TECHNISCHE UNIVERSITÄT CHEMNITZ

Fakultät Mathematik

Professur Algorithmische und Diskrete Mathematik

Diplomarbeit

Matching hochaufgelöster 3D-Scandaten von teil-deformierten Objekten an CAD-Modelle: Verfahrens-, Laufzeit- und Präzisionsoptimierung

Rüdiger Borsdorf

Chemnitz, den 2. März 2009

Betreuer:
Prof. Dr. Christoph Helmberg
Dr. Sven Herrmann

Danksagung

**Ich möchte meinem Betreuer an der TU Chemnitz
Herrn Prof. Dr. Helmberg für seine richtungsweisenden Vorgaben, sein Engagement, seine hilfreiche Unterstützung, vielerlei Hinweise und seine ständige Verfügbarkeit bei Fragen und Problemen danken. Mein Dank gilt auch der Robert Bosch GmbH für die Möglichkeit, dieses praxisnahe Thema zu bearbeiten und meinem dortigen Betreuer
Herrn Dr. Herrmann für die Betreuung, seine Ratschläge und die Erläuterung aller technischen Hintergründe. Ich möchte mich außerdem bei meinen Kollegen für die schöne gemeinsame Zeit bei Bosch, meinen Freunden Michael Döhler und Anja Lau für manchen mathematischen Austausch und schließlich meiner Familie für ihre Hilfe in jeder Hinsicht bedanken.**

Abstract

Diese Arbeit widmet sich dem Matching bzw. dem Angleichen einer Punktwolke an ein Referenzmodell im dreidimensionenalen Raum, wobei letzteres eine Oberflächentriangulierung ist. Da dieses Problem des Matchings in vielen Industriebereichen wie z. B. in der Fertigungsautomatisierung, aber auch im alltäglichen Umfeld immer mehr Anwendung findet, steigt nicht zuletzt die Nachfrage an Software, die dieses Ausrichten robust durchführt. In dieser Arbeit stellen wir einen Algorithmus vor, der zunächst die Punktwolke und das Modell mit Hilfe der Hauptkomponentenanalyse grob ausrichtet und schließlich für die Feinregistrierung ein dem ICP (Iterative Closest Point) ähnliches Verfahren anwendet. Der Unterschied ist, dass das Korrespondenzenproblem mittels eines Flussproblems minimaler Kosten modelliert wird. Zur Optimierung des hier vorgeschlagenen Verfahrens entwickeln wir desweiteren einen Algorithmus, der die Anzahl der Dreiecke des Modells reduziert und bei dem die Form des Modells weitestgehend erhalten bleibt. Somit kann eine Hierarchie von Modellen erzeugt werden. Wir vergleichen das hier vorgestellte Verfahren mit dem ICP und stellen unsere numerischen Tests vor.

This thesis concerns the problem of matching a point cloud to a reference model in three-dimensional space whereas the reference model is a triangulation of a surface. Since this problem has its applications not only in many branches of industry e.g. production automation but also in everyday life the demands on software that performs this matching robustly increase continuously. We propose an algorithm that at first registers the model with the point cloud coarsely by using the principle component analysis and finally performs the fine registration by applying a similar method to the standard ICP (Iterative Closest Point) method. The main difference is that we model the problem of finding the correspondences by a min-cost-flow-problem. To optimize this proposed algorithm we develop a method to reduce the number of triangles in the model whilst preserving the shape of the model as far as possible. Hence we can form a hierarchy of models. Furthermore, the algorithm is compared with the ICP method by providing numerical tests.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einführung

1.1 Bilderkennende Systeme

Im Zuge der immer besser werdenden Computer- und Kameratechnik wird das Feld der Anwendungen der komplexen optischen Systeme zur Bilderkennung stetig größer. Diese kombinieren im Allgemeinen optische Messgeräte mit modernster Rechentechnik, um mit Hilfe von aufgenommenen Bildern Objekte zu identifizieren. Sie werden immer häufiger eingesetzt und sind heutzutage mitunter unumgänglich.

Der große Vorteil dieser Systeme besteht darin, dass die Aufnahmen anders als bei der herkömmlichen Fotografie digitalisiert werden und damit unmittelbar jede mathematisch beschreibbare Verarbeitungsvorschrift (Algorithmus) auf die digitalen Bilddaten angewendet werden kann. Dies bedeutet, die Verarbeitung der Daten ist nicht mehr an die Beschränkungen physikalischer oder chemischer Verarbeitungsprozesse gebunden. Dabei bringt es der Fortschritt der Digitaltechnik mit sich, dass immer kompliziertere und rechenintensivere Algorithmen auf immer leistungsfähigeren Rechnern in einer akzeptablen Rechenzeit laufen können. Dies eröffnet eine Vielzahl von Möglichkeiten.

So bestehen heutige moderne Anlagen in vielen Industriezweigen, v. a. aber auch in der Autoindustrie aus einer beträchtlichen Anzahl an Kamerasystemen, die vielseitige Aufgaben übernehmen. Dies reicht von der einfachen Überwachungsfunktion, über Barcodeerkennungen bis hin zum vollautomatischen und intelligenten Erkennen und Begreifen von Objekten. In der Fertigung dienen diese Objekterkennungen z. B. der flexibleren, fehlerärmeren und im Vergleich zum Menschen schnelleren, präziseren und kostengünstigen Produktion von Erzeugnissen.

Ein wichtiges Thema spielt dabei die Qualitätskontrolle, bei der das zu prüfende Objekt aufgenommen und anschließend auf Fehler untersucht wird. Diese Fehler betreffen meist die Geometrie des zu überprüfenden Objektes. Um diese Fehler zu erkennen, ist es erforderlich, neben der präzisen Aufnahme des zu untersuchenden Objektes, effektive Algorithmen für die geschickte Segmentierung und Merkmalserkennung im Bild anzuwenden und schließlich mit Hilfe anderer Algorithmen diese Merkmale in einem Referenzdatensatz oder Referenzmodell wiederzuerkennen. Man führt einen Soll-Ist-Vergleich durch. Sind diese Algorithmen robust einsetzbar, so können diese für die automatische Qualitätskontrolle verwendet werden.

1.2 Das Matching

Bei der im letzten Abschnitt angesprochenen Qualitätskontrolle wird versucht, die Merkmale der Fehler, die in einem Referenzmodell kodiert sind, in der gleichen Form bzw. Kombination in der Aufnahme der defekten Objekte wiederzufinden. Es wird eine Zuordung der Merkmale im Modell zu den Merkmalen in der Aufnahme gesucht. Ist die Zuordung so gewählt, dass die Merkmale des Modells den zugeordneten Merkmalen in der Aufnahme entsprechen und jeweils ihre Struktur und Kombinationen untereinander bestmöglich übereinstimmen, so wurde das Modell mit der Aufnahme *gematcht*. Die Bestimmung der besten Zuordnung bezeichnen wir als *Matching*. Insbesondere im Dreidimensionalen ist dieser Vorgang oft ein Angleichen bzw. Ausrichten des Modells an die Aufnahme. Sind Modell und Aufnahme gematcht, so kann man das aufgenommene Objekt in der Regel bzgl. des Modells identifizieren.

1.3 Anwendungen

Das Modell ist dabei vielseitig beschreibbar. So kann dies z. B. in der Mustererkennung ein bestimmtes Muster sein [3], welches in der Aufnahme gesucht wird. Ebenso kann es die Form und Größe des Objektes beschreiben.

Es ist auch möglich, dass zunächst kein Modell vorhanden ist, sondern erst ein Referenzmodell zu einem Objekt erstellt werden soll, um das Objekt später wiederzuerkennen. Dabei wird z. B. das Objekt aus verschiedenen Richtungen aufgenommen bzw. gescannt und daraus ein Referenzmodell gebildet. Hierbei ist es notwendig, die Überlappungsbereiche der einzelnen Scans zu identifizieren. Werden zwei Scans miteinander verglichen, so ist ein Scan das Modell und das jeweils andere die Aufnahme.

Eine Anwendung ist auch die Objektlageerkennung, in der die Lage eines aufgenommenen Objektes bzgl. der Koordinaten des Modells durch die Anpassung der Aufnahme an das vorhandene Modell ermittelt werden kann. Im Bereich des Reverse Engineerings v. a. im Maschinenbau ist es ebenso erforderlich, ein aufgenommenes Objekt mit einem Modell zu vergleichen. Soll z. B. ein Objekt gefertigt werden, dessen Geometrie einem mittels CAD (Computer Aided Design) erstellten grafischen Modell entspricht, so kann nach der Fertigung durch die Vermessung des Objektes ermittelt werden, wie hoch der Grad der Übereinstimmung zwischen Modell und dem tatsächlich hergestellten Objektes ist.

Auch im täglichen Umfeld findet das Problem des Matchings und somit das Identifizieren von Objekten in Bezug auf das Modell Anwendung. Als Beispiel seien Fingerabdruckerkennung und Fahrerassistenzsysteme genannt.

Diese vielseitigen Anwendungsbereiche [23] unterstreichen die Bedeutung und Nachfrage an Software, die dieses Matching robust ausführt, und deuten auch auf die Komplexität dieses Themas hin. Deshalb wird mit hoher Intensität in verschiedenen Richtungen geforscht, um das Matching immer effektiver zu realisieren. Dies beinhal-

tet auch die Methoden der Segmentierung und Merkmalserkennung zu verbessern. So liegt z. B. oft das Problem der Wiedererkennung bestimmter Formen des Modells in der Identifikation und Extraktion von Merkmalen, die gegen bestimmte Faktoren, wie Lichteinflüsse, Objektrotationen und Skalierung invariant sind.

Je effektiver die Software ist, die das Matching durchführt, desto besser und leistungsfähiger sind die rechnergestützten optischen Systeme in Bezug auf die Objekt- und Merkmalserkennung und umso mehr steigt ihre Akzeptanz in der Praxis.

In dieser Arbeit konzentrieren wir uns vorwiegend auf das Finden und Optimieren von Methoden, die dem Matching einer Objektoberflächenvermessung an ein Referenzmodell dienen. Genauer werden wir jedoch die Aufgabenstellung im Kapitel 2 formulieren, in dem wir ein konkretes Problem betrachten, dessen Anwendung in den Bereich der Objektlageerkennung fällt. Dieses Problem beschreiben wir im Kapitel 4 mathematisch, stellen Lösungsansätze vor und modellieren das mathematische Modell. Im darauffolgenden Kapitel 5 wird dieses mathematische Modell gelöst und in Kapitel 6 ein resultierender Algorithmus vorgestellt. In Kapitel 7 werden wir das Verhalten unseres Algorithmus mit anderen Algorithmen mittels numerischer Tests vergleichen, bevor wir schließlich unsere Schlussfolgerungen in Kapitel 8 ziehen werden.

2 Problemformulierung

2.1 Die Aufgabenstellung

Ziel der vorliegenden Arbeit ist es, die Lage eines dreidimensionalen Objektes (siehe Abschnitt 2.2), dessen grobe Position bekannt ist, bezogen auf ein Koordinatensystem im Raum genau zu bestimmen. Dieses Problem findet vor allem Anwendung in der Fertigungsautomatisierung, in der es häufig erwünscht ist, die Lage eines vorliegenden Objektes präzise zu ermitteln, um anschließend an diesem Objekt weitere verarbeitende Schritte durchzuführen. Die Positionsbestimmung soll dabei möglichst schnell, präsize und automatisch, d. h. ohne manuelles Eingreifen, erfolgen.

Die Oberfläche des Zielobjektes liegt als grafisches Modell vor, welches einer *Triangulierung* der Objektoberfläche entspricht. Weiterhin wird mit Hilfe eines optischen Messgeräts die Objektoberfläche aus einer festgelegten Blickrichtung in der zu bestimmenden Position dreidimensional vermessen, woraus eine so genannte 3D-Punktwolke resultiert. Jene repräsentiert die reale Lage der Objektoberfläche im Raum bezogen auf ein Koordinatensystem, wobei jeder Punkt aus den entsprechenden x-, y- und z-Koordinaten besteht.

Um die Lage des gemessenen Objektes im Raum zu bestimmen, genügt es, die 3D-Punktwolke dem dreidimensionalen grafischen Modell anzupassen, d. h. zu matchen. Hierbei sei angemerkt, dass erst durch das Matching jeder Punkt der Punktwolke mit einem Teil des Objektes identifiziert wird und folglich die Lage des Objektes im Raum bestimmt ist. Das daraus resultierende Problem besteht darin, diejenige Rotation und Translation zu finden, die die Koordinaten des Modells in jene der Punktwolke überführt.

In den folgenden Abschnitten werden das Objekt, das dazugehörige grafische Modell und die aus der Vermessung erzielte Punktwolke näher beschrieben.

2.2 Das Zielobjekt

Das zu betrachtende Objekt soll eine Hülse sein, wobei nur ein Teil des Objektes von Interesse ist, da nur dieser für die Positionsoptimierung vermessen werden soll. Seine Oberfläche lässt sich im Grunde durch zwei elementare Geometrien annähern. Die Annäherung besteht aus einer Halbkugel, die am Rand fließend in einen Kreis übergeht.

2.3 Das zugehörige grafische Modell

Wie oben erwähnt, steht ein grafisches Modell der Objektoberfläche zur Verfügung. Dieses Modell ist die Triangulierung der Oberfläche (Dreiecksnetz) und liegt als STL-Datei vor. STL steht dabei für *SurfaceTesselationLanguage*, was schon darauf hinweist, dass dieses Format die Speicherung einer Oberflächentriangulierung im Dreidimensionalen unterstützt [2]. Durch zwei mit diesem Format verankerte Regeln wird gesichert, dass erstens kein Dreieckseckpunkt mitten auf einer Kante liegt und zweitens bedingt durch die festgelegte Reihenfolge der Auflistung der Dreieckseckpunkte die Orientierung eines Dreiecks bzw. die Richtung nach außen definiert ist.

Weiterhin bietet dieses Format die Codierung einer RGB-Farbe für jedes Dreieck, welche in dem vorliegenden Fall eine Gewichtung jedes Dreiecks definiert. Je höher dieser Wert ist, desto stärker soll das jeweilige Dreieck bei dem Angleichen in Betracht gezogen werden, da bei Dreiecken mit höheren Werten die zu erwartende Fluktuation der korrespondierenden Messpunkte bedingt durch den Herstellungsprozess der Hülsen geringer ist.

2.4 Die zugehörige gemessene 3D-Punktwolke

Wie am Anfang dieses Kapitels beschrieben, resultiert die Punktwolke aus einer Teilvermessung des tatsächlich hergestellten Objektes mittels eines Kamerasystems. Jedes Objekt kann bedingt durch den Herstellungsprozess von den Modelldaten innerhalb eines bekannten Toleranzspektrums abweichen, wobei die Höhe der Abweichung von der jeweiligen Stelle des Objektes abhängt. Dies führt dazu, dass die Objektaufnahmen nicht notwendigerweise deckungsgleich mit dem grafischen Modell sind und insofern auch untereinander fluktuieren.

Das Kamerasystem vermisst den Teilbereich des Objektes äquidistant entlang der x-Achse und y-Achse bezogen auf das Kamerakoordinatensystem, so dass ein Gitter entsteht, zu dem an jedem Gitterpunkt der Abstand zwischen x-y-Ebene und dem Durchstoßungspunkt bekannt ist. Dieser Punkt ist der erste Schnittpunkt zwischen der Oberfläche des Objektes und der Geraden, die durch den Normalenvektor der x-y-Ebene und dem jeweiligen Gitterpunkt definiert ist.

Falls kein solcher Schnittpunkt existiert, so ist der Abstandswert durch einen entsprechend hohen Wert gekennzeichnet. Um diese Werte und solche Messwerte, die bedingt durch das Kamerasystem ungenau sind, von den anderen Abstandswerten klar abzugrenzen, liefert das Kamerasystem neben den Messdaten eine so genannte *Qualitymap*, in der zu jedem Gitterpunkt der Wert für die Güte des entsprechenden Messwertes angegeben wird. Konkret sind die Gitterpunkte zeilenweise in einer Textdatei und die dazugehörige Qualitymap in einer Bilddatei hinterlegt.

3 Notation

Um die Arbeit weitestgehend in sich schlüssig zu halten, geben wir hier eine kurze Überblicksliste über die in den nächsten Kapiteln verwendete Notation. Wie üblich bezeichne

- $\mathbb{N} = \{1, 2, 3, \dots\}$ die Menge aller natürlichen Zahlen,
- $\mathbb{Z} = \{0, 1, -1, 2, -2, \dots\}$ die Menge aller ganzen Zahlen,
- $\mathbb{R} = (-\infty, \infty)$ die Menge aller reellen Zahlen,
- $\mathbb{R}_+ = [0, \infty)$ bzw. $\mathbb{R}_- = (-\infty, 0]$ die Menge der positiven bzw. negativen reellen Zahlen einschließlich der 0,
- $\langle \cdot, \cdot \rangle$ das Standardskalarprodukt mit $\langle x, y \rangle = x^T y$ und $\|\cdot\|_2$ die aus dem Skalarprodukt induzierte Euklidische Norm,
- $I^s = \underbrace{I \times \ldots \times I}_{s}$ für I ein Intervall und $s \in \mathbb{N}$,
- $|C|$ die Kardinalität einer Menge C,
- $\operatorname{conv} C$ die konvexe Hülle einer Menge C,
- $\lceil x \rceil$ mit $x \in \mathbb{R}$, den nächstgrößeren ganzzahligen Wert von x,
- Diag den Operator $\operatorname{Diag} : \mathbb{R}^n \mapsto \mathbb{R}^{n \times n}$ mit

$$\operatorname{Diag}(a) = \begin{pmatrix} a_1 & 0 & \dots & 0 \\ 0 & a_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & a_n \end{pmatrix},$$

- $\operatorname*{argmin}_{x \in D} f(x)$ für einen Definitionsbereich D und eine Funktion $f : D \mapsto \mathbb{R}$ die Menge

$$\mathcal{X}_* := \{x \in D : f(x) \text{ minimal}\},$$

falls $|\mathcal{X}_*| \neq 1$ und sonst das eindeutige Element aus $\mathcal{X}_*$ und analog

- $\underset{x \in D}{\operatorname{argmax}} f(x)$ die Menge

$$\mathcal{X}^* := \{x \in D : f(x) \text{ maximal}\},$$

falls $|\mathcal{X}^*| \neq 1$ und sonst das eindeutige Element aus $\mathcal{X}^*$.

Alle weiteren Symbole, die in den nächsten Kapiteln verwendet werden und nicht in der obigen Liste erscheinen, haben die in der Literatur übliche Bedeutung.

4 Die Modellierung

In diesem Kapitel wird zunächst die Problemstellung aus Kapitel 2 mathematisch formuliert, ebenso werden Grundbegriffe definiert. Im Weiteren werden Ansätze vorgestellt und diskutiert, die sich der Lösung des Problems annähern und abschließend wird das mathematische Modell beschrieben, welches die Grundlage für die weiteren Analysen in dieser Arbeit bildet. Im Kapitel 5 wird dann auf die Lösung des mathematischen Modells eingegangen.

4.1 Mathematische Formulierung

4.1.1 Grundlegende Definitionen

Um das in Kapitel 2 erwähnte grafische Modell und ebenso die aus der Messung hervorgegangene Punktwolke mathematisch beschreiben zu können, klären wir zunächst einige Grundbegriffe beginnend mit der Definition *Punkt*.

Definition 4.1.1. Sei $p \in \mathbb{R}^3$, dann nennen wir p *Punkt* und eine Menge von n Punkten $P = \{p_1, p_2, \ldots, p_n\}$ *Punktwolke* mit $n \in \mathbb{N}$.

Im Weiteren führen wir den Begriff des Dreiecks ein, um anschließend die schon in Kapitel 2 erwähnte Triangulierung definieren zu können. Darauf aufbauend ist es möglich, das grafische Modell des Zielobjektes zu beschreiben.

Definition 4.1.2. Seien q, q_1, q_2 drei Punkte. Dann wird das geordnete Tripel $D = (q, q_1, q_2)$ als *Dreieck* bezeichnet, falls die Vektoren $q_1 - q$ und $q_2 - q$ linear unabhängig sind. Die Menge

$$\triangle := \left\{ r \in \mathbb{R}^3 : r = q + \lambda_1(q_1 - q) + \lambda_2(q_2 - q) \text{ mit } \lambda_1, \lambda_2 \in \mathbb{R}_+,\ \lambda_1 + \lambda_2 \leq 1 \right\}$$

nennen wir die dem Dreieck D zugehörige *Dreiecksfläche* und den Vektor $N \in \mathbb{R}^3$ mit $N = (q - q_1) \times (q - q_2)$ den *Normalenvektor* des Dreiecks D, wobei $\times$ das Kreuzprodukt ist. Die Punkte q, q_1, q_2 heißen *Eckpunkte* des Dreiecks D.

Es sei bemerkt, dass durch die festgelegte Reihenfolge der Eckpunkte, eine Orientierung der entsprechenden Dreiecksfläche gegeben ist.

Um die Triangulierung einer Objektoberfläche (Dreiecksnetz) mathematisch beschreiben zu können, gibt es in der Literatur viele Ansätze. So wird der Begriff der Triangulierung in [8] als Paar (X, Φ) eines Polyeders X und eines Homöomorphismus

ϕ, der die geometrischen Realisierung von X in eine kompakte reguläre Fläche im dreidimensionalem Raum (Mannigfaltigkeit im dreidimensionalen Raum) abbildet, definiert. Dabei wird das Polyeder als Zerlegung der Fläche betrachtet. In [10] wird zunächst der Begriff des Polygonzugs und später daraus resultierend der Begriff der Triangulierung festgelegt. Für diese Arbeit erscheint jedoch die Definition der Triangulierung über einen Graphen, dessen Knoten Eckpunkte von Dreiecken beschreiben und dessen Kanten die Verbindungen zwischen diesen Knoten repräsentieren, am geeignetsten. Zunächst definieren wir den Begriff des ebenen Graphen und daraus ableitend den Begriff der Triangulierung.

Definition 4.1.3. Sei $\mathcal{E} \subset \mathbb{R}^3$ eine Ebene. Ein Paar $G = (V, E)$ mit einer endlichen Knotenmenge $V \subset \mathbb{R}^3$ und einer endlichen Kantenmenge E heißt *ebener Graph*, falls

- jede Kante aus E eine Teilmenge von $\mathcal{E}$ ist, die homöomorph auf das Intervall $[0, 1]$ abgebildet werden kann,
- die Bilder von 0 und 1 unter solch einem Homöomorphismus zwei Knoten (*Endpunkte* der Kante) aus V sind,
- verschiedene Kanten unterschiedliche Mengen an Endpunkten haben und
- das Innere einer Kante keinen Knoten aus V und keinen Punkt aus einer anderen Kante enthält.

Wie in [11, Abschnitt 4.2] erwähnt, wird die Ebene durch $\mathcal{E} \setminus (V \cup \bigcup E)$ in Regionen geteilt, die mit *faces* bezeichnet werden. Genau ein face ist unbeschränkt, welches in [11] *outer face* genannt wird. Die anderen faces werden *inner faces* bezeichnet. In dieser Arbeit nennen wir den Abschluss eines inner faces *ebene Fläche*.

Die Definition 4.1.3 ist eine Voraussetzung für die folgende Definition der Triangulierung, in der wir die Triangulierung als Graphen definieren, dessen Knoten Eckpunkte von Dreiecken und dessen Kanten die Verbindungen der Eckpunkte sind. Jeder Knoten besitzt dabei ein Label, so dass zu dem Knoten bekannt ist, welche Dreiecke er bildet.

Definition 4.1.4. Seien $D_1, \ldots, D_m$ Dreiecke, deren Dreiecksflächen $\triangle_1, \ldots, \triangle_m$ paarweise verschieden sind, sich nur am Rand schneiden und jeweils konsistent bzgl. der Orientierung sind. Des Weiteren bezeichne der Index von $D_1, \ldots, D_m$ eine Nummerierung der Dreiecke.

Sei $\mathcal{T} = (V, E)$ ein Graph, dessen Knotenmenge V die Menge der Eckpunkte von $D_1, \ldots, D_m$ ist und dessen Kantenmenge E durch

$$E = \{\{u, v\} : u, v \text{ Eckpunkte von } D_i \text{ für ein } i \in \{1, \ldots, m\}\}$$

bestimmt ist. Weiter sei jeder Knoten mit einem Label versehen, so dass aus dem Graphen $\mathcal{T}$ die Dreiecke $D_1, \ldots, D_m$ wieder zurückgewonnen werden können, wobei

die Eckpunkte der Dreiecke dann den entsprechenden Knoten gleichen. Seien weiter (q^i, q_1^i, q_2^i) die Eckpunkte des Dreiecks D_i für alle $i = 1, \ldots, m$ und $\mathcal{E} \in \mathbb{R}^3$ eine Ebene. Ist $\mathcal{T}$ zusammenhängend und gibt es einen Homöomorphismus

$$\Phi : \bigcup_{i=1}^{m} \triangle_i \mapsto \Phi\left(\bigcup_{i=1}^{m} \triangle_i\right) \subset \mathcal{E},$$

so dass (V^E, E^E) mit $V^E = \Phi(V)$ und E^E die Menge mit

$$E^E = \{\Phi(S) : S \text{ Strecke zwischen } u \text{ und } v, \{u, v\} \in E\}$$

ein ebener Graph ist, in dem $\Phi(\triangle_1), \ldots, \Phi(\triangle_m)$ innere Flächen sind, so heißt der Graph $\mathcal{T}$ *Triangulierung*. Wir bezeichnen die Dreiecke $D_1, \ldots, D_m$ die der Triangulierung $\mathcal{T}$ *zugehörigen Dreiecke*.

Die Triangulierung ist also ein planarer Graph, denn der ebene Graph, der durch den Homöomorphismus Φ beschrieben wird, ist offensichtlich eine konkrete Darstellung des planaren Graphen in der Ebene [11, Abschnitt 4.3]. Hierbei ist zu bemerken, dass die Dreiecke $D_1, \ldots, D_m$ den Graphen erzeugen. Somit bestimmt die Beschaffenheit der Dreiecke, ob der daraus erzeugte Graph die Eigenschaften einer Triangulierung erfüllt.

4.1.2 Die Beschreibung des grafischen Modells

Definition des Modells

Um von der Triangulierung zu der Definition des grafischen Modells zu gelangen, weisen wir jedem der Triangulierung zugehörigem Dreieck noch einen Wert aus $[0, 1]$ zu, um die im Abschnitt 2.3 erwähnten Gewichte für jedes Dreieck zu berücksichtigen.

Definition 4.1.5. Ein Tupel $M = (\mathcal{T}, \omega)$ wird *Modell* genannt, falls $\mathcal{T} = (V, E)$ eine Triangulierung, $D_1, \ldots, D_m$ die zugehörigen Dreiecke und ω eine Gewichtungsfunktion mit $\omega : \{D_1, \ldots, D_m\} \mapsto [0, 1]$ ist.

Im weiteren Verlauf dieser Arbeit werden wir das Symbol M für das Modell und $\triangle_M$ für die Vereinigung der Dreiecksflächen der zugehörigen Dreiecke verwenden. Also ist $\triangle_M$ durch

$$\triangle_M := \bigcup_{i=1}^{m} \triangle_i$$

definiert.

Transformation des Modells

Wie schon in Abschnitt 2.1 erwähnt, ist das Ziel eine Transformation zu berechnen, die die Koordinaten des Modells in die der Punktwolke überführt. Folglich ist es erforderlich, die Transformation des Modells ebenfalls mathematisch zu beschreiben. Sei dazu $R \in \mathbb{R}^{3\times3}$ eine orthogonale Matrix und $t \in \mathbb{R}^3$ ein Translationsvektor. Dann bezeichnen wir die Funktion $T^v_{R,t} : \mathbb{R}^3 \mapsto \mathbb{R}^3$ mit

$$T^v_{R,t}(w) = Rw + t$$

als *Transformation* von w für die orthogonale Matrix R und den Translationsvektor t.

Weiterführend wird nun die Transformation auf einem Dreieck $D = (q, q_1, q_2)$ durch

$$T^d_{R,t}(D) = (T^v_{R,t}(q), T^v_{R,t}(q_1), T^v_{R,t}(q_2))$$

und auf einer Triangulierung $\mathcal{T} = (V, E)$ mittels

$$T^g_{R,t}(\mathcal{T}) := \left(T^v_{R,t}(V), \left\{\{T^v_{R,t}(u), T^v_{R,t}(w)\} : \{u, w\} \in E\right\}\right),$$

definiert, wobei die Label der Knoten von $\mathcal{T}$ durch die Transformation $T^g_{R,t}$ erhalten bleiben sollen und $T^v_{R,t}(V)$ die Menge aller mittels $T^v_{R,t}$ transformierten Elemente aus V ist.

Da $T^v_{R,t}$ ein Homöomorphismus ist, gilt folglich, dass $T^g_{R,t}(\mathcal{T})$ wieder eine Triangulierung ist. Des Weiteren ändern zwar die zugehörigen Dreiecke von $T^g_{R,t}(\mathcal{T})$ ihr Lage in $\mathbb{R}^3$ gegenüber den zugehörigen Dreiecken von $\mathcal{T}$, bleiben jedoch konsistent bzgl. der Normalenvektoren, da diese Transformation nicht die Label der Knoten ändert.

Um nun abschließend die Transformation auf einem Modell zu definieren, muss nur noch die Gewichtungsfunktion berücksichtigt werden.

Definition 4.1.6. Sei $M = (\mathcal{T}, \omega)$ ein Modell, R eine orthogonale Matrix und t ein Translationsvektor, dann bezeichnen wir die Funktion

$$T_{R,t}(M) = \left(T^g_{R,t}(\mathcal{T}), \omega\left({T^d_{R,t}}^{-1}(\cdot)\right)\right)$$

als Transformation des Modells bzgl. einer orthogonalen Matrix R und eines Translationsvektors t, wobei ${T^d_{R,t}}^{-1}$ die Inverse von $T^d_{R,t}$ bezeichnet.

Projektion auf das Modell

In diesem Abschnitt definieren wie die Projektion eines Punktes $p \in \mathbb{R}^3$ auf ein Modell $M = (\mathcal{T}, \omega)$. Die Definition ist notwendig, um z. B. den nächstgelegenen Punkt auf dem Modell zu dem Punkt p zu bestimmen.

Die Projektion soll eine Funktion sein, die einen Punkt in den nächstliegenden Punkt auf dem Modell abbildet. Jedoch kann es zu einem Punkt p mehrere nächstliegende Punkte auf dem Modell geben. Man betrachte dazu Abbildung 4.1, in der

das Modell durch zwei Dreiecke dargestellt ist. Haben die Punkte d_1, d_2 den gleichen Abstand zu p, so ist die Projektion nicht eindeutig. Um dieses Problem zu vermeiden, bestimmen wir zunächst die Menge der nächstliegenden Punkte auf dem Modell zu einem Punkt p und identifizieren den Punkt der Menge, der auf der Dreiecksfläche mit dem niedrigsten Index liegt, als die Projektion des Punktes p auf das Modell.

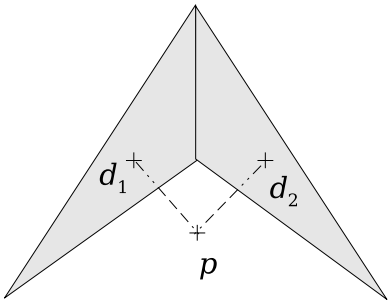


Abbildung 4.1: Veranschaulichung der Mehrdeutigkeit der Projektion

Seien dazu $\triangle_1, \ldots, \triangle_m$ die Dreiecksflächen der zugehörigen Dreiecke der Triangulierung $\mathcal{T}$ und $\mathcal{S}_p$ die Menge der nächstliegenden Punkte auf dem Modell M zu dem Punkt p, also

$$\mathcal{S}_p := \left\{ d \in \triangle_M : \|d - p\|_2 \leq \|\hat{d} - p\|_2 \ \forall \hat{d} \in \triangle_M \right\}.$$

Sei weiter $\Pi_M : \mathbb{R}^3 \mapsto \triangle_M$ eine Funktion und für ein $p \in \mathbb{R}^3$ bezeichne $k_p \in \{1, \ldots, m\}$ den kleinsten Index mit $\Pi_M(p) \in \triangle_{k_p}$. Die Funktion Π_M heißt dann *Projektion* eines Punktes p auf das Modell M, falls $\Pi_M(p) \in \mathcal{S}_p$ und kein $d \neq \Pi_M(p)$ existiert mit $d \in \mathcal{S}_p \cap \triangle_k$ für $k < k_p$.

4.1.3 Modellierung der Scandaten

Die Daten, die man aus der Vermessung des Objektes erhält und in Abschnitt 2.4 angesprochen wurden, werden wir in den folgenden Kapiteln meist als Punktwolke bezeichnen. Jedoch wird im Abschnitt 4.3.2 die Eigenschaft benötigt, dass die Messdaten äquidistant entlang der x- und y-Achse der Kamerakoordinaten aufgenommen wurden. Um diese Eigenschaft zu berücksichtigen, erweitern wir den Begriff der Punktwolke um den der *Scandaten*, wobei die Scandaten immer bezogen auf ein Modell M definiert werden. Zusätzlich wird hier auch die Fluktuation der Messdaten durch eine Zufallsfunktion, deren absoluter Betrag beschränkt ist, mit einbezogen.

Definition 4.1.7. Sei P eine Punktwolke der Größe $n = s_1 \cdot s_2$ mit $s_1, s_2 \in \mathbb{N}$ und $M = (\mathcal{T}, \omega)$ ein Modell und $\triangle_1, \ldots, \triangle_m$ die Dreiecksflächen der zugehörigen Dreiecke. Sei weiter zu einem $x, y \in \mathbb{R}$ die Gerade $\mathcal{G}_{x,y}$ mit

$$\mathcal{G}_{x,y} := \left\{ \begin{pmatrix} x \\ y \\ r \end{pmatrix} : r \in \mathbb{R} \right\}$$

gegeben und $D_{R,t} : \mathbb{R}^2 \mapsto \mathbb{R}^3$ sei eine Funktion einer orthogonalen Matrix $R \in \mathbb{R}^{3\times 3}$ und einem Translationsvektor $t \in \mathbb{R}^3$ mit

$$D_{R,t}(x,y) := \begin{cases} (x, y, \infty)^T & \text{für } \mathcal{G}_{x,y} \cap T^v_{R,t}(\triangle_M) = \emptyset, \\ \underset{g \in \mathcal{G}_{x,y} \cap T^v_{R,t}(\triangle_M)}{\operatorname{argmin}} \|g\|_2 & \text{sonst.} \end{cases}$$

Existieren x_0, y_0, t_x, $t_y > 0$, eine Zufallsfunktion $\xi : \{1, \ldots, s_1\} \times \{1, \ldots, s_2\} \mapsto \mathbb{R}$ mit $|\xi| < \varepsilon$ für $\varepsilon \geq 0$ klein und eine bijektive Funktion $G_{R,t} : \{1, \ldots, s_1\} \times \{1, \ldots, s_2\} \mapsto P$ mit

$$G_{R,t}(i,j) = D_{R,t}\Big(x_0 + (i-1)t_x, y_0 + (j-1)t_y\Big) + (0, 0, \xi(i,j))^T \quad \forall i, j$$

für eine orthogonale Matrix R und einen Translationsvektor t, so wird die Punktwolke P *Scandaten* des Modells M genannt.

Das äquidistante Gitter aus Abschnitt 2.4 wird in der Definition 4.1.7 durch $\{(x_0 + (i-1)t_x, y_0 + (j-1)t_y) : i = 1, \ldots, s_1 \text{ und } j = 1, \ldots, s_2\}$ und der jeweilige Durchstoßungspunkt durch die Funktion $D_{R,t}$ beschrieben.

4.1.4 Das Optimierungsproblem

In den letzten Abschnitten wurden alle benötigten Begriffe eingeführt, um das Optimierungsproblem dieser Arbeit zu formulieren. Seien $M = (\mathcal{T}, \omega)$ ein Modell und $\triangle_1, \ldots, \triangle_m$ die Dreiecksflächen der zugehörigen Dreiecke $D_1, \ldots, D_m$ der Triangulierung $\mathcal{T}$ und $P = \{p_1, p_2, \ldots, p_n\}$ eine Punktwolke. Sei weiter zu einer orthogonalen Matrix R und einem Translationsvektor t eine Funktion $\iota_{R,t} : P \mapsto \{1, \ldots, m\}$ gegeben, die einen Punkt $p \in P$ auf den kleinsten Index $k_p \in \{1, \ldots, m\}$ mit

$$\Pi_{T_{R,t}(M)}(p) \in \triangle_{k_p}$$

abbildet, wobei $\Pi_{T_{R,t}(M)}$ die in Abschnitt 4.1.2 definierte Projektion ist.

Dann beschreiben wir das Problem aus Kapitel 2 mathematisch durch

$$\min_{R \in \mathcal{P}, t \in \mathbb{R}^3} f(R,t) := \sum_{p \in P} \omega_{T_{R,t}}(D_{\iota_{R,t}(p)}) \|\Pi_{T_{R,t}(M)}(p) - p\|_2^2, \tag{4.1}$$

wobei $\mathcal{P}$ die Menge der orthogonalen Matrizen und $\omega_{T_{R,t}}$ die Gewichtungsfunktion des Modells $T_{R,t}(M)$ ist. In diesem Zusammenhang wird das Maß die für Anpassung zwischen Modell und Punktwolke durch den mittleren Abstand zwischen Modell und den Punkt der Punktwolke beschrieben.

Im nächsten Abschnitt werden wir Ansätze zur Lösung des Optimierungsproblems vorstellen und diskutieren.

4.2 Ansätze / Stand der Technik

Wie schon in Kapitel 1 verdeutlicht, findet das Problem (4.1) in leicht abgeänderter Form zahlreiche Anwendungen. Ebenso umfangreich ist in der Literatur die Fülle an Vorschlägen bzw. Ansätzen [5, 17, 22], dieses Problem zu lösen. Es kristallisieren sich jedoch im Wesentlichen zwei Kategorien aus diesen Ansätzen heraus [21]. Die erste befasst sich mit der Grobregistrierung. Das Problem hierbei ist, eine Rotation und Translation zu finden, so dass die Lage des transformierten Modells grob mit der der Punktwolke übereinstimmt. Meist möchte man einen „guten“ Anfangswert der Rotation und Translation erhalten, um danach ein iteratives Verfahren anzuwenden. Die zweite Kategorie optimiert dann lokal das Funktional f in (4.1), wobei bei diesen Ansätzen schon vorausgesetzt wird, dass die Lage der Punktwolke grob mit der des Modells übereinstimmt.

4.2.1 Grobregistrierung

Für die Grobregistrierung gibt es vielerlei Methoden, deren Effektivität jedoch stark von der Form des zu matchenden Objektes und der Deckungsgleichheit des Modells mit der jeweiligen Punktwolke abhängt. Deckungsgleich bedeutet in diesem Zusammenhang, dass das Funktional f im globalen Minimum null ist und es zu jedem Dreieck des Modells nach der optimalen Transformation einen Punkt der Punktwolke gibt, der in der Umgebung des jeweiligen Dreiecks liegt. Sind Modell und Punktwolke wenigstens annähernd deckungsgleich, d. h., das Funktional f ist am Minimum nahe null, so ist z. B. die Methode der Hauptkomponentenanalyse eine Möglichkeit, die Lage zwischen Punktwolke und Modell grob in Übereinstimmung zu bringen.

Bei der Hauptkomponentenanalyse werden der Schwerpunkt und die drei Hauptkomponenten (siehe Abschnitt 4.3.1) des Modells und der Punktwolke bestimmt. Als Hauptkomponente versteht man diejenige Richtung, die orthogonal zu allen anderen schon bestimmten Richtungen ist und in der ausgehend vom Schwerpunkt die Varianz bezogen auf den Abstand zu den Modellpunkten bzw. Punkten in dieser Richtung am größten ist. Die Methode ist, diejenige Transformation zu ermitteln, die die Schwerpunkte aufeinander legt und die jeweiligen Hauptkomponenten des Modells in die jeweils anderen der Punktwolke überführt.

Andere Verfahren bedienen sich der Extraktion von meist punktbezogenen lokalen Merkmalen, sogenannten *Features*. Diese beinhalten Eigenschaften, die die lokale Umgebung charakterisieren. In [20] werden z. B. die Gaußsche Krümmung und die mittere Krümmung als Feature verwendet. Diese können aber auch von anderer Natur sein. Werden z. B. Farben verwendet, so ist die Farbinformation ebenso ein geeignetes Merkmal. Oft werden Features auch so gewählt, dass markante Stellen in Punktwolke und Modell wie z. B. scharfe Kanten erkannt werden.

Hierbei ist zu bemerken, dass diese Merkmale möglichst robust sowohl in Modell als auch in der Punktwolke erkennbar sein sollten, so dass nach der Berechnung dieser Merkmale korrespondierende Features in Modell und Punktwolke ermittelt werden können. Dies bedingt letztendlich auch, dass die Features rotations- und translationsinvariant sind. Ferner weisen wir darauf hin, dass jeweils drei Punkte (linear unabhängig) aus der Punktwolke, deren korrespondierende Punkte (linear unabhängig) im Modell bekannt sind, schon hinreichend sind, um eine eindeutige Transformation zu berechnen. Siehe dazu auch den Beweis des Satzes 5.3.1 aus dem Abschnitt 5.3.2.

Um geeignete Featurepunkte und deren Korrespondenzen zu finden, gibt es eine Reihe von Ansätzen. In [20] werden, um zwei Punktwolken zu registrieren, Gebiete mit ähnlichen Features gesucht und mit Hilfe des Relaxation Labeling Verfahrens eine Zuordnung zwischen den Gebieten in der einen Punktwolke und Gebieten in der anderen bestimmt, um so letztendlich die gesuchte Transformation zu bestimmen.

Ähnlich ist das Vorgehen in [21]. Für zwei zu registrierende Triangulierungen werden Segmente bzw. Gebiete gebildet, die sich durch eine Ebene mit einem nur kleinen Fehler annähern lassen, d. h., es werden elementare Geometrien gesucht, die das Objekt beschreiben. In diesem Fall sind es Ebenen. Die erwähnten Gebiete werden mit einem vollverbundenen Graphen (G, E_G) in Beziehung gestellt, wobei jedes Gebiet ein Knoten ist und jeder Kante aus E_G eine Gewichtung zugeordnet wird, die von den Parametern der zugehörigen Ebenen abhängt. Resultierend erhält man zwei Graphen, zu denen Subgraphenisomorphismen mit mindestens drei Knoten gesucht werden. Mit Hilfe eines definierten Maßes für die Ähnlichkeit der Subgraphen werden so die am besten korrespondierenden Subgraphen bestimmt, woraus folglich auch die Transformation berechnet werden kann.

Im weiteren Verlauf stellen wir iterative Verfahren für die Feinregistrierung vor, die alle einen Startwert benötigen.

4.2.2 Feinregistrierung

Der erste Ansatz für die Feinregistrierung ist die direkte Minimierung des Funktionals. Dazu schlagen Rabbani und van den Heuvel in [23] einen Algorithmus vor, der das Funktional in (4.1) mit einem Standardverfahren für nichtlineare least squares Probleme, dem Levenberg-Marquardt-Verfahren, minimiert. Die dabei benötigten partiellen Ableitungen werden durch finite Differenzen approximiert, ebenso wird die Rotation durch Quaternionen repräsentiert. Die Autoren erhalten einen global kon-

vergenten Algorithmus. Dieses Verfahren hat den Nachteil, dass die Berechnung der Jacobi-Matrix im Levenberg-Marquardt-Algorithmus sehr kostenintensiv ist, jedoch den entscheidenden Vorteil, dass in jedem Iterationsschritt tatsächlich eine Abstiegsrichtung gewählt und das Funktional minimiert wird. In den folgenden Verfahren werden die partiellen Ableitungen nicht mehr beachtet und folglich werden auch deren Iterationsschritte weniger kostenintensiv.

Die Feinregistrierung betreffend taucht in der Literatur sehr häufig der Name des Iterative Closest Point Algorithmus (ICP) auf. Dieses Verfahren geht zurück auf Besl und McKay [6] und ist ein Verfahren, um eine Punktwolke in eine Referenzpunktwolke einzupassen. Dabei wird zu jedem Punkt der einen Punktwolke ein Punkt der jeweils nächstliegenden Punkte in der Referenzpunktwolke bestimmt und anschließend jene Transformation der einen Punktwolke berechnet, die den Abstand zum Quadrat zwischen den korrespondierenden Punkten im Mittel minimiert. Jenes Vorgehen wird iterativ angewandt. Sei $A := \{a_1, \dots, a_n\}$ eine Punktwolke und $B := \{b_1, \dots, b_m\}$ die Referenzpunktwolke. Sei weiter $A^{(i)} := \{a_1^{(i)}, \dots, a_n^{(i)}\}$ die zur Iteration i berechnete Punktwolke mit $A^{(0)} := A$. Dann wird zur Iteration $i+1$ jene Punktwolke $C^{(i)} := \{c_1^{(i)}, \dots, c_n^{(i)}\}$ berechnet, für die

$$c_j^{(i)} \in \left\{ b \in B : \left\| a_j^{(i)} - b \right\|_2 \text{ minimal} \right\} \quad \forall j = 1, \dots, n$$

gilt.

Nun wird eine Rotationsmatrix $R^{(i)}$ und ein Transformationsvektor $t^{(i)}$ bestimmt, so dass das Funktional

$$h(R,t) := \sum_{j=1}^{n} \| R a_j^{(i)} - c_j^{(i)} - t \|_2^2$$

minimal ist. Jenes Vorgehen wird mit $A^{(i+1)} := \{R^{(i)} a_1^{(i)} - t^{(i)}, \dots, R^{(i)} a_n^{(i)} - t^{(i)}\}$ wiederholt, bis die Änderung in h zweier aufeinander folgender Iterationen hinreichend klein ist. Dieses Verfahren wird als sehr genau und schnell beschrieben [12], die Güte des Verfahrens ist jedoch stark von dem Anfangswert abhängig und damit nicht robust gegenüber der anfänglichen Transformation.

Einen Ansatz, um dieses Problem zu verringern, wird in [12] vorgeschlagen. Dabei gehen die Autoren wieder von einer Punktwolke A und einer Referenzpunktwolke B aus, wobei sie annehmen, dass die Punkte aus A gegenüber den Korrespondenzpunkten in B verrauscht sind und beschreiben dies mit einer Normalverteilung $\mathcal{N}(\mu, \Sigma)$. Daraus folgernd formulieren sie das oben genannte Problem der Anpassung zweier Punktwolken A, B als Likelihood-Schätzung bezüglich der Transformation mit zusätzlichen Matching-Gewichten, die die Wahrscheinlichkeit der Korrespondenz zweier Punkte angeben. Ein Spezialfall dieser Formulierung ist das oben genannte ICP.

Nun wenden sie das Expectation-Maximization-Prinzip auf jene Likelihood-Schätzung an, in dem sie iterativ die Erwartung der Matching-Gewichte berechnen und

anschließend mit deren Hilfe die Likelihood-Schätzung bzgl. der Transformation maximieren. Sie erhalten den so genannten EM-ICP (Expectation-Maximization) als Resultat, welcher Konvergenz garantiert.

Für ein gleichmäßiges und gleichgerichtetes Rauschen wird durch die Varianz Σ mit $\Sigma = \mathrm{Diag}(\sigma, \ldots, \sigma)$ eine neue Variable σ eingeführt, welche von den Autoren als Skalierungsfaktor interpretiert wird. Bei hohen Werten von σ verhält sich der EM-ICP sehr robust, hingegen bei kleineren Werten wie der ICP. Daraus ableitend schlagen die Autoren vor, die Varianz σ anfänglich groß zu wählen und iterativ mittels eines einfachen Vorgehens zu verkleinern, um einen robusten Algorithmus zu erhalten, der jedoch ebenso genau wie der ICP ist. Zusätzlich wird noch eine Reduzierung der Anzahl der Punkte vorgenommen, die iterativ, je mehr man sich der Lösung nähert, wieder aufgehoben wird. Die Autoren erhalten einen Algorithmus, der robuster und schneller ist, wobei die Reduzierung der Punktanzahl wesentlich dazu beiträgt.

Um noch auf weitere Verbesserungsvorschläge für den ICP hinzuweisen, sei an dieser Stelle der Vorschlag aus [13] erwähnt, in dem anfänglich ein approximierter kd-Baum erstellt wird, um die Suche nach dem nächstliegenden Punkt in jeder Iteration im ICP zu beschleunigen. Ein parallele Implementierung für das ICP ist in [18] vorgeschlagen.

4.3 Die mathematische Modellierung

In diesem Abschnitt greifen wir die Ideen aus Abschnitt 4.2 auf und stellen das in dieser Arbeit behandelte mathematische Modell zur Lösung der Aufgabenstellung vor. In Anlehnung an die übliche Aufteilung der Aufgabe in Grob- und Feinregistrierung bauen wir die Modellierung ähnlich strukturiert auf. In Abschnitt 4.3.1 suchen wir zunächst nach einer guten Anfangsposition des Modells gegenüber der Punktwolke. Anschließend stellen wir den Ansatz in Abschnitt 4.3.2 vor, mit dem wir ausgehend von dieser Anfangsposition das Funktional in (4.1) minimieren. Danach widmen wir uns dem Problem des Preprocessings bzw. der Datenaufbereitung, um die Laufzeit zu verringern.

4.3.1 Grobregistrierung

In Abschnitt 4.2.1 wurden einige Vorschläge vorgestellt, wie ein guter Anfangswert für ein späteres iteratives Verfahren zur Lösung des Problems in (4.1) gefunden werden kann. Aus den Gegebenheiten erscheint uns die Hauptkomponentenanalyse als die beste Herangehensweise, da Form und Größenverhältnisse von Modell und Messung größtenteils übereinstimmt und diese Methode sowohl einfach zu implementieren, als auch schnell zu berechnen ist. Ferner erwarten wir, dass dieses Verfahren einen guten Anfangswert zur Lösung des hier beschriebenen Problems liefert. Nachteilig ist allerdings, falls z. B. das Modell nicht komplett durch die Messung abgebildet

wurde, dass die Hauptkomponenten des Modells mit denen der Messung nicht übereinstimmen müssen und so einen schlechten Anfangswert liefern. Abhilfe schaffen Featureextraktionen sowohl im Modell als auch in der Punktwolke, auf die wir in dieser Modellierung nicht eingehen.

4.3.2 Feinregistrierung

Da die direkte Minimierung des Funktionals in (4.1) zu komplex erscheint, greifen wir für die Feinregistrierung die Idee des ICP auf. Dazu sei $k : \mathbb{R}^3 \mapsto \triangle_M$ eine Funktion, die einem Punkt der Punktwolke einen korrespondierenden Punkt auf dem Modell zuordnet. Für diese werden nun $R \in \mathcal{P}$ und $t \in \mathbb{R}^3$ bestimmt, die die Summe

$$f(R,t) := \sum_{p \in P} \|Rk(p) - t - p\|_2^2,$$

minimieren. Anschließend wenden wir die Transformation mit R, t optimal auf das Modell an. Hierbei ist zu bemerken, dass keine Nachbarschaftsbeziehungen betrachtet werden, sondern zunächst nur Punkt-zu-Punkt-Korrespondenzen.

Der ICP wird, wie oben erwähnt, als sehr genau und schnell beschrieben, jedoch weniger robust, da dieser lokal optimiert und somit in lokale Minima konvergiert, die unter Umständen weit entfernt von der eigentlichen Optimallösung sind. Das Ziel ist es, die zwei ersteren Eigenschaften zu übernehmen und die letztere zu verbessern.

Eine mögliche Schwäche im ICP scheint die Wahl für k zu sein, zu einem bestimmten Punkt $p \in P$ jeweils den nächstliegenden Punkt im Modell zu nehmen. Dies bedeutet mitunter, dass mehreren Punkten aus P ein und derselbe Korrespondenzpunkt zugewiesen werden kann, was ein Widerspruch zu den eigentlichen realen Verhältnissen ist. Deshalb wählen wir für die Korrespondenzpunktsuche eine eher globale Herangehensweise.

Flussproblem minimaler Kosten

Die Idee für die Korrespondenzpunktsuche ist gegenüber der im ICP, die Punkte aus P Dreiecken aus $\mathcal{T}$ zuzuordnen, wobei die Anzahl der Punkte, die ein und demselben Dreieck zugeordnet werden, nach oben beschränkt werden sollen. Ist ein Punkt $p_i \in P$ einem Dreieck D_j zugeordnet worden, so ergibt sich der jeweilige Korrespondenzpunkt k_i von p_i aus

$$\Pi_{\triangle_j}(p_i) := \underset{d \in \triangle_j}{\operatorname{argmin}} \|d - p_i\|_2^2. \tag{4.2}$$

Da $\triangle_j \neq \emptyset$ abgeschlossen und konvex und $\mathbb{R}^3$ zusammen mit dem Euklidischen Skalarprodukt ein Hilbertraum ist, existiert dieser Korrespondenzpunkt und ist auch eindeutig [15, Abschnitt 3.1].

Wir modellieren die Korrespondenzpunktsuche mit Nebenbedingungen als Flussproblem minimaler Kosten. Dazu sei $G = (V, E)$ das in Abbildung 4.2 dargestellte

gerichtete Netzwerk mit der Knotenmenge $V = \{1, \ldots, n+m+1\}$ und der entsprechenden Kantenmenge E. Dabei repräsentiere der Knoten $i \in V$ den Punkt $p_i \in P$ für alle $i = 1, \ldots, n$ und der Knoten $n+j$ das Dreieck D_j für alle $j = 1, \ldots, m$. Die Knotenmenge E bestehe aus den Kanten (i, j) für $i = 1, \ldots, n$ und $j = n+1, \ldots, n+m$ sowie aus den Kanten $(j, n+m+1)$ für alle $j = n+1, \ldots, n+m$.

Weiter beschreibe $u \in \mathbb{R}^m$ die maximalen Kapazitäten auf den Kanten $(j, n+m+1)$ für $j = n+1, \ldots, n+m$. Dieser Vektor gibt die maximale Anzahl an Punkten an, die dem jeweiligen Dreieck zugeordnet werden darf. Sei weiter die maximale Kapazität aller weiteren Kanten gleich eins und die minimale Kapazität aller Kanten gleich null.

Wir bezeichnen mit $x_i \in \{0,1\}^m$ für alle $i = 1, \ldots, n$ den Fluss, der auf den Kanten zwischen dem Punkt i und den Dreiecken fließt, wobei $x_i(j)$ den Fluss auf der Kante $(i, j+n)$ für alle $j = 1, \ldots, m$ angibt.

Versehen wir die Kanten (i, j) für alle $i = 1, \ldots, n$ und $j = n+1, \ldots, n+m$ mit Kosten, die abhängig von der Distanz zwischen dem jeweiligen Punkt und Dreieck sind, die Knoten $1, \ldots, n$ zusätzlich mit einer Quelle mit Überschuss eins und den Knoten $n+m+1$ mit einer Senke von $-n$, so erhalten wir ein Flussproblem minimaler Kosten (min-cost-flow-problem) [4, Abschnitt 1.2]. Sei $c = (c_1^T, c_2^T, \ldots, c_n^T)^T \in \mathbb{R}^{m \cdot n}$ der Kostenvektor mit

$$c_i = \left(\|p_i - \Pi_{\triangle_1}(p_i)\|_2^2, \ldots, \|p_i - \Pi_{\triangle_m}(p_i)\|_2^2 \right)^T \quad \forall i = 1, \ldots, n$$

und $\Pi_{\triangle_j} : \mathbb{R}^3 \mapsto \triangle_j$ die Projektion aus (4.2) für alle $j = 1, \ldots, m$. Dann modellieren wir das Korrespondenzenproblem mittels des folgenden Optimierungsproblems:

$$\begin{array}{ll} \min & c^T x \\ \text{s.t.} & e^T x_j = 1 \quad \forall j = 1, \ldots, n, \\ & Dx \leq u, \\ & x = \left(x_1^T, x_2^T, \ldots, x_n^T\right)^T, x_j \in \{0,1\}^m, \end{array} \tag{4.3}$$

wobei $D \in \{0,1\}^{m \times m \cdot n}$ eine Matrix mit

$$Dx = \sum_{i=1}^{n} x_i$$

und $e = (1, 1, \ldots, 1)^T \in \mathbb{R}^m$ der Vektor, der nur Einsen enthält.

Eine offene Frage bleibt noch, wie die maximalen Kapazitäten u zweckmäßig bestimmt werden können. Hierbei gehen wir davon aus, dass die Punkte der Punktwolke auch Scandaten sind und die Koordinaten des Modells grob mit denen der Scandaten nach der Grobregistrierung übereinstimmen. Wenn die zwei Bedingungen erfüllt sind, liegt die Kameraebene bezogen auf das Modell annähernd parallel zur x-y-Ebene und folglich ist annähernd bekannt, wieviele Punkte eine bestimmte Dreiecksfläche des Modells abbilden, da die Punkte der Scandaten äquidistant entlang der x- und y-Achse liegen.

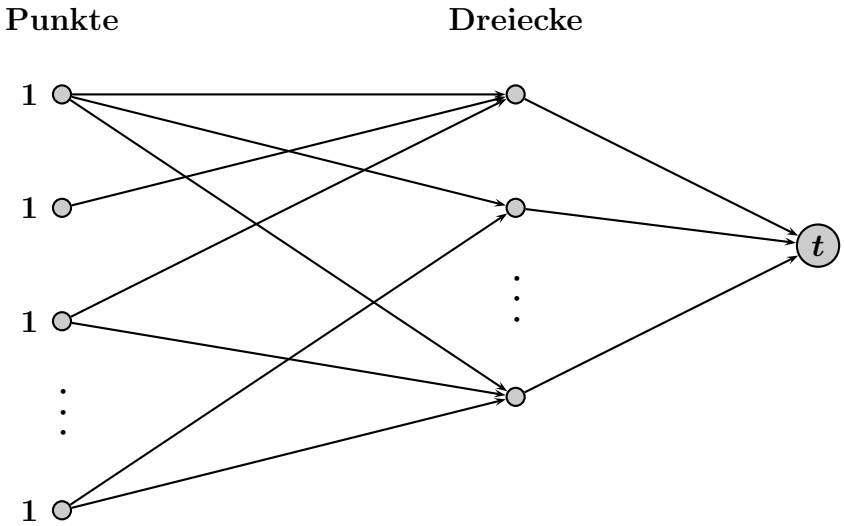


Abbildung 4.2: Darstellung des Flussproblems minimaler Kosten

Wir berechnen dabei nur die grob geschätzte Anzahl der Punkte, die auf eine bestimmte Dreiecksfläche fallen und nicht die exakte Anzahl, da anzunehmen ist, dass diese durch die grobe Anpassung von Modell und Scandaten nicht aussagekräftiger ist, jedoch höheren Rechenaufwand erfordert.

Seien $\{(x_0 + (i-1)t_x, y_0 + (j-1)t_y) : i = 1, \ldots, s_1, j = 1, \ldots, s_2\}$ die Gitterpunkte auf der x-y-Ebene für $s_1, s_2 \in \mathbb{N}$ und $t_x, t_y > 0$. Dann gibt der Wert n_s mit

$$n_s := \frac{s_1 \cdot s_2}{(s_1 - 1)t_x \cdot (s_2 - 1)t_y}$$

eine Schätzung der Anzahl der Gitterpunkte an, die auf einer Flächeneinheit liegen. Projiziert man nun mittels $\Phi : \mathbb{R}^3 \mapsto \mathbb{R}^3$ mit

$$\Phi(v) = (v(1), v(2), 0)^T$$

die Knoten der Dreiecke D_i des Modells auf die x-y Ebene und berechnet die Fläche F_i des projizierten Dreiecks, so erhält man durch

$$n_{D_i} := F_i \cdot n_s$$

die mittlere Anzahl der Punkte, die das Dreieck D_i abbilden.

Folglich drückt n_{D_i} ein Maß für die Anzahl der Punkte aus, die dem Dreieck zugewiesen werden. Um nun die maximale Kapazität u festzulegen, bestimmen wir den

nächstgrößeren ganzzahligen Wert von n_{D_i}, d. h.

$$u(i) := \lceil n_{D_i} \rceil \quad \forall i = 1, \ldots, m.$$

Falls nun die Summe der Maximalkapazitäten von u so gewählt wird, dass sie größer gleich n ist, so existiert eine zulässige Lösung des Optimierungsproblems (4.3), und da die Menge der Lösungen endlich ist, existiert auch eine Optimallösung von (4.3). Auf die Lösung des Problems (4.3) werden wir in Kapitel 5.3 eingehen.

An dieser Stelle sei noch bemerkt, dass mit obiger Herangehensweise sich die Berücksichtigung von Ausreißern einfach realisieren lässt, in dem ein zusätzlicher Knoten $\{n + m + 2\}$ zu V dazugefügt und E um die Kantenmenge

$$\widetilde{E} := \{(1, n + m + 2), \ldots, (n, n + m + 2)\} \cup \{(n + m + 2, n + m + 1)\}$$

erweitert wird. Setzt man die Kosten auf den Kanten der Menge $\widetilde{E}$ auf null und beschränkt man von oben die Kapazität der Kante $(n + m + 2, n + m + 1)$ auf die maximale Anzahl an Ausreißern, so wird erreicht, dass diejenigen Punkte, deren Zuordnung zu allen Dreiecken hohe Kosten verursachen würden, als Ausreißer deklariert werden.

Dies ist jedoch auch eine Herangehensweise, um die Zulässigkeit des Problems zu erreichen. Dazu erweitern wir ebenfalls die Knotenmenge V um den Knoten $n+m+2$ und die Kantenmenge E um $\widetilde{E}$. Falls z. B. die Bedingung

$$n \leq \sum_{i=1}^{m} u(i)$$

nicht erfüllt ist, so kann man durch das Setzen von hohen Kosten auf den Kanten $\widetilde{E} \setminus \{(n_{n+m+2}, n_{n+m+1})\}$ und durch das Setzen der maximalen Kapazität der Kante (n_{n+m+2}, n_{n+m+1}) auf einen Wert, der größer gleich $n - \sum_i u_i$ ist, die Zulässigkeit realisieren. Diese Möglichkeit werden wir für die numerischen Tests in Kapitel 7 nutzen, wenn nicht mehr jeder Punkt eine Kante zu allen Dreiecken hat. Siehe hierzu den Abschnitt 4.3.3.

Transformationsoptimierung

Aus einer zulässigen Lösung x des oben beschriebenen Optimierungsmodells erhält man die Korrespondenzpunkte auf dem Modell zu jedem Punkt der Punktwolke, in dem die Projektion $\Pi_{\triangle_{j(i)}}(p_i)$ aus (4.2) mit $j(i) \in \{j : x_i(j) = 1\}$ für alle $p_i \in P$ angewendet wird. Diese Definition ist wohldefiniert, da $|\{j : x_i(j) = 1\}| = 1$, wenn x eine zulässige Lösung ist. Nun sind die Korrespondenzpaare bekannt und das weitere Vorgehen ist wie im ICP mit dem Unterschied, dass die Abstände zu den Dreiecken unterschiedlich gewichtet werden. Wir minimieren das Funktional f mit

$$f(R, t) := \sum_{i=1}^{n} \omega(D_{j(i)}) \| Rk(p_i) - t - p_i \|_2^2, \tag{4.4}$$

und wenden die Transformation $T_{R,t}$ für R, t optimal auf das Modell M an.

4.3.3 Aufbereitung der Daten

Wie schon in [12] festgestellt, siehe dazu Abschnitt 4.2.2, bewährt es sich bei iterativen Algorithmen oft, um an Schnelligkeit und Robustheit zu gewinnen, anfänglich die Datenmenge zu verkleinern, d. h. konkret die Anzahl der Punkte in der Punktwolke und die Anzahl der Dreiecke im Modell. Um jedoch nicht an Genauigkeit zu verlieren, wird die Datenmenge sukzessive in den nächsten Iterationen, je näher man sich an dem Minimum befindet, wieder erhöht. Dabei soll das mathematische Modell mit der erhöhten Datenmenge aufgrund der in der vorhergehenden Iteration berechneten Lösung leicht lösbar sein.

Des Öfteren ist diese Herangehensweise sogar unverzichtbar, da für die Lösung des mathematischen Modells mit der vollen Datenmenge oft enorme Ressourcen beansprucht werden. Kennt man jedoch eine approximierte Lösung, so wird die Berechnung der Lösung des mathematischen Modells auch für große Datenmengen oft deutlich schneller.

Um diese Vorgehensweise in den in Abschnitt 4.3.2 beschriebenen iterativen Ablauf zu integrieren, ist nun die Idee, zunächst die Punktwolke äquidistant entlang der x- und y-Achse auszudünnen und das Modell mit nur wenigen Dreiecken zu approximieren, falls die Datenmenge zu groß ist. Für diese reduzierte Datenmenge lösen wir das in Abschnitt 4.3.2 beschriebene Flussproblem (4.3) und erhalten so eine Lösung.

In der nächsten Iteration erhöhen wir die Datenmenge, d. h., wir heben die Anzahl der verwendeten Dreiecke an. Um nun die Lösung aus der ersten Iteration zu verwenden und das in dieser Iteration zu lösende Flussproblem minimaler Kosten zu vereinfachen, entfernen wir die Kanten, von einem Punkt der Punktwolke zu all den Dreiecken, die nicht an das Dreieck grenzen, zu welchem der Punkt in der vorhergehenden Iteration zugeordnet wurde.

Genauer legen wir fest, dass ein Punkt der Punktwolke, der zur ersten Iteration dem Dreieck D zugeordnet wurde, in dem Optimierungsmodell der nächsten Iteration nur Kanten zu den Dreiecken erhält, die das Dreieck D und dessen Nachbarn in dem Modell mit der erhöhten Anzahl an Dreiecken ersetzen. Diese Vorgehensweise kann nun, in dem die Anzahl der Dreiecke sukzessiv erhöht wird, für alle weiteren Iterationen fortgeführt werden.

Um sie jedoch umzusetzen, benötigen wir eine Hierarchie von Modellen, deren erste Stufe das Modell nur grob und alle weiteren immer feiner approximiert, wobei festgelegt sein muss, welche Dreiecke in der höheren Stufe welches Dreieck in der geringern Stufe ersetzen. Die Umsetzung solch einer Hierarchie werden wir in Abschnitt 5.1 des nächsten Kapitels ausführlich beschreiben. Des Weiteren ist noch zu klären, wann zwei Dreiecke im selben Modell benachbart sind.

Definition 4.3.1. Zwei Dreiecke D_i, D_j heißen *benachbart*, falls $\triangle_i \bigcap \triangle_j \neq \emptyset$.

Ist die höchste Stufe von allen Modellen erreicht, so erhält ein Punkt der Punktwolke, der zur vorhergehenden Iteration dem Dreieck D zugeordnet wurde, in dem Optimierungsmodell der aktuellen Iteration nur Kanten zu dem Dreieck D und dessen Nachbarn.

Zusätzlich ist es auch möglich, die Anzahl der Punkte zu reduzieren und diese stufenweise wieder zu erhöhen. Jedoch betrachten wir in dieser Arbeit eine feste Anzahl von Punkten über alle Iterationen.

5 Ansätze zur Lösung des mathematischen Modells

In diesem Kapitel lösen wir das im Kapitel 4 vorgestellte mathematische Modell und bilden die Grundlage für den in Kapitel 6 aufgelisteten Algorithmus. Im nächsten Abschnitt werden wir einen Ansatz zur Erstellung einer Hierarchie von Modellen vorstellen, um die in Abschnitt 4.3.3 skizzierte Datenaufbereitung durchzuführen. In den weiteren Abschnitten geht es um die Lösung der Optimierungsprobleme (4.3) und (4.4).

5.1 Aufbau einer Hierarchie von Modellen

Der in diesem Abschnitt beschriebene Ansatz zur Erstellung einer Hierarchie basiert auf einer Idee aus [9], die jedoch kein Optimalitätskriterium erfüllt. Dabei wird aus einer vorgegebenen Triangulierung $\mathcal{T} = (V, E)$ eines Bildes im Zweidimensionalen, d. h. $V \subset \mathbb{R}^2$, iterativ eine Menge $\widetilde{V} \subset V$ ausgewählt, für die $\{q, p\} \notin E$ für alle $q, p \in \widetilde{V}$ gilt. Die Autoren bezeichnen die Menge als unabhängig.

Diese Menge und deren zugehörige Kanten werden aus der Knotenmenge V bzw. aus der Kantenmenge E entfernt. Danach werden für jeden Knoten, der aus V entfernt wurde, diejenigen Knoten ausgewählt, zu denen eine gemeinsame Kante existierte, und zwischen diesen mittels des eindeutigen Delaunay-Kriteriums [10, Abschnitt 9.2] neu trianguliert. Es entsteht eine neue Triangulierung $\hat{\mathcal{T}} = (\hat{V}, \hat{E})$, die aus weniger Knoten besteht und damit weniger Information enthält. Iterativ angewandt entsteht so eine Hierarchie von Dreiecksnetzen $\mathcal{T}_0, \mathcal{T}_1, \ldots, \mathcal{T}_k$ mit $V_0 \supset V_1 \supset \ldots \supset V_k$ und $\mathcal{T}_i = (V_i, E_i) \quad \forall i = 1, \ldots, k$ und $k \in \mathbb{N}$. Die Idee der unabhängigen Menge ist dabei, die Topologie des Modells von einer Iteration auf die andere nicht zu stark zu verändern.

Bevor ein Knoten entfernt wird, werden für jeden Knoten Kosten bestimmt. Der Knoten wird entfernt, der mit den bereits in dieser Iteration entfernten Knoten eine unabhängige Menge bildet und unter allen Knoten, die diese Bedingung erfüllen, die geringsten Kosten hat. In [9] werden die Kosten abhängig von den partiellen Ableitungen der dem Knoten zugeordneten Bildintensität und einem Ähnlichkeitsmaß der Intensität zu der jeweiligen Umgebung des Knotens bestimmt.

Diese Idee der Hierarchiebildung wird für die hier beschriebene Herangehensweise adaptiert, allerdings wird sie auf Dreiecksnetze (Oberflächen) bzw. auf einer Triangulierung im Dreidimensionalen erweitert. Außerdem wird das Kriterium zur Bestim-

mung der Kosten pro Knoten verändert. Um jenen Ansatz auf Oberflächen im $\mathbb{R}^3$ zu erweitern, bedienen wir uns dennoch der eindeutigen Delaunay-Triangulierung im Zweidimensionalen. Sei also $\mathcal{T}_i = (V_i, E_i)$ die Triangulierung zur Iteration i.

Nun ist es das Ziel, zunächst die Kosten für jeden einzelnen Knoten aus V_i zu berechnen. Dafür bestimmen wir in Abschnitt 5.1.3 ein Differenzvolumen zwischen $\mathcal{T}_i$ und einer Triangulierung, deren Knotenmenge nicht diesen Knoten, jedoch alle anderen enthält. Dieses Volumen gibt dann die Kosten für jeden einzelnen Knoten an. Nun wird die Triangulierung $\mathcal{T}_i$ durch eine Triangulierung ersetzt, deren Knotenmenge um den Knoten mit den geringsten Kosten von V_i abweicht. Zusätzlich werden alle Knoten der neuen Triangulierung, die den Nachbarknoten des Knotens mit den geringsten Kosten entsprechen, markiert, so dass sie in dieser Iteration nicht mehr entfernt werden. Diese Vorgehensweise wird wiederholt, bis nur noch Knoten, die nicht mehr entfernt werden dürfen, vorhanden sind oder eine bestimmte Anzahl zu entfernender Knoten erreicht ist. Dann bildet die Menge der entfernten Knoten die unabhängige Menge $\widetilde{V}$ und die durch Ersetzungen entstandene Triangulierung die Triangulierung zur Iteration $i + 1$.

Um aus einer Triangulierung eine neue Triangulierung zu ermitteln, die bis auf einen Knoten die gleiche Knotenmenge und bis auf alle Kanten, die zu diesem Knoten inzident sind, die ebenso gleiche Kantenmenge hat, triangulieren wir zwischen den Nachbarknoten neu und erhalten so neue Dreiecke.

Um dies zu erreichen, projizieren wir im Abschnitt 5.1.1 alle Nachbarknoten auf eine geeignete Ebene, um dann im Abschnitt 5.1.2 mittels Delaunay im Zweidimensionalen zu triangulieren.

5.1.1 Bestimmung einer geeigneten Projektion

Sei v der Knoten, der aus der Menge V_i entfernt werden soll. Dieser sei so gewählt, dass $V_i \setminus \{v\}$ und E_i ohne die inzidenten Kanten von v noch einen zusammenhängenden Graphen bilden. Dann werden zunächst zu dem Knoten v alle zugehörigen Dreiecke D_j von $\mathcal{T}_i$ bestimmt, von denen der Knoten v ein Eckpunkt ist.

Es bezeichne J_D mit $|J_D| = r$ für $r \in \mathbb{N}$ die Indexmenge dieser Dreiecke, $\triangle_j$ für alle $j \in J_D$ die zugehörigen Dreiecksflächen und $G = (V_v, E_v)$ einen Graphen mit V_v, $|V_v| > 3$, die Menge aller Eckpunkte dieser Dreiecke und E_v die Kantenmenge mit

$$E_v = \{\{u, w\} : \{u, w\} \in E_i \text{ und } u, w \in V_v\}.$$

Nun ist das Ziel, die Vereinigung der Dreiecksflächen $\triangle := \bigcup_{j \in J_D} \triangle_j$ homöomorph auf ein zweidimensionales Objekt, in dem Fall eine Ebene $\mathcal{E}$, mittels der Projektion $\Pi_{\mathcal{E}} : \mathbb{R}^3 \mapsto \mathcal{E}$ mit

$$\Pi_{\mathcal{E}}(d) = \min_{w \in \mathcal{E}} \|w - d\|_2$$

abzubilden, so dass für alle Dreiecke $D_j = \left(q^j, q_1^j, q_2^j\right)$ mit $j \in J_D$ die projizierten Dreiecke $\left(\Pi_{\mathcal{E}}\left(q^j\right), \Pi_{\mathcal{E}}\left(q_1^j\right), \Pi_{\mathcal{E}}\left(q_2^j\right)\right)$ eine Triangulierung bilden, deren Knoten sich auf

der Ebene befinden. Um die hier erwähnte Triangulierung von der Triangulierung $\mathcal{T}_i$ besser unterscheiden zu können, werden wir die Triangulierung, die nur aus einer reduzierten Anzahl der Dreiecke gebildet wird, *reduzierte Triangulierung* nennen. Später soll dann mittels Delaunay neu trianguliert werden. Zunächst betrachten wir das Problem, die Parameter der Ebene $\mathcal{E}$ entsprechend zu bestimmen. Wir fordern, dass für den Normalenvektor N der Ebene $\mathcal{E}$

$$\langle N, N_j \rangle > 0 \quad \forall j \in J_D, \tag{5.1}$$

gilt, wobei N_j der Normalenvektor des Dreiecks D_j ist. Anschaulich werden so durch die Bedingung (5.1) alle Dreiecke D_j mit $j \in J_D$ von der gleichen Seite auf die Ebene projiziert. Dies ist keine hinreichende Bedingung für die Bijektivität von $\Pi_{\mathcal{E}}|_{\triangle}$. Sie soll jedoch hier verwendet werden, um den Normalenvektor der Ebene $\mathcal{E}$ zu bestimmen.

Damit zusätzlich für die weitere Berechnung die Innenwinkel, die durch die Eckpunkte des projizierten Dreiecks bestimmt werden, nicht zu spitz sind, soll der Normalenvektor N weiter so gewählt werden, dass die Skalarprodukte in (5.1) möglichst groß werden.

Deshalb formulieren wir folgendes Optimierungsproblem, um den Normalenvektor der Ebene $\mathcal{E}$ zu bestimmen:

$$\begin{array}{ll} \max\limits_{N \in \mathbb{R}^3} & c \\ \text{s.t.} & N_j^T N \geq c \quad \forall j \in J_D \\ & \|N\|_2 \leq 1 \end{array}, \tag{5.2}$$

wobei gelten soll, dass $\|N_j\|_2 = 1$ für alle $j \in J_D$.

Dieses Optimierungsproblem ist ein second-order-cone-program [7, Abschnitt 4.4.2] und kann effizient mit dem Innere-Punkte-Verfahren gelöst werden.

Falls der aus der Lösung N^* des Problems (5.2) resultierende Zielfunktionswert c kleiner oder gleich null ist oder $\Pi_{\mathcal{E}}|_{\triangle}$ kein Homöomorphismus ist, wird der Knoten v für das Entfernen nicht gewählt. Erfüllt N^* die Bedingung (5.1) und sei w ein Punkt in V_v mit $w^T N^*$ minimal, dann beschreibt

$$\mathcal{E} = \left\{ p \in \mathbb{R}^3 : p^T N^* = w^T N^* \right\}$$

eine zweidimensionale Ebene, so dass alle weiteren Elemente aus V_v in dem Halbraum liegen, der durch die Ebene gebildet wird und in den der Normalenvektor N^* zeigt.

Auf diese Weise erhalten wir für jeden hier betrachteten Punkt aus V_i mit passendem Normalenvektor eine passende Ebene $\mathcal{E}$, so dass lokal die Dreiecke um den entsprechenden Knoten v möglichst „vorteilhaft“ projiziert werden können, und einen Homöomorphismus $\Phi : \triangle \mapsto \Pi_{\mathcal{E}}(\triangle)$ mit $\Phi \equiv \Pi_{\mathcal{E}}|_{\triangle}$.

5.1.2 Durchführung der Triangulierung

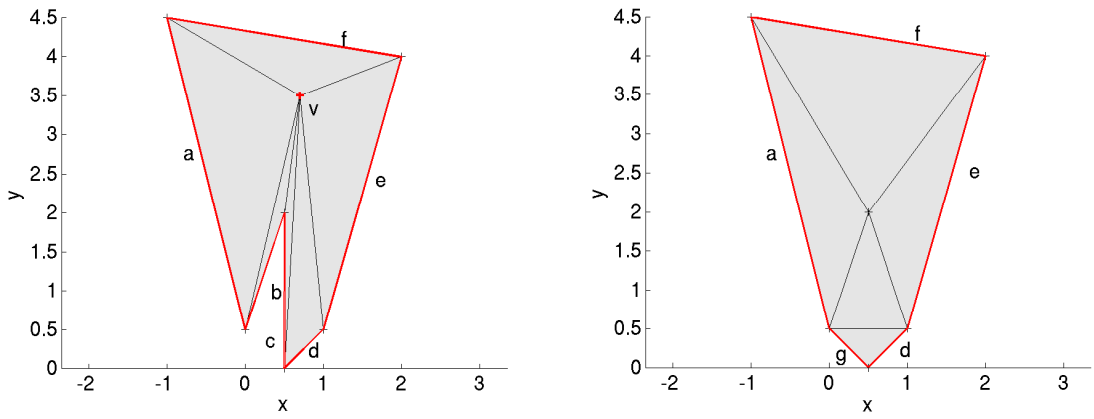
Nach der durchgeführten Projektion aus Abschnitt 5.1.1 erhalten wir nach Konstruktion Dreiecke D_j^p für alle $j \in J_D$, deren Eckpunkte auf der Ebene $\mathcal{E}$ liegen und die eine

reduzierte Triangulierung (V_v^p, E_v^p) bilden. Da die Knoten aus $V_v \setminus \{v\}$ alle auf einer Ebene liegen, kann nun zwischen diesen mittels Delaunay im Zweidimensionalen neu trianguliert werden. Seien D_j^d für alle $j \in \hat{J}_D^d$ die Dreiecke, die durch die Neutriangulierung bestimmt werden, wobei $\hat{J}_D^d$ mit $|\hat{J}_D^d| = s$ die entsprechende Indexmenge mit $s \in \mathbb{N}$ ist. Die Dreiecke D_j^d seien so beschrieben, dass alle Normalenvektoren in die gleiche Richtung wie der Normalenvektor der Ebene $\mathcal{E}$ zeigen. Des Weiteren seien $\triangle_j^d$ für $j \in \hat{J}_D^d$ die zugehörigen Dreiecksflächen. Wir nehmen an, dass es genau eine Teilmenge J_D^d von $\hat{J}_D^d$ gibt, so dass

$$\bigcup_{j \in J_D^d} \triangle_j^d \subset \Pi_{\mathcal{E}}(\triangle), \tag{5.3}$$

die Dreiecke D_j^d für alle $j \in J_D^d$ eine reduzierte Triangulierung (V^d, E^d) erzeugen und alle Kanten von E_v^p, die nicht inzident zu v sind, Elemente der Kantenmenge der reduzierten Triangulierung (V^d, E^d) sind.

Gibt es keine Teilmenge J_D^d mit den obigen Eigenschaften, so wird der Knoten v nicht für das Entfernen gewählt. In Abbildung 5.1 ist ein Beispiel dargestellt, in dem diese Indexmenge J_D^d mit den genannten Eigenschaften nicht existiert, da die Kante b im rechten Bild nicht mehr vorhanden ist.



(a) Vor Neutriangulierung

(b) Nach Neutriangulierung

Abbildung 5.1: Beispiel für die Nichtexistenz von J_D^d

Anschaulich sollen nun die Eckpunkte aller Dreiecke D_j^d für $j \in J_D^d$ mittels der Umkehrfunktion Φ^{-1} von Φ wieder in die Menge V_i abgebildet werden und die Dreiecke D_j^n für $j \in J_D^d$, die aus den abgebildeten Eckpunkten der Dreiecke D_j^d für $j \in J_D^p$ erzeugt werden, sollen zusammen mit den Dreiecken D_j für $j \notin J_D$ eine neue Triangulierung bilden. Die Dreiecke D_j^n sind dann für $D_j^d = (q^j, q_1^j, q_2^j)$ und $j \in J_D^d$ gegeben

durch

$$D_j^n = \left(\Phi^{-1}(q^j), \Phi^{-1}(q_1^j), \Phi^{-1}(q_2^j)\right).$$

Wir bezeichnen die zugehörigen Dreiecksflächen von D_j^n mit $\triangle_j^n$ für $j \in J_D^d$ und die entsprechende Vereinigung der Dreiecksflächen mit $\triangle^n$. Es ist offensichtlich, dass die Funktion $\Psi : \triangle^n \mapsto \Pi_{\mathcal{E}}(\triangle^n)$ mit $\Psi \equiv \Pi_{\mathcal{E}}|_{\triangle^n}$ wieder ein Homöomorphismus ist.

Wegen (5.3) ist die Funktion $\Phi^{-1}(\Psi(\cdot))$ über $\triangle^n$ wohl definiert. Des Weiteren gilt, dass man für jeden Homöomorphismus Γ aus der Definition 4.1.4 von $\mathcal{T}_i$ einen Homöomorphismus $\hat{\Gamma} : \triangle^n \cup \bigcup_{j \notin J_D} \triangle_j \mapsto \hat{\Gamma}\left(\triangle^n \cup \bigcup_{j \notin J_D} \triangle_j\right)$ mittels

$$\hat{\Gamma} \equiv \begin{cases} \Gamma(\Phi^{-1}(\Psi(\cdot))) & \text{über } \triangle^n \\ \Gamma & \text{sonst} \end{cases}$$

konstruieren kann, der die Menge $\triangle^n \cup \bigcup_{j \notin J_D} \triangle_j$ auf eine Ebene abbildet. Folglich ist der Graph, der durch die Dreiecke D_j^n für $j \in J_D^d$ und D_j für $j \notin J_D$ erzeugt wird, nach Konstruktion eine Triangulierung.

Hierbei sei ebenso bemerkt, dass durch das Weglassen des Punktes v eine neue Triangulierung entsteht, die jedoch im Allgemeinen weniger Details als die vorherige enthält.

5.1.3 Berechnung eines geeigneten Gütekriteriums

Wie schon am Anfang von Abschnitt 5.1 erwähnt, benötigen wir Kosten für jeden Knoten v, die die Veränderung der Triangulierung kennzeichnen, wenn der Knoten v aus $\mathcal{T}_i$ entfernt wird. Für diesen Zweck berechnen wir ein Differenzvolumen zwischen der Triangulierung $\mathcal{T}_i$ und der neuen Triangulierung.

Dazu betrachten wir zunächst das Dreieck D_j und das entsprechende Dreieck D_j^p für $j \in J_D$. Dann bilden wir die konvexe Hülle der Eckpunkte von D_j und D_j^p und bestimmen deren Volumen $\mathcal{V}_j$. Berechnet man nun das Volumen für alle $j \in J_D$, so erhält man ein Gesamtvolumen

$$\mathcal{V} = \sum_{j=1}^{r} \mathcal{V}_j.$$

Analog bestimmen wir dieses Volumen für alle D_j^d und deren entsprechende Dreiecke D_j^n für $j \in J_D^d$ und erhalten ein erneutes Gesamtvolumen $\hat{\mathcal{V}}$. Dann definieren wir den Wert $|\mathcal{V} - \hat{\mathcal{V}}|$ als Kosten für die Veränderung der Triangulierung, wenn der Knoten v entfernt wird.

Nun können für alle Knoten aus V_i Kosten berechnet und eine entsprechende Triangulierung bestimmt werden, die v nicht enthält. Folglich ist es mit dem Verfahren, welches am Anfang des Abschnitts 5.1 beschrieben wurde, möglich, eine Hierarchie von Modellen zu erstellen.

Um zusätzlich noch, wie in Abschnitt 4.3.3 angedeutet, zwischen den einzelnen Triangulierungen eine Verbindung zu erzeugen, wird ein Dreieck D_l^n für $l \in J_D^d$, das nun der Triangulierung $\mathcal{T}_{i+1}$ angehört, mit allen Dreiecken D_j für $j \in J_D$ in der Triangulierung $\mathcal{T}_i$ verlinkt, falls sie mindestens einen Eckpunkt mit D_j^n gemeinsam haben.

5.2 Grobregistrierung

In diesem Abschnitt wird beschrieben, wie die Hauptkomponenten vom Modell in die der Punktwolke überführt werden. Zunächst beschreiben wir die Hauptkomponentenanalyse im Allgemeinen, welche im Englischen mit principle component analysis bezeichnet und daher oft mit PCA abgekürzt wird.

5.2.1 PCA im Allgemeinen

Die Hauptkomponentenanalyse ist eine Transformation $H : \mathbb{R}^l \mapsto \mathbb{R}^{\hat{l}}$ mit $\hat{l} \leq l$, auch bekannt als Karhunen-Loeve-Transformation, die angewandt auf eine Menge von $k \geq 1$ Vektoren $\{a_1, \ldots, a_k\} \subset \mathbb{R}^l$, deren Mittelwert $1/k \sum_{j=1}^{k} a_j$ gleich null ist, eine Menge von Vektoren $\{b_1, \ldots, b_k\} \subset \mathbb{R}^{\hat{l}}$ liefert, deren geschätzte Kovarianzmatrix eine Diagonalmatrix Λ mit $\Lambda_{11} \geq \Lambda_{22} \geq \ldots \geq \Lambda_{\hat{l}\hat{l}}$ ist.

Sei $A \in \mathbb{R}^{l \times k}$ die Matrix mit $A = [a_1, \ldots, a_k]$. Dann ergibt sich die Transformation H aus $H(x) = [q_1, \ldots, q_{\hat{l}}]^T x$, wobei sich $[q_1, \ldots, q_l] =: Q$ orthogonal aus der Eigenwertzerlegung von der geschätzten Kovarianzmatrix $AA^T = Q^T \Lambda Q$ ergibt. Als Hauptkomponenten bezeichnet man dabei die Eigenvektoren $[q_1, \ldots, q_{\hat{l}}]$.

Wie schon aus der Transformation H erkennbar ist, kann man mit Hilfe der Hauptkomponentenanalyse eine Dimensionsreduktion multivariater Daten durchführen, was auch das Hauptanwendungsgebiet der Hauptkomponentenanalyse bildet. Wählt man $0 < \hat{l} < l$ und sind die Eigenwerte $\Lambda_{\hat{l}+1,\hat{l}+1}, \ldots, \Lambda_{ll}$ klein im Vergleich zu den restlichen Eigenwerten, so wird durch die Transformation der Informationsgehalt der Daten weitestgehend erhalten, da nur die Information verloren geht, die mit den Eigenvektoren der kleinsten Eigenwerte korrespondiert.

5.2.2 Anwendung der PCA auf das Problem

Das Augenmerk der Anwendung der Hauptkomponentenanalyse liegt in dieser Arbeit nicht auf der Dimensionsreduktion, sondern auf der Bestimmung der Hauptkomponenten der Punktwolke und des Modells, um jene aufeinander zu legen und damit die Punktwolke und das Modell gleich auszurichten. Wir bestimmen also eine Transformation $T_{MP} : \mathbb{R}^3 \mapsto \mathbb{R}^3$, die angewandt auf die Punkte $\bigcup_{i=1}^{m} \triangle_i$, eine Menge von Vektoren liefert, deren Hauptkomponenten denen der Punktwolke P entsprechen.

Dazu bestimmen wir zunächst mittels des in Abschnitt 5.2.1 beschriebenen Vorgehens die Hauptkomponenten der Punktwolke P, was die Verschiebung der Punktwolke P um den Mittelwert

$$t_p := \frac{1}{n} \sum_{i=1}^{n} p_i \tag{5.4}$$

voraussetzt. Wir erhalten eine orthogonale Matrix Q_P.

Da in den obigen Überlegungen die Hauptkomponentenanalyse nur für eine diskrete Punktmenge angewendet werden können, das Ziel aber ist, die Hauptkomponenten des Modells, also die Menge der Vektoren aus $\triangle := \bigcup_{i=1}^{m} \triangle_i$ zu bestimmen, verwenden wir die in [19] beschriebene Hauptkomponentenanalyse. Jene Analyse stellt eine Erweiterung der herkömmlichen Hauptkomponentenanalyse dar, da sie auch die Hauptkomponenten über kontinuierliche Gebiete bestimmen kann.

Zunächst berechnen wir den Mittelwert von $\triangle$ und anschließend die Hauptkomponenten, um so die orthogonale Matrix Q_M und den Translationsvektor t_M zu erhalten. Seien dafür $A^1 = [a_1^1, a_2^1, a_3^1], \ldots, A^m = [a_1^m, a_2^m, a_3^m] \subset \mathbb{R}^{3\times 3}$ Matrizen, deren Spalten die Dreiecksflächen $\triangle_1, \ldots, \triangle_m$ mittels

$$\triangle_i = \left\{ a : a = a_1^i + \mu_1 a_2^i + \mu_2 a_3^i, \mu_1 + \mu_2 \leq 1, \mu_1, \mu_2 \geq 0 \right\} \quad \forall i = 1, \ldots, m$$

erzeugen. Zusätzlich definieren wir die Menge

$$\mathbb{W} := \left\{ (\mu_1, \mu_2) \in \mathbb{R}_+^2 : \mu_1 + \mu_2 \leq 1 \right\}.$$

Dann bestimmen wir analog zu den Ausführungen in [19] den Mittelwert von $\triangle$ durch

$$\mathbb{E}(\triangle) = \frac{1}{V(\triangle)} \int_{a\in\triangle} a da,$$

wobei $V(\triangle) := \int_{a\in\triangle} 1 da$ das Volumen von $\triangle$ ist. Da die Funktionen $\phi_i : \mathbb{W} \mapsto \triangle_i$ mit $\phi_i(\mu) = a_1^i + [a_2^i, a_3^i]\mu$ bijektiv sind, folgt für

$$G_i := \begin{pmatrix} \langle a_2^i, a_2^i \rangle & \langle a_2^i, a_3^i \rangle \\ \langle a_3^i, a_2^i \rangle & \langle a_3^i, a_3^i \rangle \end{pmatrix}$$

symmetrisch und positiv definit [8, Abschnitt 3.3], dass

$$\begin{aligned} V(\triangle) &= \int_{a\in\triangle_1\cup\ldots\cup\triangle_m} 1 da \\ &= \int_{a\in\triangle_1} 1 da + \ldots + \int_{a\in\triangle_m} 1 da \\ &= \int_{\mu\in\mathbb{W}} \sqrt{\det G_1} d\mu + \ldots + \int_{\mu\in\mathbb{W}} \sqrt{\det G_m} d\mu \\ &= \left(\sqrt{\det G_1} + \ldots + \sqrt{\det G_m} \right) \cdot \int_0^1 \int_0^{1-\mu_1} d\mu_2 d\mu_1 \\ &= \frac{1}{2} \left(\sqrt{\det G_1} + \ldots + \sqrt{\det G_m} \right) \end{aligned}$$

und analog

$$\begin{aligned}\mathbb{E}(\triangle) &= \frac{1}{V(\triangle)} \int_{a\in\triangle} a da \\ &= \frac{1}{V(\triangle)} \sum_{i=1}^{m} \sqrt{\det G_i} \int_0^1 \int_0^{1-\mu_1} a_1^i + \mu_1 a_2^i + \mu_2 a_3^i d\mu_2 d\mu_1 \\ &= \frac{1}{V(\triangle)} \sum_{i=1}^{m} \sqrt{\det G_i} A^i \begin{pmatrix} \frac{1}{2} \\ \frac{1}{6} \\ \frac{1}{6} \end{pmatrix}\end{aligned} \tag{5.5}$$

ist.

Wir setzen nun den Translationsvektor $t_M := \mathbb{E}(\triangle)$ und verschieben alle Vektoren in $\triangle$ um den Vektor t_M, so dass dessen Mittelwert sich zu null ergibt. Danach erhalten wir $\widetilde{\triangle} := \{b - t : b \in \triangle\}$ und die entsprechenden Matrizen $\widetilde{A}_i = [\widetilde{a}_1^i, \widetilde{a}_2^i, \widetilde{a}_3^i]$ für alle $i = 1, \ldots, m$. Nun kann die Kovarianzmatrix mittels

$$\mathrm{Cov}\left(\widetilde{\triangle}\right) = \frac{1}{V\left(\widetilde{\triangle}\right)} \int_{a\in\widetilde{\triangle}} a a^T da$$

analog zu (5.5) ermittelt werden [19] und es ergibt sich für

$$\widetilde{G}_i := \begin{pmatrix} \langle \widetilde{a}_2^i, \widetilde{a}_2^i \rangle & \langle \widetilde{a}_2^i, \widetilde{a}_3^i \rangle \\ \langle \widetilde{a}_3^i, \widetilde{a}_2^i \rangle & \langle \widetilde{a}_3^i, \widetilde{a}_3^i \rangle \end{pmatrix},$$

dass

$$\mathrm{Cov}\left(\widetilde{\triangle}\right) = \frac{1}{V\left(\widetilde{\triangle}\right)} \sum_{i=1}^{m} \sqrt{\det \widetilde{G}_i} \widetilde{A}_i \Phi \widetilde{A}_i^T \quad \text{mit } \Phi = \begin{pmatrix} \frac{1}{2} & \frac{1}{6} & \frac{1}{6} \\ \frac{1}{6} & \frac{1}{12} & \frac{1}{24} \\ \frac{1}{6} & \frac{1}{24} & \frac{1}{12} \end{pmatrix} \tag{5.6}$$

ist.

Aus (5.6) wird mittels der Überlegungen aus Abschnitt 5.2.1 die orthogonale Matrix Q_M bestimmt, deren Spalten die Eigenvektoren von $\mathrm{Cov}\left(\widetilde{\triangle}\right)$ sind. Abschließend ergibt sich aus den obigen Betrachtungen die gesuchte Transformation T mit

$$T(a) = Q_P Q_M^T (a - t_M) + t_p. \tag{5.7}$$

5.3 Feinregistrierung

Die Aufgabe in diesem Kapitel ist es, die Optimierungsprobleme (4.3) und (4.4) aus Kapitel 4 zu lösen, wobei letzteres auf die Anwendung eines in Abschnitt 5.3.2 vorgestellten Satzes beruht. Im nächsten Abschnitt werden wir einige Lösungsverfahren vorstellen, mit denen das Problem (4.3) gelöst werden kann. Insbesondere wird dabei auf die Lösung des Problems mittels des Formens des Lagrange-Dualen, worauf wir später das so genannte Bündelverfahren anwenden, eingegangen.

5.3.1 Verfahren zur Lösung des Korrespondenzenproblems

Das in Abschnitt 4.3.2 vorgestellte Optimierungsproblem zur Bestimmung der Korrespondenzen ist ein Flussproblem und kann deshalb mit der dafür vorgesehenen Standardsoftware gelöst werden. Das wohl bekannteste Verfahren ist dabei das Simplex-Verfahren.

Das Simplex-Verfahren

Zur Lösung des Flussproblems minimaler Kosten eignet sich der Netzwerk-Simplex-Algorithmus besonders gut, siehe dazu [4, Abschnitt C.4]. Nun betrachten wir folgendes lineare Optimierungsproblem zu (4.3):

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad & e^T x_j = 1 \quad \forall j = 1, \ldots, n, \\ & Dx \leq u, \\ & x = \left(x_1^T, x_2^T, \ldots, x_n^T\right)^T, x_j \in [0,1]^m. \end{aligned} \tag{5.8}$$

Da es sich in (4.3) um ein zulässiges Flussproblem handelt, hat auch das Optimierungsproblem (5.8) eine Lösung. Da weiter die dem Flussproblem entsprechende Knoten-Kanten-Inzidenz-Matrix total unimodular ist und sowohl die Kapazitäten als auch die „Lieferanten“ und „Konsumenten“ ganzzahlig sind, hat es wegen $[0,1]^{m \cdot n}$ kompakt eine ganzzahlige Optimallösung [4, Abschnitt 11.12]. Des Weiteren ist jede zulässige Basislösung von (5.8) ganzzahlig, weshalb die mittels Simplex-Verfahren berechnete Optimallösung von (5.8) auch eine Optimallösung von (4.3) ist. Folglich lässt sich das Problem (4.3) mittels des Simplex-Verfahrens lösen.

Ein Softwarepaket, welches die Implementierung dieses Verfahrens beinhaltet, ist z. B. das ILOG CPLEX, das ursprünglich von Robert E. Bixby entwickelt und 1997 von ILOG erworben wurde [1]. Dieses Softwarepaket werden wir für numerische Tests in Kapitel 7 verwenden, um die Simplex-Methode mit anderen Verfahren zu vergleichen.

Ein weiteres Softwarepaket ist die MCF-Bibliothek (minimum-cost-flow), welche durch das Konrad-Zuse-Zentrum für Informationstechnik Berlin bereitgestellt wird. Diese Bibliothek ist eine Netzwerk-Simplex-Implementierung und dient der Lösung von Flussproblemen minimaler Kosten.

Bestimmung einer approximativen Lösung

Die bisher vorgestellten Solver lösen das Problem (4.3) exakt und liefern immer eine zulässige Lösung, falls eine existiert. Da es jedoch für den in Kapitel 4 beschriebenen Anwendungszweck ausreicht, nur eine approximative Lösung zu berechnen, ist die Idee, die Bedingung $Dx \leq u$ durch die Lagrange-Multiplikatoren in die Zielfunktion zu schreiben und das Lagrange-Duale mittels des Bündelverfahrens zu lösen.

Unter einer approximativen Lösung verstehen wir in diesem Sinne eine Lösung $\widetilde{x}$, deren Zielfunktionswert $c^T\widetilde{x} \in [c^Tx^* - \varepsilon, c^Tx^* + \varepsilon]$ ist und die Nebenbedingung $Dx \leq u$ nur approximativ erfüllt, d. h. $-\varepsilon_2 \leq D\widetilde{x} \leq u + \varepsilon_2$, wobei $\varepsilon, \varepsilon_2 \geq 0$ klein gewählt ist und x^* eine Optimallösung von (4.3) ist. Der Wert von ε bzw. ε_2 gibt somit die minimale Güte der Approximation an. Des Weiteren ist also eine approximative Lösung nicht notwendigerweise zulässig. Das Ziel ist, durch das Berechnen einer approximativen Lösung gegenüber der Berechnung der exakten Lösung weniger Operationen zu benötigen.

Es ist anzunehmen, dass die Berechnung einer approximativen Lösung für den in dieser Arbeit beschriebenen Anwendungszweck ausreichend ist, da z. B. die Kapazitätsgrenzen der Dreiecke nur geschätzte Werte sind und somit nicht zwangsläufig mit den tatsächlichen Werten übereinstimmen.

Lagrange-Relaxation

In diesem Abschnitt wird der Begriff der Lagrange-Relaxation beschrieben. Dazu betrachten wir das lineare Programm (primales Problem)

$$\begin{array}{ll} \min & c^Tx \\ \text{s.t.} & Ax \leq b \\ & x \geq 0 \end{array}$$

mit $x \in \mathbb{R}^s$, einem Kostenvektor $c \in \mathbb{R}^s$, $A \in \mathbb{R}^{t \times s}$ einer Matrix und $b \in \mathbb{R}^t$ einem Vektor für $s, t \in \mathbb{N}$.

Die so genannte *Lagrange-Funktion* ist dann definiert durch

$$L(\lambda, x) := c^Tx + (b - Ax)^T\lambda, \tag{5.9}$$

wobei $\lambda \in \mathbb{R}^t$ und $\lambda \leq 0$ die *Lagrange-Multiplikatoren* sind. Aus der Lagrange-Funktion in (5.9) ergibt sich die korrespondierende duale Funktion mittels

$$\begin{aligned} \theta(\lambda) :&= \inf_{x \geq 0} L(\lambda, x) \\ &= b^T\lambda + \inf_{x \geq 0}(c - A^T\lambda)^Tx \end{aligned} \tag{5.10}$$

und das zugehörige duale Problem aus

$$\max_{\lambda \leq 0} \theta(\lambda) \Leftrightarrow \begin{array}{ll} \max & b^T\lambda \\ \text{s.t.} & A^T\lambda \leq c \\ & \lambda \leq 0. \end{array}$$

Die Technik, die harte Nebenbedingung $Ax \leq b$ in die Zielfunktion in (5.10) zu schreiben und diese über $\lambda \leq 0$ zu maximieren, nennt man *Lagrange-Relaxation*. Das heißt, für ein bestimmtes λ wird die Lagrange-Funktion zunächst über x minimiert.

Dieses x ist nicht notwendigerweise zulässig, jedoch erhält die Lagrange-Funktion einen Bestrafungsterm, der abhängig von der „Stärke der Verletzung“ ist. Bei der Maximierung der Lagrange-Funktion über $\lambda \leq 0$ wird aber nun ein λ gewählt, so dass der Bestrafungsterm möglichst klein ist. Folglich werden bei dieser Technik „starke“ Verletzungen vermieden, jedoch muss die Nebenbedingung $Ax \leq b$ nicht mehr streng erfüllt sein.

Sei $\hat{x} \geq 0$ eine zulässige Lösung des primalen Problems und $\hat{\lambda} \leq 0$ eine zulässige Lösung des dualen Problems, dann gilt

$$c^T \hat{x} \geq (A^T \hat{\lambda})^T \hat{x} = \hat{\lambda}^T A \hat{x} \geq b^T \hat{\lambda}. \tag{5.11}$$

Aus (5.11) folgt schließlich, dass der Zielfunktionswert des dualen Problems eine untere Schranke für den Zielfunktionswert des primalen Problems angibt. Man nennt die Beziehung

$$\begin{array}{lll} \min & c^T x & \geq \quad \max \quad b^T \lambda \\ \text{s.t.} & Ax \leq b & \text{s.t.} \quad A^T \lambda \leq c \\ & x \geq 0 & \qquad\quad \lambda \leq 0 \end{array}$$

die *schwache Dualität*. Bei linearen Programmen gilt sogar die *starke Dualität* [4, Satz C.4]

$$\begin{array}{lll} \min & c^T x & = \quad \max \quad b^T \lambda \\ \text{s.t.} & Ax \leq b & \text{s.t.} \quad A^T \lambda \leq c \\ & x \geq 0 & \qquad\quad \lambda \leq 0. \end{array}$$

Nicht selten wird in der Praxis das duale Problem anstatt des primalen gelöst, da so schwierig handhabbare Ungleichungen zunächst weggelassen bzw. deren Verletzung als Bestrafung mit Hilfe der Lagrange-Multiplikatoren in der Zielfunktion berücksichtigt werden können und folglich ein leichter zu lösendes Problem resultiert. Oft ist es auch ausreichend, eine gute untere Schranke für den Zielfunktionswert des primalen Problems zu erhalten.

Bilden des Lagrange-Dualen

Nun wenden wir die Technik der Lagrange-Relaxation auf das Problem in (4.3) an. Sei dazu $\mathcal{X}$ die Menge mit

$$\mathcal{X} := \left\{ x = (x_1^T, \ldots, x_n^T)^T \in \{0,1\}^{m \cdot n} : e^T x_j = 1 \quad \forall j = 1, \ldots, m \right\}.$$

Dann gilt für $u \in \mathbb{N}^m$ mit $\sum_{i=1}^m u(i) \geq n$

$$\begin{aligned} \min_{x \in \mathcal{X}, Dx \leq u} c^T x &= \min_{x \in \mathcal{X}} \left\{ \sup_{y \leq 0} c^T x + (u - Dx)^T y \right\} \\ &\geq \max_{y \leq 0} u^T y + \left\{ \min_{x \in \mathcal{X}} (c - D^T y)^T x \right\} \\ &= \max_{y \leq 0} t(y) \end{aligned} \tag{5.12}$$

mit $t(y) := u^T y + \min_{x \in \mathcal{X}}(c - D^T y)^T x$.

Somit liefert das Lagrange-Duale eine untere Schranke für den Zielfunktionswert des primalen Problems. Die Funktion $t(y)$ ist konkav und die Menge der Subgradienten an y ist gegeben durch [15, Abschnitt 5.2 b)]

$$\operatorname{conv}\left\{s \in \mathbb{R}^m : s = u - Dx^*(y) \text{ und } x^* \in \mathcal{L}\right\},$$

wobei $\mathcal{L}$ die Menge der Lösungen des Problems $\min_{x \in \mathcal{X}}(c - D^T y)^T x$ ist.

Wie schon oben erwähnt, ist die Optimallösung des Problems (4.3) Optimallösung des linearen Optimierungsproblems (5.8) und ist endlich.

Da weiter $\mathcal{X}$ endlich ist, hat das duale Problem aus (5.12) nach dem Satz der starken Dualität [4, Satz C.4] eine endliche Optimallösung und der Zielfunktionswert stimmt mit dem der primalen Lösung überein. Folglich gilt sogar die Gleichheit in (5.12).

Weiter stellen wir fest, dass das innere Problem

$$\min_{x \in \mathcal{X}}(c - D^T y)^T x \tag{5.13}$$

für ein festes y leicht zu lösen ist, da die Nebenbedingung $Dx \leq u$ bei der Optimierung nicht mehr beachtet werden muss und folglich $(c - D^T y)^T x$ minimal wird, wenn jenes Element aus x_i für alle i gleich eins gesetzt wird, bei dem der entsprechende Eintrag von $c - D^T y$ am niedrigsten ist. Hierbei wird deutlich, dass die Lösung nicht notwendigerweise eindeutig ist, da es durchaus mehrere Elemente in x_i geben kann, die dieselben entsprechenden Werte in $(c - D^T y)_i$ haben. Um die Lösung eindeutig festzulegen, nehmen wir immer das erste Element aus x_i dessen entsprechender Wert in $(c - D^T y)_i$ am niedrigsten ist.

Da wir nun ein Element aus der Menge $\mathcal{L}$ bestimmen und folglich $t(y)$ an der Stelle y auswerten und den entsprechenden Subgradienten ermitteln können, lässt sich das Bündelverfahren auf das verbleibende Problem

$$\max_{y \leq 0} t(y) \tag{5.14}$$

anwenden.

Bündelverfahren

Das Bündelverfahren ist ein iteratives Verfahren zur Minimierung nicht glatter konvexer Funktionen. Dabei ist die zu minimierende Funktion mittels eines Orakels gegeben, welches zu einem Argument der Funktion den entsprechenden Funktionswert und einen Subgradienten aus dem Subdifferential liefert.

In jeder Iteration dieses Verfahrens wird das Orakel für einen bestimmten Punkt befragt. Anschließend wird mit Hilfe der aggregierten Subgradienten ein lokales Schnittebenen-Modell von dieser Funktion gebildet. Nun wird dieses Modell minimiert, welches zusätzlich durch einen Regularisierungsterm augmentiert wurde, und die Lösung

liefert einen neuen Punkt als Kandidaten. Zeigt die Funktionsauswertung im Kandidaten keinen guten Fortschritt, wird das Modell verbessert und ein neuer Kandidat bestimmt (ein Nullschritt), sonst wird der Kandidat das neue Zentrum des Modells (ein Abstiegsschritt). Man kann zeigen, dass die Folge der so generierten Zentrumspunkte gegen einen Minimierer der Funktion konvergiert, wenn es solche gibt. Für weitere Erläuterungen verweisen wir auf [15, 16, 24, 14].

Die Software, die wir für unsere Testzwecke in Kapitel 7 nutzen, ist eine C++-Bibliothek mit dem Namen *ConicBundle* von Herrn Prof. Dr. Helmberg, TU Chemnitz und unterliegt der GNU-Lizenz.

Eine Heuristik zur Berechnung einer ganzzahligen Lösung

Da das Bündelverfahren ein iteratives Verfahren ist, liefert es eine Näherungslösung y_N^* von (5.14) und ebenso eine Näherungslösung x_N^* des primalen Problems (5.8), die mit dem aggregierten Subgradienten korrespondiert, jedoch im Allgemeinen nicht ganzzahlig ist. Diese Näherungslösung liefert im Allgemeinen eine bessere Approximation zu der eigentlichen Optimallösung von (4.3) als eine ganzzahlige Lösung von (5.13) für y_N^*. Deshalb stellen wir hier einen Algorithmus für die Berechnung einer ganzzahligen Näherungslösung (muss nicht zwangsläufig zulässig sein) aus einer nicht ganzzahligen Näherungslösung vor.

Für die mittels Bündelverfahren berechnete Näherungslösung x_N^* gilt

$$\sum_{j=1}^{m} (x_N^*)_i(j) = 1$$

für alle $i = 1, \ldots, n$. Folglich können wir den Wert $(x_N^*)_i(j)$ als die Wahrscheinlichkeit ansehen, dass Punkt i dem Dreieck j zugehörig ist. Da die Näherungslösung x_N^* die gleiche Dimension wie der Kostenvektor c hat, könnte man $-(x_N^*)$ wieder als Kostenvektor betrachten und das Flussproblem mit diesen Kosten erneut lösen. Jedoch ließe sich das erneute Flussproblem nicht einfacher lösen als das vorhergehende. Man könnte allerdings bei dieser Vorgehensweise vorher Kanten löschen, die hohe Kosten haben, d. h., bei denen der entsprechende Wert von x_N^* niedrig ist, um so die Schwere des Problems zu verringern. Dieses Vorgehen ist jedoch sehr kostenaufwändig und garantiert keine Optimalität, weshalb wir eine Heuristik zur Bestimmung einer ganzzahligen Lösung vorschlagen, die zwar auch kein Optimalitätskriterium erfüllt, aber schnell zu berechnen ist.

Seien dazu $Z_1, \ldots, Z_m$ Mengen mit Z_j die Menge der Punktindices, deren entsprechende Punkte dem Dreieck j zugewiesen werden, und $V_1, \ldots, V_m$ Mengen mit V_j die Menge der Punktindices, deren entsprechende Punkte dem Dreieck j noch zugewiesen werden können. Zu Beginn seien die Mengen $Z_1, \ldots, Z_m$ leer und $V_j := \{1, \ldots, n\}$ für alle j.

Die hier vorgeschlagene Heuristik besteht aus zwei Phasen. Um anfänglich die Kombinationsanzahl der Korrespondenzen zu reduzieren, werden in der ersten Phase

Korrespondenzpaare ausgewählt, die eine hohe Wahrscheinlichkeit haben. Des Weiteren entfernen wir alle Indices i aus V_j $\forall j = 1, \ldots, m$, deren entsprechender Wert $(x_N^*)_i(j)$ null ist, da diese Korrespondenzen unwahrscheinlich sind.

Nun werden maximal $u(j)$ Punkteindices i aus V_j $\forall j = 1, \ldots, m$, deren entsprechende Werte $(x_N^*)_i(j)$ am größten und einen Wert von größer 0,5 haben, zu der Menge Z_j hinzugefügt und aus V_j entfernt. Wir beschränken die Anzahl der hinzugefügten Punktindices, damit die spätere Lösung zulässig bleibt. Der Wert von $(x_N^*)_i(j)$ muss größer als 0,5 sein, damit ein und derselbe Punkt auch nur genau einem Dreieck zugewiesen wird. Wir definieren also für $G_j = \{l \in V_j : (x_N^*)_l(j) \geq 0{,}5\}$ und $j = 1, \ldots, m$ die Menge

$$Q_j := G_j \cap \{l \in V_j : \nexists V \subset V_j, |V| = u(j), \text{ so dass } (x_N^*)_k(j) > (x_N^*)_l(j) \ \forall k \in V\}.$$

Dann ergeben sich die Mengen $Z_1, \ldots, Z_m$ nach der ersten Phase durch

$$Z_j \leftarrow\!\shortmid Z_j \cup Q_j$$

für alle $j = 1, \ldots, m$ und $V_1, \ldots, V_m$ mittels

$$V_j \leftarrow\!\shortmid V_j \setminus \left(\bigcup_{l=1}^{m} Z_l \right) \tag{5.15}$$

für alle $j = 1, \ldots, m$.

In der zweiten Phase wird der Punktindex $\hat{\imath}$ in V_j gesucht, dessen entsprechender Wert $(x_N^*)_{\hat{\imath}}(j)$ zwar groß ist, jedoch auch einen relativ großen Abstand zum zweitgrößten Wert in $\{(x_N^*)_i(j) : i \in V_j\}$ hat. Der Grund für dieses Vorgehen ist, dass wir es für einen solchen Punktindex in V_j als sehr wahrscheinlich betrachten, dass dieser Punkt zu Dreieck j zugehörig ist. Für die Funktion $\phi : \bigcup_{j=1}^{m} V_j \times \{1, \ldots, m\} \mapsto \mathbb{R}$ mit

$$\phi(l, j) := \begin{cases} 1 & \text{für } V_j \setminus \{l\} = \emptyset \\ \min\limits_{k \in V_j \setminus \{l\}} \frac{(x_N^*)_l(j) - (x_N^*)_k(j)}{(x_N^*)_l(j)} & \text{sonst} \end{cases}$$

lösen wir deshalb

$$\begin{aligned} \max_{l \in V_j \neq \emptyset, j} \quad & \phi(l, j) \\ \text{s.t.} \quad & (x_N^*)_l(j) \geq 0{,}7 \cdot K, \end{aligned} \tag{5.16}$$

wobei $K := \max\limits_{s \in V_j \neq \emptyset, j} (x_N^*)_s(j)$ eine Konstante ist. Das Problem (5.16) hat immer eine Lösung, falls die Vereinigung der Mengen V_j nicht leer ist, da V_j endliche Mengen sind. Im letzteren Fall wurden schon alle Punkte auf die Mengen $Z_1, \ldots, Z_m$ verteilt und man erhält folglich eine Lösung von (4.3). Die Lösung des Problems (5.16) ist allerdings nicht notwendigerweise eindeutig, weshalb wir aus der Lösungsmenge von (5.16) zunächst diejenigen Paare (l, j) betrachten, deren zweite Komponente minimal

ist, und von denen das Paar (l, j) ermitteln, dessen erste Komponente minimal ist. Wir bezeichnen dieses Paar mit $\left(\hat{l}, \hat{\jmath}\right)$. Die Menge $Z_{\hat{\jmath}}$ ergibt sich dann aus

$$Z_{\hat{\jmath}} := Z_{\hat{\jmath}} \cup \left\{\hat{l}\right\}$$

und die Mengen $V_1, \ldots, V_m$ werden wieder mittels (5.15) neu bestimmt. Gilt für $Z_{\hat{\jmath}}$, dass $|Z_{\hat{\jmath}}| = u\left(\hat{\jmath}\right)$ ist, so darf kein weiteres Element der Menge $Z_{\hat{\jmath}}$ hinzugefügt werden. Folglich kann die Menge $V_{\hat{\jmath}}$ für das erneute Starten der zweiten Phase nicht mehr einbezogen werden. Dies bedeutet letztendlich, dass sich die Wahrscheinlichkeiten in $\bigcup_{j \neq \hat{\jmath}} V_j$ für alle Punktindices in $V_{\hat{\jmath}}$ anteilig erhöhen. Deshalb verändern wir für das weitere Vorgehen die Näherungslösung x_N^* durch

$$(x_N^*)_s(j) := \left(1 + \frac{(x_N^*)_s(\hat{\jmath})}{1 - (x_N^*)_s(\hat{\jmath})}\right)(x_N^*)_s(j) \quad \forall j \neq \hat{\jmath} \text{ und } \forall s \in V_j \cap V_{\hat{\jmath}}.$$

Für alle $s \in V_{\hat{\jmath}}$ mit $s \notin \bigcup_{j \neq \hat{\jmath}} V_j$ setzen wir $Z_{\hat{\jmath}} := Z_{\hat{\jmath}} \cup \{s\}$. Dieser Schritt führt zu keiner zulässigen Lösung mehr, sondern nur zu einer approximativen Lösung in (4.3), da nun die Ungleichheitsnebenbedingung $Dx \leq u$ nicht mehr erfüllt ist. Wie schon oben erwähnt, reicht es jedoch, eine approximative Lösung für (4.3) zu berechnen, weshalb wir an dieser Stelle s dem Dreieck $\hat{\jmath}$ zuweisen und nicht z. B. seinen Nachbarn. Nach diesem Schritt wird die Menge $V_{\hat{\jmath}}$ auf die leere Menge gesetzt.

Nun wird die zweite Phase solange neu gestartet, bis $\bigcup_{j=1}^m V_j = \emptyset$ ist. Da in jeder Iteration ein Element aus $\bigcup_{j=1}^m V_j = \emptyset$ entfernt wird und nur endlich viele Elemente in $\bigcup_{j=1}^m V_j$ sind, wird diese Iteration nach endlich vielen Schritten abgebrochen. Nun lässt sich aus den Mengen $Z_1, \ldots, Z_m$ eine ganzzahlige approximative Lösung für (4.3) konstruieren, die die Bestimmung von Koorespondenzpaaren erlaubt, jedoch nicht notwendigerweise zulässig ist.

Techniken zur Verbesserung der Konvergenz

In diesem Kapitel wird eine Idee vorgestellt, um die Konvergenz des Bündelverfahrens zu verbessern. In (5.14) muss die Funktion $t(y)$ noch unter der Nebenbedingung $y \leq 0$ minimiert werden. Um diese Nebenbedingung zu erfüllen, sind zusätzliche Prüfschritte im Bündelverfahren nötig. Um diese zu vermeiden, bedarf es der Maximierung über den gesamten Raum, weshalb wir die Ungleichungsnebenbedingung im primalen Problem durch eine Gleichheitsnebenbedingung ersetzen und entsprechende konvexe Bestrafungsfunktionen $f_i : \mathbb{R}^m \mapsto \mathbb{R}$ zu der Zielfunktion addieren. Wir

erhalten

$$\begin{aligned}
\min_{\substack{x\in\mathcal{X},v\\ \text{s.t.}\quad Dx=v}} c^Tx+\textstyle\sum_{i=1}^m f_i(v_i) &= \min_{x\in\mathcal{X},v} c^Tx+\textstyle\sum_{i=1}^m f_i(v_i)+\max_{y\in\mathbb{R}^m}(v-Dx)^Ty\\
&\geq \max_{y\in\mathbb{R}^m}\min_{x\in\mathcal{X},v}(c-D^Ty)^Tx+\textstyle\sum_{i=1}^m f_i(v_i)+v^Ty\\
&= \max_{y\in\mathbb{R}^m}\hat{t}(y),
\end{aligned}$$

wobei $\hat{t}(y) := \min_{x\in\mathcal{X},v}(c-D^Ty)^Tx+\sum_i f_i(v_i)+v^Ty$ ist. Sei $s\in\mathbb{N}$, $\rho,\beta^i\in\mathbb{R}$ mit $\rho,\beta^i>0$ und $\eta_0^i,\ldots,\eta_s^i$ Stützstellen mit $\eta_0^i := u(i)$ und $\eta_j^i := \eta_{j-1}^i+\beta^i$ für alle $j=1,\ldots,s$. Dann definieren wir die Bestrafungsfunktionen f_i durch

$$f_i(v_i) := \begin{cases} 0 & \text{für } v_i\leq u(i)\\ 2\rho(\eta_j^i-u(i))\cdot(v_i-\eta_{j-1}^i)+f_i(\eta_{j-1}^i) & \text{für } v_i\in(\eta_{j-1}^i,\eta_j^i]\\ \infty & \text{sonst}\end{cases}$$

für $\forall i=1,\ldots,m$. Diese sind nach Konstruktion wie auch in Abbildung 5.2 dargestellt, stückweise linear und konvex. Die Konstanten β^i,ρ beeinflussen dabei die Höhe der Bestrafung. Wir werden diese für die in Kapitel 7.2 durchgeführten Tests empirisch bestimmen.

Es ist offensichtlich, dass

$$\begin{array}{lll}\min\limits_{x\in\mathcal{X}} & c^Tx & \geq \min\limits_{x\in\mathcal{X},v} \quad c^Tx+\sum_i f_i(v_i) \quad \geq \max\limits_{y\in\mathbb{R}^m} \quad \hat{t}(y)\\ \text{s.t.} & Dx\leq u & \text{s.t.} \quad Dx=v\end{array} \tag{5.17}$$

und für $\rho\to\infty$ die Ungleichungen in (5.17) zu Gleichungen werden. Folglich erhalten wir durch das Lösen des Problems

$$\max_{y\in\mathbb{R}^m}\hat{t}(y) \tag{5.18}$$

für ein $\rho<\infty$ eine untere Schranke für den Zielfunktionswert und gleichzeitig eine approximative Lösung des primalen Problems.

Das Bündelverfahren kann nun auf das Problem (5.18) analog zu oben angewandt werden, wobei sich die Menge der Subgradienten von $\hat{t}(y)$ an der Stelle y durch

$$\operatorname{conv}\left\{s\in\mathbb{R}^m : s=v^*(y)-Dx^*(y) \text{ und } (x^*,v^*)\in\hat{\mathcal{L}}\right\} \tag{5.19}$$

mit $\hat{\mathcal{L}}$ die Menge der Lösungen des inneren Problems

$$\min_{x\in\mathcal{X},v}(c-D^Ty)^Tx+\sum_i f_i(v_i)+v^Ty$$

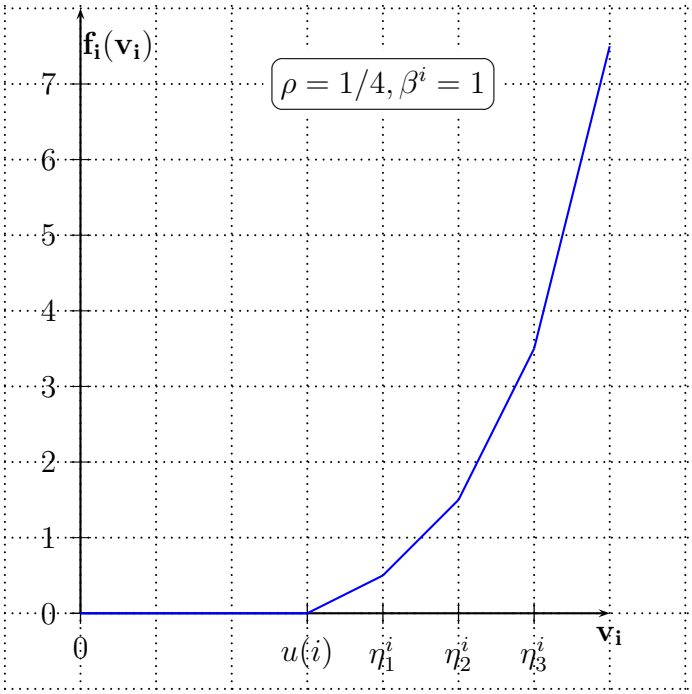


Abbildung 5.2: Grafische Darstellung einer Bestrafungsfunktion f_i

und die Auswertung von $t(y)$ sich an der Stelle y durch die obige Vorgehensweise und die zusätzliche Minimierung von $f_i(v_i) + v_i \cdot y_i$ ergibt.

Wie „gut“ die durch die Anwendung des Bündelverfahrens auf (5.18) ermittelte approximative Lösung die Lösung des eigentlichen Problems in (4.3) approximiert, werden wir durch numerische Tests in Kapitel 7 veranschaulichen.

Damit wurde ein weiteres Lösungsverfahren beschrieben, um das mathematische Modell in Abschnitt 4.3.2 zu lösen. Durch die Lösung des mathematischen Modells ist es nun möglich, die Korrespondenzpaare in jeder Iteration für die Minimierung der Summe in (4.4) zu berechnen.

5.3.2 Transformationsoptimierung

Um die Funktion in (4.4) zu minimieren, bedarf es der direkten Anwendung des folgenden Satzes [6, 5].

Satz 5.3.1. *Seien* $A = \{a_1, \ldots, a_n\}$ *und* $B = \{b_1, \ldots, b_n\}$ *Punktwolken und* $w_1, \ldots, w_n$ *Gewichte mit* $w_i \in (0, 1]$. *Sei weiter* $\mathcal{P}$ *die Menge der orthogonalen Matrizen in* $\mathbb{R}^{3\times3}$ *und* $h : \mathcal{P} \times \mathbb{R}^3 \mapsto \mathbb{R}$ *eine Funktion mit*

$$h(R, t) := \sum_{i=1}^{n} w_i \|Ra_i - b_i - t\|_2^2.$$

Dann hat die Funktion h an $R = U^T V$ und $t = R\overline{a} - \overline{b}$ mit $\overline{a} = \frac{1}{\sum_{j=1}^n w_j} \sum_{i=1}^n w_i a_i$, $\overline{b} = \frac{1}{\sum_{j=1}^n w_j} \sum_{i=1}^n w_i b_i$ ein globales Minimum, wobei U, V die aus der Singulärwertzerlegung von $N := \sum_{i=1}^n w_i (a_i - \overline{a})(b_i - \overline{b})^T = V^T \Sigma U$ resultierenden orthogonalen Matrizen sind.

Beweis. Definiere $a'_i := a_i - \overline{a}$ und $b'_i := b_i - \overline{b}$ für alle $i = 1, \ldots, n$. Dann ist

$$\begin{aligned} h(R,t) &= \sum_{i=1}^n w_i \|Ra_i - b_i - t\|_2^2 \\ &= \sum_{i=1}^n w_i \|Ra'_i - b'_i + R\overline{a} - \overline{b} - t\|_2^2 \\ &= \sum_{i=1}^n w_i \|Ra'_i - b'_i\|_2^2 + \underbrace{\sum_{i=1}^n 2w_i \left\langle Ra'_i - b'_i, R\overline{a} - \overline{b} - t \right\rangle}_{=0} + \sum_{i=1}^n w_i \|R\overline{a} - \overline{b} - t\|_2^2, \end{aligned} \tag{5.20}$$

da $\sum_{i=1}^n w_i (Ra'_i - b'_i) = 0$ nach Konstruktion von a'_i, b'_i. Aus (5.20) folgt, dass

$$\min_{R,t} h(R,t) = \min_R g(R) := \sum_{i=1}^n w_i \|Ra'_i - b'_i\|_2^2$$

und die Funktion $h(R,t)$ für $t = \overline{R}\overline{a} - \overline{b}$ bzgl. der zweiten Komponente minimal wird. Wir betrachten nun

$$\begin{aligned} g(R) &= \sum_{i=1}^n w_i \|Ra'_i - b'_i\|_2^2 \\ &= \sum_{i=1}^n w_i \|Ra'_i\|_2^2 - 2\sum_{i=1}^n w_i (Ra'_i)^T b'_i + \sum_{i=1}^n w_i \|b'_i\|_2^2 \\ &= 2\sum_{i=1}^n w_i (Ra'_i)^T b'_i + c \\ &= -2\,\mathrm{trace}\,(RN) + c, \end{aligned} \tag{5.21}$$

wobei $c = \sum_{i=1}^n w_i (\|a'_i\|_2^2 + \|b'_i\|_2^2)$ ist. Dann gilt

$$\operatorname*{argmax}_R \left(\mathrm{trace}\,(RN)\right) = \operatorname*{argmax}_R \left(\mathrm{trace}\left(URV^T\Sigma\right)\right) \quad = U^T V. \tag{5.22}$$

Gleichungen (5.21) und (5.22) implizieren

$$\operatorname*{argmin}_R g(R) = U^T V$$

und es folgt die Behauptung. □

Die Anwendung dieses Satzes liefert die beste orthogonale Matrix und den besten Translationsvektor, die nun über die Definition 4.1.6 der Transformation für die nächste Iteration auf das Modell angewandt werden können.

6 Algorithmus

Die folgende Vorgehensweise entspricht dem aus dieser Arbeit resultierenden Algorithmus zur Berechnung einer Transformation, die angewandt auf ein trianguliertes Oberflächenmodell eines Objektes die Modellkoordinaten in die einer Objektvermessung überführt, so dass Modell und die Vermessung aneinander angeglichen sind.

Algorithmus 1. Sei *max-iter* die maximale Anzahl an Iterationen für die Feinregistrierung. Dann lautet der in dieser Arbeit vorgeschlagene Algorithmus, wie folgt:

Schritt 0**:** Bilde eine Hierarchie von Modellen mittels des in Abschnitt 5.1 beschriebenen Algorithmus und verringere gegebenfalls die Punktanzahl der Messpunkte durch eine äquidistante Ausdünnung.

Schritt 1**:** Berechne den Schwerpunkt vom feinsten Modell t_M und der Scandaten t_P mittels (5.5) bzw. (5.4).

Schritt 2**:** Bestimme die Hauptkomponenten vom feinsten Modell Q_M und der Scandaten Q_P durch die Eigenwertzerlegung von (5.6) bzw. mittels der Anweisungen in Abschnitt 5.2.1.

Schritt 3**:** Berechne die Transformation T mit Hilfe von (5.7) und wende diese auf das gröbste Modell an. Setze $iter := 0$ und setze das aktuelle Modell auf das gröbste.

Schritt 4**:** Bestimme den Kostenvektor c für das Flussproblem in (4.3) für das aktuelle Modell und die ausgedünnten Scandaten.

Schritt 5**:** Löse das entsprechende Problem (5.18) mittels des Bündelverfahrens und berechne eine ganzzahlige Näherungslösung x_N^* für (4.3) durch die Anwendung der in Abschnitt 5.3.1 beschriebenen Heuristik. Bestimme die entsprechenden Korrespondenzpaare aus dieser Näherungslösung x_N^*.

Schritt 6**:** Berechne die beste Transformation $\hat{T}$ durch die Anwendung des Satzes 5.3.1.

Schritt 7**:** Setze $iter \hookleftarrow iter+1$ und $T \hookleftarrow \hat{T}\,(T(\cdot))$. Gilt für die zu $\hat{T}$ zugehörige orthogonale Matrix $\hat{R}$ und den Translationsvektor $\hat{t}$, dass $iter > max\text{-}iter$, so verlasse den Algorithmus. Die gesuchte Transformation ist T.

Schritt 8: Ist kein feineres Modell vorhanden: Wende die Transformation $\hat{T}$ auf das aktuelle Modell an und gehe zu Schritt 4. Sonst: Setze das aktuelle Modell auf das nachfolgend feinere Modell und wende die Transformation T auf das Modell an. Gehe zu Schritt 4.

7 Numerische Tests

Um das Verhalten des in Kapitel 6 beschriebenen Algorithmus zu untersuchen, werden in diesem Kapitel entsprechende numerische Tests durchgeführt. Jedoch werden sich die Tests im Wesentlichen auf die Feinregistrierung konzentrieren, da das Prinzip der Hauptkomponentenanalyse wohlverstanden ist.

Die Implemetierung der Grobregistrierung bzw. Feinregistrierung wurde in MATLAB 2007a bzw. C++ vorgenommen. Für alle hier aufgeführten Tests wurde ein 2-Kernprozessor Genuine Intel der TU Chemnitz mit 3,2 GHz und einem Arbeitsspeicher von einem GigaByte verwendet. Als Betriebsystem wurde dabei SuSE 10.3 Linux mit der Kernelversion 2.6.22.12 eingesetzt.

Im nächsten Abschnitt werden wir anfänglich die verwendeten Testbeispiele beschreiben und im darauffolgenden Abschnitt die noch nicht bestimmten Parameter des Algorithmus 1 festlegen. In den darauffolgenden Abschnitten wird der Algorithmus mit anderen Verfahren verglichen bevor wir im Abschnitt 7.6 unsere Schlussfolgerungen ziehen.

7.1 Testbeispiele

Im Grunde verwenden wir für die Tests in den nächsten Abschnitten drei Testbeispiele, d. h.drei Modelle mit den jeweils entsprechenden Scandaten unterschiedlicher Größe, die einerseits am realen Objekt aufgenommen aber auch mit Hilfe des Modells generiert wurden.

7.1.1 Testbeispiel 1

Das erste Testbeispiel ist unser Zielobjekt, welches in Abschnitt 2.2 beschrieben wurde. Für dieses Objekt liegt ein Modell vor, das aus 2800 Dreiecken besteht und dessen Ausmaße $5,6$ in der Breite, $5,7$ in der Länge und $1,3$ in der Höhe sind. Wir bezeichnen dieses Modell mit *Mreal* in nachfolgenden Versuchen.

Zusätzlich sind zu diesem real aufgenommene Scandaten vorhanden, die nach vier verschiedenen äquidistanten Ausdünnungen aus 1874, 3277, 5513 bzw. 7548 Punkten bestehen. Die Anzahl an Punkten ist teilweise ungerade, da erst die Ausdünnung der Punkte vorgenommen und danach die Qualitymap (Abschnitt 2.4) auf die ausgedünnten Scandaten angewandt wird.

Zusätzlich zu diesen aufgenommenen Messdaten generieren wir Scandaten zu dem Modell, so dass diese Daten das komplette Modell abbilden. Dazu wird in diesem

Falle ein äquidistantes Gitter auf der x-y Ebene erstellt, dessen Ausmaße größer sind als die des auf der x-y Ebene projizierten Modells. Der z-Wert ergibt sich nun durch den Schnittpunkt des Modells mit der Geraden, die durch den Punkt $(x, y, 0)^T$ und den Vektor $e_3 = (0, 0, 1)^T$ gebildet wird. Falls dieser Schnittpunkt nicht existiert, so gilt der Punkt $(x, y, 0)$ als Ausreißer und wird in die Berechnung nicht einbezogen. Außerdem gilt, dass der Schnittpunkt in diesem Testbeispiel immer eindeutig bestimmbar ist, falls dieser existiert. Nach diesem Vorgehen erhalten wir Scandaten mit den folgenden Größen: 629, 1400, 1735, 2508 und 3921.

Um die Hierarchie an Modellen zu bilden, wird das in Abschnitt 5.1 beschriebene Verfahren angewandt und wir erhalten fünf Stufen an Modellen, wobei die jeweilige Anzahl der Dreiecke in Tabelle 7.1 dargestellt ist. Das Symbol „#“ steht dabei für die Anzahl.

Stufe der Hierarchie	1	2	3	4	5
# Dreiecke Mreal	128	289	646	1531	2800

Tabelle 7.1: Dreiecksanzahl der Hierarchiestufen des Modells Mreal

7.1.2 Testbeispiel 2

Für dieses Beispiel wird ein Modell generiert. Dazu definieren wir ein äquidistantes Gitter, dessen Werte im Intervall $[-1; 1] \times [-0{,}5; 0{,}5]$ liegen. Danach wird mittels Delaunay im Zweidimensionalen zwischen den Gitterpunkten trianguliert. Sei die Reihenfolge der Eckpunkte der Dreiecke so gewählt, dass wir eine Triangulierung im Zweidimensionalen erhalten. Um nun eine Triangulierung im Dreidimensionalen zu formen, definieren wir zunächst die Funktion $h : D \mapsto \mathbb{R}$ mit

$$h(x, y) = 0{,}6\sqrt{1 - x^2 - (2y)^2} - 0{,}15e^{-20x^2} \cdot e^{-20y^2}$$

und $D := \{(x, y) \in \mathbb{R}^2 : 1 - x^2 - (2y)^2 \geq 0\}$ den Definitionsbereich von h. Diese Funktion ist eine Zusammensetzung aus einer in y- und z-Richtung gestauchten Halbkugel und der Gauß-Funktion. Sie erzeugt den Graph, der in Abbildung 7.1 zu sehen ist.

Anschließend wird an allen Eckpunkten, die im Definitionsbereich D liegen und mindestens ein Eckpunkt einer Dreiecksfläche sind, die sich komplett im Definitionsbereich befindet, der Funktionswert der Funktion h berechnet. Dieser Wert gibt den Wert für die dritte Dimension aller dieser Eckpunkte. Infolgedessen definieren die Eckpunkte mit den entsprechenden Kanten nach Definition 4.1.4 eine Triangulierung im Dreidimensionalen. Definieren wir zusätzlich noch eine Gewichtungsfunktion, die alle Dreiecke auf den Wert 1 abbildet, so erhält man ein Modell.

Ausgehend von zwei äquidistanten Gittern mit $(-1 + 0{,}1i, -0{,}5 + 0{,}05j)$ für alle $i, j = 0, \ldots, 20$ bzw. $(-1 + 0,05i, -0{,}5 + 0{,}025j)$ für $i, j = 1, \ldots, 40$ erstellen wir so zwei Modelle, die aus 674 bzw. 2624 Dreiecken bestehen. Dabei bezeichnen wir das

Modell mit der höheren Anzahl an Dreiecken als *Modell* 1 und das jeweils andere als *Modell* 2.

Wendet man nun wieder den Algorithmus zur Bestimmung einer Hierarchie von Dreiecken wie in Abschnitt 7.1.1 an, so erhält man die in Tabelle 7.2 gezeigte Dreiecksanzahl pro Stufe, wobei die Abbildung 7.1 dies für das Modell 1 illustriert.

Stufe der Hierarchie	1	2	3	4	5
# Dreiecke Modell 1	128	289	646	1531	2800
# Dreiecke Modell 2	110	272	568	674	–

Tabelle 7.2: Dreiecksanzahl der Hierarchiestufen der Modelle 1 und 2

Analog zu Abschnitt 7.1.1 werden für beide Modelle ebenso Scandaten unterschiedlicher Größe generiert. Für das Modell 1 erzeugen wir dabei Scandaten mit 89, 347, 782, 1349, 2037 und 2172 Punkten und für das Modell 2 Scandaten mit 187, 755, 1696, 2961 und 4720 Punkten.

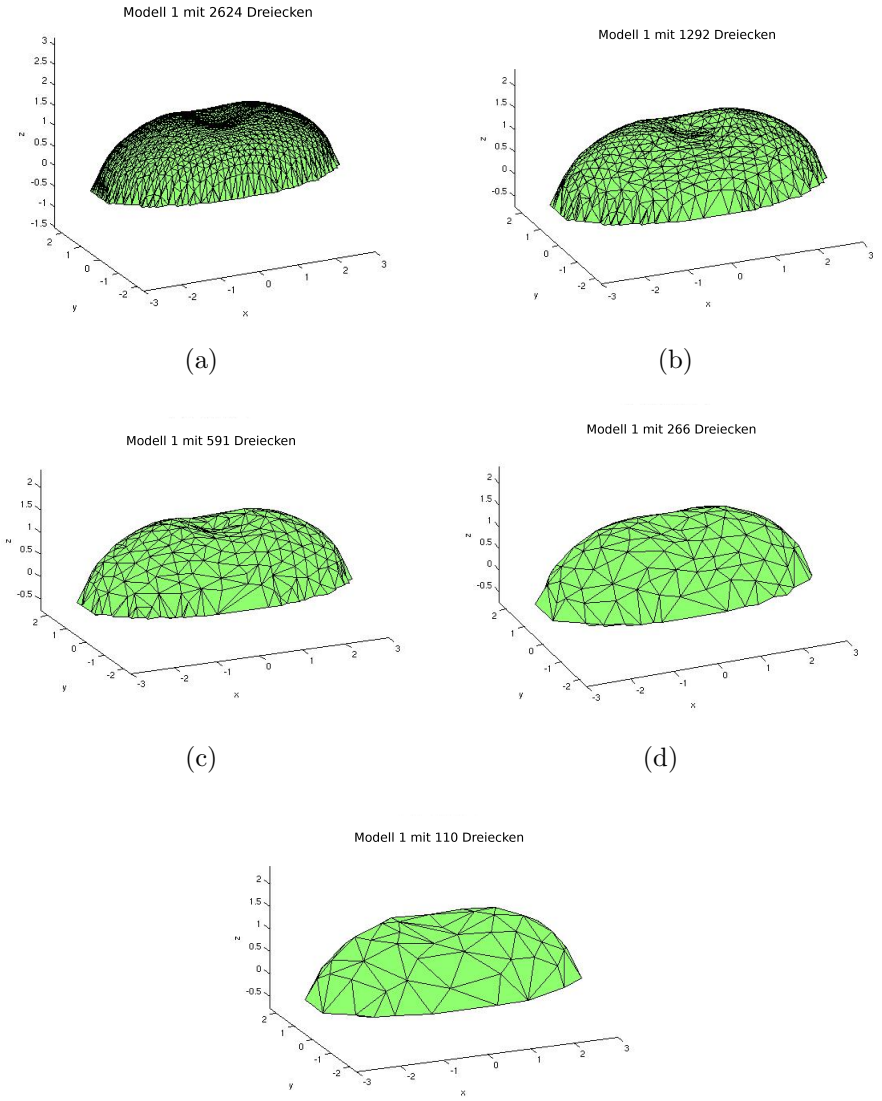
7.1.3 Testbeispiel 3

Im diesem Abschnitt gehen wir analog zu Abschnitt 7.1.2 vor, jedoch wird die Form des Modells zusätzlich durch gleichverteilte Zufallsgrößen verändert. Des Weiteren wird die Anzahl der Scandaten ebenso durch eine gleichverteilte Zufallsgröße festgelegt. Für das Generieren der zufälligen Werte nutzen wir dabei die MATLAB-Funktion `rand`.

Zunächst wird mit Hilfe einer binären gleichverteilten Zufallsgröße ausgewählt, welches von den beiden Modellen aus Abschnitt 7.1.2 für die weitere Veränderung genommen wird.

Sei also $\{(x_i, y_j) : i = 0, \ldots, s; j = 0, \ldots, t\}$ die Menge der Gitterpunkte des äquidistanten Gitters aus Abschnitt 7.1.2 mit $s, t \in \mathbb{N}$. Um nun die Form des Modells zu verändern, wird der Zielfunktionswert einer zufällig generierten Funktion im $\mathbb{R}^3$ zu dem Funktionswert der Funktion h an den Gitterpunkten addiert, die im Definitionsbereich der Funktion h liegen. Damit jedoch nicht das komplette Modell verändert wird, sollen die zufällig generierten Funktionen nur lokal wirken. Das heißt, sie haben nur auf einem beschränkten Gebiet Werte, die von Null verschieden sind.

Um das Modell noch interessanter zu gestalten, bestimmen wir außerdem mit Hilfe einer weiteren gleichverteilten Zufallsgröße $\xi \in \{1, 2, 3\}$ die Anzahl der zu addierenden zufällig generierten Funktionen. Nun bleibt noch für alle $k \in \{1, \ldots, \xi\}$ offen, die zufällig generierte Funktion zu definieren. Um die Größe des beschränken Gebiets festzulegen, betrachten wir einen gleichverteilten Zufallsvektor $\zeta = (\zeta_1, \zeta_2) \in \{x_1 - x_0, \ldots, x_s - x_0\} \times \{y_1 - y_0, \ldots, y_t - y_0\}$. Weiter sei $\nu = (\nu_1, \nu_2) \in \{x_0, \ldots, x_s - \zeta_1\} \times \{y_0, \ldots, y_t - \zeta_2\}$ ein gleichverteilter Zufallsvektor, der die Position des Gebietes



(a) (b) (c) (d) (e)

Abbildung 7.1: Darstellung einer Hierarchie von Modellen für Modell 1

angibt.

Dann können wir für einen gleichverteilten Zufallsvektor $\eta = (\eta_1, \eta_2, \eta_3) \in \{1,2\} \times \{1,2\} \times [0{,}15; 0{,}3]$ die Funktion $b^k_{\zeta,\nu,\eta} : D \mapsto \mathbb{R}$ durch

$$b^k_{\zeta,\nu,\eta}(x,y) = \begin{cases} \eta_3 \sin\left(\eta_1 \pi \frac{x-\nu_1}{\zeta_1}\right) \sin\left(\eta_2 \pi \frac{x-\nu_2}{\zeta_2}\right) & \text{für } (x,y) \in [\nu_1, \nu_1+\zeta_1] \times [\nu_2, \nu_2+\zeta_2] \\ 0 & \text{sonst} \end{cases}$$

definieren. Das Intervall [0,15; 0,3] ist so gewählt, damit die absoluten Werte von b^k größer gleich dem Maximalwert der Gauß-Funktion, jedoch kleiner gleich der Hälfte der Maximalausdehnung der Halbkugel in z-Richtung sind.

Das entstandene Modell, worauf wir ebenso das in Abschnitt 5.1 beschriebene Verfahren anwenden, um eine Hierachie bilden, bezeichnen wir mit *Mrand*.

Die Scandaten erzeugen wir analog zu Abschnitt 7.1.1, wobei für eine gleichverteilte Zufallsgröße $\theta \in \{45, 46, \ldots, 90\} \subset \mathbb{N}$ die Anzahl der Gitterpunkte des äquidistanten Gitters durch θ^2 bestimmt wird.

7.2 Empirische Parametertests

Dieser Abschnitt widmet sich vorwiegend der empirischen Bestimmung der Parameter, die für den Algorithmus 1 benötigt werden. Dabei werden wir den Wert für die relative Genauigkeit der Zielfunktion im Bündelverfahren, ebenso das verwendete Abbruchkriterium und den Wert für ρ bei der Bestimmung der Bestrafungsfunktion im Abschnitt 5.3.1 näher untersuchen. Für das Bündelverfahren verwenden wir das im selben Abschnitt erwähnte Softwarepaket.

Die hier verwendete Implementierung des Bündelverfahrens verwendet dabei eine innere und äußere Schleife, wobei für die innere Schleife ein Parameter μ für die relative Genauigkeit verlangt wird. Sie wird erfolgreich terminiert, falls

$$\frac{f_v}{1 + \hat{t}_c} < \mu$$

erfüllt ist. Es bezeichnet f_v den vorhergesagten Fortschritt für den nächsten Schritt und $\hat{t}_c$ den aktuellen Funktionswert von $\hat{t}(y)$ (siehe (5.18)).

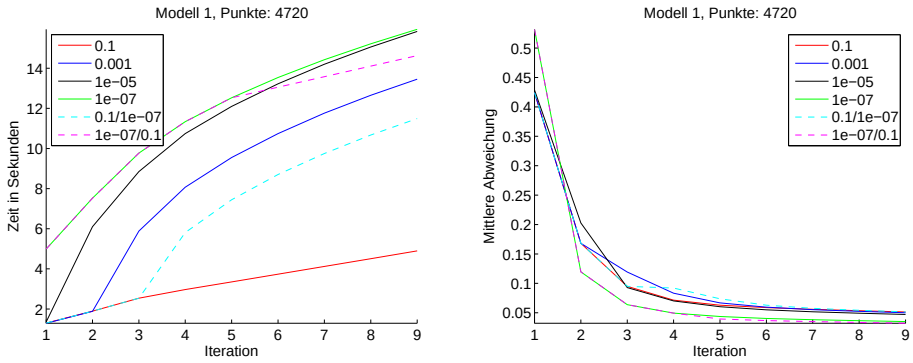
Die äußere Schleife wird abgebrochen, falls

$$\|\overline{s}\|_2 \leq k\frac{d_v}{n}, \tag{7.1}$$

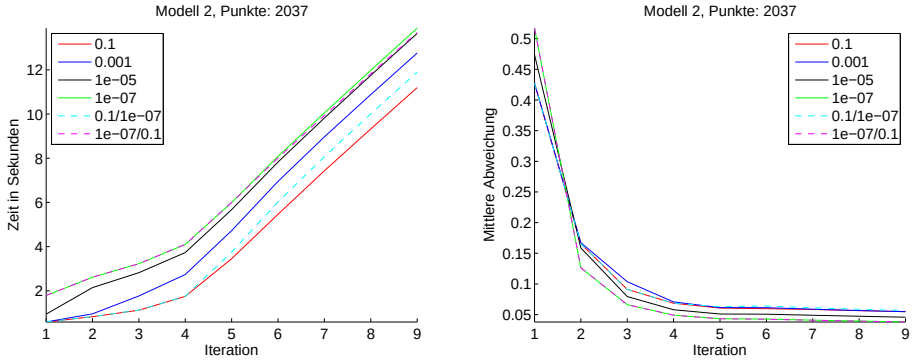
wobei k eine Konstante und d_v die Anzahl der dualen Variablen bzw. die Anzahl der verwendeten Dreiecke m und n die Anzahl der Punkte ist. Das Symbol $\overline{s}$ bezeichnet den aggregierten Subgradienten in (5.19), nachdem jedes Element des Vektors durch die Anzahl der Punkte, die Kanten zu dem entsprechenden Dreieck haben, geteilt wurde.

7.2.1 Parameter für die relative Genauigkeit

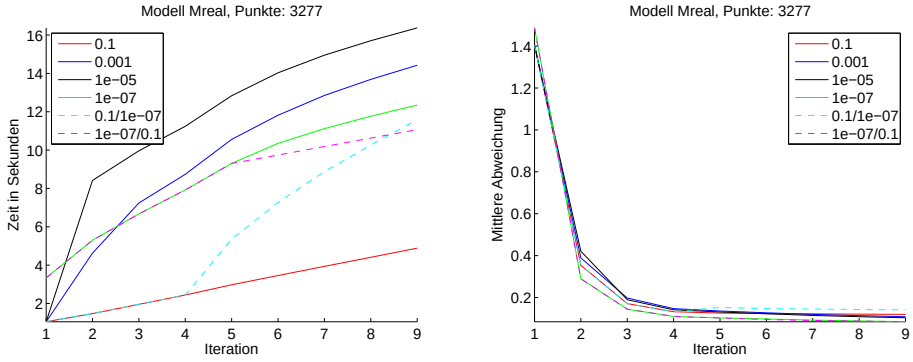
In diesem Abschnitt wird der Parameter für μ empirisch bestimmt. Dabei verwenden wir das Modell aus Abschnitt 7.1.1 und die zwei Modelle aus Abschnitt 7.1.2, wobei alle entsprechenden Scandaten genutzt werden.



(a) Modell 1, Punkte: 4720



(b) Modell 2, Punkte: 2037



(c) Mreal, Punkte: 3277

Abbildung 7.2: Ergebnisse für unterschiedliche relative Genauigkeiten

Um gleichzeitig eine Aussage über die Robustheit zu bekommen, verschieben wir das Modell nach der Grobregistrierung entlang eines äquidistanten Gitters im Intervall von $[-0{,}8; 0{,}8]^3 \subset \mathbb{R}^3$, woraus man $5^3 = 125$ Auswertungen erhält. Aus den gewonnenen Messwerten bilden wir den Mittelwert und verwenden diesen gemittelten Wert für die Diagramme in Abbildung 7.2.

Dabei betrachten wir die gewonnenen Messwerte für die erste bis neunte Iteration im Algorithmus 1 und verwenden für den Parameter μ die Werte 0,01; 10^{-2j-1} für $j = 1, \ldots, 3$. Da in den ersten Iterationen durch die verschiedenen Stufen der Hierarchie das eigentliche Modell, zu dem die Scandaten gematcht werden sollen, noch nicht verwendet wird, erscheint es sinnvoll, für die ersten Iterationen einen anderen Parameter für μ zu nutzen. Deshalb untersuchen wir auch das Verhalten des Algorithmus, wenn wir den Parameter μ für die ersten Iterationen, solange das feinste Modell der Hierarchie noch nicht verwendet wird, auf 0,01 bzw. 10^{-7} und danach auf 10^{-7} bzw. 0,01 setzen. Die Ergebnisse für die Kombination sind in den Diagrammen in Abbildung 7.2 durch eine gestichelte Linie dargestellt. Die Konstante k in (7.1) legen wir auf 0,05 und den Wert für ρ auf 6 bzw. $\beta^i = \max\{0{,}1u(i); 1\}$ fest.

Für ein Modell mit den entsprechenden Scandaten messen wir die verwendete Zeit nach der jeweiligen Iteration für die komplette Berechnung. Analog wird auch die mittlere Abweichung von Scanpunkt zu dem aktuellen Modell der Hierarchie nach der jeweiligen Iteration berechnet. In der Abbildung 7.2 ist links die benötigte Zeit und rechts die mittlere Abweichung in Abhängigkeit von der Iteration dargestellt. Wir repräsentieren die Ergebnisse in der Abbildung 7.2, wobei wir aus den neun verschiedenen Tests nur die charakteristischen Graphen zeigen.

Da die Abweichung in den ersten vier bzw. fünf Iterationen, bedingt durch die Verwendung der Hierarchie von Modellen, zu unterschiedlichen Modellen berechnet wurde, liegt das Hauptaugenmerk auf den letzten Iterationen. In der Abbildung 7.2 ist zu beobachten, dass, wenn der Parameter μ vorerst für die ersten Iterationen auf 10^{-7} und dann auf 0,01 gesetzt wird, erhält man die kleinste mittlere Abweichung nach der neunten Iteration unter allen getesteten Werten von μ. Diese Kombination benötigt sogar weniger Zeit nach der neunten Iteration, als wenn man für alle Iterationen μ auf den konstanten Wert 10^{-5} setzt. Offensichtlich erhält man bessere Ergebnisse, wenn für die ersten Iterationen eine höhere relative Genauigkeit gefordert wird.

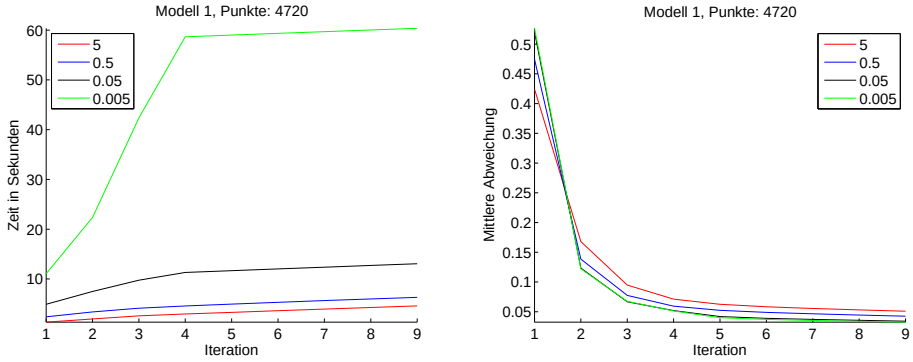
Es ist auch zu beobachten, dass große Werte von μ deutlich weniger Zeit beanspruchen und die mittere Abweichung nach der neunten Iteration dennoch recht nah der berechneten Abweichung für kleinere Werte von μ ist.

Aus den erhaltenen Ergebnissen setzen wir den Parameter μ im Algorithmus 1 für alle weiteren Tests auf Wert 10^{-7}, solange noch nicht das feinste Modell verwendet wird, und sonst auf 0,01.

7.2.2 Konstante vor der Norm des aggregierten Subgradienten

Für die Konstante k in (7.1) gehen wir analog zu den Ausführungen in Abschnitt 7.2.1 vor, wobei wir den Wert von μ auf den im selben Abschnitt vorgeschlagenen Wert setzen und das Verhalten für $k = 5 \cdot 10^{-j}$ für $j = 0, \ldots, 3$ untersuchen.

Die Ergebnisse sind dabei in Abbildung 7.3 dargestellt.



(a) Modell 1, Punkte: 4720

Abbildung 7.3: Ergebnisse für unterschiedliche Konstanten im Abbruchkriterium

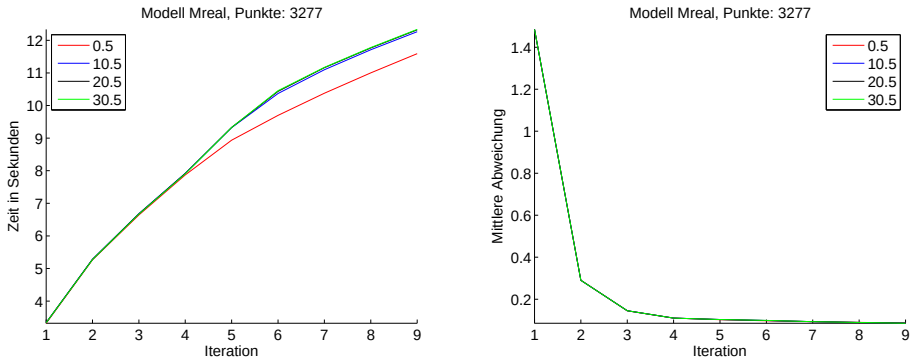
Wir beobachten in Abbildung 7.3, je kleiner die Konstante k gesetzt wird, desto mehr Zeit wird benötigt und desto kleiner sind die mittlere Abweichungen. Für $k = 0{,}005$ wird jedoch näherungsweise viermal mehr Zeit für die Berechnung der neun Iterationen benötigt als für $k = 0{,}05$ und die erreichte mittlere Abweichung unterscheidet sich kaum zu der berechneten Abweichung für $k = 0{,}05$. Um eine gute mittlere Abweichung zu erhalten und dennoch nicht zu viel Zeit zu benötigen, setzen wir den Parameter k für die weiteren Tests in den kommenden Abschnitten auf 0,05.

7.2.3 Parameter für die Bestrafung

Analog zu Abschnitt 7.2.1 bestimmen wir in diesem Abschnitt den Parameter für ρ (siehe Abschnitt 5.3.1) empirisch. Wir setzen den Wert μ, wie im selben Abschnitt vorgeschlagen, und die Konstante k auf $0,05$. In der Abbildung 7.4 sind die Ergebnisse für die Werte $\rho = 0{,}5; 10{,}5; 20{,}5; 30{,}5$ dargestellt.

Aus den Kurven in den Abbildung 7.4 ist deutlich erkennbar, dass für kleine Werte von ρ die benötigte Zeit am kleinsten ist und die berechnete mittlere Abweichung sich kaum von den berechneten mittleren Abweichungen für größere Werte von ρ unterscheidet. Je kleiner ρ desto kleiner ist der Bestrafungsterm, der zu der Zielfunktion in (5.18) dazu addiert wird, wenn die Nebenbedingung $Dx \leq u$ nicht erfüllt ist. Folglich werten wir die Ergebnisse so, dass es offensichtlich hinreichend ist, wenn die Nebenbedingung nur näherungsweise erfüllt ist. Wir setzen deshalb ρ auf 0,5 für alle weiteren Tests.

Damit sind alle Parameter für den Algorithmus 1 empirisch bestimmt und wir können nun den Algorithmus 1 mit anderen Algorithmen wie z. B. dem ICP vergleichen.



(a) Modell 1, Punkte: 4720

(b) Modell Mreal, Punkte: 3277

Abbildung 7.4: Ergebnisse für unterschiedliche Parameter für die Bestrafung

7.3 Vergleich verschiedener Algorithmen für die Feinregistrierung

In diesem Abschnitt wird anhand verschiedener Verfahren getestet, wie sich die Idee aus Abschnitt 4.3.2, die maximale Anzahl der Punkte, die einem Dreieck zugewiesen werden dürfen, bei der Koorespondenzpunktsuche in der Feinregistrierung zu beschränken, auf die Konvergenz des iterativen Verfahrens auswirkt. Dabei liegt die Konzentration im Wesentlichen auf zwei Aspekten: Wie groß die mittlere Abweichung von einem Punkt zum Modell nach jeder Iteration ist und welche Zeit für die Berechnungen der Iterationen benötigt wird.

Wir vergleichen dafür den Algorithmus 1 mit einem modifizierten Algorithmus 1, der die Korrespondenzpunksuche exakt mittels des Netzwerk-Simplex aus dem Softwarepaket CPLEX löst, und dem Standardverfahren ICP. Um zusätzlich die Information zu bekommen, ob die Verwendung der Hierarchie von Modellen lohnend in Hinsicht auf die Schnelligkeit und Genauigkeit ist, starten wir das ICP einerseits unter Einsatz der Hierarchie von Modellen und andererseits unter der bloßen Verwendung des feinsten Modells. Um die verschiedenen Algorithmen besser unterscheiden zu können, nennen wir den Algorithmus, der die Korrespondenzpunktsuche mittels des Netzwerk-Simplex löst, Algorithmus 2 und das ICP, bei dem die Hierarchie genutzt wird, ICP_H.

Für die Testzwecke verwenden wir die Testbeispiele aus Abschnitt 7.1.1, 7.1.2 und 7.1.3, wobei die Anzahl der Dreiecke des feinsten Modells und die Anzahl der entsprechenden Scandaten der Tabelle 7.3 zu entnehmen sind.

Sei $\xi \in [-0{,}15; 0{,}15]^3 \times [-0{,}8; 0{,}8]^3$ mit $\xi = (\xi_1, \ldots, \xi_6)$ ein gleichverteilter Zufallsvektor. Um wieder eine Aussage über die Robustheit zu erhalten, wird das Modell nach der Grobregistrierung zunächst um den Wert von ξ_1, ξ_2 bzw. ξ_3 um die x-,y- bzw.

z-Achse gegen den Uhrzeigersinn gedreht und anschließend um den Vektor (ξ_4, ξ_5, ξ_6) verschoben. Danach wird die Feinregistrierung für alle Verfahren durchgeführt und man erhält die benötigte Zeit und die mittlere Abweichung. Dieser Vorgang wird für jedes Modell neunmal wiederholt.

In Tabelle 7.3 geben wir die gemittelten Messwerte an, wobei die Zeit in Sekunden und die mittlere Abweichung (gekennzeichnet durch die Abkürzung „Abw." in Tabelle 7.3) von Scanpunkt zu Modell für alle verwendeten Testbeispiele für eine Auswahl der ersten 50 Iterationen (Iter.) angegeben ist.

	ICP		**ICP_H**		**Alg.** 2		**Alg.** 1	
Iter.	**Zeit**	**Abw.**	**Zeit**	**Abw.**	**Zeit**	**Abw.**	**Zeit**	**Abw.**
Modell Mrand1, **#Dreiecke:** 674, **#Punkte**: 3376								
4	21	0,118	1,88	0,12	9,63	0,0949	7,37	0,0881
5	26,2	0,0973	2,02	0,1	10,2	0,0829	8,16	0,0743
10	53,8	0,0537	2,73	0,0607	12,9	0,0646	10,9	0,0498
20	107	0,0259	4,16	0,0343	18,2	0,0574	15	0,035
30	160	0,0177	5,59	0,0254	23,5	0,0544	18,9	0,0293
40	212	0,0134	7,02	0,018	28,8	0,0526	22,6	0,0199
50	265	0,0104	8,44	0,0135	34,1	0,0514	25,8	0,0105
Modell Mrand2, **#Dreiecke:** 2624, **#Punkte**: 1417								
5	41,9	0,0629	1,08	0,0633	3,88	0,0505	3,79	0,0579
6	50,2	0,0554	1,16	0,056	4,12	0,0452	5,24	0,0533
10	83,8	0,0397	1,49	0,0403	5,1	0,0325	11	0,0405
20	168	0,0224	2,3	0,0225	7,5	0,0182	25,7	0,022
30	251	0,015	3,11	0,015	9,88	0,0122	40,8	0,015
40	335	0,0105	3,93	0,0105	12,3	0,00862	57,9	0,0107
50	419	0,00745	4,74	0,00745	14,6	0,0061	81,4	0,00776
Modell Mrand3, **#Dreiecke:** 674, **#Punkte**: 3370								
4	20,7	0,0908	1,83	0,0904	9,16	0,0574	6,98	0,061
5	25,9	0,0775	1,97	0,0773	9,69	0,0482	7,65	0,0529
10	51,8	0,0563	2,68	0,0563	12,2	0,035	10,1	0,0375
20	103	0,034	4,1	0,034	17,3	0,0277	13,7	0,0232
30	155	0,021	5,52	0,0209	22,4	0,0251	16,8	0,0151
40	207	0,0131	6,95	0,0131	27,7	0,0242	19,9	0,00974
50	259	0,00818	8,37	0,00817	33	0,0238	22,9	0,0061
Modell Mrand4, **#Dreiecke:** 2624, **#Punkte**: 7451								
5	217	0,0928	5,25	0,0923	33,2	0,0695	25,8	0,0616
6	260	0,082	5,58	0,0821	35,2	0,0611	30,2	0,0549
10	433	0,0553	6,92	0,0568	43,1	0,0484	45,5	0,0363
20	866	0,0227	10,3	0,0235	62,3	0,0389	78,7	0,0153
30	1,3e+03	0,00957	13,6	0,00987	81,6	0,0363	110	0,00679
40	1,73e+03	0,00409	17	0,00419	101	0,0356	140	0,00303

	ICP		**ICP_H**		**Alg.** 2		**Alg.** 1	
Iter.	**Zeit**	**Abw.**	**Zeit**	**Abw.**	**Zeit**	**Abw.**	**Zeit**	**Abw.**
50	2,27e+03	0,00178	20,4	0,0018	120	0,0355	175	0,00135
Modell 1 **#Dreiecke:** 674, **#Punkte**: 4720								
4	29,3	0,074	2,5	0,074	13,7	0,0551	10,6	0,0566
5	36,6	0,0657	2,69	0,0658	14,5	0,0477	11,7	0,0504
10	73,2	0,048	3,67	0,0481	18,2	0,0332	15,6	0,0351
20	146	0,0299	5,63	0,0299	25,4	0,0243	20,8	0,0226
30	219	0,02	7,6	0,02	32,6	0,021	25,1	0,016
40	293	0,0137	9,57	0,0137	39,8	0,0195	29,1	0,0117
50	366	0,00945	11,5	0,00945	47,1	0,0186	33	0,00864
Modell Mreal, **#Dreiecke:** 2800, **#Punkte**: 1874								
5	59,4	0,184	1,29	0,191	5,51	0,142	4,72	0,149
6	71,3	0,177	1,4	0,186	5,92	0,139	5,21	0,144
7	83,2	0,171	1,5	0,182	6,33	0,136	5,67	0,14
10	119	0,158	1,83	0,172	7,57	0,128	6,91	0,13
20	238	0,136	2,92	0,157	11,7	0,107	10,2	0,11
30	356	0,12	4,04	0,151	15,8	0,0904	13,2	0,0936
40	475	0,102	5,14	0,148	19,9	0,0765	16,3	0,082
50	594	0,0831	6,25	0,147	23,9	0,0649	19,4	0,0734
Modell Mreal, **#Dreiecke:** 2800, **#Punkte**: 5513								
5	175	0,191	3,63	0,202	23,3	0,133	20	0,135
6	210	0,182	3,92	0,195	24,8	0,128	22,8	0,128
10	350	0,16	5,11	0,177	30,6	0,115	30,1	0,109
20	701	0,134	8,14	0,158	45,2	0,095	42,1	0,0869
30	1,05e+03	0,115	11,2	0,149	59,9	0,0805	51,2	0,0738
40	1,4e+03	0,0902	14,3	0,145	74,6	0,069	59,3	0,0626
50	1,75e+03	0,0663	17,4	0,142	89,2	0,0596	67,2	0,0499

Tabelle 7.3: Vergleich verschiedener Algorithmen für die Feinregistrierung

Aus den Ergebnissen in Tabelle 7.3 stellen wir zunächst fest, dass die Verwendung der Hierarchie der Modelle eine erhebliche Verbessung in Bezug auf die Berechnungszeiten bringt, wenn man die benötigten Zeiten von dem ICP und ICP_H vergleicht. Dieses Resultat ist das erwartete, da unter Verwendung der Hierarchie deutlich weniger Abstandsberechnungen zwischen Modell und Punkt nötig werden. Interessanterweise können, wie für Modell Mrand3 zu sehen, unter Verwendung der Hierarchie sogar bessere Werte für die mittlere Abweichung erreicht werden.

Weiter beobachten wir, wenn man die Verbesserung der mittleren Abweichung pro Iteration für alle Verfahren für die ersten Iterationen vergleicht, dass die Idee, die Anzahl der Punkte, die einem Dreieck zugeordnet werden dürfen, zu beschränken, eine Verbesserung bringt. Jedoch ist die Berechnungzeit für jede Iteration im Algo-

rithmus 1 und 2 deutlich größer als die benötigte Zeit für die Iteration im ICP_H. Somit erhält man in Bezug auf die benötigte Zeit und mittlere Abweichung bis auf das Testbeispiel aus Abschnitt 7.1.1 die klar besseren Ergebnisse für das ICP_H.

Interessant ist auch der Vergleich zwischen Algorithmus 1 und 2. Betrachtet man die ersten Iterationen, so benötigt der Algorithmus 1 pro Iteration weniger Zeit, liefert jedoch meist größere mittlere Abweichungen. Offensichtlich liefert die approximative Lösung des Korrespondenzenproblems in Abschnitt 4.3.2 in diesem Zusammenhang schlechtere Ergebnisse in Bezug auf die mittlere Abweichung, was aber auch durch die Heuristik hervorgerufen sein kann, da diese keinem Optimalitätskriterium unterliegt. Für die letzten Iterationen ist meist zu beobachten, dass der Algorithmus 1 die besseren mittleren Abweichungen liefert.

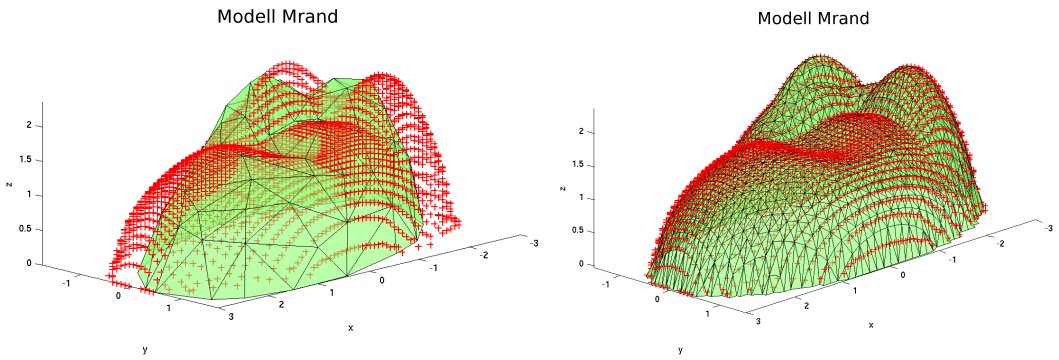
Im Allgemeinen hängt das Verhalten der Algorithmen stark von dem jeweiligen Modell ab. So liefert der Algorithmus 1 bessere Ergebnisse als das ICP_H für das Modell Mreal, jedoch nicht für alle anderen.

Es ist zu beobachten, dass die mittlere Abweichung zwar in jeder Iteration kleiner wird, wenn wir den Algorithmus 1 anwenden, aber dennoch nach der 50. Iteration deutlich größer als null ist. Da die Scandaten für die Modelle Mrand und Modell 1 generiert wurden, ist die tatsächliche mittlere Abweichung jedoch null. Um das Verhalten des Algorithmus 1 noch besser zu verdeutlichen, werden wir im nächsten Abschnitt zwei konkrete Testbeispiele betrachten.

7.4 Konvergenzverhalten für zwei Testbeispiele

Um das Verhalten des Algorithmus 1 näher zu untersuchen, werden wir in diesem Abschnitt zunächst ein Testbeispiel aus Abschnitt 7.1.3 betrachten, welches wir in Abbildung 7.5 dargestellt haben. Die Anzahl der generierten Punkte ist dabei 2303. Das Modell, welches aus 2624 Dreiecken besteht, wurde nach der Grobregistrierung 20 Grad gegen den Uhrzeigersinn um die z-Achse gedreht. Die Abbildung 7.5 stellt links diese Ausgangslage von Modell und Scandaten und rechts die Lage nach der 20. Iteration dar. Die Scandaten wurden zur besseren Illustration ausgedünnt.

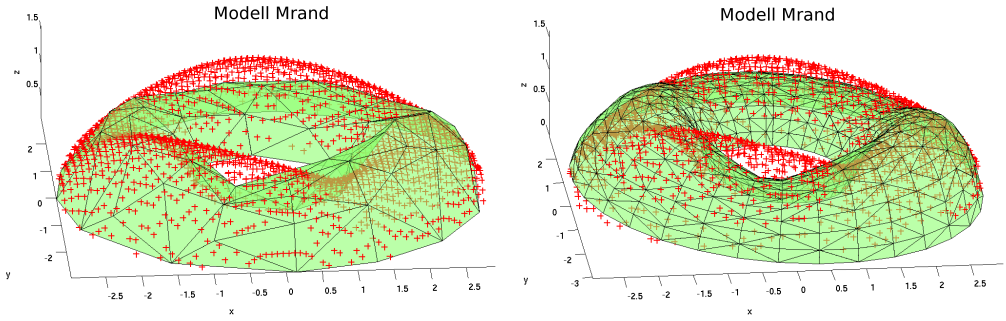
Wie man sieht, wird das Modell wieder zurückgedreht und Scandaten und Modell werden in die gleiche Ausrichtung gebracht. In Abbildung 7.6 ist ein weiteres Testbeispiel dargestellt, welches analog zu Abschnitt 7.1.3 erzeugt wurde. Der Unterschied besteht darin, dass das Modell nicht auf der Grundlage der Funktion $h(x, y)$ aus Abschnitt 7.1.2 sondern aus der oberen Hälfe eines Torus mit Innenradius 1 und Außenradius 3 gebildet wurde. Die Anzahl der Dreiecke ist dabei 832 und die Scandaten bestehen aus 3833 Punkten. Das Modell wurde diesmal nach der Grobregistrierung um 90 Grad gegen den Uhrzeigersinn gedreht, so dass, wie in Abbildung 7.6 links zu sehen, sich die Punkte der Scandaten, die zu den Erhebungen des Torus korrespondieren, genau zwischen den Erheben des Modells befinden. Rechts ist die berechnete



(a) Ausgangslage (b) Lage nach 20. Iteration

Abbildung 7.5: Konvergenzverhalten für Beispiel 1

Lage zwischen Modell und Scandaten nach der 50. Iteration im Algorithmus 1 dargestellt.



(a) Ausgangslage (b) Lage nach 50. Iteration

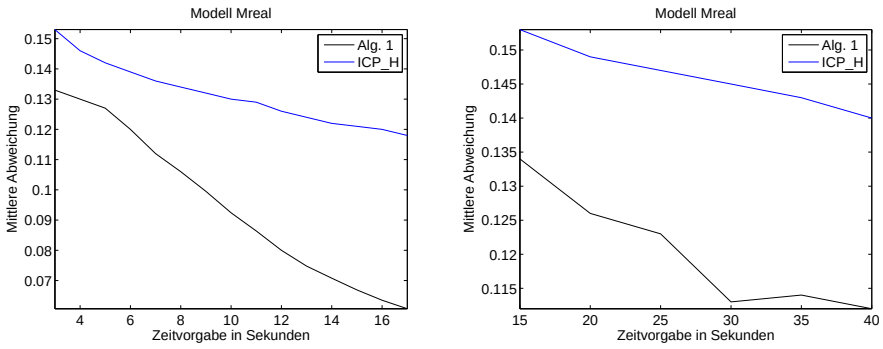
Abbildung 7.6: Konvergenzverhalten für Beispiel 2

In Abbildung 7.6 ist zu beobachten, dass Scandaten und Modell durch den Algorithmus 1 nach 50 Iterationen nicht ausgerichtet werden. Offensichtlich sind die berechneten Korrespondenzpaare so gewählt worden, dass eine Drehung um die z-Achse sowohl mit als auch gegen den Uhrzeigersinn den Funktionswert des zu minimierenden Funktionals in Satz 5.3.1 vergrößern würde. Folglich wird der mittlere Abstand zwischen Modell und Scandaten in den weiteren Iterationen nicht mehr reduziert.

7.5 Erreichbare Genauigkeiten für eine Zeitvorgabe

Für den letzten numerischen Test geben wir eine Zeit vor und berechnen die erreichte mittlere Abweichung in dieser Zeit. Dabei vergleichen wir den Algorithmus 1 mit dem ICP_H. Da die Iteration im Algorithmus 1 mehr Zeit beansprucht als die Iteration im ICP_H, wenden wir, falls die geschätzte Zeit für die nächste Iteration im Algorithmus 1 größer als die noch verfügbare Zeit ist, für alle weiteren Iterationen auch den ICP_H an. Für diese Testzwecke nehmen wir das Testbeispiel aus Abschnitt 7.1.1 mit realen Scandaten der Größe 1366 und 7548.

Um wieder die Aussage über die Robustheit zu erhalten, gehen wir analog zu Abschnitt 7.3 vor. In Abbildung 7.7 sind links die gemittelten Abweichungen nach einer vorgebenen Zeit von $3, 4, \ldots, 15$ Sekunden und rechts nach $15, 20, \ldots, 40$ Sekunden dargestellt.



(a) Punktanzahl: 1874 (b) Punktanzahl: 7548

Abbildung 7.7: Erreichbare Genauigkeit für eine Zeitvorgabe

Die gleichen Ergebnisse präsentieren wir auch in Tabelle 7.4, wobei außerdem noch die Varianz der Abweichung von Punkt zu Modell und die maximale Abweichung von Punkt zu Modell angeben ist.

	Algorithmus 1			**ICP_H**		
Zeit	**Abw.**	**max. Abw.**	**Var.**	**Abw.**	**max. Abw.**	**Var.**
Modell Mreal, # Dreiecke: 2800, **# Punkte:** 1874						
3	0,133	0,476	0,0112	0,153	0,695	0,0177
4	0,13	0,441	0,0104	0,146	0,667	0,0164
5	0,127	0,466	0,0108	0,142	0,659	0,0157
6	0,12	0,448	0,0102	0,139	0,66	0,0152
7	0,112	0,432	0,00956	0,136	0,661	0,0148

	Algorithmus 1			ICP_H		
Zeit	**Abw.**	**max. Abw.**	**Var.**	**Abw.**	**max. Abw.**	**Var.**
8	0,106	0,423	0,00902	0,134	0,663	0,0145
9	0,0995	0,42	0,00854	0,132	0,665	0,0142
10	0,0924	0,422	0,00807	0,13	0,668	0,014
11	0,0864	0,426	0,00771	0,129	0,67	0,0139
12	0,08	0,426	0,00734	0,126	0,672	0,0137
13	0,0748	0,428	0,00704	0,124	0,675	0,0135
14	0,0708	0,431	0,00676	0,122	0,677	0,0134
15	0,0669	0,435	0,00652	0,121	0,68	0,0133
Modell Mreal, # Dreiecke: 2800**, # Punkte:** 7548						
15	0,134	0,464	0,0111	0,153	0,699	0,0172
20	0,126	0,464	0,0106	0,149	0,692	0,0163
25	0,123	0,43	0,0102	0,147	0,687	0,0157
30	0,113	0,416	0,0095	0,145	0,687	0,0154
35	0,114	0,422	0,0098	0,143	0,688	0,0151
40	0,112	0,419	0,0096	0,14	0,688	0,0149

Tabelle 7.4: Erreichbare Genauigkeit für eine Zeitvorgabe

In der Tabelle 7.4 und in der Abbildung 7.7 ist eine deutliche Verbesserung in Bezug auf die Zeit und berechnete Genauigkeit des Algorithmus 1 gegenüber dem ICP_H zu sehen.

7.6 Schlussfolgerungen

Aus den numerischen Tests ist erkennbar, dass der Algorithmus 1 bezogen auf die Feinregistrierung zwar eine Alternative zum ICP darstellt, jedoch in Bezug auf die Schnelligkeit eher das Nachsehen hat. Des Weiteren bringt die Idee, die Anzahl der Punkte, die einem Dreieck zugewiesen werden, zu beschränken, eine Verbesserung. Jedoch ist die Berechnung der Lösung des Flussproblems mit den hier vorgestellten Solvern zu teuer. Folglich wird der Algorithmus 1 zu ineffektiv. Hinzu kommt, dass die Schätzung, siehe hierzu Abschnitt 4.3.2, der Anzahl der Punkte, die auf ein Dreieck kommen, sehr ungenau ist und zusätzlich voraussetzt, dass die Kameraebene bezogen auf das Modell annähernd parallel zur x-y-Ebene liegt. Letzteres ist nur der Fall, wenn das Modell bezogen auf die Scandaten nicht verkippt ist. Dies trägt zur weiteren Ineffektivität bei.

Es wird auch ersichtlich, wenn man insbesondere das zweite Testbeispiel aus Abschnitt 7.4 betrachtet, dass alle hier betrachteten Verfahren bezogen auf die Feinregistrierung nur lokal optimieren.

Der Erfolg in dieser Arbeit liegt mehr in der Bildung der Hierarchie von Modellen.

Denn die Verwendung der Hierarchie bringt einen enormen Zeitgewinn bei der Durchführung der Feinregistrierung und bringt kaum Verluste bezüglich der Genauigkeit.

8 Zusammenfassung

In den letzten Kapiteln haben wir anfänglich das in Kapitel 2 beschriebene Problem diskutiert, einen Lösungsansatz vorgestellt und dementsprechend einen Algorithmus in Kapitel 6 vorgeschlagen, der, nach den Daten aus der Tabelle 7.3 zu urteilen, das Modell des Problems aus Kapitel 2 mit den real aufgenommenen Scandaten zuverlässig und schnell matcht. Jedoch haben die Tests in Kapitel 7 auch einige Schwächen des Algorithmus aufgezeigt. Im nächsten Abschnitt werden wir noch einige abschließende Bemerkungen zu dem behandelten Algorithmus anfügen und im darauffolgenden Abschnitt noch perspektivische Vorschläge unterbreiten, um den Algorithmus zu verbessern.

8.1 Abschließende Bemerkungen

Die Hauptkomponentenanalyse, die wir für die Grobregistrierung verwendet haben, bedingt, dass das Modell und die Scandaten, annähernd die gleiche Form haben. Dies ist erforderlich, damit die Angleichung der Hauptkomponenten des Modells zu den Hauptkomponenten der Scandaten ein sinnvolles Ergebnis liefert. Sind die korrespondierenden Eigenwerte der Hauptkomponenten von ähnlicher Größe, so kann eine starke Ausdünnung der Scandaten, aber auch leichte Abweichungen der Scandaten vom Modell dazu führen, dass Modell und Scandaten nach der Grobregistrierung falsch ausgerichtet werden.

Des Weiteren kann für das hier vorgestellte Verfahren für die Feinregistrierung nachteilig sein, dass die Kosten auf den Kanten zwischen den Punkten und Dreiecken auf der Grundlage des derzeitigen Abstandes zwischen Modell und Punkt berechnet werden. Dies setzt voraus, dass Modell und Scandaten schon in einer ähnlichen Lage zueinander liegen.

Nimmt man als Beispiel einen Kugelschreiber, dessen Modell und Scandaten nach der Grobregistrierung entgegengesetzt liegen, d. h., die Scanpunkte, die zu der Klemme korrespondieren, liegen im vorderen Bereich des Kugelschreibers im Modell. Da nun die eigentlichen Korrespondenzen weit voneinander entfernt liegen, wird das Modell durch den Algorithmus 1 nicht in die optimale Lage gedreht werden.

Hierbei wird ebenso der Nachteil ersichtlich, dass das Funktional in (4.1) in dem oben beschriebenen Verfahren nicht direkt minimiert wird.

8.2 Ausblick

In diesem Abschnitt reißen wir noch einige Ideen an, die zur Verbesserung des Algorithmus 1 in Bezug auf Schnelligkeit und Robustheit führen könnten.

Zur Berechnung der Korrespondenzpunkte kann man die Anzahl der Punkte, die auf ein Dreieck beschränkt ist, auch von unten beschränken. Dies würde sicherlich zur schnelleren Konvergenz der Position des Modells in die Endlage und zur verbesserten Robustheit führen. Allerdings ist dies nur möglich, wenn das komplette Modell in den Scandaten wiederzufinden ist.

Eine weitere Verbesserungsmöglichkeit könnte sein, das Flussproblem für die Korrespondenzpunktsuche mittels des Bündelverfahrens zu lösen, jedoch die nicht ganzzahlige Näherungslösung zu nutzen, um einen gewichteten Korrespondenzpunkt zu jedem Punkt der Scandaten zu berechnen. Mit anderen Worten der Korrespondenzpunkt zu einem Scanpunkt ergibt sich aus der entsprechend der durch die Näherungslösung gewichteten Summe der nächstliegenden Punkte aller Dreiecke. Somit ist die Berechnung der Heuristik nicht mehr notwendig.

Um den in Abschnitt 8.1 angedeuteten Problemen zu entgegnen, ist es sicherlich erforderlich, für die Grobregistrierung als auch für die Feinregistrierung nicht nur die Hauptkomponenten bzw. den Abstand zwischen Modell und Scanpunkt zu betrachten. Um ein robusteres Verhalten zu bekommen, ist es notwendig, die lokalen Informationen eines jeden Punktes des Modells als auch der Scandaten zu berechnen und entsprechend zu vergleichen. Die lokalen Informationen können die in Abschnitt 4.2.1 diskutierten Merkmale wie Krümmungsinformation, Normalenvektoren oder allgemein Informationen sein, die aus den Nachbarpunkten gewonnen werden können.

Literaturverzeichnis

[1] CPLEX. `http://en.wikipedia.org/wiki/CPLEX`.

[2] STL-Schnittstelle. `http://de.wikipedia.org/wiki/STL-Schnittstelle`.

[3] S. J. Ahn, W. Rauh, H. S. Cho, and H. J. Warnecke. Orthogonal distance fitting of implicit curves and surfaces. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 24(5):620–638, May 2002.

[4] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, Englewood Cliffs, NJ, 1993.

[5] K. S. Arun, T. S. Huang, and S. D. Blostein. Least squares fitting of two 3-D point sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9:698–700, 1987.

[6] 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):239–258, February 1992.

[7] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge Univ. Press, 2004.

[8] Christian Bär. *Elementare Differentialgeometrie*. de Gruyter, 2001.

[9] Isil Celasun, Rupen Melkisetoglu, and A. Murat Tekalp. Optimal design of 2D/3D hierarchical content-based meshes for multimedia. In Eric Andres, Guillaume Damiand, and Pascal Lienhardt, editors, *DGCI*, volume 3429 of *Lecture Notes in Computer Science*, pages 45–55. Springer, 2005.

[10] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, January 2000.

[11] Reinhard Diestel. *Graph theory*. Springer-Verlag, New York, 1997. Translated from the 1996 German original.

[12] S. Granger and X. Pennec. Multi-scale EM-ICP: A fast and robust approach for surface registration. In *ECCV*, page IV: 418 ff., 2002.

[13] Michael A. Greenspan and Mike Yurick. Approximate K-D tree search for efficient ICP. In *3DIM*, pages 442–448. IEEE Computer Society, 2003.

[14] C. Helmberg and F. Rendl. A spectral bundle method for semidefinite programming. Technical report, October 20 1997.

[15] J.-B. Hiriart-Urruty and C. Lemaréchal. *Convex analysis and minimization algorithms I*, volume 305 of *Grundlehren der math. Wissenschaften*. Springer, 1993.

[16] J. B. HIRIART-URRUTY and C. LEMARECHAL. *Convex Analysis and Minimization Algorithms II*, volume 306 of *Grundlehren der mathematischen Wissenschaften*. Springer, Berlin, Heidelberg, 1993.

[17] B. Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America*, 4:629–642, 1987.

[18] C. Langis, M. Greenspan, and G. Godin. The parallel iterative closest point algorithm. In *3DIM01*, pages 195–202, 2001.

[19] Anat Levin and Amnon Shashua. Principal component analysis over continuous subspaces and intersection of half-spaces. Heyden, Anders (ed.) et al., Computer vision - ECCV 2002. 7th European conference, Copenhagen, Denmark, May 28–31, 2002. Proceedings. Part 3. Berlin: Springer. Lect. Notes Comput. Sci. 2352, 635-650 (2002)., 2002.

[20] Li N., Cheng P., Sutton M. A., McNeill S. R. Three-dimensional point cloud registration by matching surface features with relaxation labeling. *Experimental Mechanics*, 45(1):71–82, 02 2005.

[21] Maier T., Häusler G. Segmentation based fast registration of free form surfaces in the euclidian space. Technical report, Institute of Optics, Information and Photonics, University of Erlangen-Nürnberg, 2006.

[22] Jeff Phillips. Lecture 24: Iterative closest point and earth mover's distance, 4 2007. `http://de.wikipedia.org/wiki/STL-Schnittstelle`.

[23] Rabbani T., Heuvel F. A. van den. Methods for fitting csg models to point clouds and their comparison. In *The 7th IASTED International Conference on Computer Graphics and Imaging*, page 279 – 284, 2004.

[24] H. SCHRAMM and J. ZOWE. A version of the bundle idea for minimizing a nonsmooth function: Conceptual idea, convergence analysis, numerical results. *SIAM J. Optim.*, 2:121–152, 1992.

Selbstständigkeitserklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbstständig verfasst habe und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Chemnitz, den 3. März 2009

Rüdiger Borsdorf