Institut für Formale Methoden der Informatik

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Bachelorarbeit Nr. 223

Automatische Platzierung von Beschriftung in Gebieten

Moritz Knabben

Studiengang:	Informatik
Prüfer/in:	Prof. Dr. Stefan Funke
Betreuer/in:	Dipl.-Inf. Filip Krumpe
Beginn am:	22. April 2015
Beendet am:	22. Oktober 2015
CR-Nummer:	F.2.2, I.1.2, I.3.5

Zusammenfassung

Eine Beschriftung zu platzieren ist ein wesentlicher Teil, um Informationen graphisch darzustellen. Die Beschriftung von Karten ist aufwendige Arbeit, was dazu motiviert, diesen Prozess zu automatisieren. Hierfür werden einfache Gütekriterien angenommen: Die Beschriftung wird als Box repräsentiert. Es wird die größte Box innerhalb eines Polygons gesucht.

In dieser Arbeit werden zwei entwickelte und implementierte Algorithmen beschrieben und untersucht, um Gebiete zu beschriften. In einem ersten Szenario wird eine Beschriftung in einem Gebiet gesucht, die parallel zu den Koordinatenachsen ist. Mit dem entwickelten Algorithmus lassen sich Beschriftungen effizient für beliebige Gebiete erzeugen. Im zweiten Szenario wird eine Beschriftung in einem Gebiet gesucht, die beliebig rotiert sein darf. Hierbei lassen sich Beschriftungen mit erheblichem Aufwand erzeugen. Praktisch können damit nur Beschriftungen für kleine Gebiete gefunden werden.

Abstract

Label placement is a significant part to represent information graphically. Labeling of maps is an expensive task which motivates to automate the process. For this purpose simple quality criteria are assumed: The label is represented as a box. Within a polygon, the largest box is searched.

In this thesis two developed and implemented algorithms to label areas are described and tested. The first setting investigates the case of labels which are parallel to the coordinate axes. Using the algorithm labels can be generated efficiently for arbitrary areas. In the second setting an area label is searched which can be rotated arbitrarily. Here, labeling cost is considerable. In practice labels can be found only for small areas.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Algorithmenverzeichnis

1 Einleitung

Eine Beschriftung zu platzieren ist ein wesentlicher Teil, um Informationen graphisch darzustellen und trägt dazu bei, Informationen verständlicher zu machen oder ermöglicht das Verständnis. Gerade bei der Beschriftung von Karten (als Teilgebiet der Kartographie) ist eine Beschriftung essentiell. Bis [Imh75] wurden Karten nicht nach formalen Kriterien beschriftet. Formalisierte Beschriftungskriterien ermöglichen es hingegen, Beschriftungen objektiv zu vergleichen und legen die Grundlage, um automatisch zu beschriften. Grundsätzlich unterscheidet man zwischen Punkt- (Point-), Linien- (Line-Feature-) und Flächenbeschriftung (Area-Label). Aus Sicht der Informatik stellt sich die Frage, wie man algorithmisch Beschriftungen erzeugen kann, die bestimmte Kriterien einhalten. Dies ist interessant, weil Visualisierungen ohne weitere manuelle Tätigkeit erzeugt werden können. Das Problem ist abstrakt, alle Möglichkeiten aufzuzählen eine Graphik zu beschriften. Daraus wählt man die Beschriftung, die die Kriterien am besten erfüllt.

Das Point-Label-Problem ist, dass man zu einer Menge von Punkten in einer Karte Beschriftungen findet. Hierbei dürfen sich die Labels nicht überlappen. Das Erzeugen von Pointlabels ist in der Regel deterministisch nicht in polynomieller Zeit möglich. Bereits für ein vereinfachtes Problem konnte gezeigt werden, dass es NP-vollständig ist [MS91]: Gegeben sei eine Menge von Punkten in der Ebene und für jeden Punkt eine Menge von Labelpositionen sowie eine Labelgröße. Können alle Punkte beschriftet werden, ohne dass es überlappende Labels gibt?

Eine Linienbeschriftung wird auf lineare Erscheinungen in Karten angewendet. Die Beschriftung muss der linearen Form angepasst sein. In diesem Fall kann eine Beschriftung auch mehrmals auftreten [Bar01].

Beim Beschriften von Flächen sucht man dagegen *ein* Label, das die Fläche beschriftet. Gütekriterien legen fest, was als gute Beschriftung anzusehen ist. Einfache Kriterien sind, dass das Label innerhalb eines Gebiets liegt und maximal ist. Um in einem Gebiet ein maximales, achsenparalleles Rechteck zu finden, sind effiziente ($\mathcal{O}(n \log n)$) Algorithmen bekannt [DMR97; NSB94; Kna+12]. Das Problem, ein maximales rotiertes Rechteck zu finden, ist dagegen nicht effizient lösbar [Mol+12].

Diese Arbeit untersucht, wie eine Flächenbeschriftung für Gebiete erzeugt werden kann. Ein Gebiet kann als eine begrenzte Fläche verstanden werden. Eine solche Fläche kann auch Löcher enthalten; es handelt sich demnach insbesondere um ein konkaves Gebiet. Im Folgenden wird

für den Begriff Gebiet der Begriff Polygon verwendet, der im nächsten Abschnitt eingeführt wird (Abschnitt 1.1). Eine Beschriftung wird als Box repräsentiert, für die gelten muss, dass die Box vollständig innerhalb des zu beschriftenden Gebiets liegt und die Größe der Box maximal ist. Für die gesuchte Box ist das Breiten- und Höhenverhältnis bekannt. In der vorliegenden Arbeit werden zwei Szenarien untersucht. Das erste Szenario untersucht den Fall einer achsenparallelen Box und im zweiten Szenario darf die Box beliebig rotiert sein. Probleme des ersten Szenarios werden gelöst, indem das größte leere Quadrat bezüglich der Punktemenge, die sich aus den Segmentendpunkten ergibt, berechnet wird. Das zweite Szenario wird durch einen randomisieren Ansatz gelöst. Es werden zufällig gleichverteilt Segmente aus dem Polygon gewählt und damit das größte Quadrat konstruiert. Ein maximales Quadrat ist das Maximum über mehrere Proben. Durch eine Skalierung kann in beiden Szenarien das größte Rechteck für ein gegebenes Breiten- und Höhenverhältnis gefunden werden.

1.1 Grundlegende Begriffe

In diesem Abschnitt sind grundlegende Begriffe definiert.

Definition: Ein Punkt $p = (x, y) \in \mathbb{R}^2$ ist ein Element der euklidischen Ebene, wobei x die horizontale und y die vertikale Position eines Punktes ist.

Definition: Ein Segment ist ein Geradenstück, das in einem Punkt beginnt und in einem anderen endet $s := (p, p')$.

Definition: Ein Polygon ist eine Reihe von Punkten $P := (p_1, \ldots, p_n)$, wobei (p_i, p_{i+1}) für $i \in \{1, \ldots, n-1\}$ und (p_n, p_1) die Segmente eines Polygons sind. Für alle Segmente eines Polygons gilt, dass sie sich nicht schneiden.

Definition: Ein Multi-Polygon (allgemeines Polygon) ist eine Menge von Polygonen $\mathcal{P} := \{P_1, \ldots, P_l\}$, wobei es ein ausgezeichnetes Polygon $P \in \mathcal{P}$ gibt, sodass alle anderen Polygone innerhalb von P liegen. Die Fläche eines Polygons sei $\mathcal{A}(P)$ und die Fläche eines Multi-Polygons sei $\mathcal{A}(\mathcal{P}) = \mathcal{A}(P) - \sum_{P' \in \mathcal{P} \setminus \{P\}} \mathcal{A}(P')$. Das ausgezeichnete Polygon P wird auch äußeres Polygon genannt. Alle anderen Polygone $\mathcal{P} \setminus \{P\}$ werden Löcher genannt.

1.2 Gliederung

Die Arbeit gliedert sich in folgende weitere Kapitel:

Kapitel 2 – **Ähnliche Arbeiten** – gibt einen kurzen Überblick zu vergleichbaren Verfahren sowie einen Ausblick, was hinsichtlich Laufzeit und Güte zu erwarten ist.

Kapitel 3 – **Achsenparallele Beschriftung** – beschreibt ein Verfahren, um achsenparallele Beschriftungen zu erzeugen und untersucht Laufzeit und Fehler.

Kapitel 4 – **Rotierte Beschriftung** – beschreibt ein Verfahren, um rotierte Beschriftungen zu erzeugen und untersucht Laufzeit und Erfolgswahrscheinlichkeit.

Kapitel 5 – **Implementierung und Tests** – gibt einen kurzen Hinweis, wie die Algorithmen implementiert wurden und untersucht die Algorithmen hinsichtlich Laufzeit und Güte mit realen Gebieten.

Kapitel 6 **Zusammenfassung und Ausblick** – fasst diese Arbeit zusammen und beschreibt mögliche Ansatzpunkte für weitere Algorithmen und Untersuchungen.

2 Ähnliche Arbeiten

Ein häufig untersuchtes Problem ist das Maximum-Empty-Rectangle-Problem (MER) [NSB94; DMR97]. Das Problem ist wie folgt definiert: Gegeben sei eine Menge von Punkten $P = \{p_1, \ldots, p_n\}$, wobei $p_i \in \mathbb{R}^2$. Die Punktemenge ist in einer Box eingeschlossen, die durch minimale und maximale Koordinaten aus der Punktemenge bestimmt wird. Finde das größte achsenparallele Rechteck, sodass kein Punkt im Rechteck enthalten ist und das Rechteck ist vollständig in der einschließenden Box enthalten. Eine allgemeine Variante liegt vor, wenn statt der Punktemenge eine Menge von Segmenten verwendet wird. Der Algorithmus in [NSB94] verwendet einen zweistufigen Divide-and-Conquer-Ansatz. Der Algorithmus teilt die Menge der Segmente rekursiv an einer vertikalen Trennlinie auf. Um das maximale Rechteck zu finden, werden im Merge-Schritt die Segmente erneut (horizontal) aufgeteilt. Dadurch ergibt sich eine orthokonvexe Kontur um den Schnittpunkt der beiden Linien. Entlang dieser Kontur kann in linearer Zeit das größte Rechteck mittels Matrixsuche gefunden werden. Insgesamt ergibt sich eine Laufzeit von $\mathcal{O}(n \log^2 n)$. Für einen Algorithmus, der das größte Quadrat sucht, kann hingegen eine log-lineare Laufzeit erwartet werden, da man keinen zweiten Aufteilungsschritt benötigt. Tatsächlich existiert bereits eine Methode für das Problem Largest-Empty-Square mit log-linearer Laufzeit [Hwa79; LC80].

Ein weiteres Problem ist das Largest-Empty-Corner-Rectangle-Problem (LECR). Sei die Punktemenge S durch eine vertikale Linie in zwei Teilmengen S_l, S_r geteilt. Gesucht ist das größte Rechteck, sodass der untere linke Punkt in S_l und der obere rechte Punkte in S_r ist. Mittels schneller Matrixsuche kann dieses Problem in $\mathcal{O}(n \log n)$ gelöst werden, wobei $n = |S|$ [AS87]. Ein solcher Algorithmus wird in [DMR97] verwendet, um zu zeigen, dass das größte Rechteck in einem allgemeinen Polygon mit Löchern in $\mathcal{O}(n \log^2 n)$ gefunden werden kann. Hierfür muss das Polygon in zwei y-monotone Teilstücke zerlegt werden. Diese beiden Teilstücke werden orthogonalisiert. Mittels LECR kann das größte Rechteck durch ein speziell definiertes Größenmaß gefunden werden. Somit kann in der vorliegenden Arbeit ebenfalls ein effizienter Algorithmus erwartet werden, um achsenparallele Beschriftungen zu generieren.

Das Problem von rotierten Rechtecken ist allgemein eine offene Frage [NSB94]. Die beste Methode hierfür ist ein Approximationsalgorithmus mit einer Laufzeit von $\mathcal{O}(n^3)$, wobei n die Anzahl der Segmente im Polygon ist [Mol+12]. Hierfür wird das Polygon zunächst in ein Quasi-Lattice-Polygon transformiert, d. h. für jedes Segment des Polygons gibt es nur eine begrenzte Anzahl von Orientierungen. In einem Vorverarbeitungsschritt wird für alle Paare von Punkten

Tabelle 2.1 – Übersicht von Laufzeiten der vorgestellten Algorithmen und Probleme.

Objekt	Orientierung	Laufzeit	Referenz
Quadrat	achsenparallel	$\mathcal{O}(n \log n)$	[Hwa79; LC80]
Rechteck	achsenparallel	$\mathcal{O}(n \log^2 n)$	[NSB94; DMR97]
Rechteck	achsenparallel	$\Omega(n \log n)$	[DMR97]
Rechteck	rotiert	$\mathcal{O}(n^3)$	[Mol+12]

des Polygons festgestellt, ob die Linie zwischen diesen Punkten durch das Polygon geschnitten wird. Somit kann in konstanter Zeit festgestellt werden, ob ein gegebenes Punktepaar die Seite eines Rechtecks ergeben kann. Der Algorithmus geht durch alle Punktepaare. Falls die Linie eines Punktepaares innerhalb eines Polygons liegt, wird ein dritter Punkt gesucht, sodass sich das größte Rechteck ergibt. Die Laufzeit dieses Algorithmus legt somit den Rahmen fest, was in der vorliegenden Arbeit erwartet werden kann. Es ist ein schneller oder ein exakter Algorithmus denkbar, da in dieser Arbeit ein maximales Quadrat gesucht wird, was einfacher zu lösen ist. Tabelle 2.1 fasst die vorgestellen Algorithmen und Laufzeiten zusammen.

3 Achsenparallele Beschriftung

Dieses Kapitel beschreibt eine einfache und effiziente Möglichkeit, um ein Gebiet achsenparallel zu beschriften. Abbildung 3.1 ist ein Beispiel für das Problem mit Lösung.

Der Algorithmus arbeitet wie folgt: Für ein Polygon wird die Punktemenge betrachtet, die sich aus den Segmentendpunkten ergibt. Es wird das größte leere Quadrat dieser Punktemenge bestimmt. Anschließend wird geprüft, ob das Quadrat innerhalb des Polygons liegt. Falls es nicht enthalten ist, wird der Mittelpunkt des Quadrats zu der Punktemenge hinzugefügt und der Algorithmus wiederholt. Um das größte Quadrat für ein bestimmtes Breiten- und Höhenverhältnis zu bestimmen, wird die gesamte Instanz in Richtung einer der Koordinatenachsen skaliert. Da es sich um eine Approximation des größten Quadrats bezüglich der Segmente handelt, können Ecken des Quadrats außerhalb des Polygons liegen. Deswegen wird ein Schnitttest des Quadrats mit allen Segmenten am Ende durchgeführt und das Quadrat gegebenenfalls angepasst.

Dafür wird im nächsten Abschnitt zunächst das Problem formalisiert, maximale Quadrate zu finden. Abschnitt 3.2 beschreibt einen Algorithmus, um das Problem zu lösen. Der nächste Abschnitt beschreibt, wie man diesen Algorithmus verwenden kann, um maximale Rechtecke für ein gegebenes Breiten- und Höhenverhältnis zu finden. Abschnitt 3.4 erklärt, wie Beschriftungen erzeugt werden können. Der letzte Abschnitt untersucht die Laufzeit und den Fehler.

3.1 Problem

Das Problem Maximum-Empty-Square (MES) ist wie folgt definiert: Gegeben sei eine Menge von Punkten $P = \{p_0, \ldots, p_n\}$, wobei $p_i \in \mathbb{Z}^2$. Finde das größte achsenparallele Quadrat, sodass kein Punkt im Quadrat enthalten ist. Für die Punktemenge wird eine minimale Box erzeugt, sodass alle Punkte in der Box enthalten sind. Das größte Quadrat muss ebenfalls in dieser Box enthalten sein.

3.2 Algorithmus für MES

Dieser Abschnitt beschreibt einen Algorithmus zum Lösen des Maximum-Empty-Square-Problems.

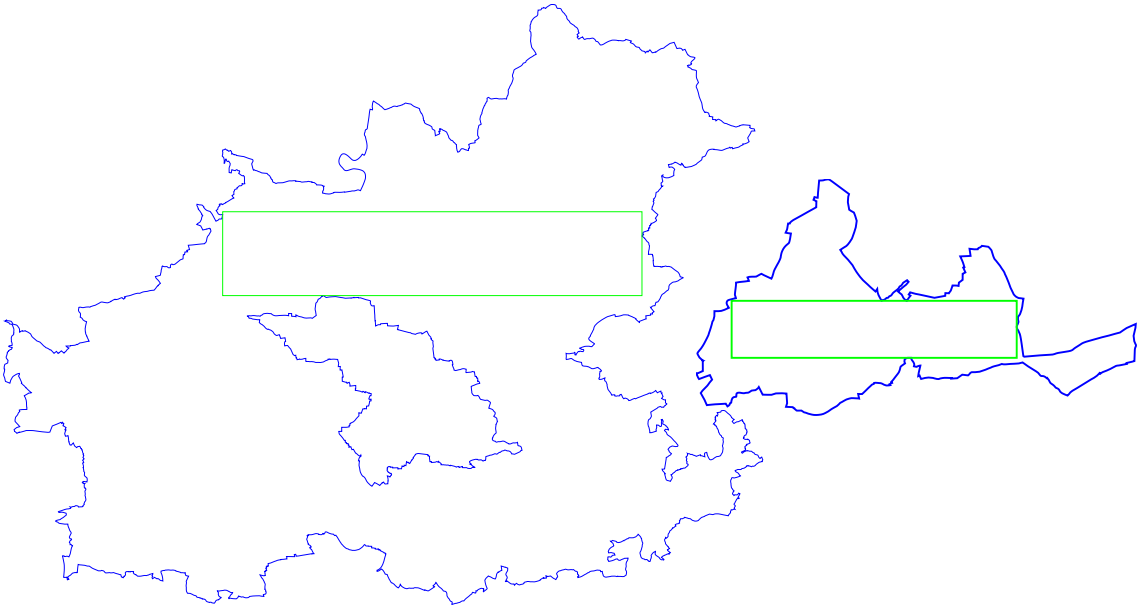

Abbildung 3.1 – Die Beispiellösungen für achsenparallel maximale Rechtecke mit einem Breiten- und Höhenverhältnis von fünf wurden mit dem Divide-and-Conquer-Algorithmus erzeugt.

3.2.1 Naiver Algorithmus

Eine erste obere Laufzeitschranke für diese Problem ist $\mathcal{O}(n^4)$, wobei n die Anzahl der Segmente oder Endpunkte der Segmente des Polygons ist. Dafür werden alle Tripel von Punkten aufgezählt, für jedes Tripel wird das größte Quadrat bestimmt und anschließend geprüft, dass kein anderer Punkt der Punktemenge im Quadrat enthalten und dass das Quadrat vollständig in der Box enthalten ist.

Das Problem ist gegeben durch drei Punkte: Finde das größte achsenparallele Quadrat, das durch diese Punkte beschränkt wird, kann wie folgt gelöst werden (siehe Abbildung 3.2). Für alle Paare von Punkten des Tripels werden zwei Ecken erzeugt. Der dritte Punkt liegt entweder in einem Bereich, der durch eine Ecke beschränkt wird oder außerhalb beider Bereiche. Im letzten Fall erzeugt dieses Tripel kein Quadrat. Ansonsten kann das größte Quadrat bestimmt werden, indem der maximale Abstand des dritten Punktes zu einer Seite der Ecke als Größe für das Quadrat genommen wird. Das maximale ist das größte Quadrat aus allen drei Möglichkeiten Paare von Punkten zu bilden.

Dieses Verfahren wird verwendet, um in einem Divide-and-Conquer-Ansatz Teilinstanzen mit konstanter Größe zu lösen.

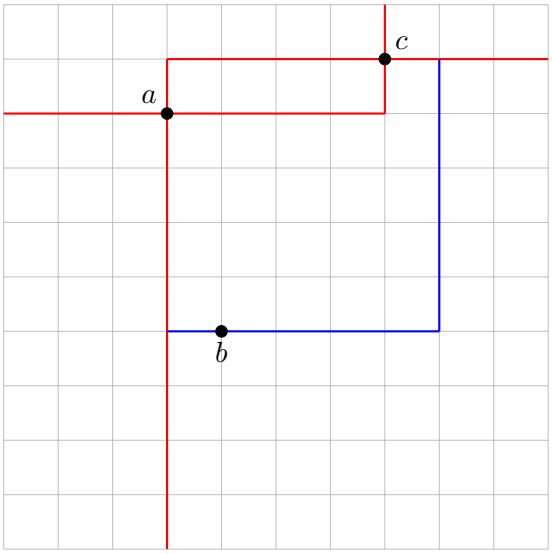

Abbildung 3.2 – Konstruktion des größten Quadrats durch drei Punkte. Die roten Geraden bilden die Ecken durch die Punkte a und c. Der Punkt b liegt in der nach unten geöffneten Ecke und bildet das Quadrat (blaue Linien).

3.2.2 Divide-and-Conquer-Algorithmus

Für den Divide-and-Conquer-Ansatz liegt die Punktemenge sortiert nach x- und y-Komponenten vor. Mit der nach der x-Komponente sortierten Liste wird der Median bestimmt, um die Punktemenge aufzuteilen. Der Median legt ebenfalls fest, an welcher Stelle die einschließende Box in zwei Boxen geteilt wird.

Der Merge-Schritt wird im Folgenden beschrieben. Gegeben sei eine Punktemenge, die bezüglich der x-Koordinate in zwei Hälften geteilt wird (siehe Abbldung 3.3). Das größte leere Quadrat liegt entweder vollständig in einer der beiden Seiten oder es liegt auf der Trennlinie l. Das bisher größte Quadrat habe die Größe s. Um das größte Quadrat auf der Trennlinie zu finden, wird eine Kontur – implizit gegeben durch eine geordnete Menge von Punkten – erzeugt, auf deren Rand eine Ecke des größten Quadrats liegt. Alle Punkte liegen sortiert bezüglich der y-Koordinate in einer Listenstruktur vor, sodass Punkte in konstanter Zeit eingefügt werden können.

3.2.2.1 Randpunkte

Für alle aufeinanderfolgenden Paare, die eine größere vertikale Distanz als das bisherige größte Quadrat haben ($s < |y_i - y_{i+1}|$), werden zwei Punkte mit derselben Höhe am äußeren Rand der einschließenden Box erzeugt und zwischen dem Paar eingefügt.

3.2.2.2 Orthogonalisierung

Im nächsten Schritt wird die Liste so erweitert, dass für jeden Punkt der nächste Punkt entweder dieselbe x- oder y-Koordinate hat (blaue Punkte in Abbildung 3.3). Dafür werden von zwei aufeinanderfolgenden Punkten $p_i = (x_i, y_i)$ und $p_{i+1} = (x_{i+1}, y_{i+1})$ der Punkt (x_i, y_{i+1}) oder (x_{i+1}, y_i) eingefügt, der von der Trennlinie horizontal weiter entfernt liegt.

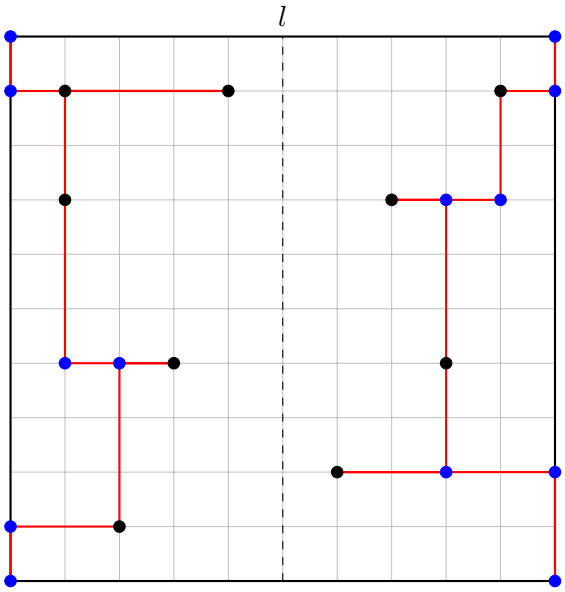

Abbildung 3.3 – Erweiterung der Punktemenge um Eckpunkte (blau). Dadurch unterscheidet sich jedes aufeinanderfolgende Paar von Punkten in höchstens einer Koordinate; es entsteht eine orthogonale Kontur.

3.2.2.3 Füllpunkte

Der nächste Schritt besteht darin, die Punkte(-menge) so zu erweitern, dass zu jedem Punkt mindestens ein Punkt existiert, der dieselbe x-Koordinate und ein Punkt der dieselbe y-Koordinate hat. Ziel ist, dass jeder Punkt auf die vertikal am weitesten entfernten Punkte auf der Kontur (oben und unten) verweist, und auf den am weitesten horizontal entfernten Punkt auf der anderen Seite. Dadurch kann die Listenstruktur durchgegangen und für jeden Punkt können vertikale und horizontale Punkte gefunden werden, die ein größtes Quadrat beschränken.

3.2.2.4 Vertikale Füllpunkte

In nächsten Schritt wird die Punktemenge so erweitert, dass zu jedem Punkt auf einer Seite ein Punkt mit derselben x-Koordinate existiert (siehe Abbildung 3.4). Für die Beschreibung

hier wird die linke Hälfte betrachtet. Die Liste von Punkten wird vorwärts (und rückwärts) durchlaufen. Für einen Punkt p_i gibt es drei Möglichkeiten, wo der nächste Punkt p_{i+1} liegt. Entweder p_{i+1} liegt auf gleicher Höhe links von p_i, rechts von p_i oder p_{i+1} liegt mit gleichem horizontalem Abstand zu l höher (oder tiefer). Durch die Erweiterung aus dem vorherigen Abschnitt ist sichergestellt, dass in den ersten beiden Fällen die Punkte auf der selben Höhe liegen und im letzten Fall ist sichergestellt, dass die Punkte dieselbe x-Koordinate haben. Liegt der nachfolgende Punkt links, bedeutet es, dass man sich auf der Kontur von der Trennlinie wegbewegt. Der Punkt p_i wird in diesem Fall auf einen Stapel gelegt. Liegt der nachfolgende Punkt rechts, werden Punkte p_s auf dem Stapel betrachtet. Ist p_s horizontal weiter von l entfernt als der nächste Punkte p_{i+1}, wird ein Punkt nach p_i und vor p_{i+1} eingefügt. Hierbei wird die y-Koordinate von p_i genommen und die x-Koordinate von p_s. Diese Schritte werden in beiden Richtungen und auf beiden Seiten durchgeführt. Danach ist sichergestellt, dass für jeden Punkt ein vertikaler Punkt vorhanden ist.

3.2.2.5 Horizontale Füllpunkte

Der nächste Schritt stellt sicher, dass für jeden Punkt auf einer Seite ein Punkt auf der anderen Seite existiert (siehe Abbildung 3.4). Wieder können mehrere Punkte möglich sein, sodass der am weitesten entfernte Punkt gewählt wird. Dafür werden beide Listen gleichzeitig durchgegangen. Seien $p_{l,i}, p_{l,i+1}$ zwei aufeinanderfolgende Punkte auf der linken und seien $p_{r,i}, p_{r,i+1}$ zwei aufeinanderfolgende Punkte auf der rechten Seite. Auf beiden Seiten wird bei gleicher y-Koordinate solange in den Listen weitergegangen, bis der am weitesten entfernte Punkt zu l gefunden wird. Falls $p_{l,i+1} < p_{r,i+1}$ wird ein Punkt auf der rechten Seite mit der y-Koordinate von $p_{l,i+1}$ eingefügt. Analog werden Punkte auf der linken Seite eingefügt. Nach diesem Schritt ist sichergestellt, dass zu jedem Punkt mindestens ein Punkt auf der anderen Seite auf gleicher Höhe liegt. Dieser Schritt beeinträchtigt nicht den vorherigen, da Punkte nur zwischen einem Punktepaar eingefügt werden, das vertikal liegt.

3.2.2.6 Verweise

Um die vertikalen Verweise zu erzeugen, wird die Liste erneut durchgegangen (siehe Abbildung 3.5). Befindet man sich bei einem horizontalen Paar und bewegt sich von der Trennlinie weg (auf der linken Seite $x_i > x_{i+1}$), wird der erste Punkt p_i auf den Stapel gelegt. Im umgekehrten Fall, wenn man sich zur Trennlinie bewegt (auf der linken Seite $x_i < x_{i+1}$), wird ein Verweis zu dem Punkt erzeugt, der auf dem Stapel liegt (und der Punkt wird vom Stapel entfernt).

Für die horizontalen Verweise werden beide Listen durchgegangen. Für eine Seite wird der äußerste Punkt einer Höhe ermittelt. Alle Punkte dieser Höhe auf der anderen Seite verweisen auf diesen Punkt.

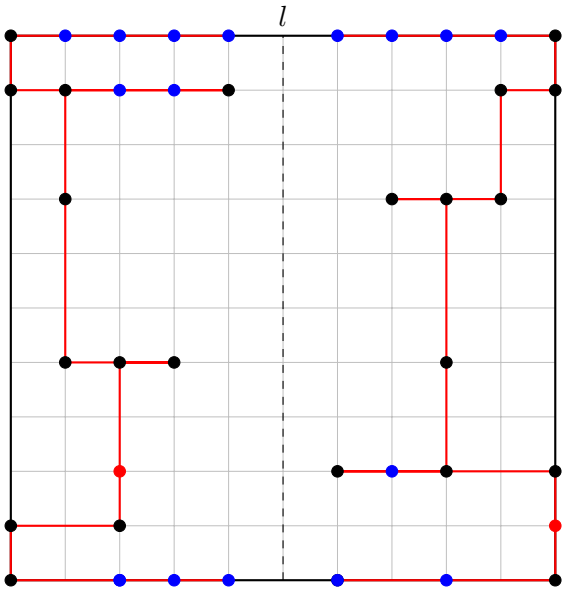

Abbildung 3.4 – Visualisierung der Listen nachdem horizontale (rot) und vertikale (blau) Füllpunkte eingefügt wurden.

Nach diesen Schritten ist sichergestellt, dass beide Listen eine orthogonale Kontur bilden (zwei aufeinanderfolgende Punkte liegen entweder horizontal oder vertikal). Für jeden Punkt in der Kontur können mittels Verweis bis zu zwei Punkte gefunden werden, die denselben horizontalen Abstand zur Trennlinie l haben und die Kontur verläuft an dieser Position näher zu l. Außerdem existiert für jeden Punkt ein Punkt auf der anderen Seite, ist mit einem Verweis verbunden und hat maximalen Abstand.

3.2.2.7 Größtes Quadrat

Das größte Quadrat kann gefunden werden, indem beide Listen durchlaufen werden. Sei p_l ein Punkt auf der linken Seite und p_r einer auf der rechten Seite. Diese Erklärung behandelt den Fall, dass p_l die untere linke Ecke eines größten Quadrats ist. Insgesamt werden die Listen viermal durchlaufen – für jede Ecke des Quadrats einmal. Für jeden Punkt p_l werden folgende Schritte durchgeführt: Zunächst wird die rechte Seite durchlaufen bis p_r auf derselben Höhe ist wie p_l und maximalen horizontalen Abstand hat. Im Beispiel in Abbildung 3.6 ist $p_l = (2, 0)$ (rot) und demnach $p_r = (10, 0)$ (rot). Von p_l wird mittels Verweis der Punkt gefunden, der maximalen Abstand nach oben hat. Dadurch wird ein horizontaler Bereich festgelegt, in dem ein Quadrat liegen kann. Im Abbildungsbeispiel ist es der Bereich $y \in [0, 4]$. Auf der linken Seite können durch die Datenstruktur keine Punkte in der Seite liegen.

Schaut man vom Punkt $p_r = (10, 0)$ mittels Verweis nach oben, wird ein Punkt gefunden, der

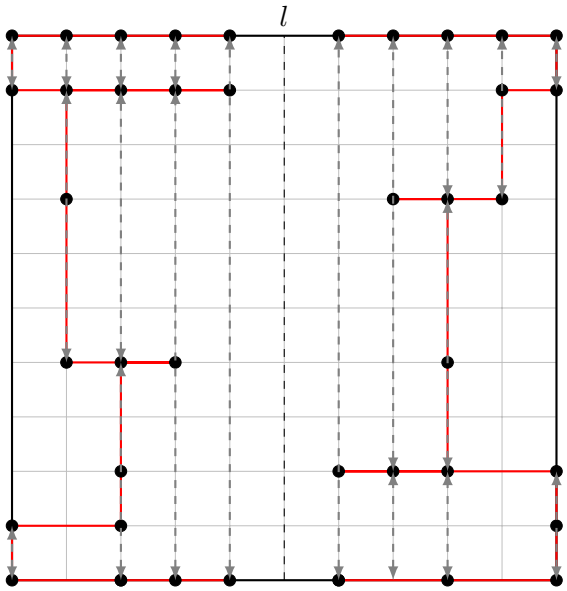

Abbildung 3.5 – Visualisierung der Listen mit vertikalen Verweisen.

einen kleinen horizontalen Bereich definiert. Die Kontur läuft demnach in das Quadrat hinein. In diesem Fall wird die rechte Seite durchlaufen, bis ein Bereich gefunden wird, der mindestens so groß ist wie der linke Bereich. Im Beispiel ist das bei $p_r = (6, 2)$ (blau) der Fall. Somit sind beide Seiten für das Quadrat bestimmt. Die Höhe ergibt sich aus dem Punkt p_l und dem Minimum der horizontalen Bereiche auf beiden Seiten.

3.3 Skalierung

Um ein maximales Rechteck mit gegebenem Breiten- und Höhenverhältnis zu bestimmen, werden alle Punkte in einem Vorverarbeitungsschritt durch die Skalierungsmatrix $S := \text{diag}(s_x, s_y)$ transformiert, wobei $s_x \neq 0, s_y \neq 0$ das Breiten- und Höhenverhältnis ist. Das größte Rechteck mit gegebenem Breiten- und Höhenverhältnis ist dann das mit S^{-1} transformierte größte Quadrat.

3.4 Beschriftung mittels MES

Ein maximales Quadrat innerhalb des Polygons kann mittels MES folgendermaßen approximiert werden: Zunächst wird das maximale Quadrat bezüglich der Punktemenge gesucht. Für ein gefundenes maximales Quadrat wird geprüft, ob dieses im Polygon enthalten ist. Hierfür wird das Even-Odd-Verfahren verwendet [HA01]. Falls es nicht enthalten ist, wird der Mittelpunkt des

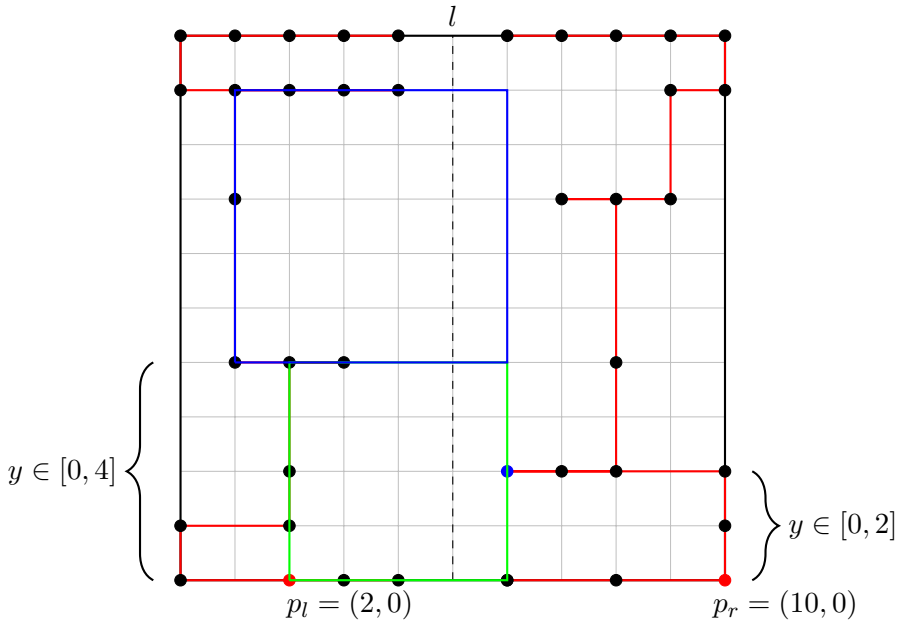

Abbildung 3.6 – Finden des größten Quadrats. Das grüne Quadrat ist das maximale Quadrat für $p_l = (2, 0)$. Ein maximales Quadrat ist das blaue Quadrat.

Quadrats zu der Punktemenge hinzugefügt und der Algorithmus wiederholt, bis das gefundene Quadrat im Polygon liegt. Dies ist in Algorithmus 1 angegeben. Hierbei ist $\mathcal{P}$ das Polygon und P die Punktemenge der Segmentendpunkte. Nach der Skalierung in Zeile 3 wird das größte Quadrat mit dem oben beschriebenen Vefahren bestimmt (Funktion LargestSquare). Die Funktion Adapt stellt sicher, dass das Quadrat vollständig im Polygon $\mathcal{P}$ liegt und MidPoint bestimmt den Mittelpunkt des Quadrats. Diese Schritte werden wiederholt, bis ein Quadrat gefunden wurde, dessen Mittelpunkt im Polygon liegt. Damit liegt insbesondere auch das Quadrat im Polygon. Der Algorithmus gibt das zurückskalierte Quadrat zurück, welches ein Rechteck mit dem festgelegten Breiten- und Höhenverhältnis ist.

Damit das Quadrat tatsächlich vollständig innerhalb des Polygons liegt, wird es mit allen Segmenten geschnitten und gegebenenfalls angepasst. Für jedes Segment werden folgende Schritte ausgeführt (siehe Abbildung 3.7): Es werden die Diagonalen des Quadrats mit dem Segment geschnitten. Der Schnittpunkt aus Diagonale und Segment ergibt bis zu vier kleinere Quadrate, die im gefundenen Quadrat liegen. Das größte Quadrat davon wird als Angepasstes verwendet.

Die Laufzeit, um ein Label zu erzeugen, ist demnach $\mathcal{O}(k \cdot ((n+k) \log(n+k))$ (für die log-lineare Laufzeit von MES siehe Abschnitt 3.5), wobei k die Anzahl der maximalen äußeren Quadrate ist und n die Anzahl der Segmente des Polygons. Die Laufzeit der Schleife (Zeile 4 bis 9) in Algorithmus 1 wird von LargestSquare dominiert.

Algorithmus 1 Algorithmus für achsenparallele Beschriftung

```
1:  function LargestSquare(P, S)
2:      P := PointSet(P)
3:      P := Scale(P, S)
4:      repeat
5:          s := LargestSquare(P)
6:          s := Adapt(P, s)
7:          p := MidPoint(s)
8:          P := P ∪ {p}
9:      until IsInside(p, OuterPolygon(P))
10:     return Scale(s, S^{-1})
11: end function
```

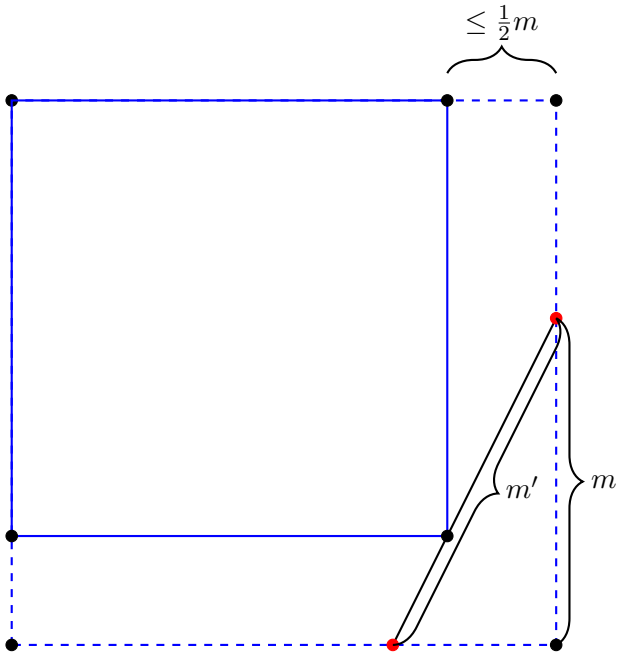

Abbildung 3.7 – Das Quadrat wird angepasst, wenn es von einem Segment geschnitten wird. Die Größe des angepassten Quadrats ist dabei abhängig vom maximalen Abstand m' zweier aufeinanderfolgender Punkte im Polygon.

3.5 Laufzeit und Fehler

Der Algorithmus hat eine Laufzeit von $\mathcal{O}(n \log n)$, wobei n die Anzahl der Punkte ist. Dies ergibt sich aus dem Mastertheorem [Cor+09]. Das Sortieren der Punkte bezüglich x- und y-Komponenten benötigt $\mathcal{O}(n \log n)$ Schritte und wird zuerst ausgeführt. Sei $T(n)$ die Zeit, um das größte Quadrat zu finden. In jedem Rekursionsschritt wird die Punktemenge in zwei

Teilprobleme halbiert. Das Finden des Medians und Aufteilen der Punktemenge erfolgt in Linearzeit.

Das Zusammenführen von zwei Teillösungen benötigt $\mathcal{O}(n)$ Schritte. Der Merge-Schritt besteht aus mehreren Teilschritten: Im ersten Schritt werden zwei Punkte am Rand der Box eingefügt. Hierbei können zwischen jedem Paar höchstens zwei Punkte hinzugefügt werden. Somit kann sich die Anzahl der Punkte verdreifachen. Beim Durchlaufen der Liste, um die Eckpunkte einzufügen, kann höchstens zwischen jedem Paar ein Punkt eingefügt werden. Insgesamt kann sich die Menge der Punkte demnach höchstens verdoppeln. Genauso verdoppelt das Einfügen der vertikalen Punkte für die Verweise höchstens die Punktemenge, da höchstens alle Punkte der Liste auf den Stapel gelegt werden können und somit auch höchstens so viele Punkte hinzugefügt werden. Beim Einfügen der horizontalen Punkte kann sich die Punktemenge ebenfalls maximal verdoppeln, wenn für jeden Punkt auf der einen Seite ein Punkt auf der anderen Seite dazukommt. Die Größe der aufgebauten Datenstruktur ist somit $\mathcal{O}(n)$.

Das Durchlaufen der Datenstruktur, um das größte Quadrat zu finden, benötigt höchstens linear viele Schritte. Hierbei sind auf jeder Seite eine Position in der jeweiligen Liste gespeichert. Eine Position bestimmt eine Ecke des Quadrats, während die andere verwendet wird, um nach einer Position in der anderen Kontur zu suchen, sodass kein Punkt im Quadrat liegt. Für eine feste Position kann somit mehrere Schritte auf der anderen Seite weitergegangen werden. Insgesamt wird die Datenstruktur aber einmal durchlaufen. Somit benötigt der Merge-Schritt linear viele Schritte. Damit ist $T(n) = 2 \cdot T(n/2) + f(n)$, wobei die Funktion f die Schritte zum Aufteilen der Punktemenge, zum Erzeugen der Datenstruktur und zur Suche des Quadrats beschreibt.

Im Allgemeinen kann der Fehler beliebig groß sein, der sich durch das Anpassen mit den Segmenten ergibt. Eine solche Instanz kann einfach gefunden werden, indem man Segmente horizontal platziert, wobei die Endpunkte auf den Rändern der Box liegen. Das größte leere Quadrat der Punktemenge hat die Größe der Box und das größte leere Quadrat der Segmentmenge den vertikalen Abstand zweier Segmente.

Für das Labeling wird angenommen, dass die Distanz von zwei aufeinanderfolgenden Punkten im Polygon kleiner als die Seitenlänge des größten Quadrats ist. Dadurch schneiden Segmente das Quadrat nur an den Ecken. Die folgenden Überlegungen sind in Abbildung 3.7 illustriert. Das Quadrat wird von einem Segment geschnitten. Der maximale Abstand von einer Ecke des Quadrats zu einem Schnittpunkt ist $m \leq m'$, wobei $m' := \max_{p_i \in P}\{\|\overrightarrow{p_i p_{i+1}}\|_2\}$ der maximale euklidische Abstand von zwei aufeinanderfolgenden Punkten ist. Die Seitenlänge des an beiden Seiten angepassten Quadrats ist mindestens $s' \geq s - 2 \cdot 0.5 \cdot m$, wobei s die Seitenlänge des ursprünglichen Quadrats ist. Somit ist das Verhältnis von angepasstem Quadrat zu maximalem Quadrat:

$$\frac{s'^2}{s^2} \geq \frac{(s-m)^2}{s^2} =: r$$

Ein maximales Quadrat in einem Polygon ist kleiner als das MES der Punktemenge $s_p \leq s$. Somit ist $s'^2 \geq r s_p^2$. Für kleine maximale Segmentlängen (oder große s') wird der Abstand $m \leq m'$ ebenfalls klein und somit weicht die Größe des gefundenen Quadrats s' wenig von s ab.

4 Rotierte Beschriftung

Dieses Kapitel beschreibt einen randomisierten Algorithmus, um das größte *rotierte* Rechteck mit gegebenem Breiten- und Höhenverhältnis in einem allgemeinen Polygon zu finden. Abbildung 4.1 ist ein Beispiel für ein Problem mit Lösung.

In Kapitel 3 konnte das Problem so reduziert werden, dass das größte Quadrat gesucht wird, indem die Instanz skaliert wird. Diese Idee wird – mit geringfügig mehr Aufwand – auch in diesem Fall angewendet. Also wird im Folgenden beschrieben, wie das größte rotierte Quadrat gefunden wird.

Der Algorithmus arbeitet wie folgt: Es werden zufällig gleichverteilt drei Segmente des Polygons gewählt. Mit diesen Segmenten wird das größte Quadrat konstruiert, das durch die Segmente begrenzt wird. Im nächsten Schritt wird für alle Segmente des Polygons geprüft, ob sie im Quadrat liegen oder es schneiden. In einem solchen Fall wird das Quadrat angepasst. Der Hauptschritt ist demnach aus drei beliebigen Segmenten das größte Quadrat zu konstruieren.

Dafür werden im nächsten Abschnitt Grundlagen beschrieben. Abschnitt 4.2 erklärt, wie ein maximales Quadrat aus drei Segmenten konstruiert wird. Der nächste Abschnitt erläutert, wie die Skalierung eingesetzt werden kann, um das größte Rechtecke zu erhalten. Abschnitt 4.4 beschreibt den gesamten Algorithmus und im letzten Abschnitt wird die Laufzeit und Erfolgswahrscheinlichkeit analysiert.

4.1 Grundlagen

Dieser Abschnitt beschreibt grundlegende Konzepte, die die Konstruktion maximaler Quadrate im nächsten Abschnitt vereinfachen.

Definition: Ein Segment ist definiert wie in Abschnitt 1.1. Die Gerade eines Segments (s, t) ist die Gerade, die durch die Punkte s und t geht.

Definition: Der Abstand zweier Segmente s_1, s_2 sei $d := \min_{p \in s_1, q \in s_2}(\|\vec{pq}\|_2)$, wobei $\| \cdot \|_2$ die euklidische Norm bezeichnet.

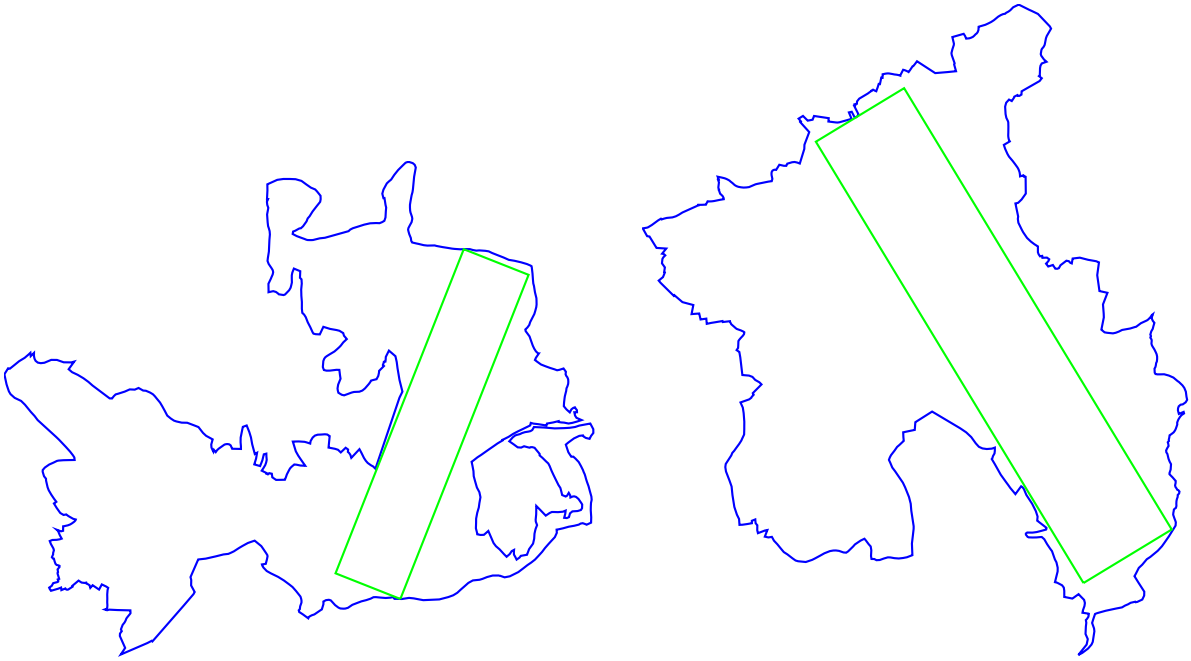

Abbildung 4.1 – Die Beispiellösungen für rotierte maximale Rechtecke mit einem Breiten- und Höhenverhältnis von fünf wurden durch $k = n^2$ viele Proben mit dem randomisierten Algorithmus erzeugt.

Definition: Ein Segment (s_b, t_b) liegt auf beiden Seiten eines anderen Segmentes, wenn der Orientierungstest für s_b und t_b unterschiedliche Seiten feststellt. Ein Segment liegt auf einer Seite, wenn der Orientierungstest für beide Punkte dieselbe Seite feststellt.

Ein Quadrat kann auf zwei Arten durch ein Segment beschränkt werden:

1. Eine flexible Begrenzung liegt vor, wenn eine Ecke des Quadrats auf einem Segment liegt und das Segment nicht parallel zu einer Seite des Quadrats ist.

2. Eine fixe Begrenzung liegt vor, wenn ein Endpunkt eines Segments auf einer Seite des Quadrats liegt oder wenn ein Segment parallel zu einer Seite des Quadrats ist und es mindestens zwei gemeinsame Punkte von einer Seite des Quadrats und dem Segment gibt.

Ein maximales rotiertes Quadrat wird durch höchstens zwei flexible Restriktionen begrenzt. Sei ein maximales Quadrat gegeben, das durch drei flexible Kontakte begrenzt wird. Es gibt zwei diagonal gegenüberliegende Kontakte. Aufgrund der flexiblen Begrenzung können die Kontakte verschoben werden. Dabei kann ein Quadrat gebildet werden, weil die Winkel an den Kontaktstellen zwischen Segmenten und Quadratseiten echt größer Null sind (siehe Abbildung 4.2).

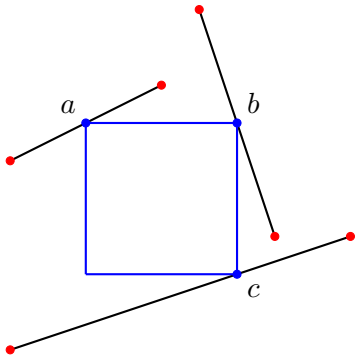

Abbildung 4.2 – Ein Quadrat, das durch drei flexible Kontakte begrenzt wird. Ein größeres Quadrat ergibt sich, indem Kontakt a nach links, gleichzeitig b nach oben und c nach rechts verschoben wird.

Ein maximales rotiertes Quadrat, das durch drei Segmente begrenzt wird, kann demnach durch drei fixe, zwei fixe und eine flexible oder eine fixe und zwei flexible Kontakte begrenzt sein. Abbildung 4.3 illustriert diese Fälle (für nicht notwendigerweise maximale Quadrate). Für fixe Kontakte wird nur ein begrenzender Punkt gezeichnet. Ein Algorithmus, der das maximale Quadrat bestimmt, ist korrekt, wenn er alle Fälle abdeckt und für jeden Fall das maximale Quadrat bestimmt.

4.2 Konstruktion maximaler Quadrate

Dieser Abschnitt beschreibt, wie ein maximal großes Quadrat konstruiert wird, wenn drei nicht-überschneidende Segmente gegeben sind. Ein Quadrat ist nur dann maximal, wenn es alle drei Segmente berührt. Basierend auf den obigen Überlegungen können drei Hauptfälle unterschieden werden: Das größte Quadrat wird durch die Endpunkte von zwei gegenüberliegenden Segmenten begrenzt (siehe Abbildung 4.4); die Geraden der Segmente bilden ein Dreieck und das größte Quadrat innerhalb des Dreiecks berührt alle Segmente (siehe Abbildung 4.5); eine Seite des größten Quadrats ist parallel zu einem Segment (siehe Abbildungen 4.6, 4.7, 4.8).

Tabelle 4.1 zeigt, welche Fälle durch die unterschiedlichen Konstruktionen abgedeckt werden.

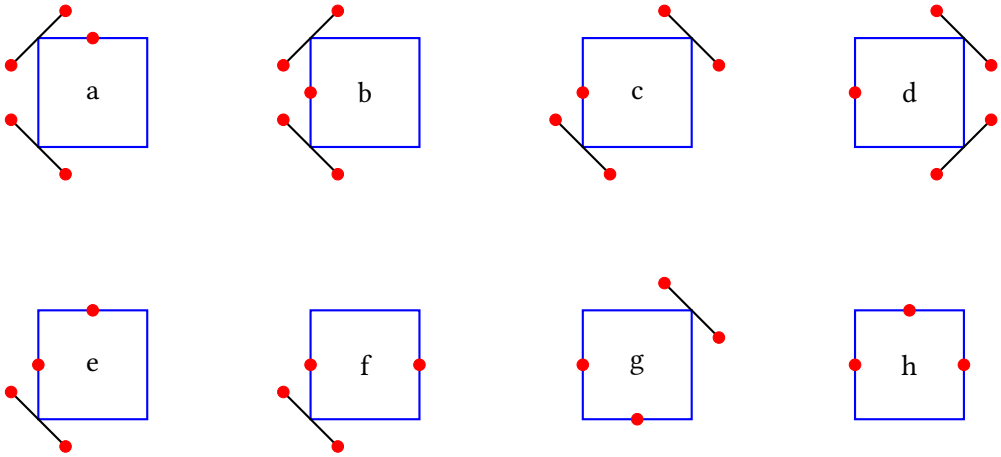

Abbildung 4.3 – Fälle von Begrenzungen von Quadraten: a, b c und d sind Fälle, die durch eine fixe und zwei flexible Begrenzungen gebildet werden, e bis einschließlich g sind Fälle, die durch zwei fixe und eine flexible Begrenzung gebildet werden. h ist das Quadrat, das durch drei fixe Begrenzungen gebildet wird. Diese Abbildung zeigt nicht alle Kombinationen; symmetrische Fälle sind nicht dargestellt.

Tabelle 4.1 – Fälle, durch die Konstruktionen abgedeckt werden.

Fall	Abgedeckte Fälle
4.2.1	h, f
4.2.2	d
4.2.3.1	b
4.2.3.2	a-e,g
4.2.3.3	a-e,g

4.2.1 Zwei fixe Begrenzungen

Der erste Fall in Abbildung 4.4 wird gelöst, indem für alle Paare von Segmenten das Paar mit minimalem Abstand gewählt wird. Die Strecke d zwischen den minimalen Endpunkten legt Größe und Rotation des Quadrats fest. An den zwei Endpunkten der Hilfsstrecke d werden zwei Hilfsgeraden c_1 und c_2 rechtwinklig zu d erzeugt. Mit diesen werden vier Fälle unterschieden. Das Segment m liegt zwischen c_1 und c_2, dann begrenzt einer der Endpunkte von m das Quadrat. Es werden beide Hilfsgeraden geschnitten, dann liegt eine flexible Begrenzung vor. Es wird nur eine Hilfsgerade geschnitten, dann kann eine flexible oder fixe Begrenzung vorliegen. Oder Segment m liegt außerhalb von beiden Hilfsgeraden. Dann wird kein Quadrat erzeugt. Am Ende wird geprüft, ob das Quadrat durch l oder n geschnitten wird. Im positiven Fall wird kein Quadrat zurückgegeben. Diese Konstruktion deckt die Fälle f und h ab.

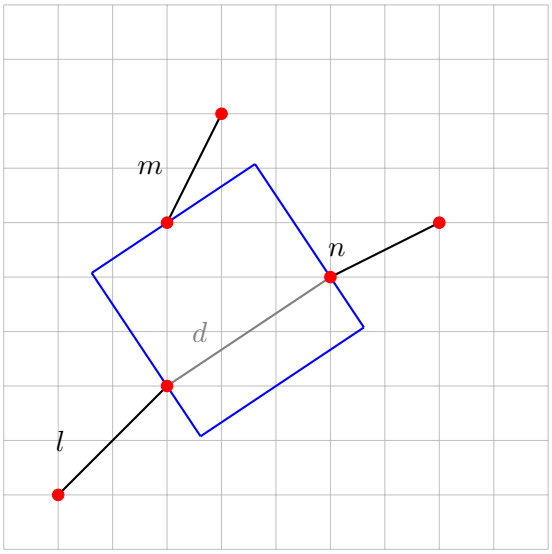

Abbildung 4.4 – Das größte Quadrat, das durch die Endpunkte von drei unterschiedlichen Segmenten begrenzt wird. Die Rotation und Größe ergibt sich aus dem Paar von Segmenten, dessen minimaler Abstand von zwei Endpunkten aus unterschiedlichen Segmenten maximal ist.

4.2.2 Die Segmentlinien ergeben ein Dreieck

Bilden die Linien der Segmente ein Dreieck, wird das größte Quadrat innerhalb dieses Dreiecks konstruiert (siehe Abbildung 4.5). Falls das größte Quadrat auch alle Segmente berührt, kann es übernommen werden. Das größte Quadrat wird mit dem Verfahren aus [LW] konstruiert und deckt Fall d ab.

4.2.3 Eine Quadratseite liegt auf einem Segment

Der letzte Fall lässt sich weiter durch die Lage der Segmente unterteilen. Allen Segmenten werden feste Bezeichnungen zugewiesen. Das Segment l ist das Segment, auf dem eine Seite des Quadrats liegt. Die Lage der übrigen Segmente m und n wird in Bezug zu l bestimmt. Durch die Konstruktionen in den Abschnitten 4.2.3.2 und 4.2.3.3 werden einige Fälle mehrfach abgedeckt. Dies ist notwendig, da die Fälle in Abbildung 4.3 durch die Kontaktstellen unterschieden werden und sich die Konstruktionen in diesem Abschnitt durch die Lage der Segmente unterscheiden.

4.2.3.1 Zwei Segmente liegen auf beiden Seiten

Für ein festes Segment l können die anderen Segmente auf beiden Seiten von l liegen (siehe Abbildung 4.6). Das größte Quadrat wird konstruiert, indem die Schnittpunkte i_m von l und m

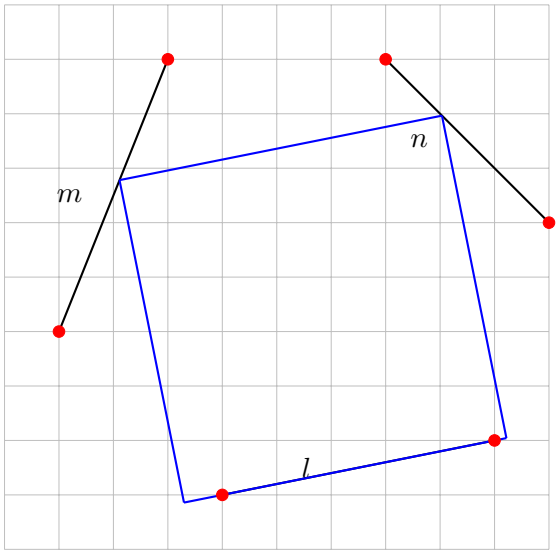

Abbildung 4.5 – Die Linien der Segmente l, m und n bilden ein Dreieck. Das größte Quadrat ist jenes, das in das Dreieck eingezeichnet werden kann und welches alle Segmente berührt.

sowie i_n von l und n als vorläufige Ecken des Quadrats genommen werden. Je nach Orientierung von m und n können Endpunkte innerhalb des bisherigen Quadrats liegen. Liegen Endpunkte eines Segments innerhalb des Quadrats, wird das Quadrat an dieser Seite verkleinert. Das maximale Quadrat aus diesem Fall ist das, welches auf beiden Seiten von l so konstruiert wird. Diese Konstruktion deckt Fall b ab.

4.2.3.2 Ein Segment liegt auf beiden Seiten

Im nächsten Fall liegt Segment m auf beiden Seiten von l. Segment n liegt auf einer Seite (siehe Abbildung 4.7). Es wird der Schnittpunkt von l und m berechnet. Dieser bildet eine vorläufige Ecke des Quadrats. Analog zu oben wird sichergestellt, dass das Segment m die obere Seite des Quadrats nicht schneidet. Ansonsten wird die Seite parallel so verschoben, dass sie durch den Endpunkt von m geht.

Das größte Quadrat wird entweder durch einen Endpunkt des Segments auf zwei verbleibenden Seiten begrenzt oder durch einen flexiblen Kontakt an einer der drei unbestimmten Ecken (rechte Seite in Abbildung 4.7). Diese Konstruktion deckt die Fälle a bis e und g ab.

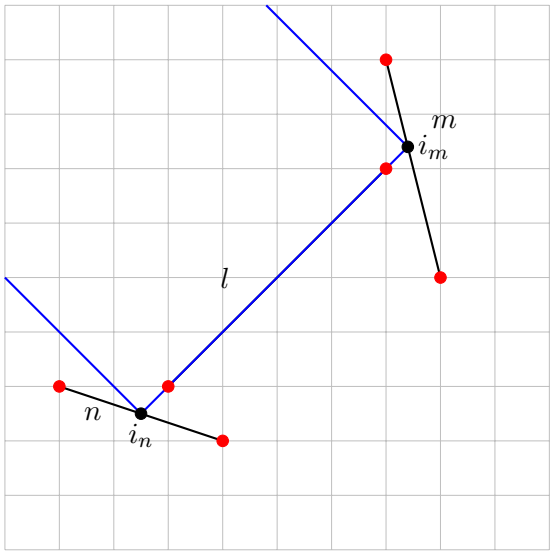

Abbildung 4.6 – Die Segmente m und n werden von der Geraden durch l geschnitten. Das maximale Quadrat wird gefunden, indem auf beiden Seiten von l das größte Quadrat konstruiert wird. Dafür wird der Schnittpunkt der Linie l mit den Segmenten m und n berechnet. Diese bilden potentielle Eckpunkte des Quadrats.

4.2.3.3 Beide Segmente liegen auf einer Seite

Liegen beide Segmente m und n auf derselben Seite bezüglich l, wird eine Ecke des Quadrats aus l und m erzeugt (siehe Abbildung 4.8). Hierbei werden mit beiden Endpunkten von m die Hilfslinien h_s, h_t konstruiert. Liegt das Segment n zwischen h_s und h_t kann kein Quadrat erzeugt werden. Ansonsten wird mittels Orientierungstest festgestellt, auf welchen Seiten der Hilfslinien ein Quadrat erzeugt werden kann.

Das maximale rotierte Quadrat für drei gegebene Segmente ist das maximale Quadrat über alle Fälle 4.2.1 bis 4.2.3.3. Da in den Fällen 4.2.2, 4.2.3.2 und 4.2.3.3 unterschiedliche Fälle mit unterschiedlich benannten Segmenten gleichzeitig vorliegen können, werden Quadrate mit allen Permutationen von Segmentbezeichnungen konstruiert und das Maximum übernommen.

4.3 Skalierung

Im Gegensatz zum achsenparallelen Fall muss die Rotation des Quadrats bekannt sein, um die Instanz zu skalieren. Abbildung 4.9 zeigt wichtige Transformationsschritte. Im ersten Bild sind die Segmente im ursprünglichen Koordinatensystem abgebildet. Um das größte skalierte Quadrat (Rechteck) zu finden, wird der Winkel zwischen einer Seite des Quadrats und der

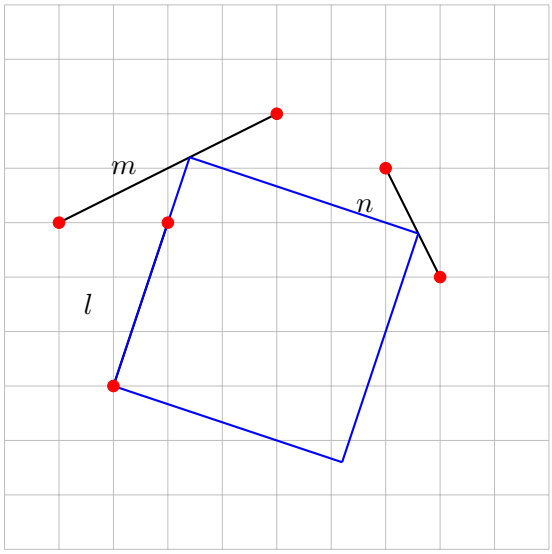

Abbildung 4.7 – Das Segment m wird von der Geraden durch l geschnitten. Das maximale Quadrat wird gefunden, indem der Schnittpunkt der Linie durch l und dem Segment m als Ecke verwendet wird. In dieser Konstellation sind zwei Fälle zu untersuchen: Entweder das größte Quadrat wird durch einen Endpunkt des Segments n begrenzt oder eine Ecke des Quadrats liegt auf dem Segment n.

x-Achse bestimmt. Der Winkel ergibt sich je nach Fall entweder aus dem Segment l oder aus der Hilfslinie d. Damit wird die Rotationsmatrix R bestimmt. Die Translationsmatrix T wird erzeugt, indem ein beliebiger invertierter Endpunkt aus d oder l gewählt wird. Die Skalierungsmatrix S ist analog zum achsenparallelen Fall. Alle drei Segmente werden mittels $T^{-1}R^{-1}SRT$ transformiert (2. Bild in Abbildung 4.9), was einer Skalierung parallel zu einer Seite des Quadrats entspricht. In diesem Koordinatensystem wird das größte Quadrat konstruiert. Das größte skalierte Quadrat (Rechteck mit festen Breiten- und Höhenverhältnis) ist das mit $(T^{-1}R^{-1}SRT)^{-1}$ transformierte größte Quadrat (3. Bild in Abbildung 4.9). Diese Transformationsdaten werden zu jedem gefundenen Quadrat gespeichert.

4.4 Algorithmus

Der randomisierte Algorithmus verwendet das oben beschriebene Verfahren und findet maximale Rechtecke in einem Polygon (siehe Algorithmus 2). Hierbei ist $\mathcal{P}$ das Multi-Polygon, k die Anzahl der Wiederholungen und a das Breiten- und Höhenverhältnis. Die Funktion Uar wählt zufällig gleichverteilt ein Segment aus dem Polygon, LargestSquare berechnet das maximale Quadrat und Adapt schneidet alle Segmente mit dem gefundenen Quadrat. Die Transformationswerte sind mit dem Quadrat gespeichert, sodass in Adapt zu prüfende Seg-

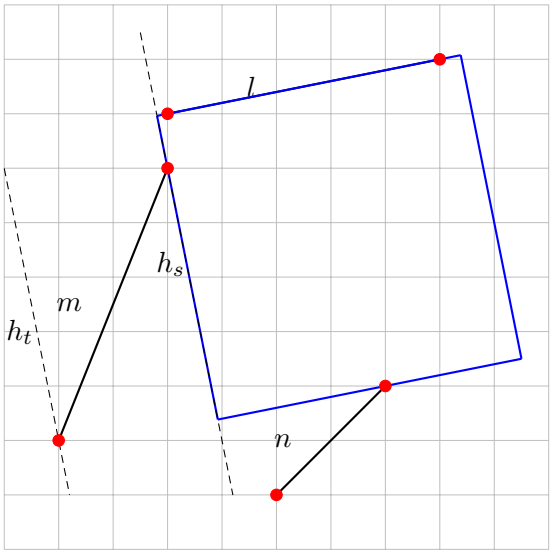

Abbildung 4.8 – Beide Segmente m und n liegen auf derselben Seite bezüglich l.

mente ebenfalls transformiert werden können. Diese Funktion behandelt zusätzlich den Fall, dass Segmentendpunkte innerhalb des Quadrats liegen. Die Funktion UNSCALE gibt schließlich das größte Rechteck zurück.

Algorithmus 2 Algorithmus für rotierte Beschriftung

```
1:  function LargestSquare(P, k, a)
2:      s_m := 0
3:      for i := 1, ..., k do
4:          l := Uar(P)
5:          m := Uar(P)
6:          n := Uar(P)
7:          s := LargestSquare(l, m, n, a)
8:          if s > s_m then
9:              s := adapt(P, s)
10:             if s > s_m ∧ Inside(s, P) then
11:                 s_m := s
12:             end if
13:         end if
14:     end for
15:     return Unscale(s_m)
16: end function
```

4.5 Laufzeit und Erfolgswahrscheinlichkeit

Eine Probe des Algorithmus hat eine Laufzeitkomplexität von $\mathcal{O}(n)$, wobei n die Anzahl der Segmente des Polygons ist. Wählt man drei Segmente zufällig gleichverteilt und konstruiert daraus das größte Quadrat, benötigt man höchstens eine konstante Anzahl von Schritten. Im nächsten Schritt wird für alle Segmente der Instanz geprüft, dass sie nicht im Quadrat liegen oder es schneiden. Hierfür wird jedes Segment des Polygons mit allen vier Segmenten des Quadrats geschnitten und für alle Endpunkte von Polygonsegmenten geprüft, ob sie im Quadrat liegen. Die Laufzeit des Punkt-im-Polygon-Test ist ebenfalls linear. Damit ergibt sich die Gesamtlaufzeit für eine Probe.

Die Erfolgswahrscheinlichkeit, für eine Probe das größte Quadrat zu finden, ist mindestens $p \geq 1/\binom{n}{3}$. Ein maximales Quadrat wird durch mindestens ein Tripel von Segmenten definiert. Da man zufällig gleichverteilt drei Segmente wählt, ergibt sich die Erfolgswahrscheinlichkeit.

Mithilfe der Binomialverteilung kann die Anzahl der Versuche bestimmt werden, sodass erwartet das größte Quadrat einmal gefunden wird. Sei $X \sim B(k, p)$ binomialverteilt, dann ist für $E(X) = 1$ die Anzahl der benötigten Versuche $k = \binom{n}{3}$. Mit der Laufzeit für eine Probe ist die Laufzeit, um erwartet das größte Quadrat zu finden, demnach $\mathcal{O}(n^4)$.

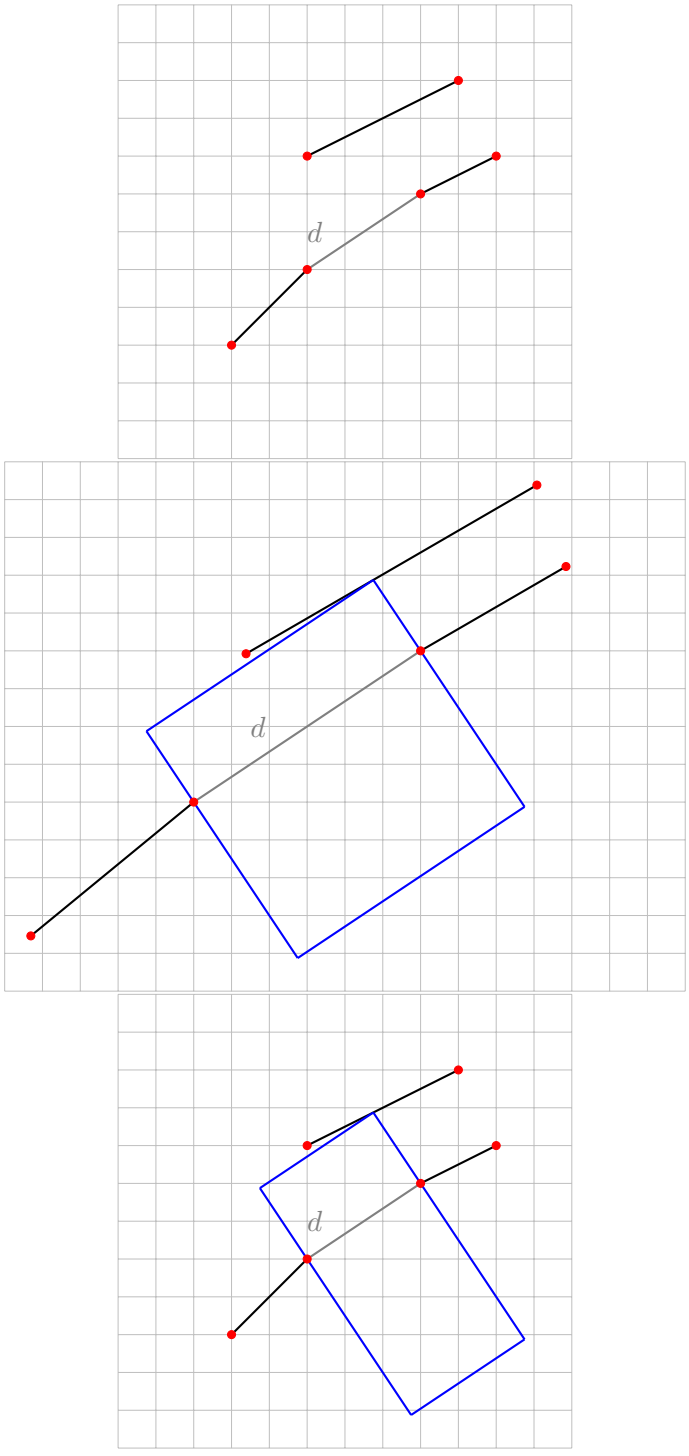

Abbildung 4.9 – Um das größte Rechteck mit einem Breiten- und Höhenverhältnis von $2:1$ zu erhalten, werden die Segmente parallel zu den Seiten des Quadrats skaliert (in diesem Fall ist das die Linie d). Im transformierten Koordinatensystem wird das größte Quadrat konstruiert (2. Bild). Das größte Rechteck ist das zurücktransformierte Quadrat im ursprünglichen Koordinatensystem (3. Bild).

5 Implementierung und Tests

Beide Algorithmen wurden in der Programmiersprache C++ implementiert, ohne Multi-Threading einzusetzen. Der Algorithmus zum Lösen der achsenparallelen Rechtecke wurde hauptsächlich in einem prozeduralen Stil programmiert. Für die Datenstruktur wurde auf Elemente der Standard-Template-Library (STL) zurückgegriffen, um die Listenstruktur mit Verweisen zu implementieren (Liste, Vektor und Stapel). Für den Algorithmus zum Lösen der rotierten Rechtecke wurde eine Vektorklasse implementiert. Durch das Operator-Überladen sind typische vektorielle Operationen leicht umsetzbar. Ebenso wurde eine Segmentklasse implementiert, wobei auch Operationen, die zur Linie eines Segments gehören, dieser Klasse zugeordnet sind. Bei grundlegenden geometrischen Operationen (Orientierung von Punkt und Gerade, Punkt auf Linie, Schnitt von zwei Linien) wurden die mathematischen Ausdrücke direkt verwendet, ohne besonderen Wert auf numerische Stabilität zu legen.

Die folgenden Tests wurden auf einem Rechner mit Intel i5-3320M Prozessor mit $2{,}2\,\mathrm{GHz}$ Taktfrequenz und $4\,\mathrm{GB}$ RAM ausgeführt. Der C++-Code wurde mit Clang 3.7.0 und der dritten Optimierungsstufe auf einem Linux-System kompiliert und ausgeführt. Die Testdaten waren eine Menge von 669 Gebieten des Landes Baden-Württemberg. In der Menge sind sowohl Gebiete mit mehreren getrennten Polygonzügen, als auch Multi-Polygone (mit Löchern) enthalten. Für den randomisierten Algorithmus wurde eine Teilmenge der kleinsten 159 Gebiete verwendet.

Für den Algorithmus für achsenparallele Rechtecke ist es interessant, wie viele Wiederholungen benötigt werden, bis das größte Rechteck innerhalb des Polygons gefunden wurde. Die Verteilung der Wiederholungen ist in Abbildung 5.1 dargestellt. Es zeigt sich, dass in 84% aller Fälle das größte Rechteck im ersten Durchlauf gefunden wird. Acht Wiederholungen sind demgegenüber in nur zwei Fällen nötig. Da die Laufzeit nicht nur von der Eingabelänge abhängt ist, es zusätzlich interessant, wie schnell der Algorithmus bei verschiedenen realen Daten ist. Abbildung 5.2 zeigt die Laufzeit des Algorithmus für achsenparallele Rechtecke und für verschiedene Instanzgrößen. Hierbei werden 96% der Instanzen in weniger als einer Sekunde gelöst und 93% in weniger als einer halben Sekunde. Insgesamt handelt es sich demnach um einen praktikablen ersten Ansatz, um achsenparallele Labels zu erzeugen.

Für den randomisieren Algorithmus wurde die Laufzeit einer Teilmenge der Gebiete gemessen. Abbildung 5.3 zeigt die Laufzeit des randomisierten Algorithmus. Hierbei wurden $k = n^3$ viele Proben erzeugt. Bereits für Instanzen mit einer Größe von 133 Segmenten benötigt der Algorithmus $4{,}5\,\mathrm{min}$. Die Größe der achsenparallelen Rechtecke ist eine untere Schranke

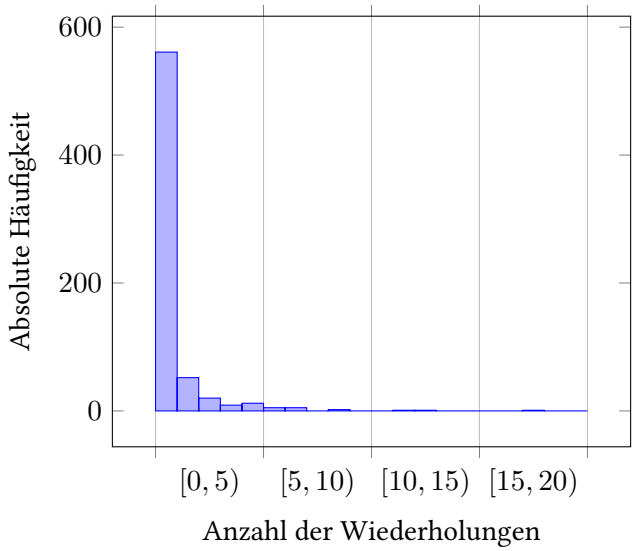

Abbildung 5.1 – Absolute Häufigkeit der notwendigen Wiederholungen des Algorithmus für achsenparallele Rechtecke bis ein Rechteck innerhalb des Polygons gefunden wird.

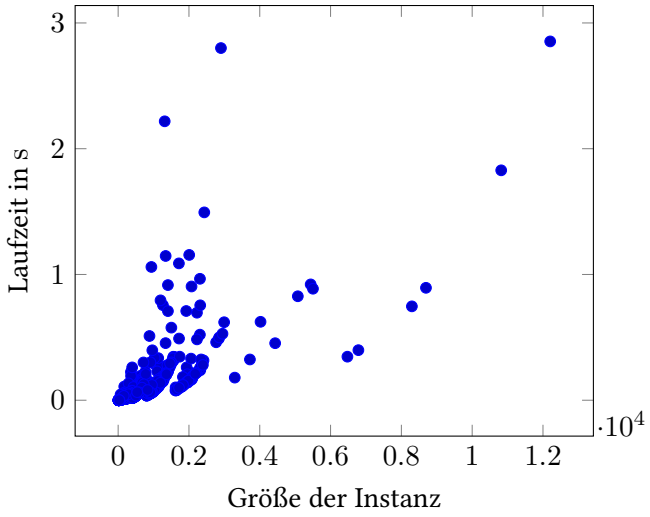

Abbildung 5.2 – Laufzeit des Algorithmus für achsenparallele Rechtecke mit unterschiedlichen Instanzgrößen.

für die Größe im rotierten Fall. Da es sich um einen randomisierten Algorithmus handelt, wurde untersucht, um viel die rotierten Rechtecke größer sind als die parallelen Rechtecke. Die Verteilung der Verhältnisse von rotiertem Rechteck zu Parallelem sind in Abbildung 5.4 illustriert. In 14% der Fälle waren die Rechtecke kleiner als das achsenparallele Rechteck. Der überwiegende Teil ergibt somit größere Rechtecke. In den meisten Fällen sind rotierte Rechtecke

um einen Faktor bis zwei größer. Aufgrund der schnell steigenden Laufzeit lässt sich dieser Ansatz nicht ohne Modifikationen praktisch einsetzen.

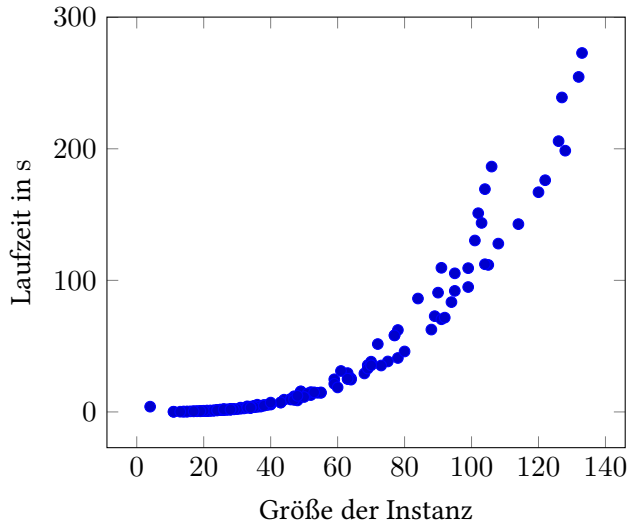

Abbildung 5.3 – Laufzeit des randomisierten Algorithmus für unterschiedliche Instanzgrößen.

Eine weitere Möglichkeit die Größe der Labels zu untersuchen ist in [Bar01] beschrieben. Dort werden gebogene Beschriftungen erzeugt. Die in der vorliegenden Arbeiten durchgeführten Tests verwenden stattdessen die Fläche des Rechtecks. Als Bewertungsmaß wurde die Wurzel aus dem Verhältnis von Rechtecks- zu Polygonfläche $\sqrt{\mathcal{A}(r)/\mathcal{A}(\mathcal{P})}$ gewählt. Durch die Wurzel erhöhen sich die Variationen zwischen kleinen Werten, sodass Beschriftungen, die wenig Fläche abdecken, besser untersucht werden können. Für die Beschriftungen wurde ein horizontaler Skalierungsfaktor von vier gewählt. Abbildung 5.5 zeigt die Ergebnisse für achsenparallele Boxen. Der häufigste Wert liegt hierbei im Intervall $[0,2;0,3)$. Somit sind in den meisten Fällen 4% bis 9% der Fläche durch eine Box bedeckt.

Abbildung 5.6 zeigt die Ergebnisse für rotierte Boxen. In diesem Fall liegt der häufigste Wert im Intervall $[0,5;0,6)$. Diese deutlich bessere Abdeckung der Polygonflächen lässt sich auch durch die wesentlich kleineren Instanzen erklären.

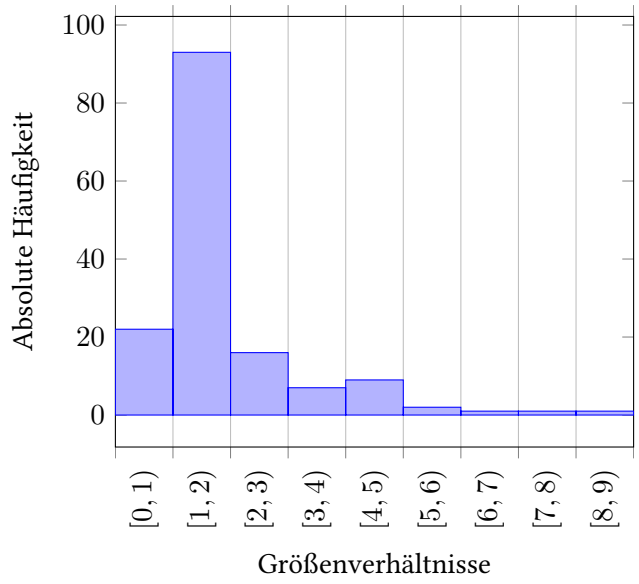

Abbildung 5.4 – Absolute Häufigkeit der Größenverhältnisse $r = s_r/s_p$ (s_r ist die Größe des rotierten Rechtecks und s_p ist die Größe des achsenparallelen Rechtecks). Für den randomisierten Algorithmus wurden $k = n^3$ Proben genommen.

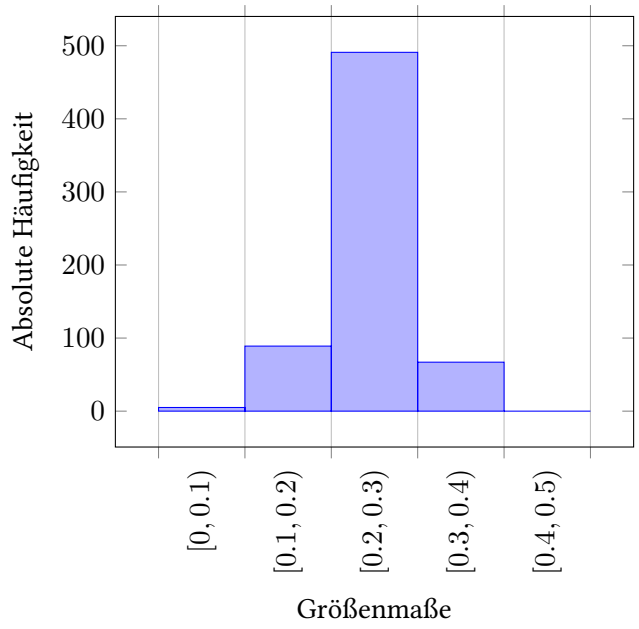

Abbildung 5.5 – Größenmaß $\sqrt{\mathcal{A}(r)/\mathcal{A}(\mathcal{P})}$ von achsenparallelen Rechtecken, wobei r die Fläche des Rechtecks ist und $\mathcal{P}$ die Fläche des Polygons. Beim Test wurden Rechtecke mit einem Breiten- und Höhenverhältnis von $4:1$ erzeugt.

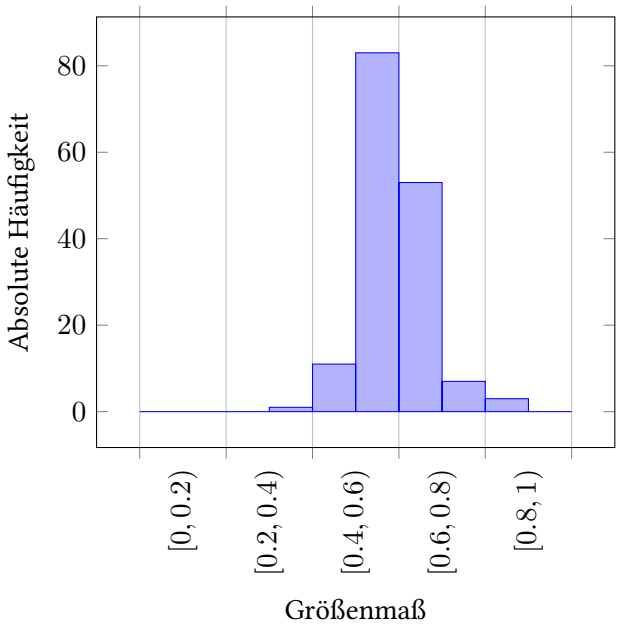

Abbildung 5.6 – Größenmaß $\sqrt{\mathcal{A}(r)/\mathcal{A}(\mathcal{P})}$ von rotierten Rechtecken, wobei r die Fläche des Rechtecks ist und $\mathcal{P}$ die Fläche des Polygons. Beim Test wurden Rechtecke mit einem Breiten- und Höhenverhältnis von $4:1$ erzeugt.

6 Zusammenfassung und Ausblick

Dieses Kapitel fasst die vorgestellten Verfahren sowie deren Ergebnisse zusammen (Abschnitt 6.1) und gibt einen Ausblick, wie die vorgestellten Ansätze weiterentwickelt werden können (Abschnitt 6.2).

6.1 Zusammenfassung

In dieser Arbeit wurden zwei Algorithmen entwickelt, um in einem allgemeinen Polygon mit Löchern das größte Rechteck zu finden, wenn das Breiten- und Höhenverhältnis bekannt ist. Der erste Algorithmus sucht das größte achsenparallele Rechteck; beim zweiten Ansatz kann das Rechteck beliebig rotiert sein. Durch das feste Breiten- und Höhenverhältnis muss die Position und Größe (und die Rotation) bestimmt werden. Durch eine Skalierung der Eingabe konnten die Instanzen so skaliert werden, dass nur das größte Quadrat gesucht werden muss.

Im ersten Szenario wurde das größte Rechteck bestimmt, indem das größte leere Quadrat in der skalierten Punktemenge bestimmt wurde, die sich aus den Segmentendpunkten des Polygons ergab. Vergleichbare Arbeiten zeigen, dass dieses Problem effizient gelöst werden kann. Hierfür wurde ein effizienter Divide-and-Conquer-Algorithmus entwickelt. Lag das gefundene Quadrat nicht im Polygon, wurde der Mittelpunkt zur Punktemenge hinzugefügt und der Algorithmus wiederholt. Für ein gefundenes Quadrat wurde geprüft, das es nicht von Segmenten geschnitten wird. Tests mit realen Gebieten zeigten, dass in den meisten Fällen keine Wiederholung des Algorithmus nötig war.

Im Szenario für rotierte Rechtecke wurde ein randomisierte Ansatz entwickelt. Dabei wurden zufällig gleichverteilt drei Segmente des Rechtecks gewählt und daraus das größte Quadrat konstruiert. Das größte Rechteck mit gegebenem Breiten- und Höhenverhältnis konnte ebenfalls durch Skalierung der Segmente ermittelt werden. Tests mit realen Gebieten zeigten, dass deutlich größere rotierte Rechtecke gefunden wurden, wobei eine deutlich höhere Laufzeit benötigt wurde.

6.2 Ausblick

Als Ausblick werden eine Reihe von Ansätzen beschrieben, wie sich die bisherigen Ergebnisse weiterentwickeln lassen. Ein wichtiger Punkt ist, den Divide-and-Conquer-Algorithmus so weiterzuentwickeln, dass das größte Quadrat bezüglich einer Menge von Segmenten gefunden wird. Dabei ist an zwei Stellen deutlich mehr Aufwand zu erwarten:

1. Im Fall von drei Segmenten muss das größte achsenparallele beschränkte Quadrat konstruiert werden.
2. Im Merge-Schritt besteht die Kontur nicht nur aus orthogonalen Segmenten. Die Datenstruktur, um das größte Quadrat zu finden, enthält Verweise auf Segmente.

Ein Vorteil dieses Algorithmus ist, dass ein Polygon so erweitert werden kann, dass immer das größte Quadrat innerhalb des Polygons gefunden wird. Die Instanz kann durch Segmente so aufgefüllt werden, dass kein Quadrat außerhalb gefunden wird (durch Segmente, die vom Polygon zur einschließenden Box gehen). Dadurch können Wiederholungen problemlos vermieden werden.

Es bedarf weiterer Entwicklungen für rotierte Beschriftungen. Interessant sind hierbei weitere eigenständige Ansätze mit möglicherweise geringerer Laufzeit. Es ergeben sich allerdings auch Möglichkeiten, bestehende Algorithmen zu verwenden. So kann die Instanz zunächst ausgedünnt oder Platzierungen ausgeschlossen werden. Dadurch können existierende Verfahren auch bei größeren Instanzen eingesetzt werden. Alternativ kann ein effizienter Algorithmus für achsenparallele Rechtecke eingesetzt werden, um in rotierten Instanzen ein größtes Rechteck zu finden.

Literaturverzeichnis

[AS87] A. Aggarwal und S. Suri. „Fast Algorithms for Computing the Largest Empty Rectangle“. In: *Proceedings of the Third Annual Symposium on Computational Geometry*. 1987, S. 278–290 (Zitiert auf Seite 5).

[Bar01] M. Barrault. „A methodology for placement and evaluation of area map labels“. In: *Computers, Environment and Urban Systems* 25.1 (2001), S. 33–52 (Zitiert auf Seiten 1, 33).

[Cor+09] Thomas H. Cormen, Clifford Stein, Ronald L. Rivest und Charles E. Leiserson. *Introduction to Algorithms*. London, 2009 (Zitiert auf Seite 15).

[DMR97] Karen Daniels, Victor Milenkovic und Dan Roth. „Finding the largest area axis-parallel rectangle in a polygon“. In: *Computational Geometry* 7.1–2 (1997), S. 125–148 (Zitiert auf Seiten 1, 5 f.).

[HA01] Kai Hormann und Alexander Agathos. „The point in polygon problem for arbitrary polygons“. In: *Computational Geometry* 20.3 (2001), S. 131–144 (Zitiert auf Seite 13).

[Hwa79] F. K. Hwang. „An O(N Log N) Algorithm for Rectilinear Minimal Spanning Trees“. In: *Journal of the ACM* 26.2 (1979), S. 177–182 (Zitiert auf Seiten 5 f.).

[Imh75] Eduard Imhof. „Positioning names on maps“. In: *The American Cartographer* 2.2 (1975), S. 128–144 (Zitiert auf Seite 1).

[Kna+12] Christian Knauer, Lena Schlipf, Jens M. Schmidt und Hans Raj Tiwary. „Largest Inscribed Rectangles in Convex Polygons“. In: *Journal of Discrete Algorithms* 13 (2012), S. 78–85 (Zitiert auf Seite 1).

[LC80] Der-Tsai Lee und Wong CK. „Voronoi Diagrams in $L_1 - (L_\infty)$ Metrics with 2-Dimensional Storage Applications“. In: *SIAM Journal on Computing* 9.1 (1980), S. 200–211 (Zitiert auf Seiten 5 f.).

[LW] Floor van Lamoen und Eric W. Weisstein. *Triangle Square Inscribing*. `http://mathworld.wolfram.com/TriangleSquareInscribing.html`. (Besucht am 19. 10. 2015) (Zitiert auf Seite 23).

[Mol+12] Rubén Molano, Pablo G. Rodríguez, Andrés Caro und M. Luisa Durán. „Finding the largest area rectangle of arbitrary orientation in a closed contour“. In: *Applied Mathematics and Computation* 218.19 (2012), S. 9866–9874 (Zitiert auf Seiten 1, 5 f.).

[MS91] Joe Marks und Stuart Shieber. *The Computational Complexity of Cartographic Label Placement*. Techn. Ber. Harvard University, 1991 (Zitiert auf Seite 1).

[NSB94] Subhas C. Nandy, Arani Sinha und Bhargab B. Bhattacharya. „Location of the largest empty rectangle among arbitrary obstacles“. In: *Foundation of Software Technology and Theoretical Computer Science*. 1994, S. 159–170 (Zitiert auf Seiten 1, 5 f.).

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Ort, Datum, Unterschrift