations $T = \{t_1, \dots, t_n\}$ minimizing*

$$\mathrm{dist}(P, Q, G, T) := \max\left(\max_{p_i \in P} \min_{q \in Q} \|(t_i + p_i) - q\|, \max_{i,j \in [n]} d_{ij}\|t_i - t_j\|\right).$$

Note, that the objective function of Problem 8 is *not convex* due to the minimum function in the distance measure in feature space. The non convexity can easily be seen by the following simple 1-dimensional example: let $P := \{0\}$ and let $Q := \{-1, 1\}$, then the objective function reduces to $\min(\|t-1\|, \|t+1\|)$ which is clearly not convex, see Figure 6.8.

This problem therefore has no convex program formulation, but we can apply some of the geometric insights of the previous sections to gain approximation algorithms for Problem 8.

6.4.1 Exact Solutions

No exact polynomial algorithms are known to solve Problem 8. A simple exact but exponential algorithm is to *guess* the assignment of the Hausdorff distance, i.e., to try for all $p \in P$ all potential nearest neighbors $q \in Q$ and to solve for the $\mathrm{O}(m^n)$ instances the convex programming problem for point sequences as presented in Section 6.1.1.

6.4.2 Complete Graphs in the Plane – with Hausdorff Distance

Here, we consider complete neighborhood graphs for point sets in the plane, as in Section 6.2.2. The approximation idea presented in Theorem 13 can be adopted to compute constant-factor approximations for Problem 8.

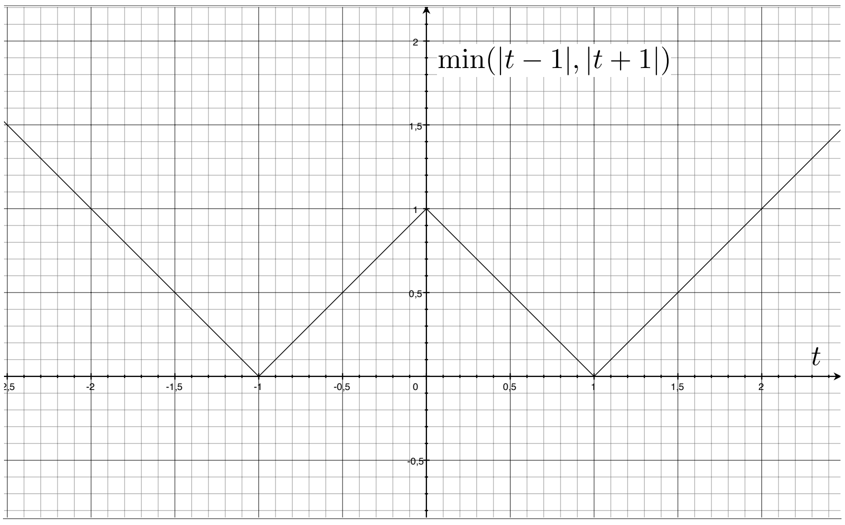


Figure 6.8: Graph of the non convex objective function of $P = \{0\}$ and $Q = \{-1, 1\}$.

Consider the point sets $S_i := \{q - p_i \mid q \in Q\}$ for $i \in [n]$ and imagine the points sets S_i to be colored in different colors. Each point $s \in S_i$ is a translation with the property that the distance of $s + p_i$ to (some point of) Q is zero, see Figure 6.9.

An optimal solution set $T = \{t_1, \ldots, t_n\}$ has the property that T itself has a diameter of OPT and that for each $t_i \in T$ there is a point $s \in S_i$ with distance $\|s - t_i\| \leqslant$ OPT. In contrast to Section 6.2.2, it is not known in advance which point $s \in S_i$ will achieve a distance of at most OPT to t_i.

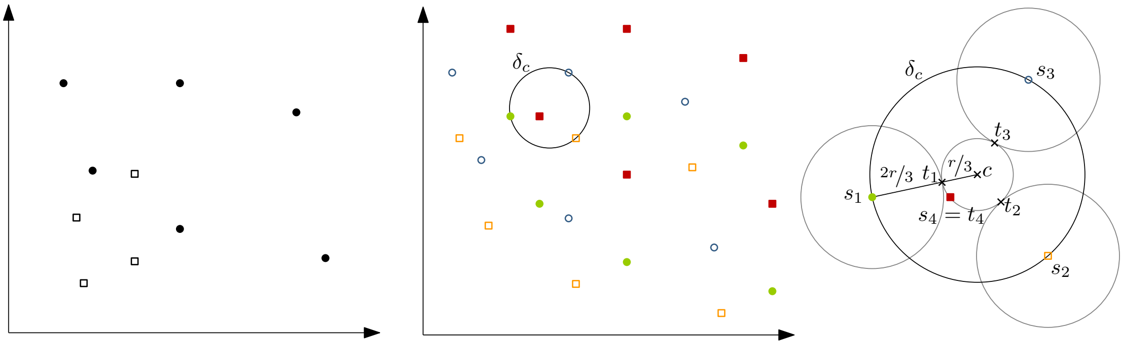


Figure 6.9: **left:** point set P (squares) and Q (discs). **middle:** point sets $S_1, \ldots, S_4$ and the smallest diameter disc δ_c that contains a point of each set S_i. **right:** the smallest diameter disc δ_c and the translation set that realizes the constant-factor approximation.

Given a smallest disc δ_c that covers at least one point of each set S_i, we can apply the approximation idea described in Theorem 13 and gain a $^2/_3\,(1 + ^1/\sqrt{3})$-approximation for the Hausdorff distance setting.

For a point set of n points in the plane that is colored with k different colors, the smallest disc that contains at least one point of each color can be computed in $O(k\,n \log n)$ time, as shown by Huttenlocher et al. [25] and Sharir and Agarwal [43, Section 8.7].

Theorem 17. *Let c be the center of the smallest disc δ_c that contains at least one point of each set $S_i = \{q - p_i \mid q \in Q\}$ and let r be radius of δ_c. For each $i \in [n]$ let s_i be a point of S_i that is covered by δ_c. Choosing t_i as the intersection o_i of the straight line $\overline{s_i\,c}$ with the circle δ_{app} with center c and radius $r/3$, or $t_i = s_i$ if δ_{app} does not intersect $\overline{s_i\,c}$, results in a ${}^2\!/_3\,(1 + {}^1\!/\sqrt{3})$-approximation that can be computed in $O\left(m\,n^2 \log m\,n\right)$ time:*

$$\operatorname{dist}(P, Q, G, (t_1, t_2, \ldots, t_n)) \leqslant {}^2\!/_3\,(1 + {}^1\!/\sqrt{3})\ \textsc{opt} \approx\ 1.05157\,\textsc{opt}\,.$$

Note, that the translation set T that is computed in Theorem 17 is optimal, given that the similarity of the translation set is measured by the diameter of the smallest enclosing disc of T.

Part II

Praxis

Chapter 7

Study on Measurement Accuracy

In the original formulation of a hybrid registration problem (see Definition 4 on page 5) one has to compute a transformation t of a given transformation class $\mathcal{T}$ that minimizes the following objective function:

$$\max\left(\vec{h}\left(t(P),\mathcal{S}\right),\vec{h}\left(t(P_c),Q_c\right)\right). \tag{7.1}$$

In the context of medical navigation systems, the characteristic points that are measured in pattern space P_c and their counterparts Q_c that are defined in the model are called *anatomical landmarks* (AL). In contrast to the anatomical landmarks, the set P of arbitrarily measured points that are taken from the measurable anatomic region of interest are called *surface or contour points* (CP).

When the quality of a registration is measured in terms of Equation 7.1, the influence of the distance between the measured and defined anatomical landmarks is assumed to be as important as the distance of the contour points to the surface. Taking a closer look at the measurement procedure suggests, however, to introduce a parameter to balance the two components.

In an actual registration workflow, both pattern features – the anatomic landmarks (P_c) as well as the contour points (P) – are *manually* measured from the region of interest using imprecise measuring devices. The positions of the anatomic landmarks in the model (the set Q_c) are determined also *manually* in the operation planning phase by marking their position in the slice images of the CT or MRT scan. During this process of measuring and defining the input data, various inaccuracies influence the actual representation (the measured positions) of the features.

With respect to the registration method, the involved errors can be classified into two categories:

Objective errors: These are inaccuracies that depend only on the imprecisions of the measuring devices that are used to determine the location of a point in the operation field.

Subjective errors: These are the inaccuracies that are caused by the surgeon or the medical staff by not defining the ALs precisely in the 3D-model during in the operation planning phase or by missing the exact position of an AL during the measurement.

The theoretical and empirical evaluation of these errors, especially the proportion of their influence on the deviation of the measured features to the *actual* positions of the features, is of great significance for developing hybrid registration algorithms.

In this chapter, different error types and their influence on hybrid registrations are investigated in theory and in experiments. Based on this studies, parameters are deduced and suggested to weight the two features in the objective function of a hybrid registration algorithm. In Chapter 5, we have presented algorithms that were especially designed to take weighting parameters into account.

7.1 Measurement Errors and Hybrid Registration Algorithms

In an ideal setting – in the absence of any errors or noise – there is a rigid body transformation $t : \mathbb{R}^3 \to \mathbb{R}^3$ so that

$$\vec{h}\,(t(P_c), Q_c) = 0 \wedge \vec{h}\,(t(P), \mathcal{S}) = 0.$$

As objective and subjective errors influence the representation of the measured features and the defined positions in real applications, the value of the objective function is usually larger than 0 at its minimum.

Let $\eta(t)$ be the error term that accounts for the distance of the transformed anatomic landmarks to their counterparts in the model and let $\mu(t)$ capture the deviation of the transformed set P to the surface $\mathcal{S}$, hence

$$\begin{aligned} \eta(t) &:= \vec{h}\,(t(P_c), Q_c) = \max_{p \in P_c} \min_{q \in Q_c} \|p - q\| \\ \mu(t) &:= \vec{h}\,(t(P), \mathcal{S}) \quad = \max_{p \in P} \min_{s \in \mathcal{S}} \|p - s\|. \end{aligned}$$

This allows us to rewrite Equation 7.1 as $\max(\mu(t), \eta(t))$. Under realistic conditions the expected magnitude of $\eta(t)$ is larger than $\mu(t)$, as $\eta(t)$ accounts for both, objective as well as the subjective errors, whereas only objective errors influence $\mu(t)$. The term $\mu(t)$ is not affected by subjective errors because a point $p \in P$ has no unique distinct target point in the model. The only constraint a rigid motion t has to fulfill with respect to a surface point p is that $t(p)$ has to be mapped close to *some* point of $\mathcal{S}$.

7.1.1 Analyzing the Influence of Noise on Measured Anatomic Landmarks and on Measured Contour Points

For this study we assume the surface $\mathcal{S}$ to be represented as a triangulated surface. Even if subjective errors could be eliminated completely, it still would be reasonable to introduce a parameter to balance the involved features in the objective, as the same deviation has a different influence on the objective function depending on whether an anatomical landmark or a contour point was measured.

Consider the experiment of measuring a point $q \in \mathbb{R}^3$ and let p be the point that was actually measured. Recall that $p = q$ only in absence of any error. We assume that the involved objective errors do not favor certain directions, i.e., that the error of measuring a single point q can be modeled as a ball b_q with center q.

for Anatomical Landmarks The effect of noise on $\eta(t)$ is equivalent to the Euclidean distance of a point p that is randomly chosen from the interior of b_q to q, see Figure 7.1 a).

for Contour Points The effect of noise on $\mu(t)$ is the distance of the measured point p to the triangulated surface $\mathcal{S}$ containing q – and not to q itself. We assume that the curvature of the surface $\mathcal{S}$ can be neglected in comparison to the diameter of b_q.

The error term $\mu(t)$ corresponds to the distance of p, randomly chosen from the interior of b_q, to a plane S that contains the triangle S_q of $\mathcal{S}$ from which q was measured. Under these assumptions, $\mu(t)$ is equal to the distance of p to its orthogonal projection $\hat{q}$ onto S, see Figure 7.1 b).

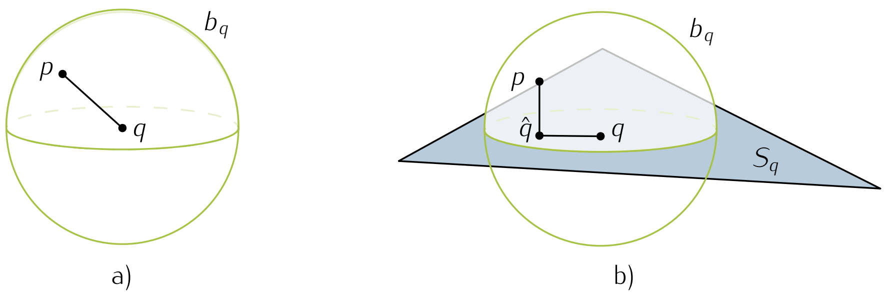


Figure 7.1: **a)** The distance of a measured characteristic point p to its target point q. **b)** The distance of a point p to the surface S.

7.2 An Error Model for Analyzing the Deviation of Measured Points

In the following, an error model will be presented that allows to analyze the effect of noise on the various feature types. For this study we assume that the measured features are in optimal position, i.e., that the identity function is optimal.

Each measured anatomical landmark $p_i \in P_c$ can be represented as the defined landmark $q_i \in Q_c$ plus a distortion vector $X_i \in \mathbb{R}^3$ that captures the effect of the objective error on the measurement of q_i. Likewise, every contour point $p \in P$ can be represented as a point $s \in \mathcal{S}$ plus a distortion vector $X_s \in \mathbb{R}^3$.

The process of measuring a point $q \in \mathbb{R}^3$ can be seen as a random experiment, where an elementary event (the distortion vector) X_q is drawn from the universe $\mathbb{R}^3$ of all possible distortion vectors. Let $Y(X_q)$ be the random variable that maps a distortion vector X_q to its length, i.e., $Y(X_q) = \|X_q\|$, where $\|\cdot\|$ denotes the Euclidean norm. In this study, we assume that the distribution of the objective errors does not favor certain directions of the distortion vectors. That is, we assume that the isosurfaces of the probability distribution function of Y are spheres with center q.

To quantify the effect of noise on the individual components $\eta(t)$ and $\mu(t)$ of the objective function, the expected magnitudes of these components within this random experiment are calculated.

Theorem 18. *The expected value of the distance of a measured anatomic landmark $p_i \in P_c$ to its corresponding landmark $q_i \in Q_c$ in the model (the expected magnitude of $\eta(t)$) is twice as large as the expected distance of any measured contour point $p \in P$ to its closest point on the plane containing the*

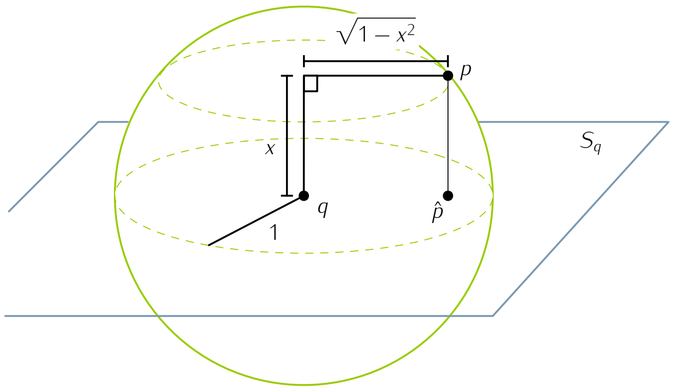


Figure 7.2: Illustration of the influence of X_q for contour points.

triangle S_p from which p was measured (the expected magnitude of $\mu(t)$):

$$\forall\, p_i \in P_c\ \forall\, p \in P \quad \mathrm{E}[\|t(p_i) - q_i\|] = 2 \cdot \mathrm{E}[\vec{h}\left(t(p), S_p\right)].$$

Proof. Let q be the point that was measured and let p be its distorted representation, i.e., $p = q + X_q$. The assumption that the isosurfaces of the probability distribution of $Y(X_q)$ are spheres around q allows us to fix the distance $\|q-p\|$ to a constant r and analyze the effect of distortion vectors of this magnitude. With respect to the described random experiment, we are now restricted to choose X_q uniformly from the sphere centered in q with radius r. The proportion of the expected values that contributes to $\eta(t)$ and $\mu(t)$ is invariant to the choice of r which allows to set r to 1.

In case of anatomic landmarks, the distortion vector contributes with its whole length to $\eta(t)$, as $\eta(t)$ is defined to be the Euclidean distance $\|q - p\|$ of the two points:

$$\mathrm{E}[\|q-p\|] = \mathrm{E}[\|X_q\|] = r = 1.$$

The analysis for a measured contour point is a bit more involved. The magnitude in which X_q contributes to $\mu(t)$ depends on the distance of p to its closest point $\hat{p}$ on the plane S_p that contains the triangle of $\mathcal{S}$ that in turn contains q, see Figure 7.2.

We need that the curvature of $\mathcal{S}$ is negligible compared to the diameter of the sphere with radius 1 around q to ensure that the distance $\|p - \hat{p}\|$ actually approximates the distance of p to $\mathcal{S}$. To analyze the expected value $\mathrm{E}[\min_{\hat{p}\in S} \|p - \hat{p}\|]$ we cut the upper half of the unit sphere around q into rings of height Δx. The area A of a ring can be expressed as $A = 2\pi\sqrt{1-x^2}\Delta l$ where $\Delta l = \frac{\Delta x}{\sqrt{1-x^2}}$ denotes the arc length of a ring and x is the height of the ring over S_p, see Figure 7.3.

We have that

$$A = 2\pi\sqrt{1-x^2}\,\Delta l = 2\pi\sqrt{1-x^2}\,\frac{\Delta x}{\sqrt{1-x^2}} = 2\pi\,\Delta x.$$

We now can express $\mathrm{E}[\min_{\hat{p}\in S} \|p - \hat{p}\|]$ as the following integral:

$$\mathrm{E}[\min_{\hat{p}\in S} \|p - \hat{p}\|] = \frac{1}{2\pi} \int_0^1 2\pi\, x\ dx = \frac{1}{2}x^2\Big|_0^1 = 1/2.$$

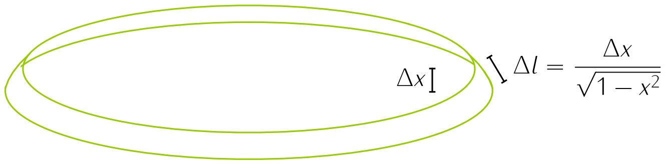


Figure 7.3: A height ring of the upper half-sphere.

□

This analysis shows that it is reasonable to introduce a weighting parameter of at least $c = 2$ to a combined objective function of the form

$$\max(c\,\mu(t), \eta(t)),$$

even if subjective errors could be ignored completely.

7.3 Empirical Experiments

The results of the theoretical studies have been verified by empirical studies. These studies have been conducted in cooperation with PD Dr. Olaf Suess and Dr. Sven Mularski from the Klinik und Hochschulambulanz für Neurochirurgie, Charité-Universitätsmedizin Berlin and Robert Günzler and Udo Warschewske from Prosurgics Ltd..

7.3.1 Experimental Setup

Preoperative
The empirical data was acquired based on routine use of neuronavigation in 15 patients with intracranial lesions. An MRT 3D navigation image dataset was generated preoperative for all patients. This dataset consists of a 3D MP rage sequence with isotropic voxels of 1mm long edges. Seven fiducial markers were previously affixed to the patient's head in a distribution that was as nonlinear as possible, using a two-component technique (self-adhesive skinpad and removable fiducial marker-fastener). The image-data processing and autosegmentation was performed with the navigation software *Guideline* (Prosurgics, Berlin, Germany). This navigation system performed the segmentation of the skin and brain surfaces, as well as the extraction of the positions of the fiducial markers. Additionally, several anatomic landmarks were located, among them three particularly suitable landmarks:

AL1	the outermost right eye corner,
AL2	the root of the nasion,
AL3	the outermost left eye corner.

Their positions were indicated on the slice image planes and on the 3D representation of the model, see Figure 7.4.

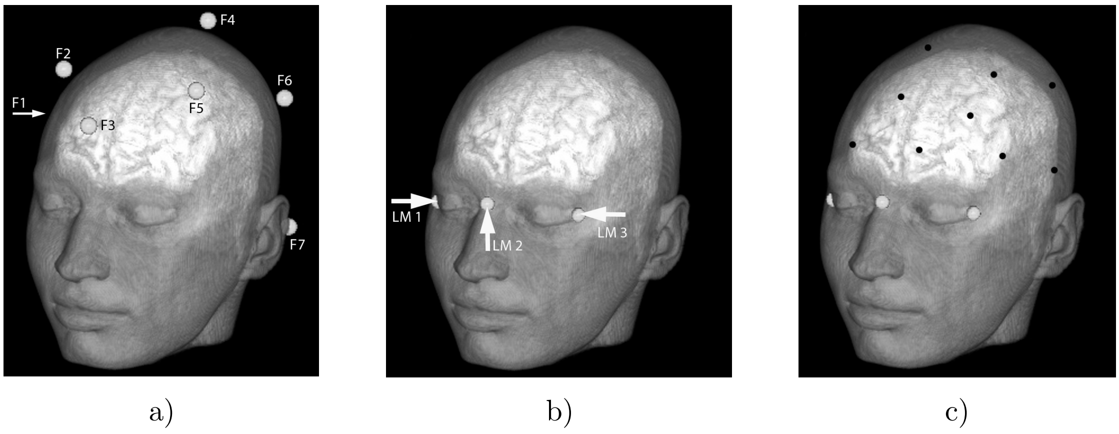


Figure 7.4: **a)** the position of the fiducial markers. **b)** the position of AL1–3. **c)** a set of suitable anatomic landmarks (gray) and contour points (black). [Illustration by permission of Prosurgics Ltd.]

Intraoperative
The intraoperative navigation was performed with an electromagnetic, sensor-based measuring technique (*miniBIRD*, Ascension Technology Corp., Burlington, Vermont, USA). After placing the patient on the operating table, the patient's head was fixed in a Mayfield support[1]. The comparative image-data registration took place in two steps:

1. a conventional fiducial marker registration (as described in Section 1.4.1) was performed by sequential measurement of the fiducial positions with a navigation stylus,
2. the anatomic landmark positions as well as several freely chosen contour points on the scalp surface were measured in a similar way.

7.3.2 Measuring Deviation

First, the measured anatomical landmarks and contour points are mapped into model space by the computed fiducial registration.

The deviation of an anatomical landmark is approximated by the Euclidean distance of its measured position in the pattern space to its defined position in the model space.

The deviation of a contour point is determined by the distance of the measured point to the closest voxel of the model that has been classified as skin in the skin segmentation process. A voxel is here modeled as a ball with a radius r that depends on the resolution of the underlying MRT image, $r = 0.5$mm for the images used in this study. The distance of a contour point p to the surface is consequently defined to be $\max(0, \|p - s\| - r)$, where s is the center of the voxel closest to p.

The adjustment parameters c that are suggested in the following section are based on the quotient of the average deviation of the measured anatomical landmarks and the average deviation of the measured

[1]a device in a ring like shape in which screws allow to fix the head of a patient so that the position remains invariant during the course of the operation

contour points.

Note on the Influence of the Aligning Fiducial Registration

The transformation with whitch the operation theatre is mapped to the model was computed based on a conventional fiducial registration method. Neither the anatomical landmarks nor the contour points have been used to compute this registration. The registration itself has influence on the measured deviations, due to inaccuracies of the involved devices that affect this registration process.

For this study, the error that is caused by the fiducial registration is regarded as an additional objective error, as it influences both, the measurement of the anatomic landmarks and the measurement of the contour points. The adjustment parameter that are suggested in the following evaluations should be regarded as pessimistic estimates because of the described inaccuracies.

7.3.3 Results

First, the results of 11 of the 15 data sets are presented that contain the *standard* landmarks AL1, AL2, and AL3, that were introduced in Section 7.3.1. In the second analysis, all data sets are considered and structured by the number of measured anatomical landmarks and the number of measured contour points.

Results for Data Sets Containing AL1, AL2 and AL3

An overview of the distances of the measured anatomic landmarks AL1, AL2, and AL3 from their defined positions in the model for 11 data sets is shown in Table 7.1 together with the average fiducial registration error (FRE) of each data set. The fiducial registration error is the root mean square distance of the measured fiducial markers after being mapped in the model space to the detected fiducial positions in the model. The FRE is an indicator for how good this particular data set has been aligned, i.e., the larger the FRE the more noise influenced the registration process. The average deviation of anatomical landmarks AL1, AL2 or AL3 from their defined position is 7.24mm over all relevant data sets.

	1	2	3	4	5	6	7	8	9	10	11	AVG
∅ FRE	3.02	2.18	1.49	2.78	1.35	2.26	2.71	2.31	2.08	1.63	2.36	2.20
AL1	4.34	8.51	2.12	7.72	7.31	13.01	7.36	15.29	14.93	11.03	6.15	8.89
AL2	9.32	3.36	2.71	10.66	0.72	8.73	3.63	5.75	7.05	6.39	4.87	5.75
AL3	7.48	3.14	2.23	6.99	4.38	10.83	7.91	11.39	6.3	8.53	8.65	7.07
AVG	7.05	5.0	2.35	8.45	4.14	10.86	6.3	10.81	9.43	8.65	6.55	**7.24**

Table 7.1: Measurement results for eleven data sets: average FRE, deviation of AL1, AL2 and AL3 (all values in [mm]).

Table 7.2 shows the deviation of these three landmarks compared to the minimal, maximal and average deviation of the contour points, differentiated by the number of measured contour points. Each row contains the values for the number of measured contour points indicated in the first column. The number in braces in the first column denotes the number of data sets that were considered in that row; a data set was considered if it contained a sufficient number of measured contour points. For each data set the average distance of the first i measured contour points to their closest surface voxel after registration was evaluated and the minimal, maximal and average values are shown in columns two to four.

	avg dist CPs surface			c = avg ALs / avg CPs		
#CPs	min	max	avg	min	max	avg
$\leqslant$ 1 [11]	0.0	6.59	1.87	0.96	655.48	85.6
$\leqslant$ 2 [11]	0.32	3.44	1.44	1.83	13.09	6.85
$\leqslant$ 3 [11]	0.21	3.04	1.41	2.61	19.64	7.76
$\leqslant$ 4 [11]	0.16	3.6	1.42	1.75	19.75	7.77
$\leqslant$ 5 [11]	0.13	3.85	1.54	1.64	18.69	7.4
$\leqslant$ 6 [11]	0.39	3.27	1.56	1.93	9.32	5.58
$\leqslant$ 7 [11]	0.34	2.8	1.49	2.25	10.5	5.75
$\leqslant$ 8 [11]	0.3	3.12	1.45	2.02	11.67	6.11
$\leqslant$ 9 [11]	0.46	2.77	1.49	2.27	13.13	5.83
$\leqslant$ 10 [11]	0.55	3.05	1.58	2.07	12.91	5.59
$\leqslant$ 11 [11]	0.53	3.21	1.62	2.27	13.28	5.59
$\leqslant$ 12 [11]	0.74	3.16	1.65	2.02	12.87	5.4
$\leqslant$ 13 [11]	0.73	2.97	1.62	2.18	9.85	5.13
$\leqslant$ 14 [11]	0.69	2.76	1.64	2.02	9.0	4.99
$\leqslant$ 15 [11]	0.65	2.59	1.61	2.16	8.77	4.91
$\leqslant$ 16 [11]	0.61	2.52	1.55	2.15	8.93	5.08
$\leqslant$ 17 [11]	0.57	2.37	1.49	2.25	8.59	5.2
$\leqslant$ 18 [11]	0.71	2.53	1.53	2.38	7.81	4.96
$\leqslant$ 19 [10]	0.68	2.39	1.52	2.33	8.25	4.98
$\leqslant$ 20 [10]	0.81	2.27	1.57	2.26	8.68	5.3
$\leqslant$ 21 [10]	0.86	2.2	1.58	2.29	8.21	5.27
$\leqslant$ 22 [10]	1.0	2.4	1.64	2.24	7.17	4.99
$\leqslant$ 23 [10]	1.01	2.48	1.67	2.26	6.97	4.84
$\leqslant$ 24 [8]	1.05	2.38	1.68	2.21	6.69	4.77
$\leqslant$ 25 [8]	1.27	2.36	1.78	2.21	6.33	4.69

Table 7.2: Min., Max. and Avg. values of the average distance of the contour points to the surface (in [mm]) and the resulting weighting parameter c. The numbers in braces in the first column indicate how many data sets were relevant for collecting the values of this row.

In columns five to six of Table 7.2 the minimal, maximal and average quotient of the average deviation of the anatomic landmarks AL1, AL2 and AL3 (see Table 7.1) and the average distance of the first i contour points is shown.

Results for Data Sets with Arbitrary Anatomic Landmarks

In clinical practice it is not always possible to measure all three described anatomic landmarks AL1, AL2 and AL3. Sometimes less than three landmarks or only landmarks at positions other than AL1, AL2 or AL3 are accessible. The resulting adjustment parameters for *arbitrary* anatomic landmarks can be discerned from Table 7.3. The columns of Table 7.3 are ordered increasingly by the number of measured anatomic landmarks and the rows are ordered increasingly by the number of measured contour points.

	#ALs				
#CPs	1	2	3	4	5
1	3.0 [15]	3.36 [13]	3.38 [12]	3.18 [8]	2.48 [5]
2	4.12 [15]	4.37 [13]	4.46 [12]	4.91 [8]	4.42 [5]
3	3.86 [15]	4.14 [13]	4.3 [12]	5.14 [8]	4.23 [5]
4	3.56 [15]	4.04 [13]	4.37 [12]	5.2 [8]	3.96 [5]
5	3.44 [15]	3.8 [13]	4.07 [12]	4.74 [8]	4.19 [5]
6	3.4 [15]	3.82 [13]	4.06 [12]	4.9 [8]	4.33 [5]
7	3.53 [15]	3.89 [13]	4.19 [12]	5.23 [8]	4.57 [5]
8	3.6 [15]	3.98 [13]	4.29 [12]	5.27 [8]	4.54 [5]
9	3.6 [15]	3.89 [13]	4.2 [12]	5.32 [8]	4.56 [5]
10	3.24 [15]	3.61 [13]	3.91 [12]	4.84 [8]	3.95 [5]
11	3.24 [15]	3.55 [13]	3.84 [12]	4.76 [8]	3.94 [5]
12	3.21 [15]	3.51 [13]	3.79 [12]	4.7 [8]	3.77 [5]
13	3.3 [15]	3.59 [13]	3.87 [12]	4.74 [8]	3.93 [5]
14	3.29 [15]	3.56 [13]	3.82 [12]	4.9 [8]	4.08 [5]
15	3.23 [15]	3.62 [13]	3.88 [12]	4.85 [8]	4.16 [5]
16	3.36 [15]	3.75 [13]	4.01 [12]	5.05 [8]	4.28 [5]
17	3.44 [15]	3.86 [13]	4.16 [12]	5.21 [8]	4.42 [5]
18	3.43 [15]	3.81 [13]	4.08 [12]	5.04 [8]	4.27 [5]
19	3.43 [15]	3.83 [13]	4.1 [12]	5.12 [8]	4.49 [5]
20	3.51 [14]	3.94 [12]	4.21 [11]	5.28 [8]	4.66 [5]
21	3.52 [14]	3.92 [12]	4.18 [11]	5.18 [8]	4.5 [5]
22	3.4 [14]	3.78 [12]	4.04 [11]	4.97 [8]	4.4 [5]
23	3.39 [14]	3.74 [12]	3.99 [11]	4.87 [8]	4.34 [5]
24	3.34 [13]	3.69 [12]	3.93 [11]	4.82 [8]	4.31 [5]
25	3.33 [11]	3.74 [10]	3.92 [9]	4.88 [7]	4.22 [4]

Table 7.3: Average quotient of the deviation for anatomic landmarks and contour points. The numbers in braces in indicate how many data sets were relevant for the cell.

Most hybrid registration algorithms that are designed for computing transformations based on exactly k anatomical landmarks, try all combinations of choosing k landmarks from the set of measured landmarks, given that more than k landmarks were measured and defined. Therefore, a cell in the i^{th} row and the j^{th} column of Table 7.3 contains the average quotient of the average distance of the best (closest) j landmarks and the average distance of the first i measured contour points over all relevant data sets. Only those data sets were considered for a cell that contained at least i measured contour points and at least j measured landmarks. The number in braces denotes the number of data sets that satisfy the constraints of a particular cell.

Chapter 8

Implementation

Along with the theoretical research that has been presented in the first part of this thesis, a framework named RegistrationToolKit has been developed to realize proof of concept implementations of most of the registration algorithms that we introduced in the previous chapters. The framework and the algorithms implemented in RegistrationToolKit currently comprise 30, 397 lines of code and provide a vast set of functionalities and methods, among them:

- a library for elementary geometric computations and operations that are common for many rigid motion registration algorithms,
- a user interface to set all parameter of the algorithms and to start and control the computation
- a visualization interface that allows to illustrate geometric primitives, triangulated surfaces and voxel models for visualizing registration results,
- introducing standardized representation formats for models, input files and registrations.

The framework itself as well as the implemented algorithms are written in the programming language `C++` using the free and platform independent Qt library [39] (developed by Nokia) to realize the user interface and OpenGL for visualizing geometric primitives.

In this chapter, we give an overview over the concept and the general design of the framework and some of its functionalities.

8.1 Requirements

The general idea that lead to the development of the RegistrationToolKit framework is to provide a consistent and standardized environment in which registration algorithms can be implemented, tested and evaluated. The focus of the concept and design is on supporting the following key functionalities:

1. to introduce standard input formats so that different algorithms or implementations can be applied on the same data set,

2. to provide an intuitive user interface that allows to test and apply the implemented algorithms,
3. to introduce common interfaces between the implemented algorithms so that algorithms can be called comfortably as subroutines within other algorithms,
4. to be able to evaluate the performance and quality of different algorithms in the same environment (e.g., by using the same library for elementary geometric operations),
5. to make the implementation of an algorithm independent of its visualization, i.e., to allow to call the algorithm without any visualization context,
6. to introduce a lightweight messaging service to allow an implementation to interact with its environment (e.g., a medical navigation system)
7. providing a suitable set of operations/functions to handle standard tasks that usually surround implementations of registration algorithms (e.g. for reading models, visualizing results, storing registrations).

8.2 The Framework

The RegistrationToolKit framework consists of several components that are shown in Figure 8.1.

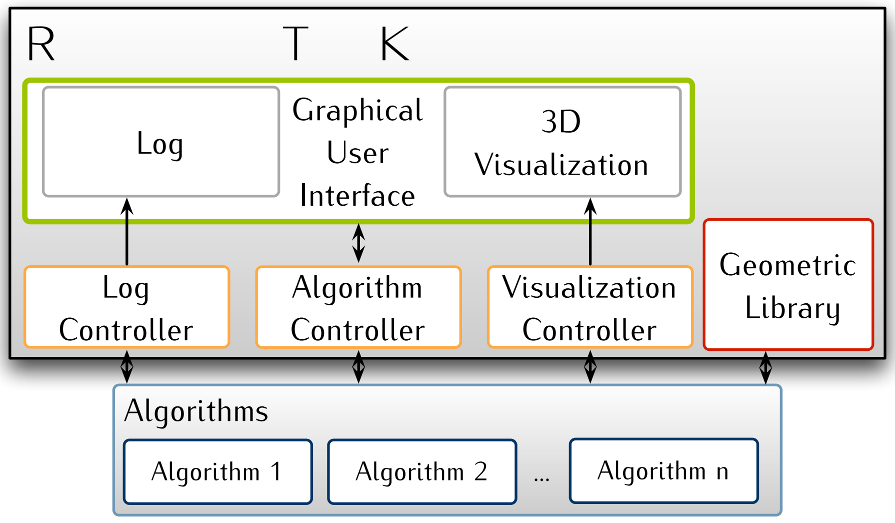


Figure 8.1: High level block diagram of the components and relations of the components of RegistrationToolKit.

The three main components are the set of implemented algorithms, the library of geometric methods and functions and the graphical user interface (GUI). The Interaction between the algorithms and the components of the user interface and some other aspects of the framework are discussed in the following section.

Input Formats

Most of the algorithms that are implemented in the framework compute point-to-surface registrations. Some of the algorithms that are presented in the theory part distinguish between triangulated surfaces and models that are represented as voxel data. In the implementation this differentiation is circumvented by introducing a common abstract (in `c++` jargon *virtual*) superclass `Model` that is extended by two classes named `TriangulatedModel` and `VoxelModel`.

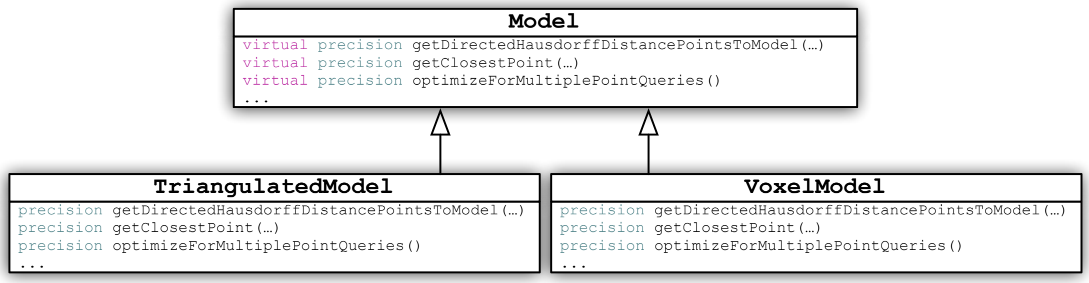

Model
virtual precision getDirectedHausdorffDistancePointsToModel(...)
virtual precision getClosestPoint(...)
virtual precision optimizeForMultiplePointQueries()
...
TriangulatedModel
precision getDirectedHausdorffDistancePointsToModel(...)
precision getClosestPoint(...)
precision optimizeForMultiplePointQueries()
...
VoxelModel
precision getDirectedHausdorffDistancePointsToModel(...)
precision getClosestPoint(...)
precision optimizeForMultiplePointQueries()
...

Figure 8.2: UML-diagram of the abstract `Model` class and two of its implementations.

All methods and functions through which algorithms can interact with models (surfaces) are defined *abstract* in the `Model` class and hence force their implementation in all classes that inherit from this class. By this abstraction it is not necessary to distinguish the type of model in an algorithm. The implementation details that require knowledge about the geometric structure of the model are hidden in the specific implementation of the respective methods of the specific model type. Another advantage of this this approach is that the implemented algorithms can directly be applied to other surface types without the need of changing the algorithmic code. All that has to be taken care of, is to implement another surface class that extends the `Model` class which has to implement all methods of `Model` that are declared *virtual*.

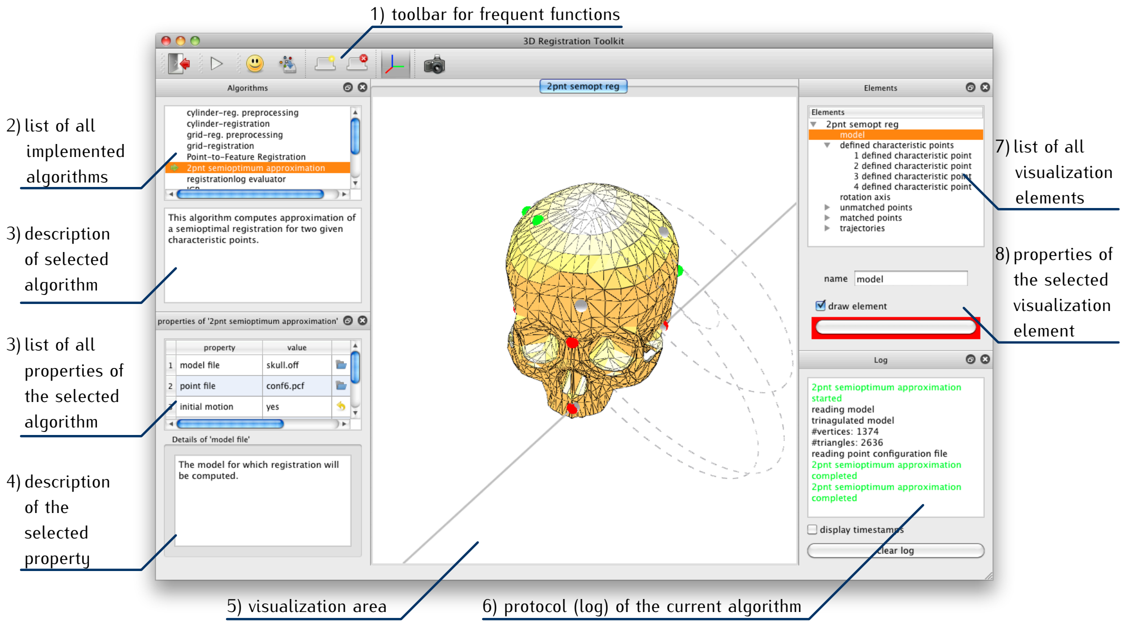
Some of the abstract geometric functions that are defined in the `Model` class are:

- `getDirectedHausdorffDistancePointsToModel(const QList<Point3D*> &points)`
  which returns the directed Hausdorff distance of a point set given by a constant reference to a list `points` to the model.
- `getClosestPoint(const Point3D &point)`
  which returns the point of the model that is closest to a query point given as a constant reference to a point `point`.
- `optimizeForMultiplePointQueries()`
  the function is called whenever multiple queries of the above listed methods are intended. Calling this function indicates that it is reasonable to invest preprocessing time in building data structures that support fast point location queries so that the aforementioned methods can be processed faster.

User Interface

The main window of the user interface of the RegistrationToolKit framework is subdivided into tree parts:

1. an area in which algorithms can be selected and their properties (e.g., approximation parameter) can be specified [see Elements 2-4 in Figure 8.3],
2. an visualization region in the center of the window [see Element 5 in Figure 8.3],
3. a section in which the elements of the visualization can be manipulated [see Elements 7,8 in Figure 8.3] as well as a log in which messages that are produced during the computation of the current algorithm are shown [see Element 6 in Figure 8.3].


1) toolbar for frequent functions
3D Registration Toolkit
Algorithms
cylinder-reg. preprocessing
cylinder-registration
grid-reg. preprocessing
grid-registration
Point-to-Feature Registration
2pnt semioptimum approximation
registrationlog evaluator
2pnt semopt reg
Elements
Log
2) list of all implemented algorithms
3) description of selected algorithm
3) list of all properties of the selected algorithm
4) description of the selected property
5) visualization area
6) protocol (log) of the current algorithm
7) list of all visualization elements
8) properties of the selected visualization element
Elements
2pnt semopt reg
model
defined characteristic points
1 defined characteristic point
2 defined characteristic point
3 defined characteristic point
4 defined characteristic point
rotation axis
unmatched points
matched points
trajectories
display timestamps
clear log

This algorithm computes approximation of a semioptimal registration for two given characteristic points.

properties of '2pnt semioptimum approximation'

	property	value
1	model file	skull.off
2	point file	conf6.pcf
3	initial motion	yes

Details of 'model file'

The model for which registration will be computed.

name: model

[✓] draw element

```
2pnt semioptimum approximation
started
reading model
trinagulated model
#vertices: 1374
#triangles: 2636
reading point configuration file
2pnt semioptimum approximation
completed
2pnt semioptimum approximation
completed
```

Figure 8.3: Screenshot of the RegistrationToolKit user interface.

An usual workflow for applying an algorithm in RegistrationToolKit is the following: after starting the program, an algorithm is chosen from the list of all algorithms [Element 2]. Selecting an algorithm from this lists causes that a detailed description of this algorithm is shown in a field [Element 3] just below the list of all algorithms. Also, all properties, variables and settings of the selected algorithm are shown in a list-style [Element 4].

A property can

- ask for a file/destination, e.g., an input file in which a (voxel-/triangulated-) model is specified or input points are stored,
- ask to specify a numeric value, e.g., the level or recursions or the approximation parameter,

- ask to set a flag, e.g., whether or not a random rigid motion should be applied to the measured points.

Default values are suggested for each property except for those properties that ask for files. When a property is clicked on, a detailed description of its influence on the algorithm is shown in a field [Element 5] below the parameter list. After selecting the desired algorithm and after specifying its parameters, the 'play' button in the toolbar [Element 1] is selected, which starts the selected algorithm.

Depending on the implementation of the selected algorithm, a progress bar at the bottom of the main window is shown to indicate the progress of the computation and the log area [Element 6] in the lower right part of the window displays status messages that are emitted by the implementation during the computation phase.

During or after the computation, geometric objects that are involved in the computation of the algorithm (e.g., the model, the measured characteristic points and the matched characteristic points) are visualized in the OpenGL component [Element 5] in the center of main window. Dragging the mouse in this field changes the viewpoint and using the mouse wheel allows to zoom onto the focused object. The element that is focused on can be selected by double-clicking the element in the list of all visualization elements [Element 7].

Library of Geometric Functions

Part of the framework is a library of geometric functions that are commonly used in the context of rigid motion registration. All functions of the class `ElementTools` (the implementation of the library) are defined *static*, implying that they can be used within any context, in particular without the need to instantiate an algorithm. Most geometric functions that compute intersections or projections are based on solving linear equations that result from a linear algebra formulation of the solution.

These are some of the functions provided by the class `ElementTools`:

`rotateAroundAxis(const Point3D &origin, const Point3D &direction, precision angle)`

returns an four by four matrix of type `Matrix44` that corresponds to the rigid motion transformation of a rotation by an angle of `alpha` around a line that contains the point `origin` and has the direction vector `direction`.

`getAligningRigitMotion(Point3D p1, Point3D p2, Point3D q1, Point3D q2)`

returns an four by four matrix of type `Matrix44` that corresponds to the rigid motion transformation t that transforms the points `p1` and `p2` so that the midpoints of the line segments $\overline{\texttt{p1},\texttt{p2}}$ and $\overline{\texttt{q1},\texttt{q2}}$ are congruent and that the line through `p1` and `p2` is congruent to the line through `q1` and `q2` and additionally $t($`p1`$)$ is closer to `q1` than to `q2`. This operation is commonly used by registration algorithms that compute (approximative) semioptima for two available characteristic points, see Chapter 3.

`getDistanceOfPointToTriangle(const Point3D &point, const Triangle &triangle)`

returns the distance of the point `point` to the triangle `triangle`, which is defined as the Euclidean distance of `point` to the closest point on the triangle. The implementation of this function is a bit involved as one has determine on which facet the closest point to `point` is. The closest point can lie in the interior of the triangle (2-facet), lie on an edge (1-facet), or be one of its corners.

`getClosestPointInTriangle(const Point3D &point, const Triangle &triangle)`

returns the point of the triangle `triangle` that realizes the shortest distance to the point `point`, see also `getDistanceOfPointToTriangle(...)`.

```
computeIntersectionsOfCircleWithBox(const Circle &circle, const AABox &box,
   QList<Point3D> &intersections)
```

returns the number of intersections of the circle `circle` with the axis aligned box `box`. The intersection points are computed by equalizing the equation representing the circle with the equations of the six rectangles that define the box. A circle can have between 0 and 12 intersections with a cube. The intersections are stored in the container `intersections`, if the circle intersects the box.

```
getIntersectionOfPlanes(const Point3D &ao, const Point3D &an, const Point3D &bo,
   const Point3D &bn, Point3D &ro, Point3D &rd)
```

returns FALSE if the plane a represented by the position vector `ao` and normal vector `an` is parallel to the plane b defined by the position vector `bo` and normal vector `bn` (i.e., if `an` and `bn` are parallel). Otherwise, the function returns TRUE and additionally stores in `ro` the position vector and in `rn` the direction vector of the line $a \cap b$.

```
getClosestPointsOnCircleToLine(const Circle &circle, const Line &line,
    QList<Point3D> &points)
```

returns the number of points on the circle `circle` that minimize the Euclidean distance of the circle to the line. These points are stored in the collection `points`. A circle can have one closest point to a line (if they touch or do not intersect) or can have up to two closest points if the circle and the line intersect. This function is for example used by the point-to-line registration algorithm described in Chapter 2.

Messaging System

The framework consists of several components (see Section 8.2) that need to interact. Deciding how to realize the control- and information flow as a common and reoccurring problem in software development. The interaction between the components in RegistrationToolKit is realized using the *observer* and the *inversion of control* design pattern [21]. These design pattern describe how the information flow is organized between a set of objects that want to react on a change of an internal state of a specific other object O. The central idea of this pattern is to provide the *observing* objects with a mechanism to register themselves at the object O that is to be observed. Whenever O changes an internal state it notifies all registrated observers.

In RegistrationToolKit, a certain variant of these pattern is used: for each specific state type (e.g. states that indicate a certain progress of the computation of a geometric algorithm) a so-called controller is introduced. Status changes of a specific type are communicated using specific implementations of the abstract `Message` class, see Figure 8.4. A change of state that for example indicates a progress change of the computation of a registration is communicated by an instance of the `AlgorithmMessage` class.

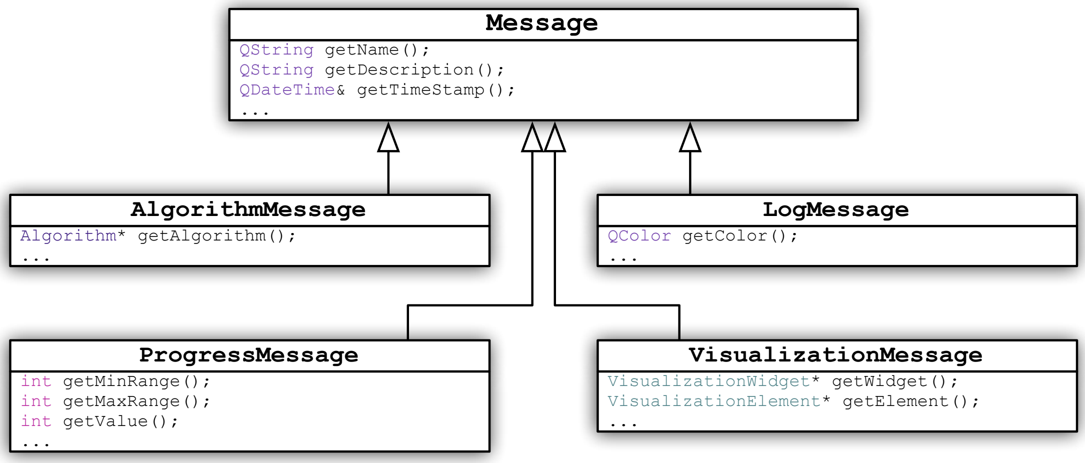


Figure 8.4: UML diagram of the `Message` class and all its implementations.

The objects that are interested in observing status changes of this specific type register to the corresponding controller. Whenever an internal state of an object O changes, a new message of the corresponding type is generated by O and delivered to the controller using the signal-slot technology provided by Qt. The controller than broadcasts this message to all of its registrated observers, see Figure 8.5.

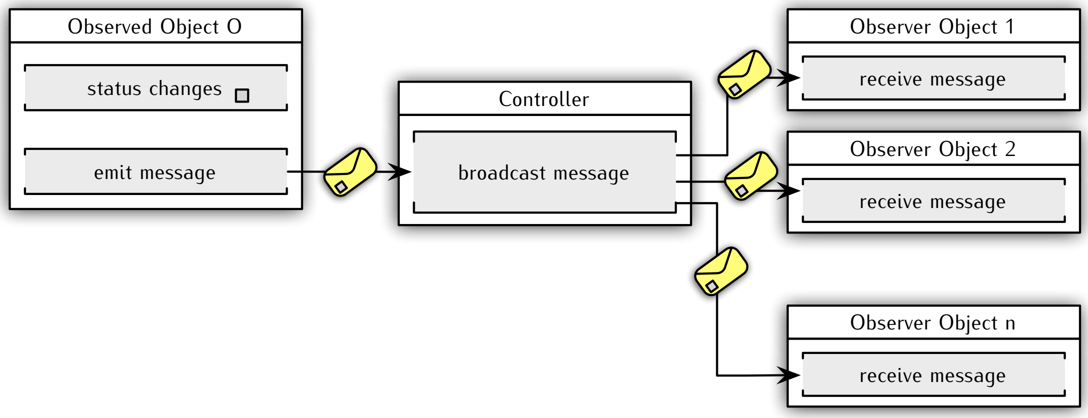


Figure 8.5: Diagram indicating the message delivery in RegistrationToolKit.

In RegistrationToolKit, four different massage types (classes) are distinguished:

`AlgorithmMessage` for communicating changes of the computation state, e.g., that the computation stated, or an error occurred, or the computation has finished.

LogMessage to communicate text messages that are to be displayed in a log.

VisualizationMessage to manage visualization elements, e.g., to add/remove/focus on an element to the OpenGL frame, see also next section.

ProgressMessage to indicate the progress of a computation, usually to control a progress bar that gives visual feedback of the current status of the computation.

Implementation vs. Visualization

An important conceptual decision in frameworks in which algorithms are implemented *and* visualized is, how to uncouple the implementation of an algorithm from its visualization. It is desirable to be able to call the implemented algorithms without any visualization context, e.g., in a command line program or capsulated in a modul or library.

One naïve solution is to introduce an internal state that stores the information whether or not a visualization context is given and only if this flag is set, specific visualization functions are called. This approach would result in code that mixes potentially highly optimized algorithmic code with visualization code which is not necessary at all to solve the algorithmic problem and is potentially never executed.

In RegistrationToolKit, visualizations are realized by exchanging messages (see previous section) between the class that contains the implementation of an algorithm and the framework (to be more precise: the visualization module of the framework). A two level uncoupling (described in the following by an example) is recommended to allow that the class that contains the actual algorithmic code for solving the specific geometric task does not contain any code for exchanging messages that deal with visualization issues. In Appendix B the implementation of an algorithm and its visualization is presented for further illustration.

Two level uncoupling by example
The framework provides an abstract class `Algorithm` that defines central functions of an algorithm, such as:

- **`run()`** to start the computation (more precise: a new thread is created that starts the algorithm) and
- **`reset()`** to reset all internal states, which is called before the algorithm is started.

This class also contains a member variable that stores a collection of `AlgorithmProperty`s which define the parameters (e.g., approximation parameters) of the algorithm.

Let `MyAlgorithm` be a class that implements an algorithm that is to be made available in the framework. In order to introduce this algorithm to the framework, `MyAlgorithm` has to extend (inherit from) the class `Algorithm`.

To visualize the geometric objects that are used by `MyAlgorithm` a new class, by convention named `MyAlgorithmVis`, is introduced that in turn inherits from `MyAlgorithm`. Instead of adding `MyAlgorithm` to the set of available algorithms of the framework, an instance of `MyAlgorithmVis` is added.

When the computation of an algorithm is triggered (see the *user interface* paragraph of this section),

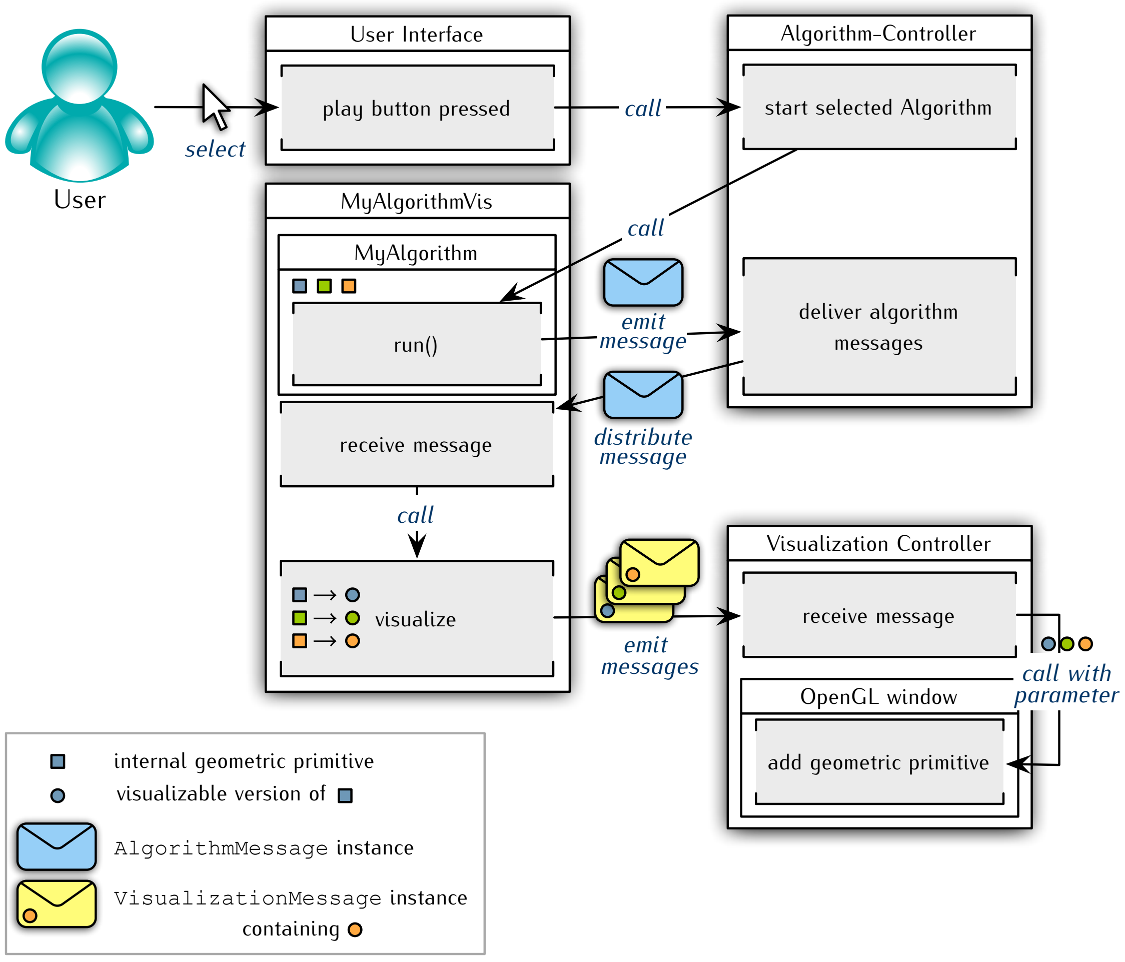


Figure 8.6: Flow diagram of the visualization process.

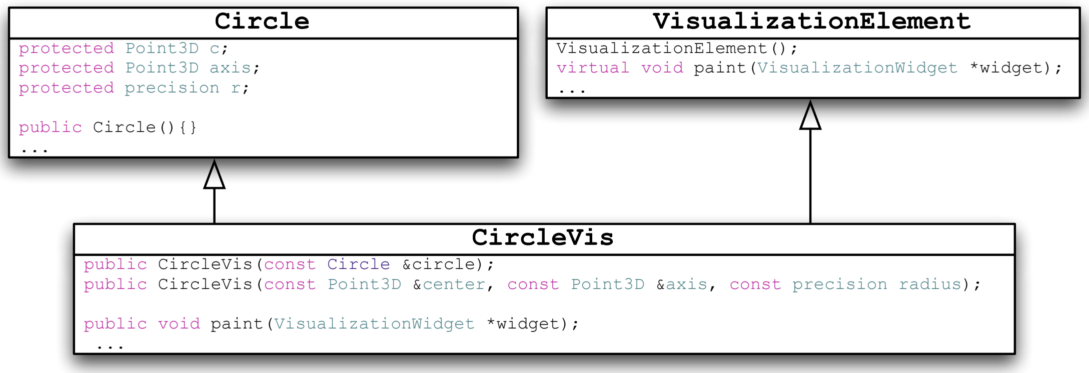
the **`run()`** function of the currently selected (in this case of **`MyAlgorithmVis`**) algorithm is called. Calling the **`run()`** method of **`MyAlgorithmVis`** actually calls the **`run()`** method of **`MyAlgorithm`** as the wrapper class (**`MyAlgorithmVis`**) by convention does not overwrite this function (or is assumed to call **`run()`** of its super class).

At the end of the geometric computation, an algorithm (here: **`MyAlgorithm`**) is expected to emit a corresponding **`AlgorithmMessage`**. This message is received and distributed by the algorithm-controller component of the framework to all registrated listeners (making use of the observer design pattern). All algorithms that are introduced to the framework are automatically registrated as listeners to this message type and hence the instance of **`MyAlgorithmVis`** also receives this message. After **`MyAlgorithmVis`** receives this message and by that detects that the geometric computation of its superclass is completed, a private function is called that takes care of the actual visualization of the geometric objects that are defined *protected* in **`MyAlgorithm`**. The visualization itself is usually done as follows: each geometric object that is stored as a *protected* member of **`MyAlgorithm`** (by that accessible by all classes that inherent from **`MyAlgorithm`**) and that is to be visualized is converted to its visualizable counterpart and send to the visualization-component of the framework by a **`VisualizationMessage`**, see Figure 8.6.

Visualization of geometric objects
Each geometric primitive (such as points, lines, triangles, circles, models, ...) that are part of the computation of a geometric algorithm has a counterpart that allows its visualization in an OpenGL context. Each class that represents a visualizable geometric object has to extend an abstract class **`VisualizationElement`** which enforces the implementation of a function **`void paint(...)`**. The object is drawn by calling this function with the OpenGL component on which it is to be drawn.

We will demonstrate the visualization capability of a geometric primitive by the example of the class **`Circle`**.

The class **`Circle`** only stores the information about its center, its radius and a normal vector of the plane in which the circle lies. The class **`CircleVis`** extends from **`Circle`** and **`VisualizationElement`** and hence implements the **`paint(...)`** function which draws the circle on the given widget, see Figure 8.7.



```
protected Point3D c;
protected Point3D axis;
protected precision r;

public Circle(){}
...
```

```
VisualizationElement();
virtual void paint(VisualizationWidget *widget);
...
```

```
public CircleVis(const Circle &circle);
public CircleVis(const Point3D &center, const Point3D &axis, const precision radius);

public void paint(VisualizationWidget *widget);
 ...
```

Figure 8.7: UML diagram of the relation between the **`Circle`**, **`CircleVis`** and **`VisualizationElement`** classes.

8.3 A Word About Precision and Data Types

Implementations of geometric algorithms have to deal with the problems of precision and numeric stability. As computers have finite storage and memory, it is not possible to store the *exact* values of all numbers that occur during a computation in any number system. Ignoring this problem by using finite precision representations will introduce rounding errors that propagate throughout the computations of an algorithm and might significantly alter the final result, e.g., by changing the topology of the result.

There are several ways to address this problem, thee typical strategies are:

1. using floating point arithmetic to represent values,
2. using a library that provides data types with variable (adaptive or definable) precision,
3. using computer algebra systems/libraries that provide methods for symbolic computing.

In RegistrationToolKit the first option – floating point arithmetic – is used to realize the computations that occur during the registration algorithms. The main reasons for this decision are:

Topology of the Involved Geometric Objects

The considered registration algorithms do not require a *stable* topology of the involved geometric objects. The objective functions that are minimized are point–to–surface distance measures; more precisely Hausdorff distances or variants of it. In this registration setting: the objectives are aggregates of point–to–feature distance functions where features here are either points, lines or triangles. The algorithms handle these features separately, i.e., even if the set of triangles constitute to a connected triangulated surface, the topology of that surface is not considered during the computations. Therefore, representing triangles (lines, points) using finite precision arithmetic coordinates could change the topology of the represented object but this does not significantly influence the necessary distance computations.

Requirements of the Cooperation Partner

A central motivation for implementing the algorithms that are presented in the first part of this thesis was to integrate them in real world medical navigation systems. The cooperation partner *Schaerer Mayfield Technologies GmbH* later taken over by *Prosurgics Ltd. Germany* required to use no additional libraries (but Qt) for license and license fee reasons. It turned out, that medical navigation systems that are currently in use internally also use finite precision arithmetic, see also next section.

Resolution and Precision of the Input Data

The algorithms that are implemented in this framework are designed to be used in real life medical navigation systems. In these settings, imprecision and rounding errors are already introduced by the coarse resolution of the input and the imprecision of the measuring devices. These errors exceed the imprecision that is caused by the use of finite precision data types by several orders of magnitude.

The unit that is internally used to store all geometric values and variables is millimeter (mm), i.e., a point with the coordinates $(2.0, 1.0, 3.0)$ is located 3mm in x-direction, 1mm in y-direction and 3mm in z-direction from the origin.

The most commonly used format for storing floating-point values is the IEEE-754 format. The *unit in last place* (ULP) of a number format is the absolute value between the *true* number and its representation in the considered format. In the IEEE-754 format in single precision (data type `float` in `C++`) the ULP of numbers that have an exponent of 0 is $\approx 10^{-7}$ and in double precision (data type `double` in `C++`) the ULP is $\approx 10^{-16}$.

In contrast, the resolution of standard clinical magnetic resonance imaging (MRI) systems is about 1mm and the resolution of common computed tomography (CT) systems is about 0.5mm. Modern (electro magnetic or optical) tracking devices with which points can be measured in the operation theatre also have a precision of about 1.8mm.

Comparing the inaccuracies that are due to imprecise inputs with the rounding errors that are caused by finite value representations shows that it is reasonable to use finite precision data types as the induced imprecision is negligible for problem instances in praxis.

Chapter 9

Conclusion and Future Work

In this thesis, a geometric problem has been studied that is of significant relevance in the context of modern computer assisted surgeries. The problem is to compute a mapping from an operation theatre to a model space that contains a 3D-model of the relevant anatomical area of a patient. Such a mapping, in this context also called *registration*, is a central component of *medical navigation devices*. These are systems that support the surgeon during an operation by projecting the currently used surgical instrument into the model at the correct relative position within the correct relative orientation. Navigated surgery is especially used during neurosurgical interventions where the tissue on which the operation is performed is typically occluded. This technique is primarily used for *rigid tissue registrations* where the point of interest (the spot where the operation is actually performed) is surrounded by rigid tissue – usually a tumor that is surrounded by the scull.

This research was motivated by the aim to enhance the reliability, to increase the stability of the computed registration, to improve the comfort for patient and medical personnel and to widen the application area of the currently used systems. Furthermore, we aimed for methods of resolution for the challenging task of computing *soft tissue registrations*. This problem is significantly harder than the problem of computing rigid tissue registrations, as one has to deal with local non-rigid tissue deformations, time dependencies (due to respiration) and the inability to fix reference points.

A conceptually new strategy has been investigated and evaluated in the light of the aforementioned applications: hybrid registration methods. These are registration techniques where the mapping problem is reduced to a series of (at least two) geometric matching problems which are interdependently and simultaneously solved to gain a solution for the initial question. The input for the individual geometric matching problems are: geometric features measured from the operation theatre on the one hand and geometric features defined in the model space on the other hand. Typically, the input differs semantically (anatomical landmarks or surface points) and with respect to the feature type (point sets or surface).

The underlying hypothesis that has been studied throughout this thesis is:

> Are hybrid registration methods appropriate techniques to solve real life registration problems as they occur in the context of navigated surgery?

9.1 Contribution

The contributions of this thesis can be classified in three groups:

1. The development and analysis of exact and approximative registration algorithms for various transformation classes and input types in $\mathbb{R}^2$ and $\mathbb{R}^3$ with focus on their applicability in the context of medical navigation systems for *rigid tissue registrations*. The focus was especially on algorithms that
   (a) guarantee quality bounds,
   (b) are able to compute *all* registrations of a certain quality,
   (c) provide robust results,
   (d) are efficient and implementable to be of practical interest.
2. The implementation of the analyzed and described strategies in an environment that is comparable to the intended application context – medical navigation systems. The realization of the algorithms in a framework that covers the relevant phases in the workflow of a medical navigation system proved that the presented approaches are indeed of practical relevance.
3. A new geometric strategy has been developed to address the problem of soft tissue registrations: *non-uniform geometric matchings*. This generalization of geometric matching problems allows to compute mappings that consist of *several* mutually dependent and locally valid transformations (instead of a single transformation). With this approach, it is possible to match several areas of interest individually by simultaneously considering their spacial and/or chronological neighborhood (to cover local deformations and time dependencies).

The concept of reducing the registration problem to hybrid geometric matching problems has been proven to lead to stable, efficient (in theory as well as in praxis) and reliable algorithms that provide quality guarantees for the computed solutions. Considering semantically different geometric sets as input for hybrid registrations allows to combine several geometric matching techniques from the field of computational geometry.

Consider the formulation of a hybrid registration problem, where a point-to-point matching problem is solved for the anatomical landmarks together with a point-to-surface matching problem for the surface points and the surface. For this setting it is possible to combine the reduction of the dimension of the search space (due to the point-to-point matching) with the stability of the comparably harder point-to-surface matching problem.

The presented algorithms cover a majority of the clinically relevant application fields. They are proven to meet the quality and computation time constraints; their proof-of-concept implementations in the developed framework support the theoretical results.

In addition to the contributions mentioned above, a further step has been made to close the gap between the theoretical analysis of hybrid registration algorithms and their actual use in medical navigation systems. In an empirical study about measurement inaccuracies of anatomical landmarks and surface points, weighting parameters have been deduced, see Chapter 7. These parameters are a crucial component for every system that uses hybrid matching algorithms to compute registrations.

9.2 Future Work

Further Theoretical Research

The concept of hybrid registration problems can be seen as a general framework where several details need to be specified to meet the requirements of a specific application. To solve a specific problem instance, one has to specify these components. To clarify the choices that have been made in this thesis, we repeat the formal definition of a hybrid registration problem.

Definition 2 (Hybrid Registration Problem).
Given:

$\mathcal{G}$	*a class of geometric objects*
$\mathrm{dist}_{\mathcal{G}} : \mathcal{G} \times \mathcal{G} \to \mathbb{R}^+$	*a distance measure on* $\mathcal{G}$
$\mathcal{P} = P_1, \dots, P_m$	*a sequence of geometric features with* $P_i \in \mathcal{G}$ *for* $i \in [m]$
$\mathcal{Q} = Q_1, \dots, Q_m$	*a sequence of geometric features with* $Q_i \in \mathcal{G}$ *for* $i \in [m]$
$\mathcal{T}$	*a set of admissible transformations of* $\mathcal{G}$
$f : \mathbb{R}^m \to \mathbb{R}$	*an aggregate function*

Task: *Compute a transformation* $t \in \mathcal{T}$ *minimizing*

$$f\left(\left(\mathrm{dist}_{\mathcal{G}}\left(t(P_i), Q_i\right)\right)_{i \in [m]}\right)$$

For the class of problem instances that have been considered in this thesis, the

class of geometric objects $\mathcal{G}$ was fixed to points, point sets and triangulated surfaces,
the distance measure $\mathrm{dist}_{\mathcal{G}}$ **in object feature space** was the directed Hausdorff distance or variants of it,
the transformation class $\mathcal{T}$ was fixed to rigid motions or translations,
the aggregate function f was fixed to the maximum of the individual matching result values.

Further research could extend the presented results to e.g., *almost rigid* affine transformations, i.e., affine maps with a transformation matrix that have a determinant that is almost 1. Also other distance measures such as the root mean squared distance as well as other aggregate functions such as the average value of the m individual matching problems could be considered.

For the challenging task of computing hybrid registration problems with a single anatomical landmark (Chapter 4), the problem of computing an exact rigid motion registration in $\mathbb{R}^3$ remains open as well as the problem of computing a polynomial-time approximation scheme.

The weighted directed Hausdorff distance as introduced in Chapter 5 is defined as the maximum of m Hausdorff distances, where each Hausdorff distance is multiplied by a scalar. As reasonable modifications of this measure, also the average of the m sub-measures could be considered. Also non-linear weighting methods could be considered for the individual Hausdorff distances that are involved. Again, computing rigid motion or almost rigid affine transformation registrations in $\mathbb{R}^3$ remains a challenging task with many applications in theory and praxis.

A natural candidate for extensions and further research is the promising approach of non-uniform matchings that has been introduced to deal with local deformations in rigid body registrations or with soft tissue registrations.

Open questions and tasks are:

1. to find a purely geometric algorithm (not a convex programming formulation) to compute non-uniform matchings for translations of arbitrary neighborhood graphs in the plane or in $\mathbb{R}^3$,
2. extend the results to rigid motion and almost rigid motion registrations in the plane and in $\mathbb{R}^3$, where important subproblems are
   (a) to formulate an adequate similarity measure for these transformation sets to compare transformations that are adjacent in the neighborhood graph,
   (b) (optionally) to *generate* a suitable partition of the pattern space so that there exists a non-uniform geometric matching that satisfies a given quality bound or optimizes the quality.

Non-uniform registrations could also be extended to consider time dependencies. A direct way to consider time in a non-uniform registration is to extend neighborhood graphs (here described for translations) to time dependent neighborhood graphs as illustrated in Figure 9.1. A time dependent neighborhood graph consists of copies of regular neighborhood graphs – one for each considered time step. For a time step t, the set S^t encodes the object space distances of the registrated features. A transformation t_i^t of this time step is connected by edges to the transformations t_i^{t-1} and t_i^{t+1} of the transformation in the adjacent time steps. These edges have to be sufficiently weighted to balance the constraints of similarity between a specific transformation in consecutive time steps and between transformations in the same time step (the weight should also depend on the length of the time span between two time steps).

Further Empirical Research

Two interesting tasks remain open that – due to the cessation of business of a cooperation partner – could not be completed.

Comparison of Methods on a Common Significant Data Basis
The algorithms presented in this thesis have been theoretically analyzed and their implementations have been tested in the framework that has been designed for this purpose. It remains open to compare the results presented here with the quality of registrations that are computed by medical navigation systems that are currently used in praxis. In this context, it would also be interesting to compare these approaches with other known techniques such as iterative closest point techniques (ICP), especially testing whether a downstream ICP phase significantly increases the quality of a registration.

To be able to do the intended study it is necessary to

1. collect a significantly large set of clinical data (CT/MRT images) along with a measurement of all geometric features that are required to compute registrations with all considered registration methods,
2. to perform a skin segmentation/surface extraction on the raw data to gain the model to which the measured features are registrated to (for hybrid registrations),
3. to define meaningful measures to compare the individual results as the different registration techniques optimize different objective functions.

As we believe that these algorithms indeed improve upon the existing systems, the final step is to integrate and test the developed algorithms in an actual navigation system. This step can only be realized in close cooperation with an industry partner that is established in the field.

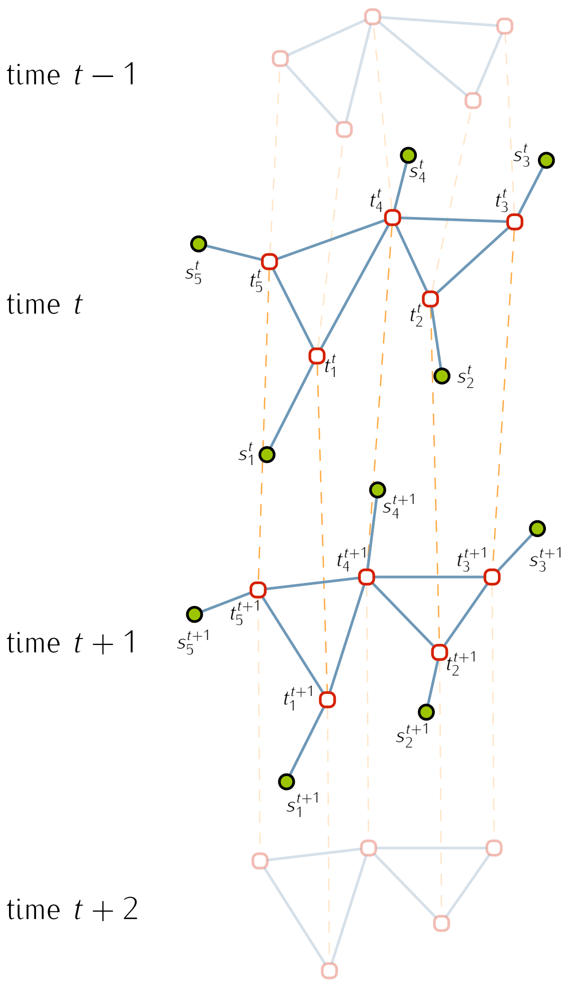


Figure 9.1: Illustration of a time dependent neighborhood graph for time steps $t-1$, t, $t+1$ and $t+2$. The dashed edges connect transformations between successive time steps.

Appendix A

Algorithms Implemented in RegistrationToolKit

In the following, several algorithms are presented that were implemented in RegistrationToolKit. A simple example of an implemented algorithm is given in Appendix B.

Point-to-Feature Registration (Chapter 2)

Parameter to be specified by the user:

input file path to the text file in which the rotation center/points/lines (or line segments) are stored.
line registration boolean flag indicating that a point-to-line registration (or if set to FALSE a point-to-line segment registration) is performed.

The point-to-feature approximation algorithm that is described in Chapter 2 has been implemented for features that are lines or line segments. The point-to-line approximation is rather simple. The central task is to compute all critical angles which are either defined by the closest point on a trajectory to a line, or their intersections.

If line segments are considered, the computation is slightly more involved. This is because the critical points (angles) have to be computed based on several case distinctions. It has to be distinguished whether

- the trajectory intersects/touches/does not intersect the line segment,
- wether none/one or both endpoints lie within the trajectory and
- whether or not the orthogonal extensions of the endpoints of the line segment intersect the trajectory.

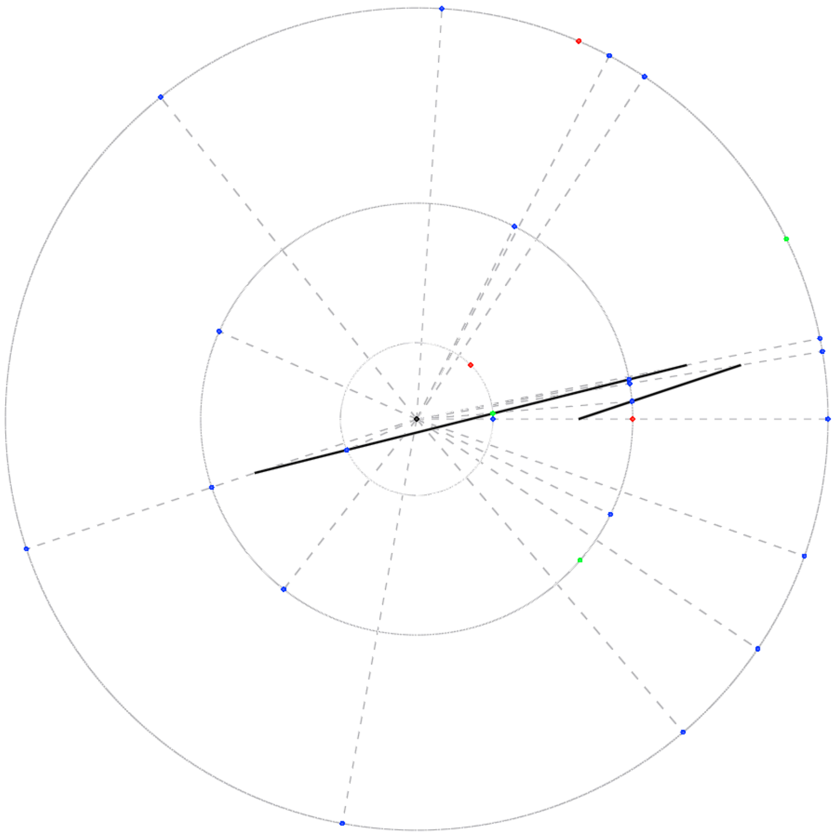


Figure A.1: Screenshot of the visualization of the point-to-feature registration.

Absolute-Error Approximations of Semioptima (Chapter 3)

In total four algorithms have been implemented to cover the absolute-error approximations that are described in Chapter 3. Two of these algorithms perform the preprocessing, one algorithm computes registrations based on the gird discretization technique and the forth algorithm computes registrations based on the cylinder discretization technique.

Parameter to be specified by the user for the preprocessing of the grid based registration algorithm:

model file path to the file in which the (triangulated or voxel based) model is described.
output file path to the file in which the preprocessed data is to be saved.
μ the discretization value.
dist. threshold optional parameter t; if set to $t > 0$ all directed Hausdorff distances $\vec{h}(c, \mathcal{S})$ of cells c that satisfy $\vec{h}(c, \mathcal{S}) \geqslant t$ are set to ∞ to ensure that only registrations are considered that result in a quality less than t.
safety the amount (in %) by which the bounding box (unit cube) is extended to cover surrounding space around the model. Example: if set to 100% all eight unit length boxes that surround the unit cube are also sampled.

As $O(1/\mu^3)$ point-to-model distance requests have to be performed during the preprocessing phase, the `optimizeForMultiplePointQueries()` function of the model is called. This function allows the implementation of the model to build internal data structures (e.g., quad-trees) that support fast point-to-model distance queries. The actual implementation of this function depends on the specific representation of the model.

Parameter to be specified by the user for the algorithm that computes a registration based on a cubical grid discretization:

model file path to the file in which the (triangulated or voxel based) model is described, for visualization purposes only. The surface information that is needed to compute the registration has been extracted in the preprocessing phase of the algorithm.
pp. data file path to the file containing the pre-processed data.
point file path to the file containing the landmarks and surface point information.
char. quality flag indicating whether the objective function is the maximum of the Hausdorff distance of the characteristic points in semioptimal position multiplied by a constant p and the best found position of the surface points to the surface. If set to FALSE only the Hausdorff distance of the registrated surface points to the surface is considered.
p see *char. quality*.
refinement flag indicating whether perturbations of the rotation axis should be considered to potentially gain better registration results. If set to FALSE only semioptimal configurations are tested. The perturbations are defined by introducing a cubical grid around the first anatomical landmark and a spherical grid around the second. All combinations of grid point pairs (one sample from the spherical grid one from the cubical grid) are tried as alternative positions for the characteristic points of the model, see Figure A.2.
cube length, cube step, angle range, angle step parameter that are relevant, if *refinement* is set to TRUE. The *cube length* w specifies the edge length of the discretization and the angle range α

the width of the spherical grid, see Figure A.2. *Cube step* defines in how many samples the cube is subdivided in each dimension and *angle step* defines how many samples are to be introduced along the latitude and longitude of the spherical grid.

The central function of this algorithm that has to be carefully implemented is the method that computes the intersections of a trajectory with a cube of the spherical grid in $\mathbb{R}^3$. The intersections have to be ordered according to the rotation angle that moves the measured point onto a boundary of a cell. Furthermore, it has to be ensured that all cubes are followed in the *same direction*, i.e., clockwise around the rotation axis for all trajectories.

1

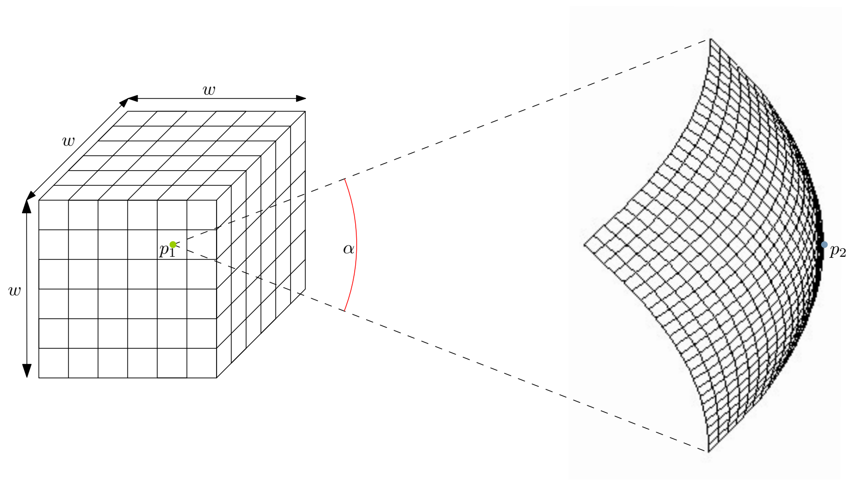


Figure A.2: Parameter of the perturbation option of the cubic grid registration algorithm for two defined and measured characteristic points.

Parameter to be specified by the user for the preprocessing of the cylinder based registration algorithm:

model file path to the file in which the (triangulated or voxel based) model is described.

point file path to the file in which the point sets are described (as the positions of the anatomical landmarks have to be known at preprocessing time).

output file path to the file in which the preprocessed data is to be saved.

μ the discretization value.

safety the amount (in %) by which the *outermost cylinder* is extended. A value of 10% e.g. extends the length rotation axis of the outer most cylinder as well as its radius by a factor of 1.1.

header only flag indicating weather or not only the header information of the preprocessing should be written to the output file.

As for the grid based preprocessing algorithm, the `optimizeForMultiplePointQueries()` of the model is called to optimize it for several point-to-model distance queries.

Parameter to be specified by the user for the algorithm that computes a registration based on a cylindric grid discretization:

model file path to the file in which the (triangulated or voxel based) model is described, for visualization purposes only. The surface information that is needed to compute the registration has been extracted in the preprocessing phase of the algorithm.
pp. data file path to the file containing the pre-processed data.
point file path to the file containing the landmarks and surface point information.

The central algorithmic task of this algorithm is:

1. to compute the slice and ring and the index of the initial sample for each measured surface point,
2. to find the angle intervals for which the largest sample-to-surface distance is minimal,
3. to combine the initial aligning motion with the corresponding rotations around the axis.

Registration with a Single Characteristic Point (Chapter 4)

This algorithm has been implemented twice. The first variant is described in Chapter 4.3.2 and only considers virtual characteristic points that are taken from a sphere centered in the defined characteristic point in the model space with radius $\|p - \hat{p}\|$.

Here, an implementation is described that follows the description given in Chapter 4.3.1.

Parameter to be specified by the user for the approximation heuristic for a single characteristic point:

model file path to the file in which the (triangulated or voxel based) model is described.
point file path to the file containing the landmarks and surface point information.
initial motion flag indicating whether or not the measured points are initially transformed by a random rigid motion.
max # of iterations if set to > 0: max number of iterations that are performed, i.e., bound on the number of tested sampled.
quality threshold registrations with a quality below this threshold are included in the result set.
subdivision threshold cells of the octree that organizes the subdivision of the annulus $\circledcirc$ that have an edge length below this threshold are not further subdivided.

A general outline of this heuristic is

1. pick a virtual characteristic point $\hat{q}$ in model space,
2. call an algorithm $\mathcal{A}_2$ that computes registrations for two characteristic points by adding $\hat{q}$ to the input,
3. exclude/include the neighborhood around $\hat{q}$ based on the quality of the registration,
4. pick a new virtual characteristic point and repeat.

In this implementation, the simple semioptimal approximation algorithm that is described in the next paragraph is used as the nested algorithm $\mathcal{A}_2$. The annulus and its subdivisions and refinements is organized in an octree where only cells are considered, that fall within or intersect the annulus. The next virtual characteristic point is chosen as the center of the first leaf (in BFS order starting at the root) cell of the octree that is not covered by an in/excluding area and whose center falls within the annulus. This criterion selects the largest (edge length wise) leafs first. If the first leaf that is encountered which is not covered by an in/excluding area has a center that lies outside of the annulus, this leaf is subdivided and the BFS is restarted.

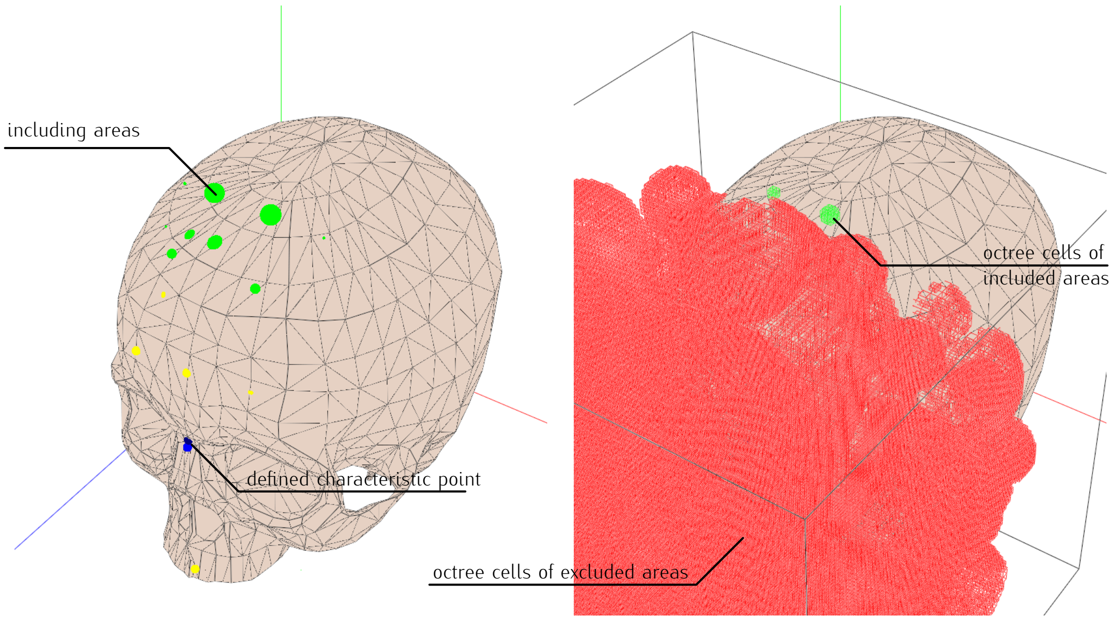

including areas
defined characteristic point
octree cells of included areas
octree cells of excluded areas

Figure A.3: Screenshots of the one point approximation heuristic: **left:** the model and the matched points and the regions that result in a registration value 0.05, **right:** cells of the octree that approximate inclusion/exclusion areas.

Simple Semioptimal Approximation Algorithm

This section describes the implementation of a simple algorithm that approximates semioptimal registrations for two defined and measured characteristic points. The quality of the computed registration depends on a parameter specified by the user and on a geometric property of the input, i.e., the length of the longest distance of any measured surface point to its orthogonal projection onto the line through the two defined characteristic points.

A registration is computed by first applying an initial rigid motion that alines the anatomical landmarks as defined in Section 3.1.2. Then, $u + 1$ positions of the measured point set are tested by rotating the measured points by angles of $i\,2\pi/u$ for $i = 0, \ldots, u$ around the rotation axis, for a user defined parameter u. The best position is then taken as the registration.

Parameter to be specified by the user for this algorithm:

model file path to the file in which the (triangulated or voxel based) model is described.
point file path to the file containing the landmarks and surface point information.
initial motion flag indicating whether or not an initial random rigid motion shall be applied to relocate the measured points.
number of angles the parameter *u* that defines how many positions (different rotation angles) will be tested.
assigned flag indicating whether or not the characteristic points are assigned, i.e., whether the first defined characteristic point is matched to the first measured characteristic point. If set to TRUE , only the configuration in which the first (second) measured characteristic point is matched the to first (second) defined characteristic point is tested. If set to FALSE all possible assignment combinations are tested.

Iterative Closest Point – ICP

The iterative closes point (ICP) method is a well known and studied heuristic for computing point-to-point and point-to-surface matchings, see also Section 1.2.1. The basic idea of this method is that starting from a unspecified initial position the nearest neighbor of each pattern point in the model (point set / surface) is determined. Then, the rigid motion that – if applied to the pattern – minimizes the least squares distance between the pattern and the determined nearest neighbors is computed and applied to the pattern. This process (determining the nearest neighbors, computing the minimizing rigid motion, applying the motion) is usually iterated until one of the following criteria apply:

- a certain threshold is reached,
- a certain number of iterations is reached,
- the nearest neighbor assignment does not chance after applying a motion (i.e., a local minimum is reached),
- the decrease of distance between two successive iteration steps is below a certain threshold.

The implementation of the ICP algorithm in RegistrationToolKit allows to take the characteristic points into consideration to influence the (important) initial positions. If specified by the user, the measured points are aligned in way as described for the previous algorithm: first moved to a semioptimal position and rotated around the axis through the first two characteristic points.

Parameter to be specified by the user for the ICP algorithm:

model file path to the file in which the model is described.
point file path to the file containing the landmarks and surface point information.
max # iterations maximal number of ICP steps that are performed per initial position
initial alignment flag indicating whether the anatomical landmarks shall be used to bring the measured points in potentially good initial positions. If set to TRUE, the first two characteristic points are used to bring the measured points in semioptimal position, *u* (the parameter that is described next) positions are tested by rotating the point set around the axis defined by the first two defined characteristic points. The point set is rotated by multiples of $2\pi/u$. If set to FALSE the initial position is the position of the measured points as described in the *point file*.
# initial positions the parameter *u* that describes how many rotation positions are to be tested.

distance threshold optional termination parameter. If set to > 0 the ICP process is terminated if the current directed Hausdorff distance of the point sets to their features is terminated.

trim target optional parameter v describing how many measured points shall be considered. If set to > 0 only the first v measured surface points are considered in the ICP process (reasonable for very large measured point sets).

Appendix B

Example Implementation of an Algorithm

In this chapter a simple algorithm is presented to illustrate how geometric algorithms are implemented and visualized in RegistrationToolKit, see also Chapter 8.

The algorithm takes a point p and a line l in $\mathbb{R}^3$ as input and computes the orthogonal projection $p \perp l$ of p onto l. The coordinates of p as well as the position of l can be specified by the user. In the visualization p, l and $p \perp l$ are shown as well as a (dotted) line segment from p to $p \perp l$, see Figure B.1.

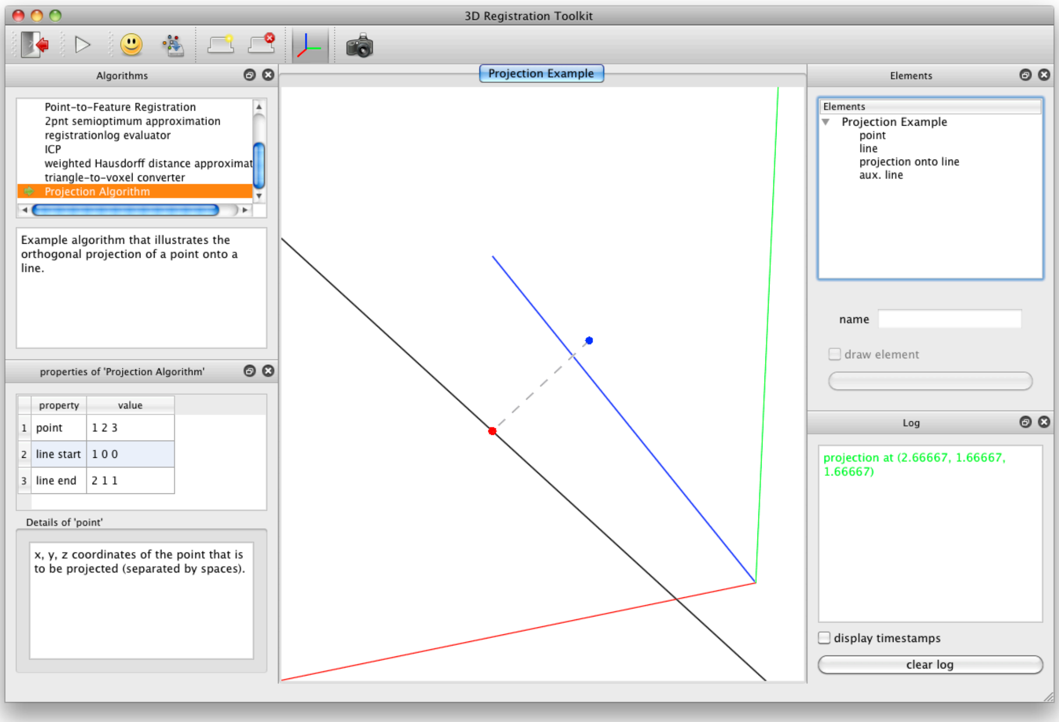
The implementation consists two classes, **`TestAlgorithm`** and **`TestAlgorithmVis`**, defined in four files:

`TestAlgorithm.h` defining the member variables and methods of the **`TestAlgorithm`** class.
`TestAlgorithm.cpp` containing the implementation of the **`TestAlgorithm`** class, containing the algorithmic code.
`TestAlgorithmVis.h` defining the member variables and methods of the **`TestAlgorithmVis`** class.
`TestAlgorithmVis.cpp` specifying the implementation of the **`TestAlgorithm`** class, responsible for visualizing the result of the algorithm.

The class **`TestAlgorithm`** contains the actual implementation of the algorithm whereas the class **`TestAlgorithmVis`** (which extends the **`TestAlgorithm`** class) takes care of the visualization of the geometric objects. In the following the four files are listed and their function is summarized.

`TestAlgorithm.h`

The geometric objects will be visualized later are stored as 'protected' members in the **`TestAlgorithm`** class. The definition also contains the definition of descriptive members like the name and a description of the algorithm that are shown in the user interface.



Point-to-Feature Registration
2pnt semioptimum approximation
registrationlog evaluator
ICP
weighted Hausdorff distance approximat
triangle-to-voxel converter
Projection Algorithm

Example algorithm that illustrates the orthogonal projection of a point onto a line.

	property	value
1	point	1 2 3
2	line start	1 0 0
3	line end	2 1 1

x, y, z coordinates of the point that is to be projected (separated by spaces).

Elements
- Projection Example
  - point
  - line
  - projection onto line
  - aux. line

```
projection at (2.66667, 1.66667,
1.66667)
```

Figure B.1: Screenshot of the *projection algorithm* example (with coordinate system).

```
#ifndef TESTALGORITHM_H
#define TESTALGORITHM_H

#include "Algorithm.h"
#include <QString>

/**
 * Definition of the 'TestAlgorithm' class.
 * The algorithm computes the orthogonal projection
 * of a point onto a line.
 **/
class TestAlgorithm: public Algorithm
{
  Q_OBJECT

  public:
    // the name of the algorithm (required by the Algorithm class)
    static const QString NAME;

    // a string that describes the algorithm (required by the Algorithm class)
    static const QString DESCRIPTION;

  private:
    // properties of the algorithm, i.e., all parameter
    // that can be influenced by the user

    // string representation of the coordinates of the point
    AlgorithmProperty *pointPosProperty;

    // string representation of the first point of the line
    AlgorithmProperty *lineStartProperty;

    // string representation of the second point of the line
    AlgorithmProperty *lineEndProperty;

  protected:
    // geometric members that are used by the algorithm and
    // by the visualization

    // the point that is to be projected
    Point3D* point;

    // its projection
    Point3D* projection;

    // the line onto which the 'point' ist to be projected
    Line* line;

  public:
    /**
     * Constuctor of the TestAlgorithm class
     *
     * factory: factory for generating geometric elements,
     * required by the superclass (Algorithm)
     **/
    TestAlgorithm(ElementFactory *factory);

    /**
     * Starts the algorithm (required by the Algorithm class)
     **/
    void run();

  signals:
    /**
     * Signal passing function to emmit messages to all listeners.
     *
     * msg: the message that is to be distributed
     **/
    void sendMessage(Message* msg);

  private:
    /**
     * Generates a point based on the string representation of the property
     * 'prop' that specifies the coordinates of the 'point'.
     *
     * prop: reference to the property that contains the string representation
     * of the coordinates
     * point: pointer to the pointer at which the new point is to be stored.
     *
     * return: 'true' if the property contains a string that is a proper
     * coordinate representation, 'false' otherwise.
     **/
    bool PropertyToPoint(const AlgorithmProperty &prop, Point3D **point);
};

#endif
```

TestAlgorithm.cpp

In this implementation, the visualization takes place *after* the algorithmic computations are finished. This is only one way of realizing the visualization. An alternative is to send a visualization message during the computation to indicate that a geometric object should be added/removed/focused on/changed in the visualization.

```
#include "TestAlgorithm.h"
#include "ElementTools.h"
#include "AlgorithmMessage.h"
#include "LogMessage.h"
#include "Model.h"

const QString TestAlgorithm::NAME = "Projection Algorithm";
const QString TestAlgorithm::DESCRIPTION = "Example algorithm that illustrates the orthogonal
      projection of a point onto a line.";

TestAlgorithm::TestAlgorithm(ElementFactory *factory):
  Algorithm(NAME, DESCRIPTION, factory)
{
  // setting up the algorithm properites ...
  pointPosProperty = new AlgorithmProperty(QString("point"), QString("x, y, z coordinates of
       the point that is to be projected (separated by spaces)."), QString("1 2 3"));
  lineStartProperty = new AlgorithmProperty(QString("line start"), QString("x, y, z
       coordinates of one point of the line (separated by spaces)."), QString("1 0 0"));
  lineEndProperty = new AlgorithmProperty(QString("line end"), QString("x, y, z coordinates
       of another point of the line (separated by spaces)."), QString("2 1 1"));
  // ... and adding them to the list of properties that are shown in the GUI
  properties.append(pointPosProperty);
  properties.append(lineStartProperty);
  properties.append(lineEndProperty);
}

void TestAlgorithm::run()
{
  // the actual computation starts here by sending a message to the environment that the
       algorithm started
  emit sendMessage(new AlgorithmMessage("algorithm started", AlgorithmMessage::
       ALGORITHM_STARTED, this));

  // temporary variables to store the points on the line
  Point3D *pnt1, *pnt2;

  // get features from properties
  if (!PropertyToPoint( *pointPosProperty, &point)) return;
  if (!PropertyToPoint( *lineStartProperty, &pnt1)) return;
  if (!PropertyToPoint( *lineEndProperty, &pnt2)) return;

  // get the line from the two points
  line = new Line(*pnt1, *pnt2);

  // compute the orthogonal projection using code from the geometric library
  projection = new Point3D(ElementTools::getOrthogonalProjectionOnLine(*point, *line));

  // send a string representation of the projection to the log
  emit sendMessage(new LogMessage(QString("projection at %1").arg(projection->toString()), Qt
       ::green));

  // computation is done. Last step: broadcasting this information to the environment
  emit sendMessage(new AlgorithmMessage(
           QString("algorithm finished"),
           QString("orthogonal projection computet."),
           AlgorithmMessage::COMPUTATION_FINISHED,
           this));
}


bool TestAlgorithm::PropertyToPoint(const AlgorithmProperty &prop, Point3D **point)
{
  QStringList coordinates = prop.getStringValue().split(" ");
  if (coordinates.length() != 3)
  {
      emit sendMessage(new AlgorithmMessage(
           QString("Wrong format of %1 property").arg(prop.getName()),
           QString("Algorithm aborted."),
           AlgorithmMessage::CRITICAL_ALGORITHM_ERROR_OCCURRED,
           this));
    return false;
  }

  (*point) = new Point3D(coordinates[0].toDouble(), coordinates[1].toDouble(), coordinates
       [2].toDouble());
  return true;
}
```

TestAlgorithmVis.h

The definition of the **TestAlgorithmVis** class contains a member variable to store the OpenGL window in which the elements are to be displayed. The class definition also contains methods for triggering the visualization and for requesting OpenGL windows. The class requires to implement a slot **receiveMessage(...)** to be able to detect relevant status changes, e.g., to recognize that the algorithmic computation has finished and the visualization can be started.

```
#ifndef TESTALGORITHMVIS_H
#define TESTALGORITHMVIS_H

#include "TestAlgorithm.h"

class Message;
class VisualizationWidget;

/**
 * Definition of the 'TestAlgorithmVis' class.
 * This class is responsible of visualizing the
 * geometric objects of the 'TestAlgorithm' class
 * after the algorithmic computation has finished.
 */
class TestAlgorithmVis: public TestAlgorithm
{
  Q_OBJECT

  private:
    // pointer to the message by which an OpenGL window is requested
    Message *widgetRequestMessage;

    // pointer to the OpenGL window that is assigned to this algorithm
    VisualizationWidget *widget;

  public:
    /**
     * Constuctor of the TestAlgorithmVis class
     *
     * factory: factory for generating geometric elements,
     * required by the superclass (Algorithm, superclass of TestAlgorithm)
     **/
    TestAlgorithmVis(ElementFactory *factory);

  private:
    /**
     * Emmits a message to the visualization controller and requests
     * an own window in which the elements are visualized.
     **/
    void requestOwnWidget();

    /**
     * Called after 'TestAlgorithm' sends the message that the
     * computation is completed. Takes care of visualizing the
     * geometric 'protected momber' features of 'TestAlgorithm'.
     **/
    void visualize();


  public slots:
    /**
     * Slot for recieving messages. Required to implement the
     * listener interface.
     *
     * msg: pointer to the message that is sended.
     **/
    void receiveMessage(Message* msg);
};

#endif
```

TestAlgorithmVis.cpp

The `TestAlgorithmVis` class extends the `TestAlgorithm` class. As there is no implementation of the `run()` method in `TestAlgorithmVis`, the corresponding method of `TestAlgorithm` is triggered when the algorithm is started. In the last line of the `run()` method of `TestAlgorithm` (see Section B) a message is emitted that is received by the `receiveMessage(...)` method of this `TextAlgorithmVis` class. After detecting this situation, the algorithm requests an OpenGL window from the visualization controller. As soon as this window is constructed (communicated by a visualization message), the visualization is started.

```
#include "TestAlgorithmVis.h"

#include "Message.h"
#include "VisualizationMessage.h"
#include "AlgorithmMessage.h"
#include "VisualizationWidget.h"
#include "AllVisualizationElements.h"

TestAlgorithmVis::TestAlgorithmVis(ElementFactory *factory):
  TestAlgorithm(factory),
  widgetRequestMessage(NULL)
{
}


void TestAlgorithmVis::requestOwnWidget()
{
  // message stored in member variable to be able to
  // remember the request, when it is answered.
  widgetRequestMessage = new VisualizationMessage("Projection algorithm requests new widget",
        "Projection Example", VisualizationMessage::REQUEST_VIZUALIZATION_WIDGET);

  emit sendMessage(widgetRequestMessage);
}

void TestAlgorithmVis::visualize()
{
  VisualizationElement *element;

  // visualization of the point
  element = new Point3DVis(*point);
  element->setName(QString("point"));
  element->setColor(Qt::blue);
```

```
    element->setRadius(0.05);
    emit sendMessage(new VisualizationMessage(VisualizationMessage::VISUALIZATION_ELEMENT_ADDED
         , widget, element));

    // visualization of the line
    element = new LineVis(*line);
    ((LineVis*) element)->setExtendToInfinity(true);
    element->setName(QString("line"));
    element->setColor(Qt::black);
    emit sendMessage(new VisualizationMessage(VisualizationMessage::VISUALIZATION_ELEMENT_ADDED
         , widget, element));

    // visualization of the projection
    element = new Point3DVis(*projection);
    element->setName(QString("projection onto line"));
    element->setColor(Qt::red);
    element->setRadius(0.05);
    emit sendMessage(new VisualizationMessage(VisualizationMessage::VISUALIZATION_ELEMENT_ADDED
         , widget, element));

    // dotted line from point to projection
    element = new LineVis(*point, *projection);
    element->setName(QString("aux. line"));
    element->setColor(Qt::gray);
    element->setLineStyle(VisualizationProperties::DASHED);
    emit sendMessage(new VisualizationMessage(VisualizationMessage::VISUALIZATION_ELEMENT_ADDED
         , widget, element));
    emit sendMessage(new VisualizationMessage(VisualizationMessage::FOCUS_ON_ELEMENT_REQUEST,
         widget, element));
  }

  void TestAlgorithmVis::receiveMessage(Message* msg)
  {
    if (msg->getType() == Message::ALGORITHM_MESSAGE)
    { // an algorithm message is received
      AlgorithmMessage *aMsg = (AlgorithmMessage*) msg;

      if ( aMsg->getContent() == AlgorithmMessage::COMPUTATION_FINISHED
          && aMsg->getAlgorithm() == this)
      { // this TextAlgorithm that was extended has finished
        // ask for an OpenGL widget
        requestOwnWidget();
      }
    } else if (msg->getType() == Message::VISUALIZATION_MESSAGE)
    { // a visualization message is received
      VisualizationMessage *vMsg = (VisualizationMessage*) msg;

      if ( vMsg->getContent() == VisualizationMessage::VIZUALIZATION_WIDGET_GENERATED &&
             widgetRequestMessage != NULL &&
             vMsg->getRelatedMsg() == widgetRequestMessage)
      { // it is the answer to the request for an OpenGL window
        widget = vMsg->getWidget();
        // the visualization can be triggered
        visualize();
      }
    }
  }
```

List of Figures

Bibliography

[1] H. Alt and L. Guibas. Discrete Geometric Shapes: Matching, Interpolation, and Approximation. In *Handbook of Computational Geometry*, pages 121–153. Elsevier B.V., 2000.

[2] H. Alt, L. Scharf, and S. Scholz. Probabilistic Matching and Resemblance Evaluation of Shapes in Trademark Images. In *Proceedings of the ACM International Conference on Image and Video Retrieval (CIVR)*, pages 533–540, Amsterdam, The Netherlands, July 2007.

[3] F. Aurenhammer. Voronoi Diagrams – a Survey of a Fundamental Geometric Data Structure. *ACM Computing Surveys*, 23(3):345–405, September 1991.

[4] F. Aurenhammer and H. Edelsbrunner. An Optimal Algorithm for Constructing the Weighted Voronoi Diagram in the Plane. *Pattern Recognition*, 17(2):251–257, 1984.

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

[6] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, March 2004.

[7] O. Cheong, J. Gudmundsson, H.-S. Kim, D. Schymura, and F. Stehn. Measuring the Similarity of Geometric Graphs. In *Proc. 8th International Symposium on Experimental Algorithms*, volume 5526 of *Lecture Notes in Computer Science*, pages 101–112, Dortmund, Germany, 2009. Springer.

[8] G. Cox, G. de Jager, and B. Warner. A New Method of Rotation, Scale and Translation Invariant Point Pattern Matching Applied to the Target Acquisition and Guiding of an Automatic Telescope. *2nd South African Workshop on Pattern Recognition*, pages 167–172, 1991.

[9] R. H. Davis and J. Lyall. Recognition of Handwritten Characters – a Review. *Image and Vision Computing*, 4(4):208–218, 1986.

[10] B. M. Dawant. Non-Rigid Registration of Medical Images: Purpose and Methods, a Short Survey. In *Proceedings of the 2002 IEEE International Symposium on Biomedical Imaging*, pages 465–468, Washington, DC, USA, June 2002. IEEE.

[11] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. *Computational Geometry: Algorithms and Applications*. Springer-Verlag TELOS, 3 edition, 2008.

[12] D. Dimitrov, R. Günzler, C. Knauer, K. Kriegel, S. Mularski, F. Stehn, O. Suess, and U. Warschewske. The Precision of Measuring Anatomic Landmarks and its Influence on Hybrid Registration Algorithms. Technical Report B 07-17, Institute of Computer Science, Freie Universität Berlin, Berlin, Germany, 2007.

[13] D. Dimitrov, C. Knauer, and K. Kriegel. Registration of 3D-Patterns and Shapes with Characteristic Points. In *Proceedings of International Conference on Computer Vision Theory and Applications - VISAPP 2006*, pages 393–400, Setúbal, Portugal, 2006.

[14] D. Dimitrov, C. Knauer, K. Kriegel, and F. Stehn. Approximation Algorithms for a Point-to-Surface Registration Problem in Medical Navigation. In *Proc. Frontiers in Algorithmics Workshop*, volume 4613 of *Lecture Notes in Computer Science*, pages 26–37, Lanzhou, China, 2007. Springer.

[15] D. Dimitrov, C. Knauer, K. Kriegel, and F. Stehn. Approximate Point-to-Surface Registration with a Single Characteristic Point. In *Proceedings of International Conference on Computer Vision Theory and Applications - VISAPP*, pages 188–195, Funchal, Portugal, 2008.

[16] A. C. Evans, S. Marrett, D. L. Collins, and T. M. Peters. Anatomical-Functional Correlative Analysis of the Human Brain using Three-Dimensional Imaging Systems. In *Proceedings of the International Society of Optical Engineering (SPIE): Medical Imaging III*, volume 1092, pages 264–274, May 1989.

[17] P. W. Finn, L. E. Kavraki, J. C. Latombe, R. Motwani, C. Shelton, S. Venkatasubramanian, and A. Yao. RAPID: Randomized Pharmacophore Identification for Drug Design. In *In Proceedings of the International Symposium on Computational Geometry*, pages 324–333, 1997.

[18] J. M. Fitzpatrick, J. B. West, and C. R. Maurer. Predicting Error in Rigid-Body Point-Based Registration. *IEEE Transactions on Medical Imaging*, 17(5):694–702, 1998.

[19] L. M. G. Fonseca and M. H. M. Costa. Automatic Registration of Satellite Images. In *Brazilian Symposium on Computer Graphics and Image Processing*, volume 10, pages 219–226, Campos do Jordao , Brazil, 1997.

[20] K. P. Gall, L. J. Verhey, and M. Wagner. Computer-Assisted Positioning of Radiotherapy Patients Using Implanted Radiopaque Fiducials. *Med Phys*, 20(4):1153–1159, 1993.

[21] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns*. Addison Wesley, Reading, MA, 1995.

[22] R. Hartmann. Registrierung von Punktmengen auf Flächen mit 3 Charakteristischen Punkten. Master's thesis, Freie Universität Berlin, Berlin, Germany, 2007.

[23] J. E. Hershberger. Finding the Upper Envelope of n Line Segments in $O(n \log n)$ time. *Inf. Process. Lett.*, 33(4):169–174, 1989.

[24] F. Hoffmann, K. Kriegel, S. Schönherr, and C. Wenk. A Simple and Robust Geometric Algorithm for Landmark Registration in Computer Assisted Neurosurgery. Technical Report B 99-21, Institute of Computer Science, Freie Universität Berlin, Berlin, Germany, December 1999.

[25] D. P. Huttenlocher, K. Kedem, and M. Sharir. The Upper Envelope of Voronoi Surfaces and its Applications. In *SCG '91: Proceedings of the Seventh Annual Symposium on Computational Geometry*, pages 194–203, New York, NY, USA, 1991. ACM.

[26] A. E. Johnson and M. Hebert. Surface Registration by Matching Oriented Points. In *NRC '97: Proceedings of the International Conference on Recent Advances in 3-D Digital Imaging and Modeling*, pages 121–128, Washington, DC, USA, 1997. IEEE Computer Society.

[27] C. Knauer, K. Kriegel, and F. Stehn. Minimizing the Weighted Directed Hausdorff Distance between Colored Point Sets under Translations and Rigid Motions. In *Proceedings of the 25th European Workshop on Computational Geometry (EuroCG)*, pages 45–48, Brussels, Belgium, March 2009.

[28] C. Knauer, K. Kriegel, and F. Stehn. Minimizing the Weighted Directed Hausdorff Distance Between Colored Point Sets Under Translations and Rigid Motions. In *Proc. Frontiers in Algorithmics Workshop*, volume 5598 of *Lecture Notes in Computer Science*, pages 108–119, Hefei, China, 2009. Springer.

[29] C. Knauer, K. Kriegel, and F. Stehn. Minimizing the Weighted Directed Hausdorff Distance Between Colored Point Sets under Translations and Rigid Motions. *Theoretical Computer Science*, 2010.

[30] C. Knauer, K. Kriegel, and F. Stehn. Towards Non-Uniform Geometric Matchings. In *Proceedings of the 26th European Workshop on Computational Geometry (EuroCG)*, pages 257–260, Dortmund, Germany, 2010.

[31] P. J. M. Laarhoven and E. H. L. Aarts, editors. *Simulated Annealing: Theory and Applications*. Kluwer Academic Publishers, Norwell, MA, USA, 1987.

[32] N. Li, P. Cheng, M. Sutton, and S. McNeill. Three-dimensional Point Cloud Registration by Matching Surface Features with Relaxation Labeling Method. *Experimental Mechanics*, 45:71–82, 2005.

[33] J. B. A. Maintz and M. A. Viergever. A Survey of Medical Image Registration. *Medical Image Analysis*, 2(1):1–36, 1998.

[34] J. Matousek. *Lectures on Discrete Geometry*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2002.

[35] C. R. Maurer, Jr. and J. M. Fitzpatrick. A Review of Medical Image Registration. In *Interactive Image Guided Neurosurgery*, pages 17–44, 1993.

[36] C. R. Maurer, Jr., J. M. Fitzpatrick, M. Y. Wang, R. L. Galloway, R. J. Maciunas, and G. S. Allen. Registration of Head Volume Images Using Implantable Fiducial Markers. *IEEE Transactions on Medical Imaging*, 16:447–462, 1997.

[37] N. Megiddo. On the Ball Spanned by Balls. *Discrete & Computational Geometry*, 4:605–610, 1989.

[38] N. J. Mitra, N. Gelfand, H. Pottmann, and L. Guibas. Registration of Point Cloud Data from a Geometric Optimization Perspective. In *SGP '04: Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, pages 22–31, New York, NY, USA, 2004. ACM.

[39] Nokia. The Qt cross platform UI framework `http://qt.nokia.com`.

[40] G. Rote. The Convergence Rate of the Sandwich Algorithm for Approximating Convex Functions. *Computing*, 48(3-4):337–361, 1992.

[41] W. Rucklidge. Lower Bounds for the Complexity of the Graph of the Hausdorff Distance as a Function of Transformation. *Discrete & Computational Geometry*, 16(2):135–153, 1996.

[42] S. Rusinkiewicz and M. Levoy. Efficient Variants of the ICP Algorithm. In *Proceedings of the Third International Conference on 3D Digital Imaging and Modeling*, pages 145–152, 2001.

[43] M. Sharir and P. K. Agarwal. *Davenport-Schinzel Sequences and their Geometric Applications*. Cambridge University Press, New York, NY, USA, 1996.

[44] Y. Sun, J. Paik, A. Koschan, and M. A. Abidi. Point Fingerprint: A New 3-D Object Representation Scheme. *IEEE Transaction on Systems, Man, and Cybernetics–Part B: Cybernetics*, 33:712–717, 2003.

[45] G. T. Toussaint. A Simple Linear Algorithm for Intersecting Convex Polygons. *The Visual Computer*, 1:118–123, 1985.

[46] P. A. van den Elsen, E. J. D. Pol, and M. A. Viergever. Medical Image Matching – a Review with Classification. *Engineering in Medicine and Biology Magazine, IEEE*, 12(1):26–39, 1993.

[47] J. Vleugels and R. Veltkamp. Efficient Image Retrieval through Vantage Objects. *Pattern Recognition*, 35:69–80, 1999.

[48] Y. Wang, B. S. Peterson, and L. H. Staib. Shape-based 3D Surface Correspondence using Geodesics and Local Geometry. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2:644–651, 2000.

[49] S. M. Yamany and A. A. Farag. Surfacing Signatures: An Orientation Independent Free-Form Surface Representation Scheme for the Purpose of Objects Registration and Matching. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(8):1105–1120, 2002.

Erklärung

Hiermit versichere ich, **Fabian Stehn** , dass ich die vorliegende Arbeit selbständig angefertigt und ohne fremde Hilfe verfasst habe, keine außer den von mir angegebenen Hilfsmitteln und Quellen dazu verwendet habe und die den benutzten Werken inhaltlich oder wörtlich entnommenen Stellen als solche kenntlich gemacht habe.

Berlin, den 5. Oktober 2010

Fabian Stehn