Institut für Parallele und Verteilte Systeme
Abteilung Anwendersoftware
Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Diplomarbeit Nr. 2847

Preprocessing von Positionsdaten mit Algorithmen zur Polygonzugvereinfachung

Steffen Reimann

Studiengang:	Informatik
Prüfer:	Prof. Dr. Bernhard Mitschang
Betreuer:	Dipl.-Inf. Nicola Hönle
begonnen am:	1. November 2008
beendet am:	30. April 2009
CR-Klassifikation:	H.2.8, C.2.4, F.2.2, G.1.2

Kurzfassung

Mobile Objekte, die mit Positionssensoren ausgerüstet sind, liefern kontinuierliche Positionsdatenströme. Dabei entstehen große Datenmengen, die entweder direkt verarbeitet oder für die spätere Auswertung gespeichert werden. Um den Verarbeitungsaufwand zu reduzieren und Speicherplatz zu sparen ist es notwendig, die Positionsdatenströme zu komprimieren. Dadurch sollen redundante Daten herausgefiltert und die Datenmenge insgesamt verkleinert werden.

Von Kaya wurde in [Kay05] gezeigt, dass sich Positionsdatenströme effizient komprimieren lassen. Auf diesen Ergebnissen aufbauend zeigt diese Arbeit, wie Positionsdatenströme mithilfe von Algorithmen zur Polygonzugvereinfachung komprimiert werden können, ohne eine vorgegebene Abweichung gegenüber dem ursprünglichen Datenstrom zu überschreiten. Dazu werden verschiedene Algorithmen zur Polygonzugvereinfachung untersucht und mit den Algorithmen von Kaya verglichen.

Außerdem wird ein Verfahren vorgestellt, das den Positionsdatenstrom mithilfe einer digitalen Straßenkarte komprimiert. Dazu müssen die Positionen mit der Straßenkarte abgeglichen und zusammengefasst werden.

Untersucht wurde in beiden Fällen insbesondere die Datenreduktion und die experimentelle Laufzeit der Algorithmen. Dazu wurden Simulationen auf echten Positionsdaten durchgeführt. Die Simulationsdaten stammen aus dem Infati-Projekt von Prof. Jensen u. a. von der Universität Aalborg (Dänemark) [Jen+04].

Die Simulationsergebnisse zeigen, dass Algorithmen zur Polygonzugvereinfachung die verwendeten echten Positionsdatenströme um bis zu 97 % komprimieren können, ohne dabei mehr als 50 Meter von den ursprünglichen Datenströmen abzuweichen.

Interessanterweise unterscheidet sich die experimentelle Laufzeit der Algorithmen trotz gleicher Kompressionsrate um mehrere Größenordnungen. Das hat erhebliche Auswirkungen auf die praktische Nutzbarkeit der Verfahren.

Das kartenbasierte Verfahren erreicht im Vergleich mit den Verfahren zur Polygonzugvereinfachung eine niedrigere Kompressionsrate, die hauptsächlich vom Aufbau der Straßenkarte abhängt.

Inhaltsverzeichnis

Kurzfassung **3**

1. Einleitung **7**
1.1. Positionsbestimmung bei mobilen Objekten 7
1.2. Positionsdatenströme 8
1.3. Kompression von Datenströmen 8
1.4. Der Sonderforschungsbereich NEXUS 9
1.5. Bearbeitete Problemstellung 10
1.6. Struktur der Arbeit 10

2. Vorverarbeitung durch einen Operator **11**
2.1. Das NEXUS Distributed Stream Processing Framework 11
2.2. Architektur eines Operators 11
2.3. Anpassung der bestehenden Pufferkomponente 13

3. Definition der Eigenschaften der Algorithmen **15**
3.1. Maximal erlaubter Fehler 15
3.2. Zeitkomplexität 17
3.3. Arbeitsweise 18
3.4. Alterungsstabilität 20
3.5. Fehlerbeschränktheit bei mehrfacher Vereinfachung 20
3.6. Beschränkung des maximalen Fehlers bei Abfragen auf vereinfachten Daten . 21

4. Eigenschaften der bereits untersuchten Algorithmen **23**
4.1. Richtung und Geschwindigkeit 23
4.2. Lineare Extrapolation 25
4.3. Erster und letzter Punkt 25
4.4. Lineare Regression 27
4.5. Distanzbasiert 29

5. Polygonzugvereinfachung **31**
5.1. Polygonzüge 31
5.2. Verwandte Ansätze im Bereich der Polygonzugvereinfachung 32

5.3. Anwendbarkeit auf räumlich-zeitliche Polygonzüge 34

6. Ausgewählte Algorithmen zur Polygonzugvereinfachung 35
6.1. Verwerfend-strombasiertes Verfahren: PLAZA 35
6.2. Verfeinernd-fensterbasiertes Verfahren von Douglas und Peucker 41
6.3. Dezimierend-fensterbasiertes Verfahren von Boxer, Leu und Chen 45

7. Kartenbasierte Algorithmen 53
7.1. Kartenabgleich 53
7.2. Kompression der abgeglichenen Positionen 54
7.3. Verwandte Ansätze im Bereich der kartenbasierten Algorithmen 54
7.4. Angepasster kartenbasierter Algorithmus von Jensen und Čivilis 57
7.5. Vorhersage der Bewegungsrichtung mithilfe der Karte 60

8. Simulation 63
8.1. Verwendete Datensätze 63
8.2. Simulationsparameter Polygonzugvereinfachung 65
8.3. Simulationsparameter beim kartenbasierten Algorithmus 67

9. Ergebnisse 69
9.1. Algorithmen zur Polygonzugvereinfachung 69
9.2. Kartenbasierter Algorithmus 83

10. Zusammenfassung und Ausblick 87
10.1. Algorithmen zur Polygonzugvereinfachung 87
10.2. Kartenbasierter Algorithmus 88

Danksagung 91

A. Koordinatentransformation der INFATI-Daten 93
A.1. Vorgehensweise 93
A.2. Umwandlung der INFATI-Pfade in (x,y,t)-Koordinaten 93
A.3. Umwandlung der UTM-Koordinaten in geodätische Koordinaten 94

B. Koordinatentransformation innerhalb des Operators 97
B.1. Gründe für die Umwandlung in ein metrisches System 97
B.2. Umwandlung der Koordinaten 97

C. Details zu den verwendeten Daten 99
C.1. Geschwindigkeitsverteilung der INFATI-Datensätze 99
C.2. Kartenauschnitt aus OpenStreetMaps 108

Kapitel 1

Einleitung

In diesem Kapitel wird zuerst auf die Positionsbestimmung bei mobilen Objekten eingegangen. Danach werden Kompressionsmöglichkeiten für Positionsdatenströme analysiert und es wird auf die in dieser Arbeit betrachtete Problemstellung eingegangen.

1.1. Positionsbestimmung bei mobilen Objekten

Zur Bestimmung der Position eines mobilen Objekts (Auto, Fußgänger mit Mobiltelefon, etc.) lassen sich verschiedene Verfahren einsetzen:

Die erste Möglichkeit ist die *Koppelnavigation* (engl. dead reckoning), bei der die Positionsbestimmung nur durch das mobile Objekt erfolgt. Dabei werden, ausgehend von einer bekannten Position, nachfolgende Positionen anhand der Bewegungsrichtung und der Geschwindigkeit des mobilen Objekts bestimmt. Diese Art der Positionsbestimmung hat eine hohe Messungenauigkeit, da sich Fehler immer weiter aufaddieren.

Zur Koppelnavigation benötigt das mobile Objekt Sensoren zur Bestimmung der Bewegungsrichtung und der Geschwindigkeit. Diese Art der Positionsbestimmung ist daher hauptsächlich in Fahrzeugen (PKWs und LKWs) anzutreffen, da diese bereits über einen Sensor zur Geschwindigkeitsmessung verfügen, bei Fußgängern ist die Geschwindigkeitsbestimmung ohne satellitengestützte Systeme deutlich schwieriger.

Als zweite Möglichkeit können *terrestrische Positionsbestimmungssysteme* verwendet werden. Diese Verfahren berechnen die aktuelle Position eines Empfängers mithilfe der Signallaufzeiten unterschiedlicher terrestrischer Sendeanlagen. Dazu können beispielsweise die Signale von Mobilfunkmasten genutzt werden.

Im Nahbereich oder in Gebäuden kann die Position mit Ultraschall- oder Infrarotsystemen bestimmt werden, da hier die Positionsbestimmung mithilfe von Mobilfunksystemen häufig fehlerhaft oder sogar unbrauchbar ist.

Als dritte Möglichkeit bieten sich *satellitengestützte Positionierungssysteme* wie Galileo[1], NAVSTAR-GPS[2] (meist nur GPS genannt), Compass[3], IRNSS[4], QZSS[5] oder GLONASS[6] an [Gre02]. Sie können teilweise weltweit zur Bestimmung der eigenen Position eingesetzt werden. Von den genannten System ist das GPS bisher als einziges im Einsatz. Es kann Positionen mit einer Abweichung zwischen 10 und 30 Metern bestimmen. Das Galileo-System befindet sich momentan noch im Aufbau und die anderen Systeme sind entweder nur in den jeweiligen Ländern verfügbar oder noch nicht vollständig nutzbar.

An dieser Stelle wird auf Hofmann-Wellenhof und Lichtenegger verwiesen, die in [HWL08] ausführlich auf die technischen Details globaler satellitengestützter Positionierungssysteme eingehen. Insbesondere werden dort die mathematischen Grundlagen und die Eigenheiten der oben genannten Systeme detailliert behandelt.

1.2. Positionsdatenströme

Mobile Objekte, die mit Positionssensoren ausgestattet sind, liefern einen Strom von Positionsdaten. Die Datenmenge steigt dabei proportional zur Aktualisierungsfrequenz. Für hohe Aktualisierungsfrequenzen ist deshalb sowohl für die Echtzeit-Verarbeitung als auch für die Speicherung ein großer Rechenaufwand nötig. Da die mobilen Objekte Daten häufig in einer nicht benötigten Genauigkeit liefern, ist es möglich, diese vor der Weiterverarbeitung zu komprimieren.

Durch die Kompression kann sowohl die zu speichernde Datenmenge, als auch die Anzahl der zu sendenden Positionsaktualisierungsnachrichten eines mobilen Objekts reduziert werden.

1.3. Kompression von Datenströmen

Bei der Kompression von Datenströmen unterscheidet man zwischen syntaktischen und semantischen Kompressionsverfahren [Kay05]:

1.3.1. Syntaktische Kompression

Syntaktische Kompressionsalgorithmen verarbeiten einen Datenstrom ausschließlich binär. Da die Art und der Aufbau des Datenstroms zur Kompression nicht betrachtet werden ist

[1] Galileo: europäisches Satellitennavigationssystem
[2] NAVSTAR-GPS: Navigational Satellite Timing and Ranging – Global Positioning System, us-amerikanisches Satellitennavigationssystem
[3] Compass: chinesisches Satellitennavigationssystem
[4] IRNSS: Indian Regional Navigation Satellite System, indisches Satellitennavigationssystem
[5] QZSS: Quasi-Zenit-Satelliten-System, japanisches Satellitennavigationssystem
[6] GLONASS: Globalnaja Nawigazionnaja Sputnikowaja Sistema, russisches Satellitennavigationssystem

der Informationsgehalt eines syntaktisch komprimierten Datenstroms nicht sofort ersichtlich. Erst nach der Dekomprimierung ist der Inhalt des Datenstroms wieder verfügbar. Außerdem können syntaktische Kompressionsalgorithmen nicht verlustbehaftet komprimieren, da sie den Datenstrom dadurch unbrauchbar machen würden.

1.3.2. Semantische Kompression

Semantische Kompressionsalgorithmen verwenden Informationen über den Inhalt eines Datenstroms. Dadurch ist es möglich verlustbehaftete Kompression durchzuführen, ohne dabei den Datenstrom unbrauchbar zu machen. Bekannte verlustbehaftete semantische Kompressionsverfahren sind z. B. die Bildkompression des JPEG-Formats und die Kompression von Tonaufnahmen mithilfe des MP3-Formats.

Semantisch Kompressionsalgorithmen erreichen ein deutlich höheres Kompressionsverhältnis, als syntaktische Verfahren. Einige verlustbehaftete semantische Kompressionsverfahren garantieren zudem eine vorgegebene maximale Abweichung vom ursprünglichen Datenstrom.

1.4. Der Sonderforschungsbereich NEXUS

Die vorliegende Arbeit wurde im Rahmen des Sonderforschungsbereichs 627 „NEXUS – Umgebungsmodelle für mobile kontextbezogene Systeme“ [Nex09; Hön+08; Cip+09] der Universität Stuttgart erstellt.

Ziel von NEXUS ist es, die reale Welt und digitale Informationsräume (das WWW, digitale Bibliotheken, usw.) miteinander zu verknüpfen. Das dadurch entstehende, angereicherte Umgebungsmodell soll sowohl Beschreibungen realer Objekte, als auch vollständig virtuelle Objekte enthalten [Rot08].

Dieses angereicherte Umgebungsmodell dient als Basis für kontextbezogene Anwendungen. Kontextbezogene Anwendungen nutzen die Umgebung, in der sich ein mobiles Objekt oder ein Benutzer befindet, um ihm gezielt Informationen anzuzeigen oder spezielle Funktionalität zur Verfügung zu stellen.

In vielen kontextbezogenen Anwendungen spielen der Ort und die Zeit eine große Rolle. Durch Kenntnis der aktuellen Position eines Benutzers innerhalb einer Stadt kann beispielsweise ein digitaler Reiseführer Sehenswürdigkeiten in der Umgebung des Benutzers anzeigen. Kennt eine Anwendung das aktuelle Datum und die Uhrzeit, kann sie dem Benutzer aktuelle Fahrpläne anzeigen oder ihn auf Veranstaltungen hinweisen.

Der Benutzer erhält damit Informationen, die in seiner aktuellen Umgebung bzw. dem aktuellen Kontext von Nutzen sind.

1.5. Bearbeitete Problemstellung

Die Position eines mobilen Objekts wird im Allgemeinen in zeitlich-räumlichen (x,y,t)-Koordinaten angegeben. Die Höhe als dritte Raumdimension wird dabei vernachlässigt. Verbindet man die einzelnen Positionen eines Positionsdatenstroms durch Geradenabschnitte, entsteht ein Polygonzug.

Nachdem von Kaya in [Kay05] gezeigt wurde, dass sich Positionsdatenströme effizient komprimieren lassen, wird in dieser Arbeit untersucht, ob auch Algorithmen zur Polygonzugvereinfachung zur Kompression verwendet werden können. Dazu werden die Polygonzugvereinfachungsalgorithmen „PLAZA“, „Douglas-Peucker“ und „Boxer, Leu und Chen“ analysiert und mit den von Kaya untersuchten Vereinfachungsalgorithmen „Distanzbasiert“, „Erster und letzter Punkt“, „Lineare Extrapolation“ und “Lineare Regression“ verglichen.

Die Algorithmen zur Polygonzugvereinfachung werden außerdem mit einem kartenbasierten Algorithmus verglichen. Dieser erreicht die Kompression dadurch, dass er die Positionen der mobilen Objekte mit einer Karte abgleicht und zusammenfasst.

Zur Bestimmung der experimentellen Laufzeit und der Kompressionsrate werden für die Simulation die Positionsdaten des Infati-Projekts verwendet [Jen+04]. Da es sich dabei um echte GPS-Logs handelt werden sie generierten Datensätzen vorgezogen.

Die entwickelte Komponente zur Kompression der Positionsdatenströme wird für das Nexus-Projekt als Operator implementiert, um sie im Rahmen des „Nexus Distributed Stream Processing Framework“ nutzen zu können.

1.6. Struktur der Arbeit

In Kapitel 2 wird auf das in Nexus verwendete Operatorenkonzept eingegangen.

Die Kapitel 3 und 4 definieren wichtige Eigenschaften der Algorithmen und fassen die bereits von Kaya in [Kay05] untersuchten Algorithmen zusammen.

In den Kapiteln 5 und 6 werden Verfahren zur Polygonzugvereinfachung diskutiert, themenverwandte Quellen erläutert und die in dieser Arbeit untersuchten Algorithmen analysiert.

Das Kapitel 7 beschreibt kartenbasierte Verfahren, sowie den ausgewählten Algorithmus und dessen Eigenschaften.

Abschließend werden in den Kapiteln 8 und 9 die Simulationsergebnisse vorgestellt und ausgewertet. Die Arbeit schließt mit einer Zusammenfassung in Kapitel 10.

Aufwendige Koordinatenumrechungen und eine Analyse der verwendeten Infati-Datensätze finden sich in den Anhängen A bis C ab Seite 93.

Kapitel 2

Vorverarbeitung durch einen Operator

Im Sonderforschungsbereich NEXUS wird das „NEXUS Distributed Stream Processing Framework“ (NEXUS-DSPF) als Middleware für die strombasierte Bearbeitung von Daten aus Sensornetzwerken entwickelt. Innerhalb dieses Frameworks sollen virtualisierte Operatoren als Softwarekomponenten eingesetzt werden, die die Sensordatenströme manipulieren [Cip+09].

2.1. Das NEXUS Distributed Stream Processing Framework

Kontextbezogene Anwendungen können logische Abfragen an das NEXUS-DSPF stellen. Diese Abfragen ähneln Abfragen an Datenbanksysteme, beziehen sich aber auf kontinuierliche strombasierte Daten. Das NEXUS-DSPF wandelt diese logischen Abfragen über mehrere Zwischenschritte in einen physischen Abfragegraphen im verteilten Sensornetzwerk um.

Dieser Abfragegraph beinhaltet die Datenquellen und die Operatoren die die Datenströme manipulieren. Diese Komponenten liefern in Zusammenarbeit das Ergebnis der Abfrage kontinuierlich an eine Senke, meist die abfragende Anwendung. Die Operatoren weisen standardisierte Schnittstellen auf und lassen sich daher nahezu beliebig miteinander kombinieren, um mehrere Verarbeitungsschritte hintereinander auszuführen.

Die anfragende kontextbezogene Anwendung muss sich dadurch nicht selbst um die Verknüpfung der Datenströme, die dem Sensornetzwerk zugrunde liegende Netzwerktopologie und andere technische Details kümmern. Dadurch kann sie die Sensordaten komfortabel nutzen.

2.2. Architektur eines Operators

Ein Operator wird in NEXUS in einem Container ausgeführt. Der Container stellt dem Operator die Eingangswarteschlangen (IQ_1, $IQ_2 \ldots IQ_n$) und einen Inputmanager zur Verfügung. Der Inputmanager kann dabei verschiedene Verarbeitungssemantiken realisieren. Einfache

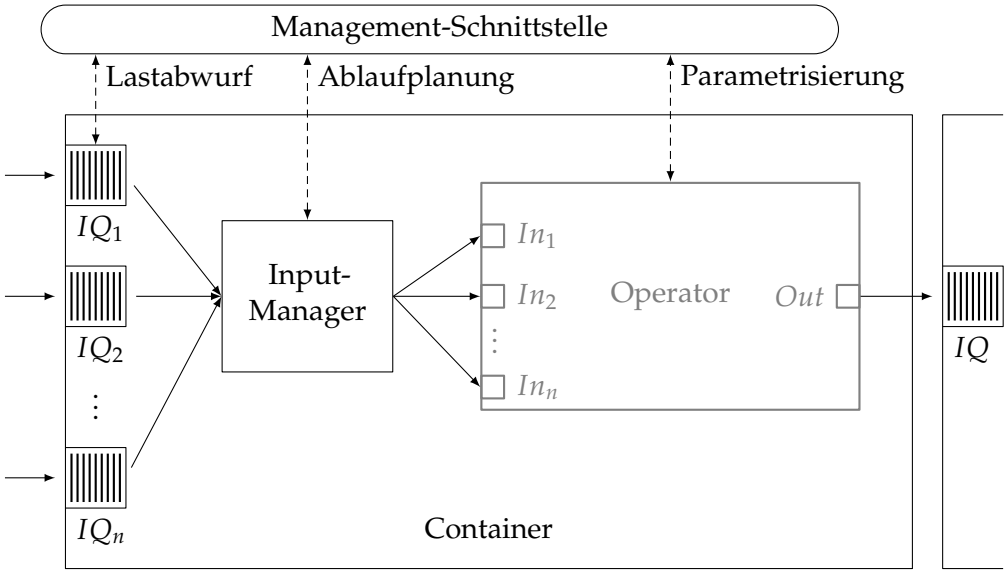


Abbildung 2.1.: Architektur eines Containers mit Operator.

Verarbeitungssemantiken können durch den Inputmanager realisiert werden und z. B. auf Objekte in mehreren Eingangswarteschlangen warten. Komplexere Verarbeitungssemantiken müssen im Operator selbst implementiert werden.

Im einfachsten Fall ist ein Operator passiv, d. h. der Inputmanager holt sich ein Objekt aus einer Eingangswarteschlage und übergibt es dem Operator zur Verarbeitung. Dieser verarbeitet es und legt das Ergebnis in die Eingangswarteschlange des nächsten Containers (siehe Abbildung 2.1).

Der Operator lässt sich über eine Managementschnittstelle parametrisieren. Außerdem kann durch eine globale Ablaufsteuerung in die Arbeit des Inputmanagers eingegriffen werden oder es können bei kritischer Überlastung Objekte aus den Eingangswarteschlangen entfernt werden.

2.2.1. Vorteile der Implementierung als Operator

Die Implementierung als Operator hat gegenüber der bisherigen Implementierung einige Vorteile. Die Komponente besitzt dadurch eine bekannte Architektur und standardisierte Schnittstellen. Außerdem lässt sie sich als passiver Operator realisieren, der beliebig mit anderen Operatoren kombiniert und hintereinander ausgeführt werden kann. Zudem ist die Komponente dadurch leicht austauschbar.

2.2.2. Nachteile der Implementierung als Operator

Nachteilig an der Implementierung als Operator ist, dass die Komponente damit nicht eigenständig benutzt werden kann, sondern innerhalb des Frameworks ausgeführt werden muss.

2.3. Anpassung der bestehenden Pufferkomponente

Im Rahmen dieser Arbeit wird die von Kaya in [Kay05] entwickelte Pufferkomponente angepasst. Die ursprüngliche Pufferkomponente wurde als eigenständige Komponente entwickelt und enthält deshalb einige Teilkomponenten, die bei der Implementierung als Operator nicht mehr benötigt werden.

Beibehalten wurden zum größten Teil die Algorithmen- und die Zeitgeberkomponente. Entfernt wurden die Teilkomponenten die Daten empfangen und senden, sowie die Verwaltungskomponente um Datenströme zu registrieren, da die von ihnen angebotene Funktionalität durch den Container bzw. das Nexus-DSPF bereitgestellt wird.

Kapitel 3

Definition der Eigenschaften der Algorithmen

In dieser Arbeit werden Algorithmen zur Kompression von Positionsdatenströmen untersucht und verglichen. Dazu wird in diesem Kapitel eine Klassifikation der Algorithmen anhand ihrer Arbeitsweise angegeben und es werden einige Eigenschaften der Algorithmen definiert.

3.1. Maximal erlaubter Fehler

Um einen dreidimensionalen Positionsdatenstrom zu komprimieren, muss dessen Punktanzahl reduziert werden. Damit die Genauigkeit des komprimierten Positionsdatenstroms garantiert werden kann, wird eine Fehlerschranke ϵ eingeführt.

Der komprimierte Positionsdatenstrom soll an jedem Punkt nur einen maximalen Abstand ϵ zum unkomprimierten Datenstrom haben. Durch diesen *maximal erlaubten Fehler* lässt sich die Abweichung des komprimierten Positionsdatenstroms vom ursprünglichen Datenstrom nach oben beschränken.

Zur Berechnung des Abstands im dreidimensionalen räumlich-zeitlichen (x,y,t)-Raum, können verschiedene Distanzfunktionen eingesetzt werden. Die Wichtigsten werden von Cao, Wolfson u. a. in [CWT06] definiert und im Folgenden vorgestellt (siehe Abbildung 3.1):

Für alle vorgestellten Distanzen gilt: Sei $p_m = (x_m, y_m, t_m)$ ein Punkt und $\overline{p_i p_j}$ der Geradenabschnitt zwischen den Punkten $p_i = (x_i, y_i, t_i)$ und $p_j = (x_j, y_j, t_j)$.

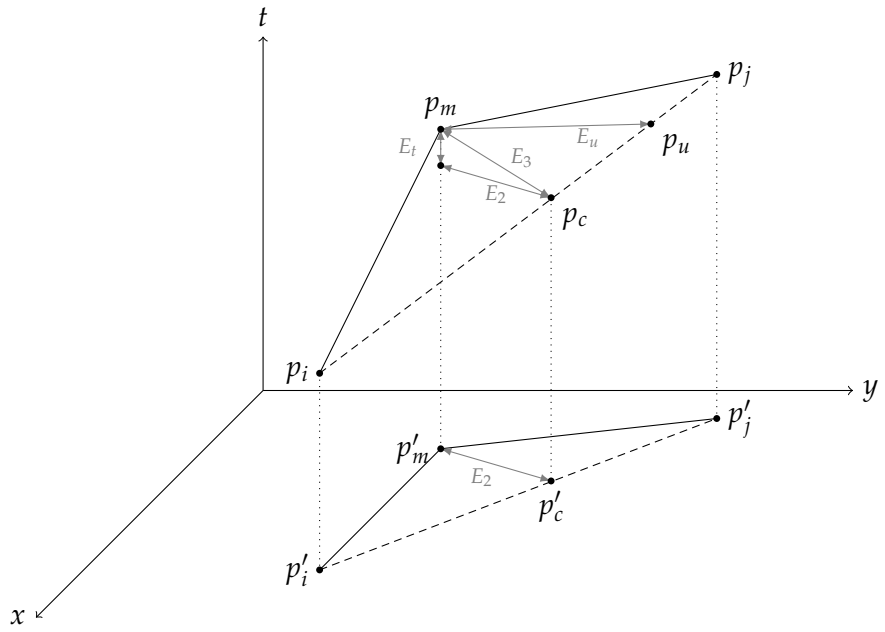


Abbildung 3.1.: Verschiedene Distanzmaße zwischen dem Punkt p_m und dem Geradenabschnitt $\overline{p_i p_j}$: Die zweidimensionale euklidische Distanz E_2, die dreidimensionale euklidische Distanz E_3, die Zeitdistanz E_t und die Zeitgleichheitsdistanz E_u.

3.1.1. Euklidische Distanz im (x,y,t)-Raum

Zweidimensional

Von Cao, Wolfson u. a. wird die zweidimensionale euklidische Distanz E_2 zwischen p_m und $\overline{p_i p_j}$ für den dreidimensionalen (x,y,t)-Raum wie folgt definiert:

$$E_2(p_m, \overline{p_i p_j}) = \sqrt{(x'_m - x'_c)^2 +(y'_m - y'_c)^2}$$

Der Punkt $p'_m = (x'_m, y'_m, 0)$ ist die Projektion des Punktes p_m auf die (x,y)-Ebene. Der Punkt $p'_c = (x'_c, y'_c, 0)$ ist der Punkt auf $\overline{p'_i p'_j}$, der im Sinne der „normalen“ zweidimensionalen euklidischen Distanz am nächsten an p'_m liegt.

Dreidimensional

Die dreidimensionale euklidische Distanz E_3 zwischen p_m und $\overline{p_i p_j}$ wird definiert als:

$$E_3(p_m, \overline{p_i p_j}) = \sqrt{(x_m - x_c)^2 +(y_m - y_c)^2 + (t_m - t_c)^2}$$

Der Punkt $p_c = (x_c, y_c, t_c)$ ist der Punkt auf $\overline{p_i p_j}$, der am nächsten an $p_m = (x_m, y_m, t_m)$ liegt. Der Winkel zwischen den Vektoren $\overline{p_i p_c}$ und $\overline{p_c p_m}$ beträgt in diesem Fall 90°.

3.1.2. Zeitdistanz

Die Zeitdistanz E_t zwischen p_m und $\overline{p_i p_j}$ wird definiert als:

$$E_t(p_m, \overline{p_i p_j}) = |t_m - t_c|$$

Die Zeit t_c ist die Zeit des Punktes p_c auf $\overline{p_i p_j}$, der bezüglich der zweidimensionalen euklidischen Distanz E_2 am nächsten an p_m liegt. Gibt es mehrere Punkte mit derselben minimalen Distanz, so wird derjenige ausgewählt, der $|t_m - t_c|$ maximiert.

3.1.3. Zeitgleichheitsdistanz

Die Zeitgleichheitsdistanz E_u zwischen p_m und $\overline{p_i p_j}$ wird definiert als:

$$E_u(p_m, \overline{p_i p_j}) = \sqrt{(x_m - x_u)^2 + (y_m - y_u)^2}$$

Wobei $p_u = (x_u, y_u, t_u)$ der Punkt ist, dessen Zeit dieselbe ist, wie die von p_m, also $t_u = t_m$. Gibt es mehrere Punkte mit derselben Zeit (die Zeit ist monoton und nicht streng monoton), so wird derjenige ausgewählt, der die Distanz minimiert. Dieser Fall tritt auf, wenn die Zeit der Punkte p_i, p_j und p_m dieselbe ist, also $t_i = t_j = t_m$. Der Punkt p_u liegt dann entweder auf p_i oder auf p_j.

3.1.4. Kombinierte Distanzen

Distanzmaße lassen sich auch kombinieren. Die Verknüpfung zweier Distanzmaße E_a und E_b zwischen den Punkten p und p' wird definiert als:

$$E_a \wedge E_b(p, p') = max(E_a(p, p'), E_b(p, p'))$$

Diese kombinierte Distanz wird von Cao, Wolfson u. a. in [CWT06] mit den Distanzen E_t und E_u verwendet, um den maximalen Fehler bei bestimmten Abfragen an die komprimierten Positionsdaten zu beschränken. Im Kapitel 3.6 dieser Arbeit wird auf diese Abfragen näher eingegangen.

3.2. Zeitkomplexität

Zur Analyse der Algorithmen wird deren asymptotische Laufzeit bestimmt. Sie ist im gegebenen Anwendungsfall der strombasierten Datenkompression aber kaum von Bedeutung, da hier nur Punktmengen von wenigen hundert Punkten betrachtet werden.

3.3. Arbeitsweise

Die untersuchten Algorithmen werden zur Kompression bzw. Vereinfachung von Positionsdatenströmen eingesetzt. Im Folgenden werden sie deshalb anhand ihrer Arbeitsweise auf diesen Datenströmen klassifiziert. Grundlegend unterscheiden sich die Verfahren darin, wie sie mit neu eintreffenden Punkten umgehen. Es lassen sich strombasierte und fensterbasierte Verfahren unterscheiden.

3.3.1. Strombasierte Verfahren

Strombasierte Verfahren bearbeiten neu eintreffende Punkte sofort und vergleichen diese mit dem aktuellen Vereinfachungsabschnitt. Abhängig vom eingesetzten Verfahren, wird dann der Vereinfachungsabschnitt beibehalten, angepasst oder ein neuer Vereinfachungsabschnitt erzeugt. Diese Verfahren lassen sich weiter in verwerfende und puffernde Verfahren unterteilen.

Verwerfend-strombasierte Verfahren

Verwerfend-strombasierte Verfahren bearbeiten den neu eingetroffenen Punkt und verwerfen ihn im Anschluss. Sie speichern daher immer nur den aktuelle Vereinfachungsabschnitt und eine im Voraus bekannte feste Anzahl an Punkten. Dadurch haben sie einen geringen Speicherverbrauch. Da jeder Punkt genau einmal betrachtet und mit dem aktuellen Vereinfachungsabschnitt verglichen wird, weisen verwerfend-strombasierte Verfahren eine lineare Zeitkomplexität auf.

Puffernd-strombasierte Verfahren

Puffernd-strombasierte Verfahren speichern eine Punktmenge zwischen und berechnen bei jedem Eintreffen eines neuen Punktes den Vereinfachungsabschnitt neu. Dazu verwenden sie alle älteren gepufferten Punkte, die dadurch mehrfach betrachtet werden. Puffernd-strombasierte Verfahren weisen durch dieses mehrfache Betrachten der Punkte eine nicht-lineare Zeitkomplexität auf. Zudem benötigen sie mehr Speicherplatz als verwerfend-strombasierte Verfahren, da sie Teile der betrachteten Punkte zwischenspeichern.

3.3.2. Fensterbasierte Verfahren

Fensterbasierte Verfahren sammeln eine bestimmte Anzahl Punkte und berechnen anhand dieser einen oder mehrere Vereinfachungsabschnitte. Sie unterscheiden sich von strombasierten Verfahren darin, dass sie mit der Berechnung erst bei Erreichen der Fenstergröße beginnen und die Vereinfachung nicht jedes Mal anpassen, wenn ein neuer Punkt eintrifft.

Für die Angabe der Fenstergröße gibt es zwei Möglichkeiten. Entweder wird eine Punktanzahl vorgegeben oder die Punktanzahl wird indirekt über eine Zeitschranke festgelegt.

Vorgegebene Punktanzahl

Die Angabe einer festen Punktanzahl hat den Vorteil, dass die Laufzeit der Vereinfachungsberechnung gut im Voraus abschätzbar ist. Außerdem beschränkt sie den benötigten Speicherplatz. Der zeitliche Abstand zwischen der Ausgabe von Vereinfachungspunkten ist in diesem Fall aber nicht vorhersagbar.

Zeitschranke

Setzt man eine Zeitschranke ein, um die Fenstergröße zu beschränken, erzwingt man damit beim Erreichen dieser Zeitschranke einen zusätzlichen Vereinfachungspunkt. Dadurch kann sich, in Abhängigkeit der Rate in der die Positionsdaten ankommen, die Punktanzahl im Fenster verändern.

Durch die Zeitschranke wird eine kontinuierliche Antwort des Kompressionsalgorithmus erzwungen. Dadurch eignet sich der Einsatz einer Zeitschranke im Gegensatz zur festen Punktanzahl dann, wenn beispielsweise ein aktueller Positionsdatenstrom komprimiert und anschließend ohne Verzögerung direkt weiterverarbeitet werden soll.

Die Punktanzahl innerhalb des Fensters wird durch die Zeitschranke nicht beschränkt. Dadurch ist nicht abschätzbar wieviel Speicherplatz die puffernden oder fensterbasierten Verfahren benötigen. Das kann sowohl bei puffernden als auch bei fensterbasierten Verfahren zu sehr langen Laufzeiten führen.

3.3.3. Weitere Klassifikationsmöglichkeiten

Neben der oben angegebenen Klassifikation nennt McMaster in [McM87] weitere Klassifikationsmöglichkeiten, die auf anderen Kriterien basieren. So klassifiziert McMaster die Algorithmen anhand der mathematischen Vorgehensweise (z. B. Auswahl der Punkte, Erkennung von Winkeln, Epsilon Filterung . . .) oder basierend auf der Ausdehnung der gleichzeitig vom Algorithmus betrachteten Punkte (z. B. einzelne Punkte, lokale Nachbarschaft, globale Punktmenge).

Für die Vereinfachung von Positionsdatenströmen sind diese Klassifikationen weniger relevant als die Arbeitsweise der Algorithmen. Deshalb wird auf diese Klassifikationsmöglichkeiten nicht weiter eingegangen.

3.4. Alterungsstabilität

Häufig werden ältere Daten nicht mit derselben Genauigkeit benötigt, wie aktuelle Daten. Deshalb kann man die Datensätze schrittweise „altern“ lassen, indem man sie immer weiter vereinfacht.

Cao, Wolfson u. a. stellen in [CWT06] einen Alterungsmechanismus und das Konzept der Alterungsstabilität vor. Ein Verfahren $\mathcal{A}$ gilt als alterungsstabil bezüglich einer Distanzfunktion E, wenn die Ausführung des Verfahrens auf einer Folge von Positionsdaten P zuerst mit kleiner und dann mit großer Fehlerschranke ϵ zum selben Ergebnis führt, wie die direkte Ausführung mit der größeren Fehlerschranke.

Sei $\mathcal{A} : P \times \epsilon \times E \longmapsto P_{vereinfacht}$ ein Vereinfachungsverfahren und ϵ eine Fehlerschranke. Ein Vereinfachungsverfahren ist alterungsstabil, wenn gilt:

$$\mathcal{A}(\mathcal{A}(P, \epsilon_{klein}, E), \epsilon_{gro}, E) = \mathcal{A}(P, \epsilon_{gro}, E)$$

Alterungsstabile Verfahren haben gegenüber nicht alterungsstabilen Verfahren den Vorteil, dass man einen vereinfachten Pfad später weiter vereinfachen kann und dabei denselben Pfad erhält, wie wenn man die Vereinfachung mit der letzten Fehlerschranke auf dem Ursprungsdatenstrom durchgeführt hätte. Bei nicht alterungsstabilen Verfahren ist der vereinfachte Pfad nach mehrfacher Vereinfachung verschieden von dem bei direkter Vereinfachung mit dem größeren Fehler entstehenden Pfad.

Palpanas u. a. stellen in [Pal+04] sogenannte vergessliche Funktionen („amnesic functions“) vor, die eine inkrementelle Vereinfachung des Pfades bei Alterung ermöglichen. Hier „altert“ der komplette Pfad und muss somit im Hauptspeicher vorliegen. Da in der Arbeit von Palpanas u. a. mit Positionsdatenströmen gearbeitet wird, die direkt im Hauptspeicher weiterverarbeitet werden, scheidet das dort vorgestellte Verfahren für unseren Anwendungsfall aus.

3.5. Fehlerbeschränktheit bei mehrfacher Vereinfachung

Da die meisten Verfahren nicht alterungsstabil sind, wird zusätzlich untersucht ob diese bei mehrfach nacheinander ausgeführter Vereinfachung den Fehler beschränken.

Im Gegensatz zur Alterungsstabilität ist dabei der Verlauf des mehrfach vereinfachten Pfades unwichtig, solange der maximal erlaubte Fehler gegenüber dem Originalpfad eingehalten wird. Für die Fehlerbeschränktheit können Einschränkungen bezüglich der Zeitschranken (eventuell müssen bei erneuter Vereinfachung die Fenster dieselben sein, wie bei der ersten Vereinfachung) oder der Fehlerschranke (z. B. muss die neue Fehlerschranke ein Vielfaches der vorhergehenden Fehlerschranke sein) nötig sein.

3.6. Beschränkung des maximalen Fehlers bei Abfragen auf vereinfachten Daten

Die vereinfachten Daten der Positionsdatenströme können zur späteren Nutzung und Analyse in Datenbanken abgelegt werden. Typische Abfragen an diese gespeicherten zeitlich-räumlichen Daten können, in Abhängigkeit der bei der Vereinfachung verwendeten Distanzfunktion, auf den vereinfachten Positionsdaten zu einem unbeschränkt großen Fehler führen. Dazu werden im Folgenden einige typischen Abfragen an zeitlich-räumliche Daten vorgestellt [CWT06].

3.6.1. Typische Abfragen

Sei $P = p_1, p_2 \dots p_n$ mit $p_i = (x_i, y_i, t_i)$ für $i \in [1..n]$ eine Folge von Positionsdaten.

where_at

Die Abfrage where_at(P, t) gibt die erwartete Position zum Zeitpunkt t zurück. Wenn $t < t_1$ oder $t > t_n$, dann ist das Ergebnis undefiniert. Da die Zeitdimension monoton, aber nicht streng monoton ist, kann das Ergebnis der Anfrage mehrere Punkte beinhalten.

when_at

Die Abfrage when_at(P, x, y) gibt die Zeit t zurück, zu der das Objekt an Position (x, y) war. Da das Objekt mehrere Male an derselben Position sein kann, kann auch hier das Ergebnis der Anfrage mehrere Punkte beinhalten.

intersect

Die Abfrage intersect(P, $Polygon$, t_1, t_2) ist wahr, wenn der Pfad P das zweidimensionale $Polygon$ in der (x,y)-Ebene zwischen den Zeitpunkten t_1 und t_2 schneidet. In [CWT06] wird diese Abfrage nur für konvexe Polygone definiert und untersucht.

nearest_neighbour

Die Abfrage nearest_neighbour(P, $Objekte$, t) gibt das Objekt $O \in Objekte$ zurück, das zum Zeitpunkt t am nächsten an P war.

	where at	when at	intersect	nearest neighbour
E_2	nein	nein	nein	nein
E_3	nein	nein	nein	nein
E_u	ja	nein	ja	ja
E_t	nein	ja	nein	nein
$E_u \wedge E_t$	ja	ja	ja	ja

Tabelle 3.1.: Beschränkung des maximalen Fehler bei typische Abfragen an zeitlich-räumliche Daten bei verschiedenen Distanzmaßen.

3.6.2. Wahl der Distanzfunktion anhand der Beschränktheit des Fehlers

Die Distanzfunktionen werden von Cao, Wolfson u. a. in [CWT06] auf die Beschränktheit des Fehlers bezüglich der oben genannten Abfragen hin untersucht. Das Ergebnis ist in Tabelle 3.1 zusammengefasst.

Die beiden euklidischen Distanzmaße sind nicht in der Lage den maximalen Fehler bei den vorgestellten Abfragen zu beschränken. Die Zeitgleichheitsdistanz E_u beschränkt den Fehler bei allen räumlichen Anfragen, die Zeitdistanz E_t beschränkt ihn für die zeitliche when_at() Abfrage. Die kombinierte Distanz aus Zeit- und Zeitgleichheitsdistanz $E_u \wedge E_t$ schafft es, den Fehler bei allen vier Abfragen zu beschränken.

Bei der Implementierung der Verfahren im Rahmen dieser Arbeit wurde die Zeitgleichheitsdistanz E_u verwendet, da die kombinierte Distanz nicht bei allen Verfahren einsetzbar ist. Zudem ist unklar, wie die Distanzmaße E_u und E_t bei der Bestimmung des Maximums zu gewichten sind, da die Zeit- und Raumdimensionen verschiedene Einheiten und ihre Werte verschiedene Größenordnungen haben.

Kapitel 4

Eigenschaften der bereits untersuchten Algorithmen

In diesem Kapitel werden die bereits von Kaya untersuchten und implementierten Algorithmen und ihre Eigenschaften vorgestellt. Zudem werden die Algorithmen auf Alterungstabilität und ihre Fehlerbeschränktheit bei mehrfacher Vereinfachung untersucht. Für eine detaillierte Analyse der Arbeitsweise der Algorithmen sei auf das Kapitel 4.7 in [Kay05] verwiesen.

Die Algorithmen arbeiten, soweit dies möglich ist, mit der Zeitgleichheitsdistanz E_u. Bei den fensterbasierten Verfahren wird mit einer maximalen Zeitdauer gearbeitet, um die maximale Fenstergröße zu beschränken. Ist diese erreicht, wird der betrachtete Teil des Positionsdatenstroms vereinfacht und das Ergebnis gesendet.

Bei den strombasierten Verfahren wird auch mit einer maximalen Zeitdauer gearbeitet. Diese erzwingt, dass spätestens nach Erreichen dieser Zeitdauer gesendet wird. Ist schon vor Erreichen der maximalen Zeitdauer das Senden eines Vereinfachungspunktes möglich, wird der Zeitgeber zurückgesetzt und das Verfahren erhält wieder die maximale Zeitdauer, bis es zum nächsten Mal senden muss.

4.1. Richtung und Geschwindigkeit

4.1.1. Verfahren

Das vektorbasierte Verfahren „Richtung und Geschwindigkeit“ stellt den vereinfachten Positionsdatenstrom durch Vektoren dar. Dabei bildet die Vereinfachung keinen zusammenhängenden Pfad, sondern enthält Sprünge zwischen den Enden der Vektoren und den Anfängen des jeweils nächsten Vektors.

Das Verfahren benötigt als Eingabe neben den Punktkoordinaten auch die Bewegungsrichtung und die Geschwindigkeit des mobilen Objekts.

4.1.2. Eigenschaften

Arbeitsweise

Das Verfahren „Richtung und Geschwindigkeit“ arbeitet verwerfend-strombasiert. Er speichert den ersten Punkt und dessen Richtung und Geschwindigkeit als Vereinfachunsgvektor und testet für alle folgenden Punkte, ob diese vom Vereinfachungsvektor ausreichend genau angenähert werden. Dazu wird der Vereinfachungsvektor in jedem Schritt bis zur Zeit t des aktuellen Punktes verlängert. Ist die Annäherung ausreichend, wird der untersuchte Punkt verworfen. Für die Abstandsberechnung wird die Zeitgleichheitsdistanz E_u verwendet.

Komplexität

Da das Verfahren jeden Punkt nur einmal betrachtet, hat es eine Zeitkomplexität von $O(n)$.

Maximaler Fehler

Das Verfahren garantiert einen vorgegebenen maximalen Fehler des komprimierten Positionsdatenstroms gegenüber dem ursprünglichen Positionsdatenstrom.

Alterungsstabilität

Das Verfahren ist für keine der verwendeten Distanzfunktionen alterungsstabil, da die Punkte auf den Vektoren der Vereinfachung andere Richtungsvektoren haben als die ursprünglichen Punkte. Dadurch kommt es bei der schrittweisen Vergrößerung von ϵ zu anderen Vereinfachungsvektoren, als durch die direkte Wahl eines größeren Wertes für ϵ.

Fehlerbeschränktheit bei mehrfacher Vereinfachung

Damit das Verfahren den Fehler bei mehrfacher Vereinfachung beschränkt, muss der maximal erlaubte Fehler des vorhergehenden Vereinfachungsschrittes bekannt sein. Um den mit ϵ_1 vereinfachten Positionsdatenstrom mit einem maximal erlaubten Fehler ϵ_2 weiter zu vereinfachen, muss man den maximal erlaubten Fehler $\epsilon_{max} = \epsilon_2 - \epsilon_1$ verwenden. Der dadurch entstehende vereinfachte Positionsdatenstrom weist dann gegenüber dem ursprünglichen Datenstrom einen maximalen Fehler von ϵ_2 auf.

Ein bereits vereinfachter Datenstrom lässt sich demnach nur weiter vereinfachen, wenn $\epsilon_2 > \epsilon_1$, da sonst keine weitere Vereinfachung eintritt.

4.2. Lineare Extrapolation

Da nicht alle Datenquellen die Geschwindigkeit und die Richtung des mobilen Objekts liefern, muss das vektorbasierte Verfahren für diese Datenquellen angepasst werden.

Die lineare Extrapolation berechnet aus den ersten zwei Punkten eines Positionsdatenstroms einen Vereinfachungsvektor. Bei allen weiteren Punkten wird geprüft, ob diese durch den Vereinfachungsvektor ausreichend genau angenähert werden können. Ist das der Fall, werden die Punkte jeweils verworfen. Ist die Abweichung zu groß, wird aus dem aktuellen und dem darauf folgenden Punkt ein neuer Vereinfachungsvektor berechnet.

Das Verfahren unterscheidet sich von dem Verfahren „Richtung und Geschwindigkeit“ ausschließlich darin, wie der Vektor der Bewegungsrichtung und die Geschwindigkeit berechnet wird. Da GPS und andere Positionierungssysteme die Richtung und die Geschwindigkeit des mobilen Objekts ebenfalls anhand zweier Punkte berechnen, verhalten sich die beiden Verfahren bezüglich der untersuchten Eigenschaften gleich.

4.3. Erster und letzter Punkt

4.3.1. Verfahren

Das Verfahren berechnet die Vereinfachung durch den ersten und letzten Punkt eines Intervalls. Es startet mit den ersten beiden Punkten eines Positionsdatenstroms. Bei jedem weiteren Punkt wird zuerst der Vereinfachungsgeradenabschnitt durch den ersten und den aktuellen Punkt bestimmt und dann der Abstand aller bisher gesammelten Punkte zu diesem Geradenabschnitt bestimmt. Sind alle diese Abstände kleiner als ϵ, fährt das Verfahren fort. Hat ein Punkt einen größeren Abstand, wird mit dem aktuellen und dem vorherigen Punkt ein neuer Vereinfachungsgeradenabschnitt gebildet und der alte zum vereinfachten Positionsdatenstrom hinzugefügt.

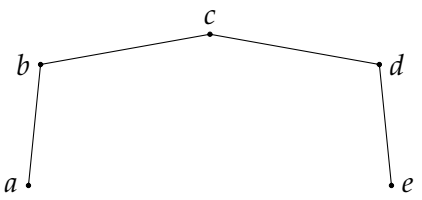
4.3.2. Eigenschaften

Arbeitsweise

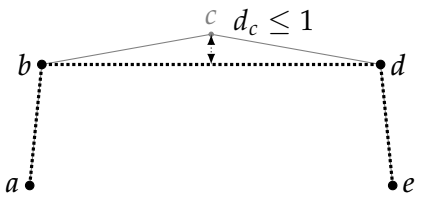
Das Verfahren arbeitet puffernd-strombasiert und berechnet bei jedem neuen Punkt die Abstände zur Vereinfachungsgerade mithilfe der Zeitgleichheitsdistanz E_u neu.

Komplexität

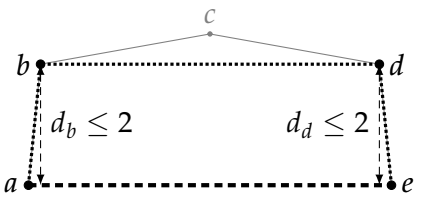
Da das Verfahren beim Eintreffen eines neuen Punktes zur Abstandsberechnung alle bisher gesammelten Punkte durchlaufen muss, hat es eine Zeitkomplexität von $O(n^2)$.



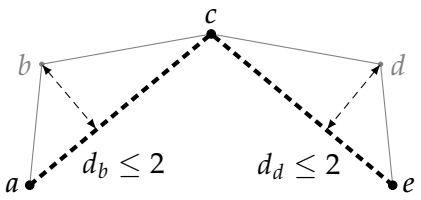
(a) Ursprünglicher Pfad



(b) Vereinfachung mit $\epsilon = 1$



(c) Wiederholte Vereinfachung mit $\epsilon = 1$ und $\epsilon = 2$



(d) Direkte Vereinfachung mit $\epsilon = 2$

Abbildung 4.1.: Gegenbeispiel zur Alterungsstabilität des Verfahrens „Erster und letzter Punkt“ im $\mathbb{R}^2$.

Maximaler Fehler

Da für alle Punkte berechnet wird, ob die Vereinfachungsgerade diese mit einem Abstand kleiner ϵ annähert, garantiert das Verfahren den vorgegeben maximal erlaubten Fehler.

Alterungsstabilität

Das Verfahren ist für keine der beschriebenen Distanzfunktionen alterungsstabil. Die Abbildungen 4.1a bis 4.1d auf dieser Seite zeigen ein Beispiel im $\mathbb{R}^2$, in dem ein Positionsdatenstrom einmal schrittweise und einmal direkt vereinfacht wird. Das Beispiel ist aber auch für den räumlich-zeitlichen (x,y,t)-Raum verwendbar, indem man eine der Raumdimensionen auf Null setzt und die erste Koordinate als Zeitdimension betrachtet.

Die Punkte haben die folgenden Koordinaten:
$a = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$, $b = \begin{pmatrix} 0{,}2 \\ 2 \end{pmatrix}$, $c = \begin{pmatrix} 3 \\ 2{,}5 \end{pmatrix}$, $d = \begin{pmatrix} 5{,}8 \\ 2 \end{pmatrix}$ und $e = \begin{pmatrix} 6 \\ 0 \end{pmatrix}$.

Zuerst wird der Pfad *abcde* mit einer Fehlerschranke von $\epsilon = 1$ vereinfacht (siehe Abbildung 4.1b). Dadurch entsteht der Pfad *abde*. Danach wird der bereits vereinfachte Pfad mit einer Fehlerschranke von $\epsilon = 2$ weiter vereinfacht. Es entsteht der Pfad *ae* (siehe Abbildung 4.1c).

Wird der ursprüngliche Pfad *abcde* direkt mit $\epsilon = 2$ vereinfacht, entsteht mit *ace* ein anderer Pfad, als bei der mehrfachen Vereinfachung (siehe Abbildung 4.1d).

Fehlerbeschränktheit bei mehrfacher Vereinfachung

Das Verfahren ist bei mehrfacher Vereinfachung im Allgemeinen nicht fehlerbeschränkt. Das ist in Abbildung 4.1c auf der vorherigen Seite gut am Abstand des Geradenabschnitts $\overline{ae}$ zum Punkt c sehen kann, der größer als 2 ist.

Um die Fehlerbeschränktheit zu erreichen, gelten dieselben Einschränkungen, wie für das Verfahren „Richtung und Geschwindigkeit“ (siehe Kapitel 4.1.2).

4.4. Lineare Regression

4.4.1. Verfahren

Regressionsverfahren bilden für eine gegebene Punktmenge eine Ausgleichskurve. Bei der linearen Regression wird versucht, eine Ausgleichsgerade durch eine Punktmenge zu legen, die einen möglichst geringen Fehler erzeugt. In [Kay05] wird beschrieben, wie die dreidimensionale Regression in zwei zweidimensionale Regressionsprobleme zerlegt werden kann. Bei dieser Art der Regression auf dreidimensionalen (x,y,t)-Daten verschieben sich die Koordinaten der Punkte in der Ebene, ihr Zeitanteil bleibt aber gleich. Im Prinzip wird die Zeitgleichheitsdistanz E_u (siehe Kapitel 3.1.3) verwendet.

Die durch diese Regressionsgerade dargestellte Vereinfachungsgerade wird durch Stützpunkte beschrieben, die nicht mehr auf der ursprünglichen Kurve liegen müssen.

Für jeden neu hinzukommenden Punkt wird die Regressionsgerade neu berechnet und getestet, ob diese die Punktmenge mit einem maximalen Fehler ϵ annähert. Ist dies der Fall, kann der nächste Punkt hinzugefügt werden, wenn nicht, wird ein neues Intervall eröffnet, in dem erneut eine Regressionsgerade erzeugt wird.

4.4.2. Eigenschaften

Arbeitsweise

Das Verfahren arbeitet puffernd-strombasiert, da es die vorausgehenden Punkte für die Berechnung der Regressionsgerade benötigt.

Komplexität

Da bei jedem neu hinzukommenden Punkt die Regressionsgerade neu berechnet und der Abstand aller bisherigen Punkte zu dieser Gerade berechnet wird, hat das Verfahren eine Zeitkomplexität von $O(n^2)$.

Maximaler Fehler

Durch die Berechnung der Regressionsgerade und den anschließenden Test auf den maximal erlaubten Fehler ϵ, beschränkt das Verfahren den Fehler der Vereinfachung.

Alterungsstabilität

Von Palpanas u. a. wird in [Pal+04] gezeigt, dass man mehrere Regressionsgeraden zu einer zusammenfassen kann, wenn der Abstand der Messpunkte auf der Zeitachse immer derselbe ist. Das ist beispielsweise gegeben, wenn das mobile Objekt im Sekundentakt seine Position sendet.

Ist dieser konstante Zeitabstand bekannt, lassen sich die Projektionen der Ursprungspunkte auf die Regressionsgerade berechnen. Aus den projizierten Punkten mehrerer Regressionsgeraden kann dann die gemeinsame Regressionsgerade berechnet werden ohne die ursprünglichen Punkte zu kennen. Dieselbe Gerade erhält man, wenn die ursprünglichen Punkte genutzt werden.

Um bei bekanntem konstanten Zeitabstand der Positionsdaten die Alterungsstabilität des Regressionsverfahrens zu erreichen, muss das Verfahren angepasst werden. Da die Regressionsgeradenabschnitte immer durch zwei Punkte beschrieben werden, muss der angepasste Algorithmus immer zwei Punkte (Anfang und Ende der aktuellen Regressionsgerade) gleichzeitig verarbeiten.

Dadurch erhält man zwei Varianten des Regressionsalgorithmus. Eine einfacher Algorithmus, der die erste Vereinfachung erzeugt und ein angepasster Algorithmus für weitere Vereinfachungsschritte. Das Verfahren ist in seiner einfachen Form also für keine der verwendeten Distanzfunktionen alterungsstabil. Die alterungsstabile Variante des Verfahrens stellt zudem Anforderungen an den Datenstrom, die im Allgemeinen nicht erfüllt sind.

Fehlerbeschränktheit bei mehrfacher Vereinfachung

Fasst man mehrere Regressionsgeraden zusammen, lässt sich der maximale Fehler wie beim Verfahren „Richtung und Geschwindigkeit“ durch Subtraktion der maximalen Fehler beschränken (siehe Kapitel 4.1.2). Da aber, wie bei der Alterungsstabilität beschrieben, das Regressionsverfahren für die erneute Kompression angepasst werden müsste, ist eine Fehlerbeschränktheit bei mehrfacher Vereinfachung im Allgemeinen nicht möglich.

4.5. Distanzbasiert

4.5.1. Verfahren

Das distanzbasierte Verfahren arbeitet wie ein Filter auf dem Positionsdatenstrom. Solange neu hinzukommende Punkte innerhalb der ϵ-Umgebung um den ersten Punkt liegen, werden diese verworfen. Liegt ein neuer Punkt außerhalb dieser Umgebung, wird er zum vereinfachten Positionsdatenstrom hinzugefügt und das Verfahren wiederholt sich. Dieses Vorgehen führt bei stehenden mobilen Objekten dazu, dass für diese theoretisch nur ein Punkt gespeichert werden muss.

4.5.2. Eigenschaften

Arbeitsweise

Das Verfahren arbeitet verwerfend-strombasiert, da es alle Punkte innerhalb der ϵ-Umgebung des aktuell ausgewählten Punktes verwirft.

Komplexität

Da für jeden Punkt nur einmal geprüft werden muss, ob dieser in der ϵ-Umgebung des aktuell ausgewählten Vereinfachungspunktes liegt, hat das Verfahren eine Zeitkomplexität von $O(n)$.

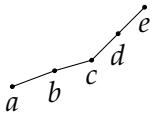
Maximaler Fehler

Das Verfahren erzeugt Vereinfachungen mit einem garantierten maximalen Fehler von ϵ, da alle verworfenen Punkte innerhalb der ϵ-Umgebung eines Punktes der komprimierten Punktmenge liegen. Zur Distanzberechnung wurde die zweidimensionale Euklidische Distanz E_2 im dreidimensionalen (x,y,t)-Raum verwendet.

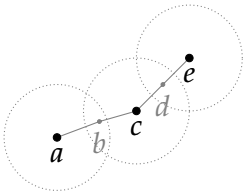
Alterungsstabilität

Das Verfahren ist für die verwendete E_2 Distanz nicht alterungsstabil. Einen Beispielpfad zeigen die Abbildungen 4.2a bis 4.2d auf der nächsten Seite.

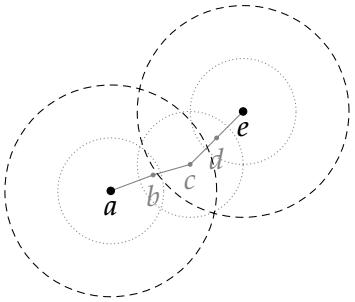
Die Punkte haben die folgenden Koordinaten:
$a = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$, $b = \begin{pmatrix} 0,8 \\ 0,3 \end{pmatrix}$, $c = \begin{pmatrix} 1,5 \\ 0,5 \end{pmatrix}$, $d = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$ und $e = \begin{pmatrix} 2,5 \\ 1,5 \end{pmatrix}$.



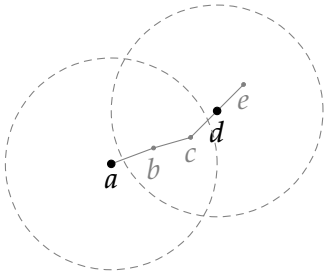
(a) Ursprünglicher Pfad



(b) Mit $\epsilon = 1$ vereinfachter Pfad



(c) Wiederholt mit $\epsilon = 1$ und $\epsilon = 2$ vereinfachter Pfad



(d) Direkt mit $\epsilon = 2$ vereinfachter Pfad

Abbildung 4.2.: Gegenbeispiel zur Alterungsstabilität des Verfahrens „Distanzbasiert“ im $\mathbb{R}^2$.

Wird der ursprüngliche Pfad *abcde* mit einem maximal erlaubten Fehler $\epsilon = 1$ vereinfacht, bleiben die Punkte *ace* übrig (siehe Abbildung 4.2b). Wird dieser Pfad dann mit einem maximal erlaubten Fehler von $\epsilon = 2$ erneut vereinfacht, bleiben die Punkte *a* und *e* übrig. Die direkte Vereinfachung mit $\epsilon = 2$ führt zu den Punkten *a* und *d*. Die beiden vereinfachten Pfade unterscheiden sich, somit ist das Verfahren nicht alterungsstabil.

Fehlerbeschränktheit bei mehrfacher Vereinfachung

Für die Fehlerbeschränktheit gelten dieselben Eigenschaften wie für das Verfahren „Richtung und Geschwindigkeit“ (siehe Kapitel 4.1.2).

Kapitel 5

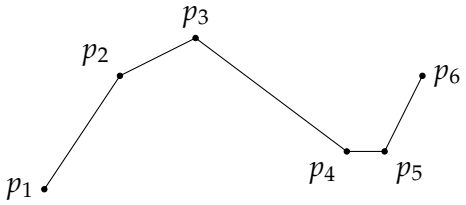
Polygonzugvereinfachung

5.1. Polygonzüge

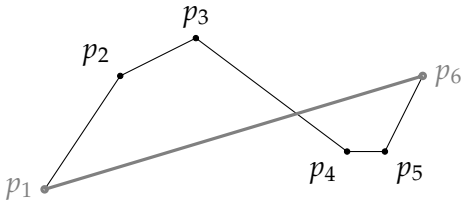
Ein Polygonzug ist die Vereinfachung einer Kurve, die durch eine Punktmenge $\{p_1, \ldots, p_n\}$ beschrieben wird, welche im $\mathbb{R}^2$, $\mathbb{R}^3$ oder einem beliebigen mehrdimensionalen Raum liegt. Die Punkte sind jeweils durch Geradenabschnitte $(\overline{p_i p_{i+1}} \mid \forall i \in \{1..(n-1)\})$ verbunden (siehe Abbildung 5.1a). Polygonzüge können Kurven beliebig genau annähern.

Algorithmen zur Polygonzugvereinfachung versuchen die Punktmenge eines gegebenen Polygonzuges P zu reduzieren (siehe Abbildung 5.1b). Dabei kann man entweder die Punktanzahl bei vorgegebenem maximalen Fehler minimieren oder den Fehler bei vorgegebener Punktanzahl minimieren. Es lassen sich also die folgenden Problemstellungen unterscheiden [AG96; II86]:

Das min-# Problem: Finde einen vereinfachten Polygonzug $P_{vereinfacht}$ mit einer minimalen Anzahl an Punkten, so dass der Abstand zum Polygonzug P den vorgegebenen maximal erlaubten Fehler ϵ nicht überschreitet.



(a) Ein Polygonzug P mit den Punkten p_1 bis p_6.



(b) Ein Polygonzug P mit den Punkten p_1 bis p_6 und der Vereinfachung $\overline{p_1 p_6}$.

Abbildung 5.1.: Beispiel für einen Polygonzug P und eine Vereinfachung $P_{vereinfacht}$

Das min-ϵ Problem: Finde einen Polygonzug $P_{vereinfacht}$ mit einer gegebenen Anzahl von Punkten m, der den Polygonzug P mit minimalem Fehler ϵ annähert.

In dieser Arbeit sollen Positionsdatenströme mithilfe von Algorithmen zur Polygonzugvereinfachung komprimiert bzw. vereinfacht werden. Als Eingabe dient ein Positionsdatenstrom, der als Polygonzug interpretiert werden kann, wenn man die einzelnen Punkte mit Geradenabschnitten verbindet. Die Ausgabe soll ein komprimierter Datenstrom (Polygonzug) sein, der einen vorgegebenen maximal erlaubten Fehler einhält. Diese Aufgabe entspricht dem min-# Problem, daher wird das min-ϵ Problem im Folgenden nicht weiter betrachtet.

Das min-# Problem lässt sich auf einfache Weise optimal in exponentieller Zeit (in $O(2^n)$) berechnen, indem alle Vereinfachungsmöglichkeiten berechnet werden und daraus die kleinste ausgewählt wird. Durch Ausnutzen geometrischer Eigenschaften oder durch dynamische Programmierung lässt sich die Komplexität auf Polynomialzeit senken. Typische Implementierungen für optimale Vereinfachung haben quadratische oder kubische average-case Komplexität [HG97].

Heuristische Verfahren weisen dagegen teilweise lineare und häufig nur im schlechtesten Fall quadratische Zeitkomplexität auf. Ein lineares und zwei quadratische Verfahren werden in dieser Arbeit betrachtet.

5.2. Verwandte Ansätze im Bereich der Polygonzugvereinfachung

Algorithmen zur Polygonzugvereinfachung werden seit Mitte des 20. Jahrhunderts in der Computergrafik und der Kartographie eingesetzt und finden Anwendung auf dem Gebiet der Mustererkennung.

5.2.1. Computergrafik

In der Computergrafik werden Polygonzüge zur Darstellung beliebiger Kurven eingesetzt. Meist werden solche Kurven in einem Detaillierungsgrad gespeichert, der die Ausgabegenauigkeit des Ausgabemediums (z. B. ein Bildschirm) bei weitem übersteigt oder in der aktuellen Ansicht (z. B. verkleinert) nicht benötigt wird. Um die zu verarbeitende Datenmenge zu reduzieren, können Verfahren zur Polygonzugvereinfachung eingesetzt werden, mit denen die Datenmenge reduziert wird, ohne das Anzeigeresultat zu verändern [BB82; AG96]. Die Polygonzüge können beispielsweise mit einem maximal erlaubten Fehler komprimiert werden, der kleiner als die Größe eines Pixels des darstellenden Bildschirms ist. Die Abweichungen von der ursprünglichen Kurve sind dann nicht darstellbar und daher auch nicht sichtbar.

5.2.2. Kartographie

Zur Vereinfachung von Flussläufen, Straßen, Ländergrenzen usw. werden Verfahren zur Polygonzugvereinfachung in der Kartographie eingesetzt. Hauptziel ist es, die Datenmenge der Karte zu reduzieren, um nachfolgende Berechnungen zu beschleunigen. Man erreicht durch die Reduktion der Datenmenge aber auch eine bessere Bildschirmdarstellung und ein besseres Druckbild der Karten, da die Ausgabe einer Karte mit vielen Details mit einer geringen Auflösung zu Überschneidungen und Unlesbarkeiten führen kann [McM87; Aga+05].

In den letzten 40 Jahren wurden in der Kartographie viele Verfahren entwickelt. Diese unterscheiden sich u. a. darin, dass der vereinfachte Polygonzug entweder nur aus Punkten des ursprünglichen Polygonzuges oder auch aus neuen Punkte bestehen kann:

Polygonzugvereinfachung auf Basis bestehender Punkte

Ein Teil der Verfahren beschränkt die Punkte der Vereinfachung auf die Punktmenge des ursprünglichen Polygonzuges. Diese Verfahren können weiter in dezimierende und verfeinernde Verfahren unterteilt werden.

Die *dezimierenden Verfahren* gehen von der gesamten Punktmenge aus und reduzieren diese, bis keine Vereinfachung mehr möglich ist, ohne einen vordefinierten maximal erlaubten Fehler ϵ zu überschreiten (siehe Kapitel 3.1). Die zurückgebliebenen Punkte beschreiben dann den vereinfachten Polygonzug. Beispiele für dezimierende Verfahren finden sich in [II86; CC96; AV00; Aga+05].

Verfeinernde Verfahren wählen solange Punkte aus der ursprünglichen Punktmenge, bis der vordefinierte maximal erlaubte Fehler ϵ unterschritten wird. Das bekannteste Verfahren stammt von Douglas und Peucker [DP73; Whi85] und wird in Kapitel 6.2 auf Seite 41 beschrieben und untersucht. Cao und Wolfson [CWT06] untersuchen das Verfahren unter Verwendung anderer Distanzmaße im räumlich-zeitlichen Raum. Ali Abam u. a. [AA+07] erweitern das Verfahren um ein Fehlerorakel und senken damit die Platzkomplexität des Algorithmus.

Polygonzugvereinfachung auf Basis freier Punktwahl

Bei einigen Verfahren können neue Punkte hinzugenommen werden, die nicht aus der Punktmenge des Ursprungspolygonzuges stammen. Regressionsverfahren, wie sie Palpanas u. a. [Pal+04] und Manis u. a. [MPT97] beschreiben und untersuchen, sind eine Klasse von Verfahren, die neue Punkte verwenden. Kaya implementierte in [Kay05] ein Regressionsverfahren (siehe Kapitel 4.4).

Imai und Iri [II88] beschreiben ein weiteres Verfahren, bei dem neue Punkte verwendet werden. Bei ihnen werden die Punkte durch Rechtecke der Breite 2ϵ abgedeckt, innerhalb dieser dann der vereinfachte Teilpolygonzug liegen muss.

5.2.3. Mustererkennung

In der Mustererkennung ist die Darstellung von Formen von grundlegender Bedeutung. Sie dient als Basis für die weiteren Schritte wie z. B. das Erkennen von Formen [LC88]. Verfahren zur Polygonzugvereinfachung werden hier eingesetzt, um die Formen so zu vereinfachen, dass deren Charakteristik erhalten bleibt. Dabei kommen ähnliche Verfahren wie in der Kartographie zum Einsatz. Ein dezimierendes Verfahren wird von Leu und Chen [LC88] vorgestellt und von Boxer u. a. [Box+93] verfeinert. Dieses Verfahren wird in Kapitel 6.3 auf Seite 45 beschrieben und untersucht.

5.3. Anwendbarkeit auf räumlich-zeitliche Polygonzüge

Fast alle der genannten Verfahren arbeiten auf Polygonzügen im zwei- oder dreidimensionalen euklidischen Raum ($\mathbb{R}^2$ und $\mathbb{R}^3$). Daher ist es möglich, die Distanzen mit der euklidischen Abstandsformel zu berechnen.

In Nexus wird dagegen mit Positionsdaten im $\mathbb{R}^2$ gearbeitet, die durch die Zeit als dritte Dimension erweitert werden. Dadurch ergibt sich ein dreidimensionaler Raum, in dem spezielle räumlich-zeitliche Distanzmaße verwendet werden können (siehe Kapitel 3.1). Die speziellen Distanzmaße können insbesondere berücksichtigen, dass die Zeitdimension eine andere Einheit hat als die beiden anderen, und dass die Zeitdimension monoton ist.

Aufgrund des speziellen räumlich-zeitlichen (x,y,t)-Raums sind die meisten Verfahren aus der Literatur nicht direkt auf räumlich-zeitliche Polygonzüge anwendbar. Einige Verfahren lassen sich aber leicht anpassen, indem räumlich-zeitliche Distanzmaße wie die Zeitgleichheitsdistanz E_u eingeführt werden. Für andere Verfahren muss der dreidimensionale (x,y,t)-Raum in zwei zweidimensionale (x,t) und (y,t)-Räume aufgeteilt werden. Diese Teilräume werden dann getrennt voneinander betrachtet.

Kapitel 6

Ausgewählte Algorithmen zur Polygonzugvereinfachung

In diesem Kapitel werden die von mir implementierten und analysierten Algorithmen zu Polygonzugvereinfachung beschrieben und klassifiziert.

6.1. Verwerfend-strombasiertes Verfahren: Piecewise Linear Approximation with Zoning Angle

Das PLAZA (Piecewise Linear Approximation with Zoning Angle) Verfahren von Soroush u. a. [SWP08] ist auf zweidimensionale Sensordaten ausgelegt, von denen eine Dimension die Zeit ist. Diese Dimension muss monoton sein. Soroush u. a. komprimieren mit dem PLAZA Verfahren Sprachdatenströme von verteilten Mikrofonen.

Das Verfahren erzeugt aus Punkten des ursprünglichen Polygonzuges einen Winkel, in dem ein Abschnitt des vereinfachten Polygonzuges liegen muss, um für alle Punkte den maximal erlaubten Fehler nicht zu überschreiten. Mit jedem neu hinzukommenden Punkt kann dieser Winkel kleiner werden. Kann ein neuer Punkt nicht mit einem Abschnitt innerhalb des Winkels dargestellt werden, wird der Abschnitt gespeichert und ein neuer Winkel mit einem neuen Abschnitt wird erzeugt.

Im Folgenden wird zuerst der zweidimensionale Algorithmus beschrieben. Später wird erläutert, wie man das zweidimensionale Verfahren anpassen muss, um es auf den zeitlich-räumlichen (x,y,t)-Raum anwenden zu können.

6.1.1. Verfahren

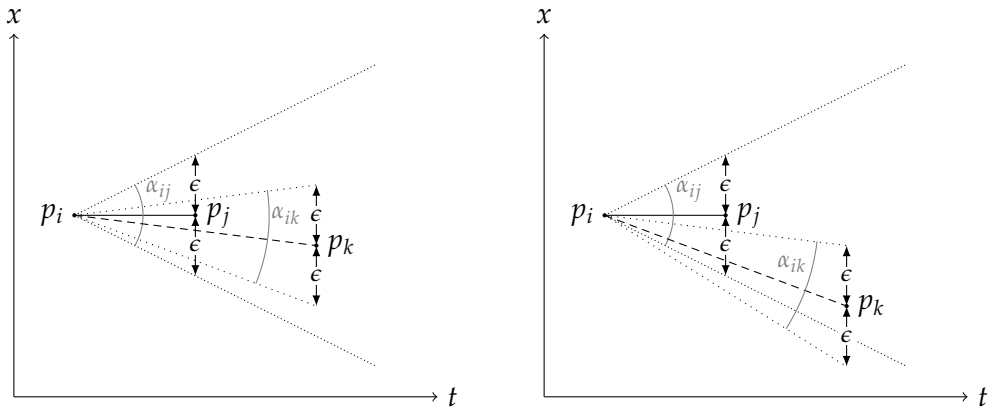
Durch zwei Punkte p_i und p_j des Polygonzuges und den maximal erlaubten Fehler ϵ an p_j entsteht ein Winkel α_{ij}, innerhalb dessen der Abschnitt des vereinfachten Polygonzuges liegen muss.

Nimmt man einen dritten Punkt p_k dazu, können folgende fünf Situationen auftreten:

1. p_k liegt zusammen mit seiner ϵ-Umgebung innerhalb von α_{ij}.
2. p_k liegt innerhalb von α_{ij}, die ϵ-Umgebung ragt auf einer Seite teilweise darüber hinaus.
3. p_k liegt außerhalb von α_{ij}, die ϵ-Umgebung ragt auf einer Seite teilweise hinein.
4. p_k liegt zusammen mit seiner ϵ-Umgebung außerhalb von α_{ij}.
5. p_k liegt innerhalb von α_{ij}, die ϵ-Umgebung ragt auf beiden Seiten darüber hinaus.

1. Fall und 2. Fall

Liegt der Punkt p_k innerhalb von α_{ij} und seine ϵ-Umgebung teilweise oder komplett innerhalb von α_{ij}, dann schränkt er den Winkel ein. Der neue Winkel ist dann $\alpha_{ij} \cap \alpha_{ik}$. Die Vereinfachung wird durch den Abschnitt zwischen den Punkten p_i und p_k gebildet (siehe Abbildungen 6.1a bis 6.1b auf dieser Seite).

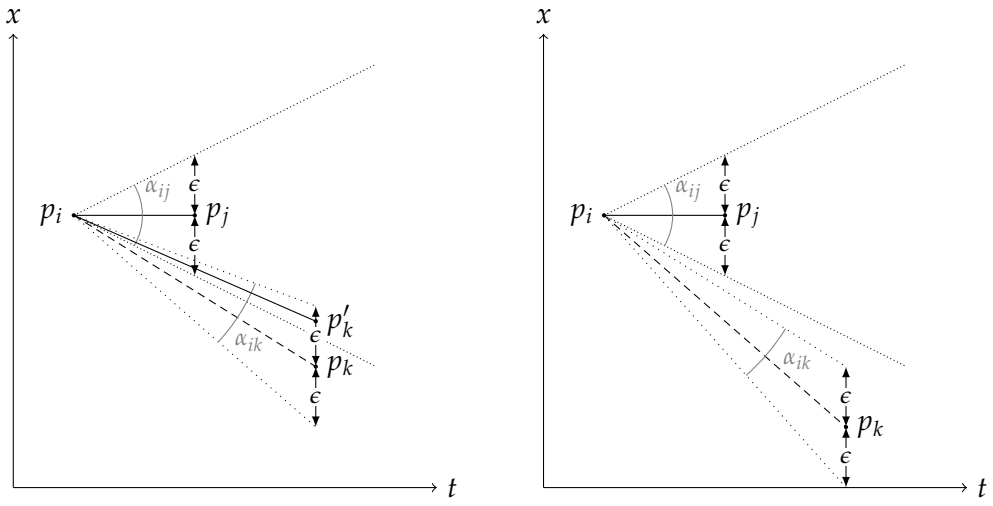


(a) 1. Fall (b) 2. Fall

Abbildung 6.1.: Der erste und der zweite Fall des PLAZA Algorithmus.

3. Fall

Liegt ein Punkt p_k außerhalb von α_{ij}, seine ϵ-Umgebung aber teilweise innerhalb des Winkels muss der Winkel speziell angepasst werden. Der Bereich, in dem die Vereinfachung liegen darf, bildet sich durch den Schnitt des aktuellen Winkels α_{ij} mit dem Winkel α_{ik}, der durch den Punkt p_i und den aktuellen Punkt p_k mit seiner ϵ-Umgebung gebildet wird. Da der Punkt p_k außerhalb des neuen, erlaubten Winkels $\alpha_{ij} \cap \alpha_{ik}$ liegt, muss ein anderer Stützpunkt für den Abschnitt des vereinfachten Polygonzuges gefunden werden. Dieser Punkt p'_k liegt genau auf der Winkelhalbierenden des neuen Winkels. Der bisherige Endpunkt der vereinfachten Polygonzuges p_j kann nun entfernt und durch p'_k ersetzt werden (siehe Abbildung 6.2a).



(a) 3. Fall

(b) 4. Fall

Abbildung 6.2.: Der dritte und der vierte Fall des PLAZA Algorithmus.

4. Fall

Liegt der Punkt p_k mit seiner ϵ-Umgebung außerhalb des Winkels α_{ij}, so kann er durch den aktuellen Abschnitt nicht dargestellt werden (siehe Abbildung 6.2b). Nun wird der Abschnitt zwischen den Punkten p_i und p_j ausgegeben und mit einem neuen Winkel begonnen. Startpunkt ist dabei p_j, der zusammen mit p_k den neuen Winkel α_{jk} bildet.

5. Fall

Liegt der Punkt p_k innerhalb des Winkels α_{ij}, seine ϵ-Umgebung aber vollständig außerhalb dieses Winkels, muss der Winkel nicht weiter angepasst werden. Der aktuelle Abschnitt der Vereinfachung wird dann durch den neuen Abschnitt zwischen den Punkten p_i und p_k gebildet (siehe Abbildung 6.3).

6.1.2. Anpassungen für den dreidimensionalen räumlich-zeitlichen Raum

Das PLAZA-Verfahren arbeitet nur auf zweidimensionalen Daten im (x,t)-Raum, daher muss es für die Verwendung im zeitlich-räumlichen (x,y,t)-Raum angepasst werden. Zwei Anpassungsmöglichkeiten werden im Folgenden beschrieben:

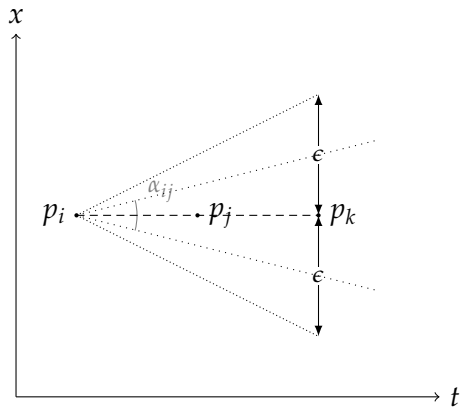


Abbildung 6.3.: Der fünfte Fall des PLAZA Algorithmus.

Dreidimensionale Kegel

Im ersten Fall stellt man die Winkel im dreidimensionalen räumlich-zeitlichen (x,y,t)-Raum mithilfe von Kegeln dar. Zur Beschränkung des Winkels müssen dann Kegel- und Kreisschnitte berechnet werden. Als Ergebnis dieser Schnittoperationen entstehen komplexe Flächen im dreidimensionalen (x,y,t)-Raum, die sich nicht durch geschlossene Formeln darstellen lassen. Daher ist diese Variante zu komplex und rechenintensiv und wird nicht weiter verfolgt.

Zerlegung in zwei zweidimensionale Räume

Die zweite Variante zerlegt den (x,y,t)-Raum in die zwei zweidimensionalen (x,t)- und (y,t)-Räume und arbeitet in diesen mit dem ursprünglichen PLAZA Algorithmus. Das entspricht der Verwendung von Pyramiden anstelle von Kegeln.

Durch diese Zerlegung des (x,y,t)-Raums vereinfacht sich die Berechnung des Winkels in dem die Vereinfachung liegen muss. Dabei tritt aber ein Problem auf:

Setzt man die beiden zweidimensionalen Winkel zusammen, entsteht im dreidimensionalen räumlich-zeitlichen (x,y,t)-Raum eine Pyramide mit einer rechteckigen Grundfläche, deren maximale Kantenlänge jeweils durch 2ϵ beschränkt ist.

Betrachten wir zunächst den Spezialfall einer quadratischen Grundfläche dieser Pyramide (siehe Abbildung 6.4):

In diesem Fall ist die Kantenlänge des Quadrats 2ϵ. Der Inkreis dieses Quadrates hat den Radius ϵ. Liegt ein Punkt in der zweidimensionalen Darstellung innerhalb beider Winkel, kann er zusammen mit seiner ϵ-Umgebung im dreidimensionalen räumlich-zeitlichen (x,y,t)-Raum teilweise im Bereich außerhalb des Inkreises liegen (beispielsweise der Punkt p_k in Abbildung 6.4 auf der nächsten Seite). Sein Abstand zum vereinfachten Polygonzug ist dann größer als der maximal erlaubte Abstand ϵ.

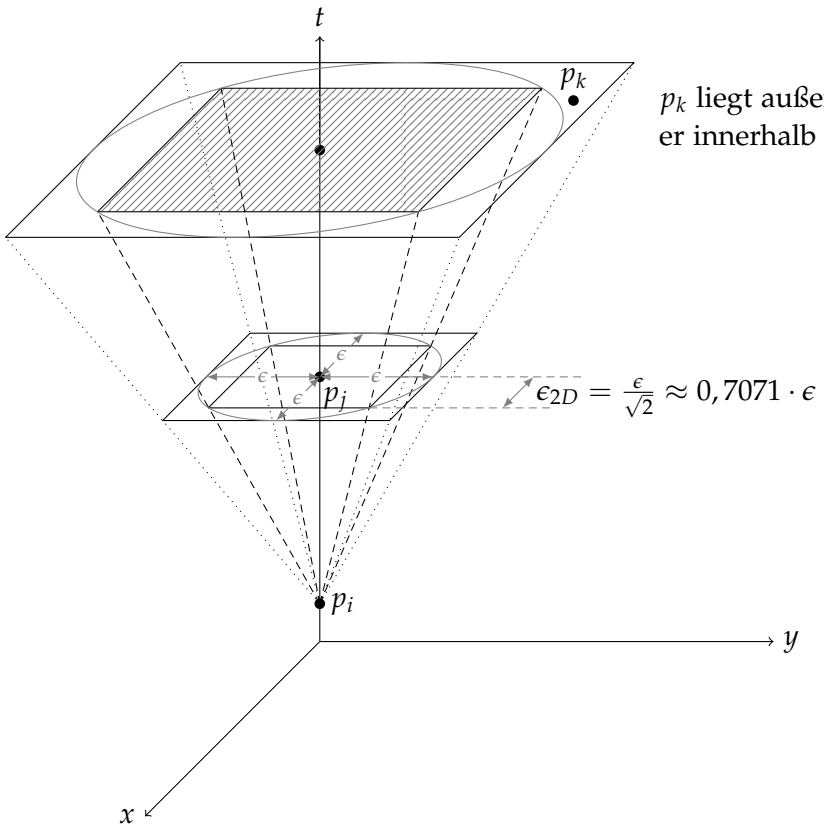


Abbildung 6.4.: Probleme des PLAZA Algorithmus

Das Problem lässt sich dadurch lösen, dass man einen kleineren Wert für ϵ wählt. Möchte man eine Vereinfachung mit einem maximal erlaubten Fehler ϵ erzeugen, so wählt man als maximal erlaubten Fehler in der zweidimensionalen Darstellung $\epsilon_{2D} = \frac{\epsilon}{\sqrt{2}}$.

Dadurch erhält man im dreidimensionalen räumlich-zeitlichen (x,y,t)-Raum ein Quadrat, dass im Kreis mit dem Radius ϵ eingeschlossen ist (das schraffierte Quadrat in Abbildung 6.4). Man verliert dadurch allerdings $1 - \frac{\pi}{4} \approx 21,5\,\%$ der Fläche des Quadrates. Dadurch werden einige Punkte ungünstigerweise aussortiert. Der vereinfachte Polygonzug enthält daher eventuell mehr Punkte als nötig.

Für den allgemeinen Fall einer rechteckigen, nicht quadratischen Grundfläche entstehen durch die Rechtecke der maximalen Kantenlänge $2\epsilon_{2D} = 2\frac{\epsilon}{\sqrt{2}} = \sqrt{2}\,\epsilon$ rechteckige Schnittflächen. Diese garantieren ebenso, dass der maximal erlaubte Fehler ϵ eingehalten wird.

6.1.3. Eigenschaften des dreidimensionalen PLAZA Algorithmus

Arbeitsweise

Das Verfahren arbeitet verwerfend-strombasiert. Es testet, ob ein Punkt mit dem aktuellen Abschnitt darstellbar ist und schränkt den aktuellen Winkel ein, wenn dies nötig ist. Danach wird der Punkt verworfen. Das Verfahren speichert nur den Start- und Endpunkt des aktuellen Abschnitts und den oberen und unteren Winkel. Dadurch belegt es wenig Speicherplatz.

Komplexität

Da jeder Punkt nur einmal betrachtet wird, hat das Verfahren eine Zeitkomplexität von $O(n)$.

Maximaler Fehler

Das Verfahren erzeugt Vereinfachungen mit einem garantierten maximalen Fehler von ϵ. Da bei der Aufteilung des (x,y,t)-Raums in die beiden zweidimensionalen Räume der maximal erlaubte Fehler ϵ immer orthogonal zur Zeitachse überprüft wird, entspricht dieses Distanzmaß der Zeitgleichheitsdistanz E_u (siehe Kapitel 3.1.3).

Alterungsstabilität

Das PLAZA Verfahren ist nicht alterungsstabil, da das Verfahren für die Vereinfachung teilweise Punkte erzeugt, die nicht auf dem Ursprungspolygonzug liegen. Durch einen größeren maximal erlaubten Fehler werden deshalb völlig andere Punkte erzeugt, als durch schrittweise Vereinfachung.

Fehlerbeschränktheit bei mehrfacher Vereinfachung

Das Verfahren berechnet die Abstände mit der Zeitgleichheitsdistanz E_u. Um einen beschränkten Fehler bei mehrfacher Vereinfachung zu garantieren, muss der maximal erlaubte Fehler des vorausgegangenen Vereinfachungsschritts bekannt sein.

Das PLAZA Verfahren beschränkt den maximalen Fehler genau dann bei mehrfacher Vereinfachung, wenn bei der erneuten Vereinfachung der maximal erlaubte Fehler der vorhergehenden Vereinfachung vom aktuellen maximal erlaubten Fehler abgezogen wird (siehe auch Kapitel 4.1.2 auf Seite 24).

6.2. Verfeinernd-fensterbasiertes Verfahren von Douglas und Peucker

6.2.1. Allgemeines Prinzip

Der Algorithmus von Douglas und Peucker [DP73] ist ein verfeinernd-fensterbasiertes Verfahren. Es komprimiert jeweils den Teilpolygonzug der innerhalb des Fensters liegt.

Der Algorithmus startet damit, dass er den ersten und letzten Punkt des Teilpolygonzuges zum vereinfachten Teilpolygonzug hinzufügt. Nun werden die Abstände aller Punkte zum Geradenabschnitt durch den ersten und letzten Punkt betrachtet. Der Punkt, der den größten Abstand zu diesem Geradenabschnitt hat, wird zum komprimierten Teilpolygonzug hinzugefügt, wenn sein Abstand größer als ϵ ist. Dadurch teilt er den Polygonzug in zwei Teile, auf denen das Verfahren erneut gestartet wird.

Das Douglas-Peucker Verfahren arbeitet mit einer beliebigen Distanzfunktion. Der Einfachheit halber wird er im Folgenden unter Verwendung der zweidimensionalen euklidischen Distanz im $\mathbb{R}^2$ beschrieben.

6.2.2. Verfahren

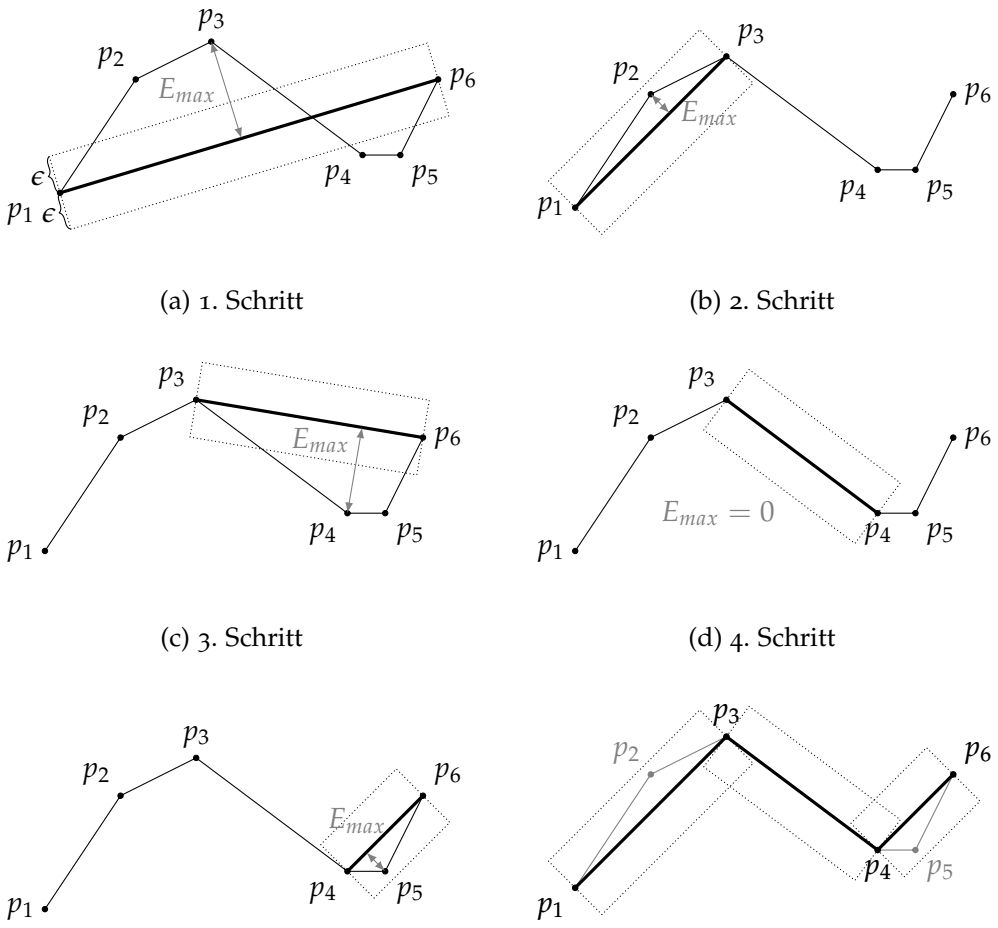
Man wählt den ersten und letzten Punkt des Fensters als Start- und Endpunkt der Vereinfachung (siehe Abbildung 6.5a) und berechnet für alle Punkte dazwischen den Abstand E zu dieser Vereinfachung. Dann bestimmt man den Punkt mit dem maximalen Abstand E_{max}. Ist dieser Punkt näher als ϵ an der Vereinfachung, nähert diese den Teilpolygonzug ausreichend gut an. Ist der Abstand größer als ϵ, wird dieser Punkt zur Vereinfachung hinzugefügt und der Algorithmus auf der linken und rechten Seite (siehe Abbildungen 6.5b bis 6.5c auf der nächsten Seite) erneut gestartet [DP73].

Das Verfahren terminiert, wenn alle Punkte einen Abstand kleiner als ϵ von der Vereinfachung haben (siehe Abbildung 6.5f).

6.2.3. Verbesserungsmöglichkeiten

In dieser Arbeit wurde eine einfache rekursive Variante des Douglas-Peucker Verfahrens implementiert. Für das Verfahren gibt es in der Literatur aber eine Vielzahl verschiedener Implementierungen. So wird z. B. in [HS92] die worst-case Komplexität auf $\Theta(n\ log_2\ n)$ gesenkt, indem die Punkte mit maximalen Abstand zur aktuellen Vereinfachung auf der konvexen Hülle des Pfades gesucht werden.

Zudem kann die Verwendung von quadrierten Distanzen und quadrierten ϵ-Werten die Berechnung zusätzlich beschleunigen, da keine Quadratwurzeln berechnet werden müssen.



(a) 1. Schritt (b) 2. Schritt

(c) 3. Schritt (d) 4. Schritt

(e) 5. Schritt (f) Ergebnis

Abbildung 6.5.: Durchführung des Douglas-Peucker Algorithmus mit der euklidischen Distanz im $\mathbb{R}^2$ an einem Polygonzug mit sechs Punkten.

6.2.4. Anpassungen für den dreidimensionalen räumlich-zeitlichen Raum

Das Douglas-Peucker Verfahren schreibt keine Distanzfunktion vor und lässt sich daher im dreidimensionalen (x,y,t)-Raum mit der Zeitgleichheitsdistanz E_u (siehe Kapitel 3.1.3) einsetzen [CWT06].

6.2.5. Eigenschaften des dreidimensionalen Douglas-Peucker Algorithmus

Arbeitsweise

Der Algorithmus arbeitet fensterbasiert. Bei Erreichen der Fenstergröße wird der Teilpolygonzug innerhalb des Fensters vereinfacht. Der komprimierte Teilpolygonzug wird ausgegeben und das Fenster geleert. Danach sammelt das Verfahren wieder Punkte bis zum Erreichen der Fenstergröße.

Komplexität

Der Algorithmus hat eine worst-case Zeitkomplexität von $O(n^2)$. Diese kommt dann zustande, wenn in jedem Teilungsschritt der Rekursion der Punkt mit dem größten Abstand zum komprimierten Geradenabschnitt an einem Ende des zu vereinfachenden Polygonzuges liegt und der Abstand dieses Punktes größer als ϵ ist. Dann muss für jeden Teilungsschritt für alle Punkte die Distanz zu einem neuen Geradenabschnitt neu berechnet werden. Der Algorithmus ähnelt in seinem Aufbau stark dem Quicksort-Algorithmus und weist wie dieser eine average-case Zeitkomplexität von $O(n \log n)$ auf.

Maximaler Fehler

Das Verfahren erzeugt Vereinfachungen mit einem garantierten maximalen Fehler von ϵ.

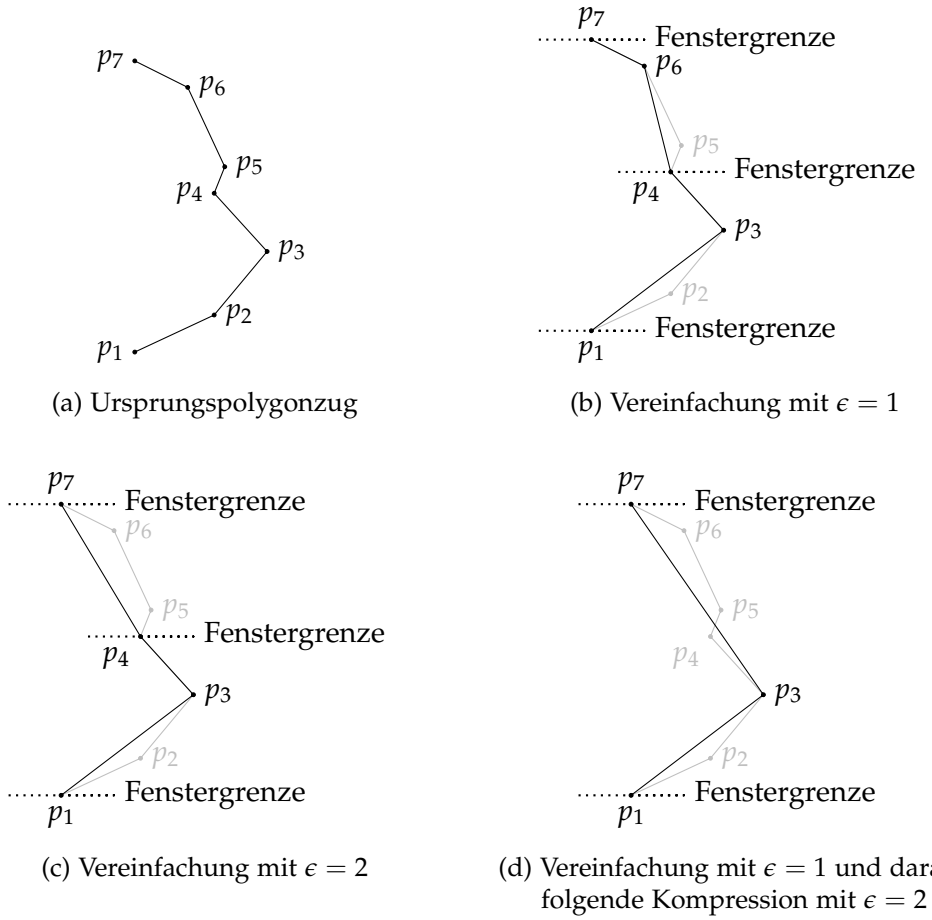
Alterungsstabilität

Cao, Wolfson u. a. beweisen in [CWT06], dass das Douglas-Peucker Verfahren auf dem gesamten Polygonzug für alle Distanzfunktionen alterungsstabil ist.

Nutzt man das Verfahren mit einer Fenstergröße, die kleiner als die Größe des Polygonzuges ist, ist das Verfahren nicht mehr alterungsstabil.

Bei der Vereinfachung des ursprünglichen Polygonzuges mit einem kleinen und einem großen maximal erlaubten Fehler werden zwei vereinfachte Polygonzüge erzeugt. Das Verfahren arbeitet dabei in beiden Fällen auf denselben Fenstern. Der Polygonzug, der mit dem großen Fehler erzeugt wurde, beinhaltet aufgrund der Vorgehensweise des Algorithmus alle Punkte des mit dem kleinen Fehler erzeugten Polygonzuges (siehe Abbildungen 6.6a bis 6.6c auf der nächsten Seite)

Wird nun der mit dem kleinen Fehler erzeugte Polygonzug erneut vereinfacht, verschieben sich die Grenzen zwischen den Fenstern (siehe Abbildung 6.6d). Nun ist es möglich, dass bei der Kompression mit dem großen Fehler die Punkte ausgelassen werden, die im ersten Vereinfachungsschritt Randpunkte der Fenster waren, beispielsweise der Punkt p_4 in Abbildung 6.6. Das ist möglich, da diese Punkte durch die Verwendung der Fenster in den



(a) Ursprungspolygonzug

(b) Vereinfachung mit $\epsilon = 1$

(c) Vereinfachung mit $\epsilon = 2$

(d) Vereinfachung mit $\epsilon = 1$ und darauffolgende Kompression mit $\epsilon = 2$

Abbildung 6.6.: Alterung eines Polygonzuges mit dem Douglas-Peucker Verfahren

vereinfachten Polygonzug aufgenommen wurden und nicht durch zu großen Abstand zu einem Geradenabschnitt.

Fehlerbeschränktheit bei mehrfacher Vereinfachung

Das Verfahren ist bei mehrfacher Vereinfachung fehlerbeschränkt. Bei einer Fenstergröße, die größer als die Größe des Polygonzuges ist, ist das Verfahren alterungsstabil und damit auch fehlerbeschränkt bei mehrfacher Vereinfachung. Durch kleinere Fenstergrößen werden dem vereinfachten Polygonzug an den Fenstergrenzen zusätzliche Punkte hinzugefügt. Dadurch verliert das Verfahren zwar seine Alterungsstabilität, die Fehlerbeschränktheit wird durch dieses Verhalten aber nicht beeinflusst, da der vereinfachte Polygonzug sogar aus mehr Punkten besteht, als für die Fehlerbeschränktheit nötig wären.

6.3. Dezimierend-fensterbasiertes Verfahren von Boxer, Leu und Chen

Das dezimierend-fensterbasierte Vereinfachungsverfahren von Boxer, Leu und Chen [LC88; Box+93] kann eine beliebige Distanzfunktion für die Abstandsberechnung verwenden und ist deshalb für die Zeitgleichheitsdistanz geeignet. Es wird in der Literatur zur Vereinfachung von Polygonzügen zur Mustererkennung genutzt.

Durch diesen Fokus arbeitet das Verfahren ursprünglich auf geschlossenen zweidimensionalen Polygonzügen. Es lässt sich aber einfach auf offene Polygonzüge im dreidimensionalen (x,y,t)-Raum anwenden. Der Einfachheit halber wird das Verfahren im Folgenden mithilfe der zweidimensionalen euklidischen Distanz im $\mathbb{R}^2$ beschrieben.

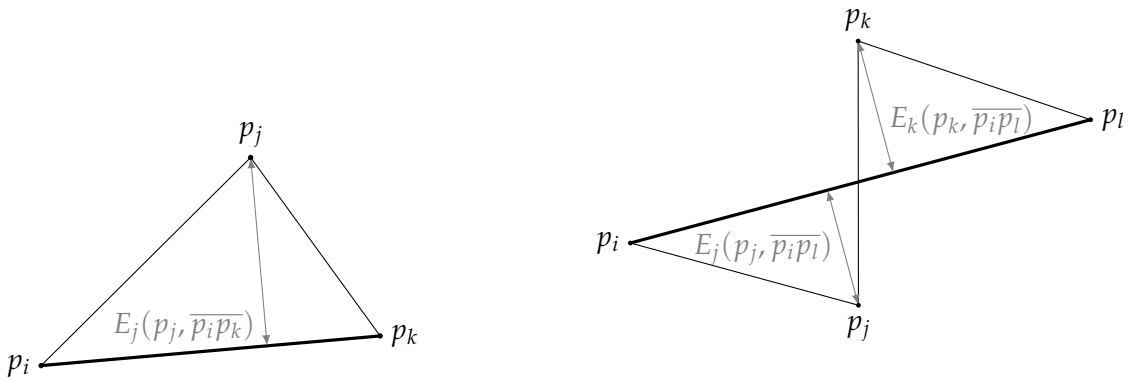
6.3.1. Definitionen

Vor der Erläuterung des Verfahrens müssen einige Begriffe definiert werden.

Maximale Abweichung des Polygonzuges zur Vereinfachung

Maximale Abweichungen des Polygonzuges zu einer Vereinfachung treten immer an den äußeren Eckpunkten auf. Die Abbildungen 6.7a bis 6.7b auf dieser Seite zeigen einen Polygonzug mit zwei bzw. drei Kanten. Seien $E_j(p_j, \overline{p_i p_l})$ und $E_k(p_k, \overline{p_i p_l})$ die kürzesten Abstände der Punkte p_j und p_k von der Vereinfachungskante $\overline{p_i p_l}$, dann ist die maximale Abweichung des Polygonzuges zur Vereinfachungskante definiert als:

$$E_{max} = max(E_j, E_k)$$



(a) zwei Kanten (b) drei Kanten

Abbildung 6.7.: Abstand der Vereinfachungskante zu den Eckpunkten.

Für einen Polygonzug mit zwei Kanten ist der Abstand $E_{max} = E_j$. Die Formel lässt sich auch für größere Punktmengen anpassen, in diesem Verfahren wird sie aber nur für ein und zwei Punkte benötigt.

Benachbarte Abschnitte

Man betrachtet einen Abschnitt A des Polygonzuges. Die Nachbarabschnitte $A_{Nachbar}(A)$ sind definiert als die Abschnitte, die sich vollständig oder teilweise mit dem Abschnitt A überlappen:

(1) Besteht A aus drei Kanten $\overline{p_ip_j}$, $\overline{p_jp_k}$ und $\overline{p_kp_l}$, dann sind die Nachbarabschnitte $A_{Nachbar}(A) = \{ (\overline{p_{i-2}p_{i-1}}, \overline{p_{i-1}p_i}, \overline{p_ip_j}), (\overline{p_{i-1}p_i}, \overline{p_ip_j}), (\overline{p_{i-1}p_i}, \overline{p_ip_j}, \overline{p_jp_k}), (\overline{p_ip_j}, \overline{p_jp_k}), (\overline{p_jp_k}, \overline{p_kp_l}), (\overline{p_jp_k}, \overline{p_kp_l}, \overline{p_lp_{l+1}}), (\overline{p_kp_l}, \overline{p_lp_{l+1}}), (\overline{p_kp_l}, \overline{p_lp_{l+1}}, \overline{p_{l+1}p_{l+2}})\}$ (siehe Abbildung 6.8).

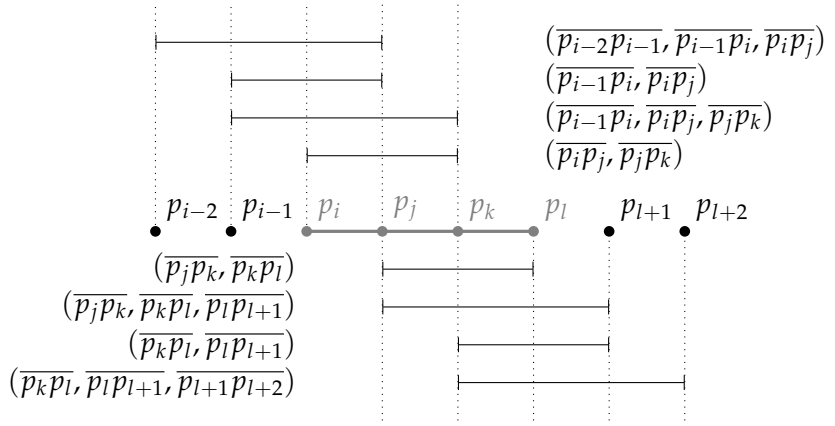


Abbildung 6.8.: Die Nachbarabschnitte von $A = \{(\overline{p_ip_j}, \overline{p_jp_k}, \overline{p_kp_l})\}$.

(2) Besteht A aus zwei Kanten $\overline{p_ip_j}$ und $\overline{p_jp_k}$, dann sind die Nachbarabschnitte $A_{Nachbar}(A) = \{ (\overline{p_{i-2}p_{i-1}}, \overline{p_{i-1}p_i}, \overline{p_ip_j}), (\overline{p_{i-1}p_i}, \overline{p_ip_j}), (\overline{p_{i-1}p_i}, \overline{p_ip_j}, \overline{p_jp_k}), (\overline{p_ip_j}, \overline{p_jp_k}, \overline{p_kp_{k+1}}), (\overline{p_jp_k}, \overline{p_kp_{k+1}}), (\overline{p_jp_k}, \overline{p_kp_{k+1}}, \overline{p_{k+1}p_{k+2}})\}$ (siehe Abbildung 6.9).

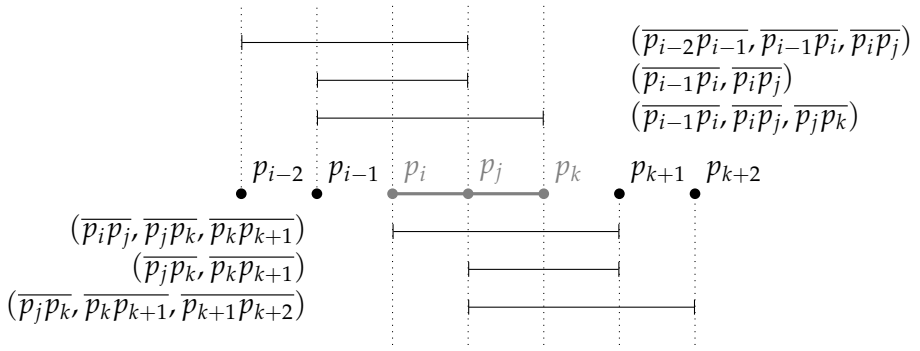


Abbildung 6.9.: Die Nachbarabschnitte von $A = \{(\overline{p_ip_j}, \overline{p_jp_k})\}$.

Abschnitt mit der kleinsten lokalen Abweichung (AklA)

Ein Abschnitt A mit zwei oder drei Kanten wird Abschnitt mit der kleinsten lokalen Abweichung (AklA) genannt, wenn sein maximaler Abstand zur Vereinfachungskante kleiner ist, als der seiner Nachbarabschnitte $A_{Nachbar}(A)$.

6.3.2. Anpassung für offene Polygonzüge

Bei geschlossenen Polygonzügen hat jeder Abschnitt auf beiden Seiten Nachbarabschnitte. Damit lässt sich sehr einfach der Abschnitt mit der kleinsten lokalen Abweichung bestimmen, da es keine Sonderfälle gibt. Ist der Polygonzug offen, haben am Anfang und Ende des Polygonzuges einige Abschnitte auf einer Seite keine oder zu wenige Nachbarabschnitte.

Bei diesen Abschnitten können nur die vorhandenen Nachbarabschnitte genutzt werden, sie müssen daher als Sonderfälle separat behandelt werden.

6.3.3. Verfahren

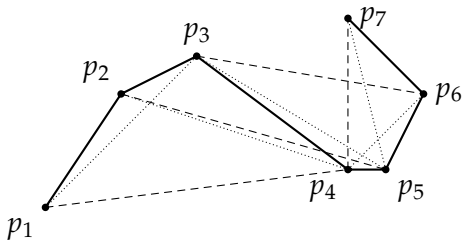
Das Verfahren iteriert über die Kanten des Polygonzuges. Bei jeder Kante wird für jeden mit dieser Kante beginnenden und aus zwei oder drei Kanten bestehenden Abschnitt überprüft, ob es sich um einen Abschnitt mit der kleinsten lokalen Abweichung handelt. Ist dies der Fall und ist der Abstand kleiner als der vorgegebene maximal erlaubte Fehler ϵ, wird der Abschnitt zur Vereinfachung vorgemerkt.

Ist man mit allen Kanten fertig, werden alle vorgemerkten Abschnitte vereinfacht, wobei mit den aus drei Kanten bestehenden Abschnitte begonnen wird, da diese zu einer schnelleren Reduktion der Kantenanzahl führen. Auf der neuen Kantenmenge wird das Verfahren erneut ausgeführt, bis sich von einer Iteration zur nächsten nichts mehr ändert [LC88].

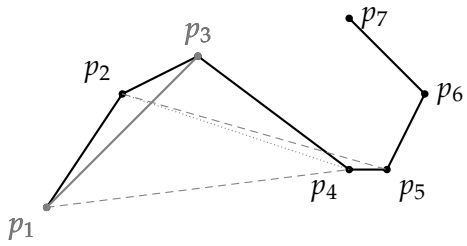
6.3.4. Beispiel

In den Abbildungen 6.10a bis 6.11f auf den Seiten 48–49 wird die erste Iteration an einem Beispielpolygonzug durchgeführt. Zuerst werden alle Abschnitte mit zwei und drei Kanten erzeugt (siehe Abbildung 6.10a). Nun werden von $\overline{p_1p_2}$ ausgehend alle Kanten betrachtet. Man untersucht ob es sich bei einem der beiden Abschnitten $((\overline{p_1p_2}, \overline{p_2p_3}, \overline{p_3p_4}), (\overline{p_1p_2}, \overline{p_2p_3}))$ um einen Abschnitt mit der kleinsten lokalen Abweichung (AklA) handelt (siehe Abbildung 6.10b).

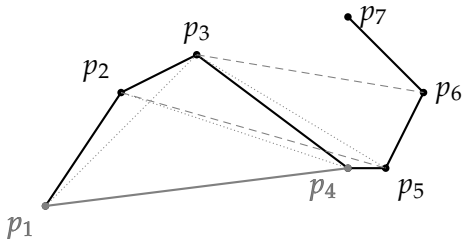
Der Abstand von p_2 zu $\overline{p_1p_3}$ ist kleiner, als der Abstand der zu den sie einschließenden Nachbarabschnitten $A_{Nachbar}(A) = \{(\overline{p_1p_2}, \overline{p_2p_3}, \overline{p_3p_4}), (\overline{p_2p_3}, \overline{p_3p_4}, \overline{p_4p_5}), (\overline{p_2p_3}, \overline{p_3p_4})\}$. Außerdem ist der Abstand kleiner als ϵ. Deshalb ist der Abschnitt $(\overline{p_1p_2}, \overline{p_2p_3})$ ein AklA. Er wird daher zur Vereinfachung vorgemerkt.



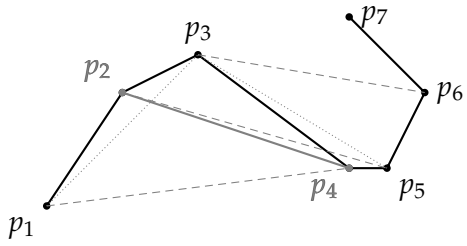
(a) 1. Schritt: Die aus zwei oder drei Kanten bestehenden Abschnitte



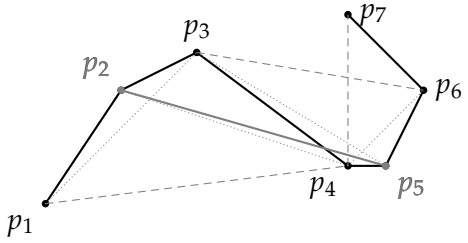
(b) 2. Schritt: $\overline{p_1p_3}$ ist ein AklA.



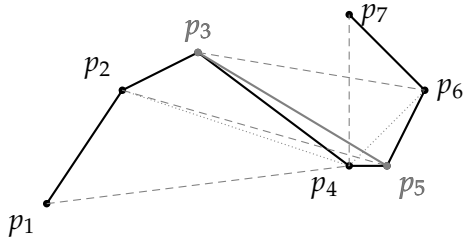
(c) 3. Schritt: $\overline{p_1p_4}$



(d) 4. Schritt: $\overline{p_2p_4}$



(e) 5. Schritt: $\overline{p_2p_5}$

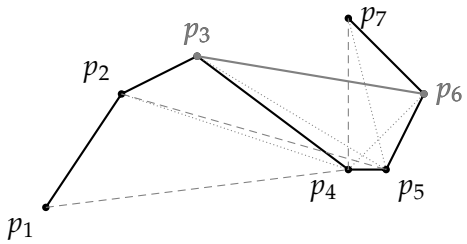


(f) 6. Schritt: $\overline{p_3p_5}$ ist ein AklA.

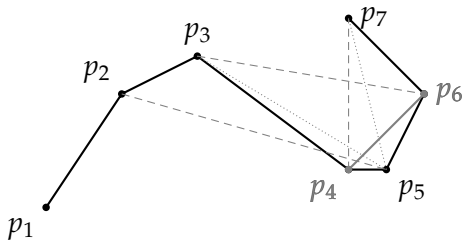
Abbildung 6.10.: Durchführung des Algorithmus von Boxer, Leu und Chen (1). Beispiel an einem Polygonzug mit sechs Punkten mit der euklidischen Distanz im $\mathbb{R}^2$.

Dieser Schritt wird für alle weiteren Abschnitte durchgeführt (siehe Abbildungen 6.10c bis 6.11d auf den Seiten 48–49).

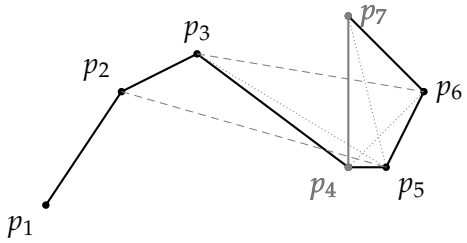
Die beiden in der vorangegangenen Schritten vorgemerkten AklA $(\overline{p_1p_2}, \overline{p_2p_3})$ und $(\overline{p_3p_4}, \overline{p_4p_5})$ können nun in einem nachfolgenden Schritt vereinfacht werden (siehe Abbildung 6.11e). Auf dem dadurch vereinfachten Polygonzug (siehe Abbildung 6.11f) wird nun wieder von vorne begonnen. Das Verfahren bricht ab, wenn keine weitere Vereinfachung mehr stattgefunden hat.



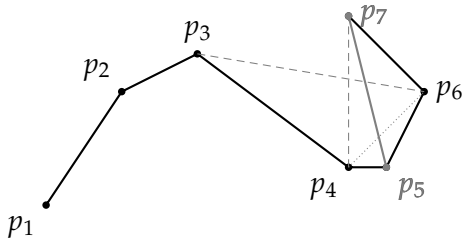
(a) 7. Schritt: $\overline{p_3p_6}$



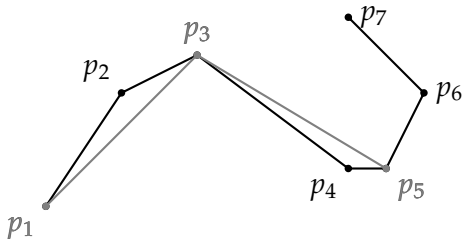
(b) 8. Schritt: $\overline{p_4p_6}$



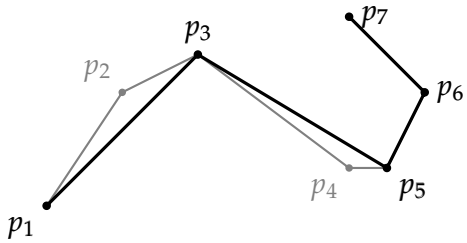
(c) 9. Schritt: $\overline{p_4p_7}$



(d) 10. Schritt: $\overline{p_5p_7}$



(e) 11. Schritt: Zur Vereinfachung markierte Abschnitte



(f) 12. Schritt: Der Polygonzug nach dem ersten Durchlauf

Abbildung 6.11.: Durchführung des Algorithmus von Boxer, Leu und Chen (2). Beispiel an einem Polygonzug mit sechs Punkten mit der euklidischen Distanz im $\mathbb{R}^2$.

6.3.5. Verbesserungsmöglichkeiten

In [Box+93] wird das Verfahren durch einen Flächenschwellenwert erweitert, der dafür sorgt, dass die Vereinfachung genauer bleibt, wenn durch die Vereinfachung ein zu großer Flächenunterschied zwischen ursprünglichem und vereinfachtem geschlossenem Polygonzug entsteht. Das ist im Fall eines offenen Polygonzuges nicht möglich, da keine geschlossene Fläche existiert.

Man könnte aber die Flächen zwischen dem ursprünglichen Polygonzug und seiner Vereinfachung berechnen. Durch die Erweiterung um einen Flächenschwellenwert wird der vereinfachte Polygonzug größer als dies für die Einhaltung des maximal erlaubten Fehlers nötig wäre. Deshalb wurde diese Idee nicht weiter verfolgt.

6.3.6. Anpassung für den dreidimensionalen räumlich-zeitlichen Raum

Da das Verfahren die verwendbare Distanzfunktion nicht einschränkt, lässt es sich im zeitlich-räumlichen (x,y,t)-Raum mit der Zeitgleichheitsdistanz E_u (siehe Kapitel 3.1.3) einsetzen.

6.3.7. Eigenschaften des dreidimensionalen Algorithmus von Boxer, Leu und Chen

Arbeitsweise

Der Algorithmus arbeitet fensterbasiert. Bei Erreichen der Fenstergröße wird der Teilpolygonzug im Fenster vereinfacht. Der vereinfachte Teilpolygonzug wird ausgegeben und das Fenster geleert. Danach sammelt das Verfahren wieder Punkte bis zum Erreichen der Fenstergröße.

Komplexität

Der Algorithmus iteriert solange immer wieder über die Punktmenge, bis kein Abschnitt mehr vereinfacht werden kann. In jeder Iteration muss er alle, noch vorhandenen Abschnitte darauf hin untersuchen ob diese ein Abschnitt mit der kleinsten lokalen Abweichung sind. Der Algorithmus hat damit eine Zeitkomplexität von $O(n^2)$.

Boxer u. a. erwähnen in [Box+93], dass es möglich ist, den Algorithmus so zu implementieren, dass die Zeitkomplexität auf $\Theta(n)$ gesenkt wird. Diese Anpassung ist jedoch sehr aufwendig und wurde in dieser Arbeit nicht durchgeführt.

Maximaler Fehler

Das Verfahren erzeugt Vereinfachungen mit einem garantierten maximalen Fehler von ϵ.

Alterungsstabilität

Das Verfahren ist nicht alterungsstabil. Im Gegensatz zum Douglas-Peucker Verfahren vereinfacht es den Polygonzug nicht von den Extremwerten ausgehend, sondern von den lokalen Abweichungen ausgehend. Beim Entfernen von Punkten aus dem Polygonzug entfernt das Verfahren zuerst die aus drei Kanten bestehenden Abschnitte und dann die noch möglichen Abschnitte, die aus zwei Kanten bestehen.

Durch diese lokale Vereinfachung ist es möglich, dass je nach maximal erlaubtem Fehler Punkte entfernt oder beibehalten werden. Das führt dazu, dass das Verfahren nicht alterungsstabil ist.

	Arbeitsweise	Komplexität	Alterungsstabilität	Fehlerbeschr. bei mehrfacher Vereinfachung
Richtung und Geschw.	verwerf.-stromb.	$O(n)$	nein	möglich
Lineare Extrapolation	verwerf.-stromb.	$O(n)$	nein	möglich
Erster und letzter Punkt	puffernd-stromb.	$O(n^2)$	nein	möglich
Regression	puffernd-stromb.	$O(n^2)$	nein	nein
Distanzbasiert	verwerf.-stromb.	$O(n)$	nein	möglich
PLAZA	verwerf.-stromb.	$O(n)$	nein	möglich
Douglas-Peucker	fensterbasiert	$O(n^2)$	nein	ja
Boxer, Leu und Chen	fensterbasiert	$O(n^2)$	nein	möglich

Tabelle 6.1.: Übersicht über die Eigenschaften der untersuchten Algorithmen.

Fehlerbeschränktheit bei mehrfacher Vereinfachung

Da das Verfahren den Datenstrom lokal vereinfacht, muss der maximal erlaubte Fehler der vorhergehenden Vereinfachung bekannt sein, um Fehlerbeschränktheit bei mehrfacher Vereinfachung zu garantieren.

Das Verfahren von Boxer, Leu und Chen beschränkt den maximalen Fehler wie das PLAZA Verfahren genau dann, wenn bei der erneuten Vereinfachung der maximal erlaubte Fehler der vorhergehenden Vereinfachung vom gewünschten maximal erlaubten Fehler abgezogen wird (siehe auch Kapitel 4.1.2 auf Seite 24).

Zusammenfassung

Die Eigenschaften der Algorithmen zur Polygonzugvereinfachung und der von Kaya implementierten Verfahren sind in Tabelle 6.1 zusammengefasst.

Kapitel 7

Kartenbasierte Algorithmen

Kartenbasierte Algorithmen nutzen eine Karte um Positionsdatenströme zu komprimieren. In diesem Kapitel wird beschrieben, wie die Positionen mit der Karte abgeglichen und danach komprimiert werden können. Nach einer Diskussion der verwandten Literatur wird das Verfahren von Jensen, Čivilis u. a. analysiert.

7.1. Kartenabgleich

Zur Kompression des Positionsdatenstroms mit einem kartenbasierten Verfahren werden im ersten Schritt die (x,y,t)-Koordinaten, die das mobile Objekt liefert, mit der Karte abgeglichen (Englisch: map-matching). Durch den Abgleich wird die Position auf der Karte bestimmt, an der sich das mobile Objekt am Wahrscheinlichsten befindet.

Der Kartenabgleich ist nötig, da sowohl die Positionsdaten des mobilen Objekts, als auch die Kartenrepräsentation fehlerbehaftet sind, und daher voneinander abweichen. Zudem ist die Abtastrate des Positionssensors entscheidend für die Genauigkeit des Kartenabgleichs, da zwischen zwei bestimmten Positionen keine gesicherte Aussage über die Position des mobilen Objekts getroffen werden kann [Pfoo8].

Nach dem Kartenabgleich werden die (x,y,t)-Koordinaten der Positionen des mobilen Objekts in (Straße, Stelle auf der Straße, t)-Koordinaten transformiert, um die Positionen relativ zur Karte anzugeben. Dabei handelt es sich aber weiterhin um dreidimensionale Koordinaten, eine Datenkompression ist daher noch nicht erreicht.

Für den Kartenabgleich gibt es mehrere Ansätze, die in Laufe dieses Kapitels vorgestellt werden.

7.2. Kompression der abgeglichenen Positionen

Um die Daten zu komprimieren wird die Positionsanzahl reduziert indem beispielsweise nur die erste Position auf einer Straße gespeichert wird und alle weiteren auf derselben Straße verworfen werden. Dadurch behält man nur die Information wann ein mobiles Objekt auf eine Straße gefahren ist und wann es diese wieder verlassen hat. Durch Berechnung der Durchschnittsgeschwindigkeit auf diesem Straßenabschnitt, kann eine Abschätzung über die Position des mobilen Objekts in der Zeit zwischen dem Einfahren und dem Verlassen der Straße getroffen werden.

Genauere Abschätzungen erreicht man durch Beschleunigungs- oder Geschwindigkeitsprofile der einzelnen Straßen, die zusammen mit der Karte gespeichert sind. Durch diese lässt sich die Position des mobilen Objekts entsprechend genauer abschätzen [ČJP05].

7.3. Verwandte Ansätze im Bereich der kartenbasierten Algorithmen

7.3.1. Kartenabgleich

Algorithmen zum Kartenabgleich gliedern sich nach Miyashita u. a. [MTN08] in statische Algorithmen und Echtzeitalgorithmen. Statischer Kartenabgleich liefert erst am Ende einer Route den Pfad auf der Karte und kann daher globale Informationen zum Kartenabgleich nutzen. Kartenabgleich in Echtzeit dagegen liefert den Pfad „live“ und eignet sich daher für den Einsatz in Navigationsgeräten oder zur Verarbeitung strombasierter Positionsdaten. Er ist allerdings auf lokale Informationen angewiesen und daher ungenauer.

Nach Greenfeld [Gre02] gibt es für den Kartenabgleich mit einer (Straßen-)Karte drei Schwierigkeitsstufen:

Vorgegebene Route

Das mobile Objekt bewegt sich auf festen Routen (z. B. Busrouten, U-Bahn, S-Bahn, ...). Die Aufgabe besteht darin, das Segment der Route zu finden auf dem sich das mobile Objekt momentan bewegt. Dadurch ist der Suchraum sehr stark eingeschränkt und der Kartenabgleich dementsprechend schnell und genau.

Vorgeschlagene Route

Nutzt ein mobiles Objekt ein Navigationsgerät, kann dieses annehmen, dass sich der Benutzer auf der vorgeschlagenen Route bewegt. Dadurch kann der Algorithmus mit einer größeren Fehlertoleranz arbeiten und den Suchraum für den Kartenabgleich einschränken, da ihm bekannt ist, auf welcher Straße sich der Benutzer bewegen sollte. Bewegt sich der Benutzer

aber beispielsweise auf einer Parallelstraße, die sehr nah an der vorgeschlagenen Route liegt, kann das Navigationsgerät dies nicht oder nur schwer erkennen.

Allgemeiner Kartenabgleich

Beim allgemeinen Kartenabgleich gibt es keine Annahmen über die Route des mobilen Objekts. Das Verfahren nutzt nur die Positionsdaten des mobilen Objekts und Informationen aus der Karte, um die Position auf dieser zu bestimmen.

7.3.2. Verfolgung von mobilen Objekten (Client-Server-Kommunikation)

Jensen u. a. [TJ08; TJ07; JP07] beschreiben Verfahren mithilfe derer mobile Objekte, die sich auf festen Routen (z. B. Buslinien) bewegen, ihre Position mit dem Server austauschen können. Der Austausch dieser Informationen soll aus Kostengründen mit möglichst wenigen Nachrichten geschehen.

Die Verfahren dienen dazu, in Echtzeit Vorhersagen über die zukünftige Position eines mobilen Objekts zu treffen, z. B. um zukünftige Abfahrtszeiten an Bushaltestellen anzuzeigen. Dazu werden sowohl von der Serverkomponente, als auch von der Komponente des mobilen Objekts Vorhersagen gemacht und ausgetauscht.

Da beiden Komponenten die gemeinsame Vorhersagefunktion bekannt ist, findet ein Austausch erst statt, wenn entweder die aktuelle Position zu stark von der gemeinsamen Vorhersagefunktion abweicht (das mobile Objekt wird aktiv) oder sich die Servervorhersage stark ändert (der Server wird aktiv). Die Servervorhersage ändert sich beispielsweise aufgrund externer Gegebenheiten, z. B. Stauvorhersagen oder schlechtem Wetter.

7.3.3. Datenreduktion eines GPS-Pfades

Cao, Wolfson u. a. [CW05] beschreiben einen Algorithmus, der einen Pfad auf eine Karte abbildet und diesen dann vereinfacht. Der Algorithmus arbeitet auf räumlich-zeitlich Daten im (x,y,t)-Raum. Er arbeitet nicht strombasiert, sondern bearbeitet GPS-Pfade im Nachhinein.

Der Pfad wird dabei zuerst mit der Karte abgeglichen. Dazu werden die Positionen auf alle in ihrer Nähe liegenden Kartensegmente projiziert. Aus diesen Punkten wird ein Graph erzeugt, dessen Kanten aus den Straßensegmenten bestehen. Der Graph kann dabei Knoten enthalten, die keine eingehenden oder ausgehenden Kanten haben. In diesem Graph wird dann ein Pfad vom Start zum Endpunkt gesucht.

Der projizierte Pfad, der potentiell größer ist als der Originalpfad, da er zusätzlich Straßen enthält, wird dann verkleinert. Dabei werden Punkte auf Segmenten der Karte weggelassen, wenn die Ungenauigkeit, die dadurch entsteht, kleiner einer Fehlerschranke ϵ ist. Zur Distanzberechnung wird die Zeitgleichheitsdistanz E_u (siehe Kapitel 3.1.3) verwendet.

7.3.4. Verwendete Informationen aus der Karte

Greenfeld [Greo2] und White u. a. [WBKoo] beschreiben mehrere Möglichkeiten wie Informationen aus der Karte genutzt werden können:

Geometrische Informationen

Einige Verfahren zum Kartenabgleich nutzen nur geometrische Informationen um das Straßensegment zu bestimmen, auf dem sich ein mobiles Objekt bewegt. Geometrische Informationen beinhalten in diesem Zusammenhang die Distanz der Positionen zu Segmenten oder Punkten der Karte, die Bewegungsrichtung des mobilen Objekts und die Richtung der Straße, sowie deren Winkel zueinander.

Geometrische und topologische Informationen

Andere Verfahren zum Kartenabgleich betrachten zudem auch die Verbindungen der Segmente untereinander. Dabei wird die Tatsache ausgenutzt, dass ein mobiles Objekt nicht sprunghaft seine Position ändern kann. Somit wird eine neue Position auf der Karte nur auf dem aktuellen oder einem daran anschließenden Straßensegment gesucht. Durch die Nutzung von topologischen Informationen kann der tatsächliche Weg des mobilen Objekts besser verfolgt werden.

Probleme treten auf, wenn die Abstände der Punkte so groß sind, dass Straßensegmente übersprungen werden. Dies kann durch die Betrachtung weiterführender Straßensegmente, die an das aktuelle Straßensegment anschließen, eingeschränkt werden. Dieser Erweiterung macht das Verfahren aber aufwendiger.

Ein weiteres Problem stellen Folgefehler beim Kartenabgleich dar. Wenn beispielsweise zu Beginn des Kartenabgleichs die „falsche“ Parallelstraße gewählt wurde, ist es möglich, dass dieser Fehler nicht oder sehr spät bemerkt wird.

7.3.5. Vorverarbeitung

Karte

Die Repräsentation der Karte hat einen großen Einfluss auf den Kartenabgleich. Deshalb kann das Kompressionsverhalten des kartenbasierten Verfahrens durch die Vorverarbeitung der Karte verbessert werden. Prinzipiell gibt es zwei Möglichkeiten eine Karte digital zu beschreiben:

Eine Möglichkeit ist, dass *jeder Straßenabschnitt* zwischen zwei Kreuzungen *eine Straße* darstellt. Da dadurch aber sehr viele kurze Straßen benötigt werden, ist es sinnvoll Straßen über mehrere Kreuzungen hinweg führen zu lassen. Die zweite natürlichere Möglichkeit

besteht demnach darin, dass *Straßen* einer Karte *längere Straßen über Kreuzungen hinweg* repräsentieren. Das entspricht der Verwendung von Straßennamen in der Realität.

Čivilis u. a. [Čiv+04a; ČJP05; ČJP07] beschreiben Algorithmen mit deren Hilfe die Anzahl der Straßensegmente reduziert werden kann. Ein Algorithmus versucht dabei, Hauptstraßen, Autobahnen und andere wichtigen Straßen vor Nebenstraßen zu bevorzugen. Die wichtigeren Straßen werden dabei aus vielen Segmenten zu einem Polygonzug zusammengefasst. Dadurch wird die Position eines mobilen Objektes, das sich z. B. auf der Hauptstraße bewegt, beim Kartenabgleich immer auf derselben Straße positioniert. Diese Daten lassen sich besser komprimieren, da weniger Wechsel der Straßensegmente stattfinden.

Ein anderes Verfahren besteht darin, aufeinanderfolgende Segmente einer Straße zu einem größeren Segment zusammenzufassen, wenn dadurch ein vorgegebener maximal erlaubter Fehler nicht überschritten wird. Dadurch beschleunigt man den Kartenabgleich, da weniger Straßensegmente überprüft werden müssen.

Daten

White u. a. [WBK00, S. 100 ff.] erläutern Probleme beim Kartenabgleich im Kreuzungsbereich. Im Kreuzungsbereich häufen sich, aufgrund der normalerweise geringen Geschwindigkeit des mobilen Objekts, die Punkte. Daher häufen sich hier auch die Fehler, da Ungenauigkeiten der Positionsbestimmung hier im Verhältnis zur Bewegung überwiegen. Durch Verfahren zur Polygonzugvereinfachung (siehe Kapitel 5) kann die Punktanzahl reduziert und der Pfad geglättet werden. Dadurch wird sowohl der Kartenabgleich im Kreuzungsbereich verbessert, als auch der Aufwand für diesen reduziert.

7.3.6. Nachbereitung

Durch den Kartenabgleich auf Straßensegmente ist es möglich, dass ein (auf der Karte) unzusammenhängender Pfad entsteht. Zudem ist es möglich, dass das mobile Objekt an Kreuzungen von einer Straße zu nächsten zu springen scheint. Diese „Fehler“ lassen sich in der Nachbereitung korrigieren.

Nicht zusammenhängende Pfade können verbunden werden, indem beispielsweise der fehlende Teil auf dem kürzesten Weg verbunden wird. Fehlerhafte Segmente in Kreuzungsbereichen können entfernt werden. Durch diese Maßnahmen wird die Qualität des aufgezeichneten Pfades verbessert.

7.4. Angepasster kartenbasierter Algorithmus von Jensen und Čivilis

Jensen, Čivilis u. a. [ČJP05; JP07; Čiv+04a; Čiv+04b; ČJP07; PJ03] beschreiben ein Verfahren zum Kartenabgleich, dass strombasiert arbeitet. Das Verfahren wurde im Rahmen dieser

Arbeit angepasst und wird im Folgenden erläutert. Dabei wird auf die vorgenommenen Änderungen eingegangen.

7.4.1. Aufbau der Karte

Die Karte besteht aus Straßen, die als Polygonzüge gespeichert sind. Ein Abschnitt einer Straße, der genau aus zwei Punkten und dem dazwischenliegenden Geradenabschnitt besteht, ist ein Straßensegment. Ein Straßensegment ist gerichtet, d. h. die Straßenkarte stellt einen gerichteten Graphen dar. Dadurch ist jeder Straße eine eindeutige Richtung zuweisbar.

Für diese Arbeit wurde eine einfache Kartenverwaltungskomponente erstellt. Diese hält die Kartendaten vollständig im Arbeitsspeicher. Die Kartenrepräsentation besitzt keinen räumlichen Index (Geo-Index) und kann somit keine räumlichen Abfragen verarbeiten. Daher müssen im Folgenden immer alle Straßen betrachtet werden. Das Verfahren benötigt deshalb viel Rechenzeit.

7.4.2. Projektion der Position auf die Karte

Zuerst müssen die Positionen auf die Segmente der umliegenden Straßen abgebildet werden. Um den Ausschnitt der Karte einzuschränken, in dem die Positionen auf die Straßensegmente abgebildet werden, wird ein Radius δ angegeben. Alle Straßen, die den kreisförmigen Ausschnitt mit dem Radius δ schneiden, werden untersucht. Die gegebene Position wird auf die Segmente aller Straßen innerhalb dieses Kreises abgebildet. Dabei wird sie orthogonal auf den Geradenabschnitt projiziert. Liegt der projizierte Punkt auf der Verlängerung des Geradenabschnitts wird er auf den Endpunkt des Geradenabschnitts abgebildet, der ihm am nächsten liegt.

7.4.3. Auswahl der wahrscheinlichsten Position auf der Karte

Aus den projizierten Punkten muss nun derjenige ausgewählt werden, der am wahrscheinlichsten der echten Position entspricht. Dazu wird ein Kartenabgleichsgewicht berechnet. Der projizierte Punkt, der das geringste Gewicht hat, wird ausgewählt.

Für die Berechnung der Gewichte werden zwei gewünschte Eigenschaften der zu suchenden Kartenposition genutzt. Einerseits soll der abgeglichene Punkt so nah wie möglich an der ursprünglichen Position liegen, andererseits sollte die Bewegungsrichtung des mobilen Objekts möglichst gut mit der Richtung des Straßensegments übereinstimmen, auf das die Position abgeglichen wird. Die beiden Maße werden mit ihren Maximalwerten normalisiert und anschließend zum Kartenabgleichsgewicht addiert.

7.4.4. Speicherung der Position auf der Karte

Die mit der Karte abgeglichenen Positionen werden in (Straße, Stelle auf der Straße, t)-Koordinaten gespeichert. Um die Datenmenge zu reduzieren, wird immer nur die erste Position auf einer Straße gespeichert (siehe Kapitel 7.2).

7.4.5. Veränderungen gegenüber dem Algorithmus von Jensen, Čivilis u. a.

Der Algorithmus von Jensen, Čivilis u. a. arbeitet auf einem ungerichteten Graphen. Dadurch kann die Bewegungsrichtung eines mobilen Objekts nie mehr als 90 Grad von der Straßenrichtung abweichen. Zudem werden Positionen von mobilen Objekten, die sich in oder gegen die Straßenrichtung bewegen, auf dieselbe Straße abgeglichen.

Da die verwendeten Kartendaten Informationen über Einbahnstraßen enthalten, wird das Verfahren auf einen gerichteten Graphen erweitert. Dabei werden für Straßen die in beide Richtungen befahrbar sind zwei Polygonzüge erstellt, die denselben Verlauf aber eine entgegengesetze Richtung aufweisen. Die Positionen des mobilen Objekts lassen sich damit auf das Straßensegment abgleichen, das die korrekte Fahrtrichtung hat. Dadurch kann die Bewegungsrichtung eines mobilen Objekts um bis zu 180 Grad von der Fahrtrichtung der Straße abweichen.

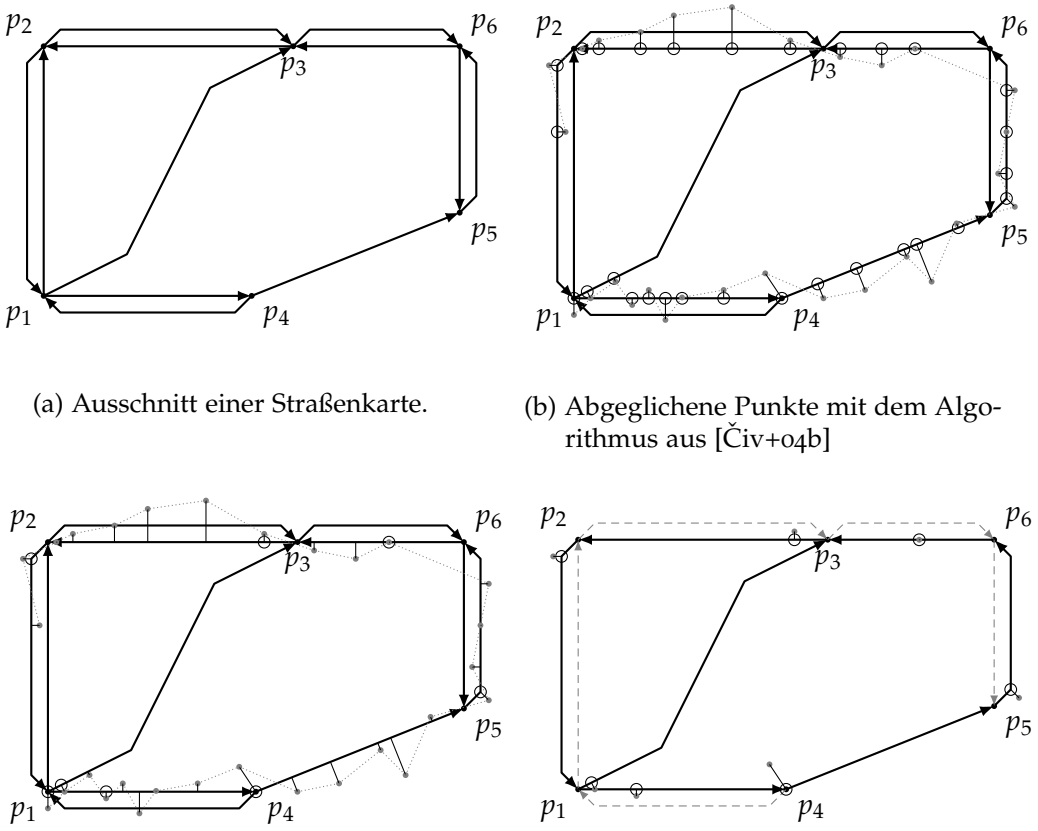
Der Algorithmus hat zwei Parameter, um die beiden Teilmaße des Kartenabgleichsgewichts zu gewichten. Das Maß für die Übereinstimmung der Winkel zwischen der Bewegungs- und der Fahrtrichtung wird mit dem Parameter α multipliziert, das Abstandsmaß mit dem Parameter β. Beide Parameter werden auf denselben Wert $\alpha = \beta = 1$ gesetzt.

7.4.6. Beispiel

Gegeben ist eine Straßenkarte (siehe Abbildung 7.1a) mit zwölf gerichteten Straßen, die aus Polygonzügen aufgebaut sind. Zum besseren Verständnis wurden Straßen, die in beide Richtungen befahrbar sind, nebeneinander dargestellt. Auf diese Karte wurden 29 Punkte abgeglichen (siehe Abbildung 7.1b).

Die kleinen grauen Punkte sind die ursprünglichen Punkte, die abgeglichenen Punkte sind als große Kreise markiert. Zur Kompression wurden nur die ersten Punkte auf Straßensegmenten behalten (siehe Abbildung 7.1c) und dadurch die entsprechenden Straßensegmente ausgewählt (siehe Abbildung 7.1d). Dabei fällt auf, dass das diagonale Straßensegment $\overline{p_1 p_3}$ durch den Algorithmus falsch ausgewählt wurde.

Da die Bewegungsrichtung an einem Punkt aus dem aktuellen und letzten Punkt berechnet wird, kann für den Punkt a noch keine Richtung angegeben werden. Diese wird auf den Standardwert gesetzt, ein Vektor senkrecht nach oben (90° in Polarkoordinaten). Dadurch wird er auf das Segment $\overline{p_1 p_2}$ abgeglichen (siehe Abbildung 7.2) bei der Auswahl der Straßensegmente vernachlässigt.



(a) Ausschnitt einer Straßenkarte.

(b) Abgeglichene Punkte mit dem Algorithmus aus [Čiv+04b]

(c) Positionen auf der Karte

(d) Straßensegmente

Abbildung 7.1.: Ausschnitt einer Straßenkarte.

7.5. Vorhersage der Bewegungsrichtung mithilfe der Karte

Eine weitere Möglichkeit die Kartendaten zu nutzen besteht darin, das Wissen über die Straße, auf der sich das mobile Objekt bewegt, zu nutzen, um dessen zukünftige Bewegungsrichtung vorherzusagen.

Dieses Wissen könnte im Rahmen eines Polygonzugvereinfachungsverfahrens genutzt werden, um bei der Wahl des aktuellen Vereinfachungspunktes auf den weiteren Verlauf einzugehen, ohne diesen bereits genau zu kennen. Dadurch könnte bei strombasierten Verfahren eine bessere Annäherung und damit eine größere Kompression erreicht werden. Das strombasierte Verfahren könnte durch die bekannte Position auf der Karte ein Stück in die erwartete Zukunft schauen und diese Informationen zur Vereinfachung des Pfades verwenden.

Zur Vorhersage des weiteren Verlaufs könnten beispielsweise Vorhersagen wie von Tiešyté und Jensen [TJ08; TJ07] vorgeschlagen, oder Richtungsinformationen wie von White u. a. [WBK00] betrachtet, verwendet werden.

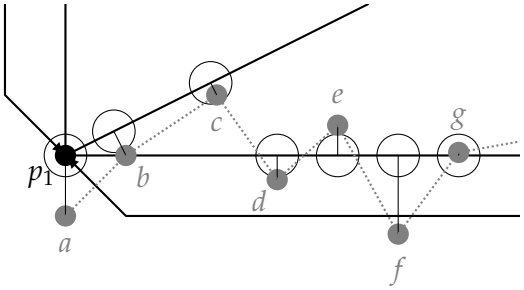


Abbildung 7.2.: Vergrößerung eines Auschnitts der Straßenkarte mit allen abgeglichen Positionen.

Diese Art der Vereinfachung erweitert strombasierte Verfahren so, dass diese den Polygonzug nicht mehr nur lokal komprimieren. Dadurch könnte eine höhere Kompressionsrate erreicht werden. Die betrachteten fensterbasierten Verfahren komprimieren einen Polygonzug ebenfalls nicht nur lokal und es zeigt sich, dass einfache strombasierte Verfahren eine ähnliche Kompressionsrate erreichen wie fensterbasierte Verfahren. Da durch die Vorhersagen der Bewegung zudem z. B. im Kreuzungsbereich fehlerhafte Annahmen gemacht werden, ist nicht zu erwarten, dass sich durch die Vorhersagen die Kompressionsrate drastisch verbessert.

Deshalb wurde auf eine weitere Untersuchung dieses Ansatzes verzichtet.

Kapitel 8

Simulation

Zur Simulation wurde eine Simulationsumgebung erstellt. Diese besteht aus den folgenden Komponenten:

- Einem einfachen Positionsdatensender, der die Simulationsdaten einliest und an den Operator weiter reicht.
- Einem einfachen Positionsdatensender, der die vereinfachten Daten entgegen nimmt und in Dateien schreibt.
- Dem Operator selbst, der die Vereinfachung des Positionsdatenstroms vornimmt.
- Einer einfachen Kartenverwaltung.
- Sowie einigen Hilfsklassen, die die Simulation steuern.

8.1. Verwendete Datensätze

8.1.1. INFATI-Daten

Zur Simulation wurden die INFATI-Datensätze von Jensen u. a. [Jen+04] genutzt. Diese Datensätze enthalten 20 GPS-Pfade von PKWs. Sie wurden im Rahmen eines Geschwindigkeitsanpassungs-Projekts über einen Zeitraum von Dezember 2000 bis März 2001 gesammelt. Die Datensätze enthalten insgesamt über 1,8 Millionen Messpunkte.

Die Daten wurden innerhalb der Stadt Aalborg (Dänemark) und ihrer Umgebung aufgezeichnet [JS04]. Die Datensätze unterscheiden sich darin, wie groß der Anteil der Messpunkte in der Stadt, auf Landstraßen oder auf der Autobahn ist. Da die Daten im Rahmen eines Geschwindigkeitsanpassungs-Projekts aufgezeichnet wurden, enthalten sie auch Informationen über die geltenden Geschwindigkeitsbeschränkungen an den Messpunkten.

Die Geschwindigkeitsbeschränkungen sind aber nicht für alle Messpunkte verfügbar, da aufgrund von Empfangsproblemen nicht alle Positionen korrekt bestimmt und auf die Straßenkarte abgeglichen werden konnten. Dadurch enthalten die Datensätze fehlerhafte

Punkte sowie Punkte mit einer Geschwindigkeitsbegrenzung von 0 km/h. In Dänemark gilt allgemein eine Geschwindigkeitsbegrenzungen, auch auf Autobahnen.

Die Geschwindigkeitsverteilungen der einzelnen Datensätze finden sich im Anhang C.1 auf Seite 99, eine Übersicht in Tabelle C.1 auf Seite 107.

Da die Daten nicht in dem für diese Arbeit benötigten Format vorliegen, müssen sie umgewandelt werden. Dabei muss sowohl das Speicherformat, als auch das verwendete Geokoordinatensystem gewechselt werden. Details zu den einzelnen Umwandlungsschritten finden sich in Anhang A ab Seite 93.

8.1.2. Kartendaten von OpenStreetMaps

Das für den Kartenabgleich benötigte Kartenmaterial wurde vom OpenStreetMaps-Projekt [Ope08] bezogen. Die Karten dieses Projekts werden von vielen freiwilligen Helfern erstellt. Dabei wird ähnlich dem Wiki-Prinzip bei Wikipedia gemeinsam eine immer genauere Karte erstellt, indem Routen mit GPS-Empfängern abgefahren oder abgelaufen werden und die daraus erzeugten GPS-Pfade in die gemeinsame Karte eingepflegt werden.

Die Kartendaten von Aalborg sind im Vergleich zu kommerziellen Kartendaten noch nicht sehr detailliert, aber für eine Bewertung des Kartenabgleichsalgorithmus im Rahmen dieser Arbeit ausreichend.

Kartenausschnitt

Als Kartenausschnitt wurde eine rechteckige Region um Aalborg gewählt, die dem Kartenausschnitt in [JS04] ungefähr entspricht (siehe Abbildung 8.1). Eine große Darstellung des Kartenausschnitts befindet sich im Anhang C auf Seite 108.

Der Kartenausschnitt enthält 1829 Straßen die in beide Richtungen und 180 Straßen die nur in eine Richtung befahrbar sind. Das ergibt insgesamt 3838 gerichtete Straßen.

Die Karte wurde am 8. April 2009 mit dem folgenden Aufruf an die OpenStreetMaps-API abgerufen.

```
http://www.informationfreeway.org/api/0.5/
   way[highway=motorway|motorway_link|trunk|trunk_link|primary|primary_link
            |secondary|tertiary|unclassified|road|residential|living_street
            |service|track|pedestrian|services|bus_guideway]
      [bbox=9.649,56.859,10.152,57.165]
```

Über die API wird der Kartenbereich mithilfe von Geokoordinaten als „bounding box“ ausgewählt `[bbox=9.649,56.859,10.152,57.165]`. Da nur die Straßen für den Kartenabgleich relevant sind wurde über die `highway` Auswahl eine Liste aller Straßenelemente übergeben. Für eine detaillierte Beschreibung der API und ihrer Parameter sei auf [Ope08] verwiesen.

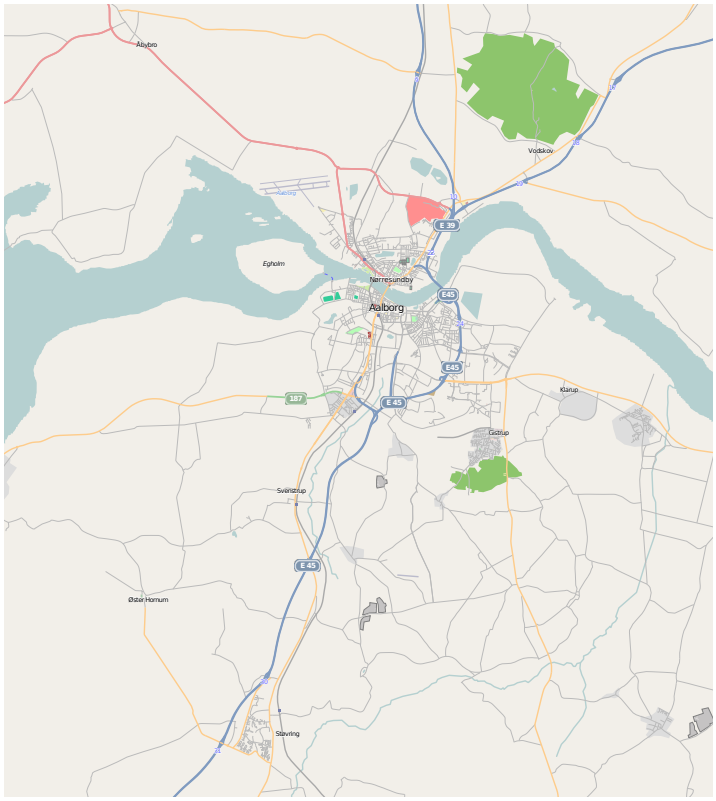


Abbildung 8.1.: Kartenausschnitt von Aalborg (Dänemark) aus OpenStreetMaps.

Import

Beim Import in die Vereinfachungskomponente werden alle Daten in ein metrisches System umgerechnet um die Distanzberechnungen zu erleichtern. Die Umrechnung beim Import wird in Anhang B ab Seite 97 beschrieben.

8.1.3. Simulationsumgebung

Die Simulation wurde auf einem IBM Thinkpad T61 mit einer 1,83 GHz DualCore CPU und 1GB RAM durchgeführt. Die CPU war auf 1,0 GHz heruntergetaktet, um Überhitzung zu vermeiden. Außerdem nutzte die Simulation nur einen Kern der CPU.

8.2. Simulationsparameter Polygonzugvereinfachung

Die Simulationen wurden für alle 20 Datensätze aus den INFATI-Daten durchgeführt. Dabei wurde sowohl der Wert für den maximal erlaubten Fehler ϵ, als auch die maximale Fenstergröße variiert.

8.2.1. Maximal erlaubter Fehler

Der Einfuss des maximal erlaubten Fehlers ϵ auf die Vereinfachung wurde untersucht. Dazu wurden Simulationen mit den folgenden Werten durchgeführt: 10 m, 25 m, 50 m, 100 m, 250 m, 500 m, 750 m und 1000 m.

8.2.2. Maximale Fenstergröße

Die maximale Fenstergröße, eine Zeitraum, hat einen Einfluss auf die Vereinfachung. Sie bestimmt zusammen mit der Updaterate bei den fensterbasierten Algorithmen nach wievielen Punkten die Vereinfachungsberechnung gestartet wird. Bei den strombasierten Algorithmen bestimmt sie, wann auf jeden Fall ein Vereinfachungspunkt gesendet werden muss, auch wenn das aus der Sicht des Vereinfachungsverfahrens noch nicht nötig wäre.

Die Punktanzahl, die bis zum Erreichen der Fenstergröße betrachtet wird, berechnet sich aus:

$$\text{Punktanzahl} = \text{Updaterate} \cdot \text{Fenstergröße}$$

Im Operator wird die maximale Fenstergröße durch eine Zeitgeberkomponente überwacht. Da ein solcher Zeitgeber bei jeder Simulation unterschiedliche Werte liefern kann, wird er für die Simulation deaktiviert. Stattdessen wird die Anzahl der Punkte eines Fensters fest vorgegeben. Dadurch können reproduzierbare Werte erreicht werden und die Dauer der Simulation verkürzt sich, da die Simulation dadurch zeitunabhängig wird.

Die Simulationen wurden mit den folgenden Punktanzahlen durchgeführt: 5, 10, 15, 20, 50, 100, 1000, 10 000 und 1 000 000 Punkte. Da kein Datensatz mehr als 270 000 Punkte umfasst, dient der letzte Wert dazu zu simulieren, was passiert, wenn das Fenster den gesamten Datensatz umfasst. Dadurch ist ein Einfluss der Fenstergröße auf die Vereinfachung in diesem Fall vollständig ausgeschlossen.

Währen den Simulationsläufen stellte sich heraus, dass das Verfahren „Boxer, Leu und Chen“ für große Fenstergrößen sehr viel Zeit benötigt. Eine einzelne Simulation mit einer Fenstergröße, die größer als die Punktmenge des Polygonzuges ist, benötigte teilweise mehr als fünf Stunden. Wegen der sehr langen Simulationszeiten bei großen Fenstergrößen wurde das Verfahren „Boxer, Leu und Chen“ nur bis zu einer Fenstergröße von 10 000 Punkten simuliert. Bereits bei dieser Fenstergröße ist das Verfahren signifikant langsamer als die anderen Verfahren. Neue Erkenntnisse sind bei einer Simulation des Verfahrens von Boxer, Leu und Chen mit einer Fenstergröße von 1 000 000 Punkten nicht zu erwarten.

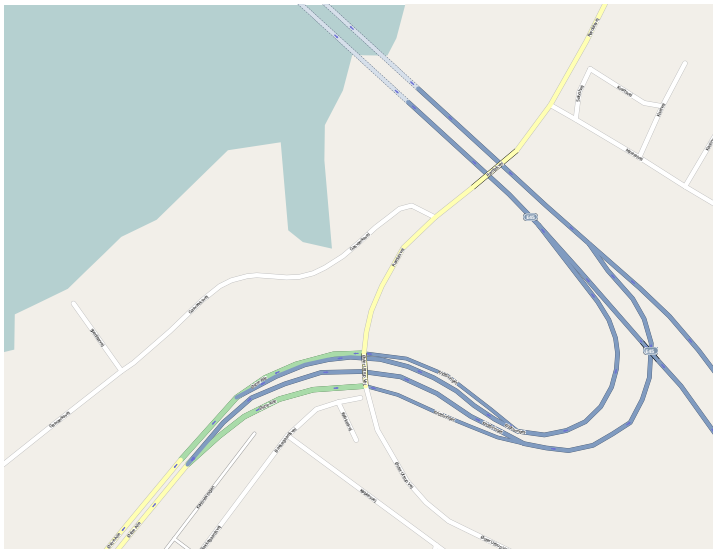


Abbildung 8.2.: Kartenausschnitt

8.3. Simulationsparameter beim kartenbasierten Algorithmus

8.3.1. Suchradius

Beim kartenbasierten Algorithmus wurde auf die Verwendung eines Fensters verzichtet. Die Straße, auf die ein Punkt abgeglichen werden kann, wird im Umkreis von $\delta = 500$ Metern um den Punkt gesucht. Das stellt sich als guter Kompromiss zwischen Laufzeit und Treffsicherheit des Kartenabgleichs heraus. Durch die teilweise fehlenden Straßen in den verwendeten Daten muss mit 500 Metern ein relativ hoher Wert gewählt werden.

Die Straßendaten des OpenStreetMaps Projekts sind teilweise noch sehr unvollständig. Das führt in Verbindung mit dem Suchradius von $\delta = 500$ dazu, dass bei einzelnen Datensätzen bis zu 10 % der Punkte nicht mit der Karte abgeglichen werden können.

Kommerzielle Straßenkarten sind meist deutlich vollständiger. Dadurch sind dort Werte für den Suchradius δ im Bereich von 50 bis 200 Meter sinnvoller, da diese kleineren Werte die Laufzeit des Kartenabgleichs reduzieren.

Um die experimentelle Laufzeit für den kartenbasierten Algorithmus auf einer Karte mit Geo-Index zu simulieren wurde ein kleiner Kartenausschnitt für den Kartenabgleich gewählt (siehe Abbildung 8.2). Dieser wurde mit der „bounding box“ `[bbox=9.95,57.04851,9.96893,57.05642]` ausgewählt.

8.3.2. Vorverarbeitung der Daten

Im Kreuzungsbereich werden aufgrund der geringen Fahrzeuggeschwindigkeit viele Punkte aufgezeichnet (siehe Kapitel 7.3.5). Diese Punkthaufen führen beim kartenbasierten Algorithmus zu fehlerhaftem Kartenabgleich und einer langen Laufzeit. Um dem entgegen zu wirken, wird einer erweiterten Version des kartenbasierten Verfahrens der distanzbasierte Algorithmus zur Polygonzugvereinfachung vorgeschalten (siehe Kapitel 4.5).

Kapitel 9

Ergebnisse

9.1. Algorithmen zur Polygonzugvereinfachung

Mit den im vorherigen Kapitel vorgestellten INFATI-Datensätzen wurden die Algorithmen „Lineare Extrapolation“ (siehe Kapitel 4.2), „Erster und letzter Punkt“ (siehe Kapitel 4.3), „Lineare Regression“ (siehe Kapitel 4.4), „Distanzbasiert“ (siehe Kapitel 4.5), „PLAZA“ (siehe Kapitel 6.1), „Douglas-Peucker“ (siehe Kapitel 6.2) und „Boxer, Leu und Chen“ (siehe Kapitel 6.3) simuliert.

Da sich die Verfahren auf allen INFATI-Datensätzen ähnlich verhalten, werden in diesem Kapitel die durchschnittlichen Ergebnisse vorgestellt.

9.1.1. Datenkompression

Hauptziel der Arbeit ist es, den Positionsdatenstrom zu komprimieren. Bei der Auswahl des Kompressionsverfahrens ist deshalb die Größe des komprimierten Polygonzuges ein wichtiges Kriterium. Im Folgenden wird die Größe der Kompression für verschiedene Fenstergrößen, verschiedene maximal erlaubte Fehler und für alle vorgestellten Algorithmen dargestellt.

Allgemeine Beobachtungen

Die Kompressionsrate ist vom maximal erlaubten Fehler ϵ und der Fenstergröße abhängig.

Die Fenstergröße beschränkt die Größe des vereinfachten Polygonzuges nach unten, da sie eine minimale Punktanzahl des vereinfachten Polygonzuges vorgibt, die nicht unterschritten werden kann. Die minimale Größe des vereinfachten Polygonzuges berechnet sich aus:

$$\text{Min. Größe des vereinfachten Polygonzuges} = \frac{\text{Größe des ursprünglichen Polygonzuges}}{\text{Fenstergröße}}$$

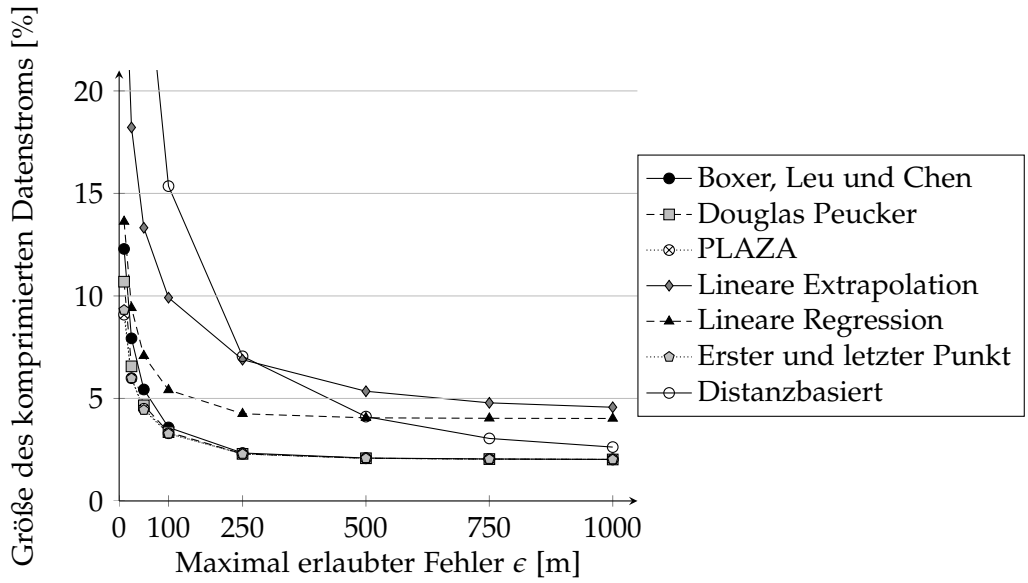


Abbildung 9.1.: Vergleich der durchschnittlichen Größe des komprimierten Polygonzuges bei einer Fenstergröße von 100 Punkten.

Ist die Fenstergröße beispielsweise fünf Punkte, kann der Pfad maximal auf ein Fünftel seiner ursprünglichen Größe komprimiert werden. Für Fenstergrößen < 50 Punkten dominieren diese die Größe des komprimierten Polygonzuges. Vergrößert man die Fenster, nimmt der Einfluss immer weiter ab und der maximal erlaubte Fehler dominiert das Kompressionsverhalten.

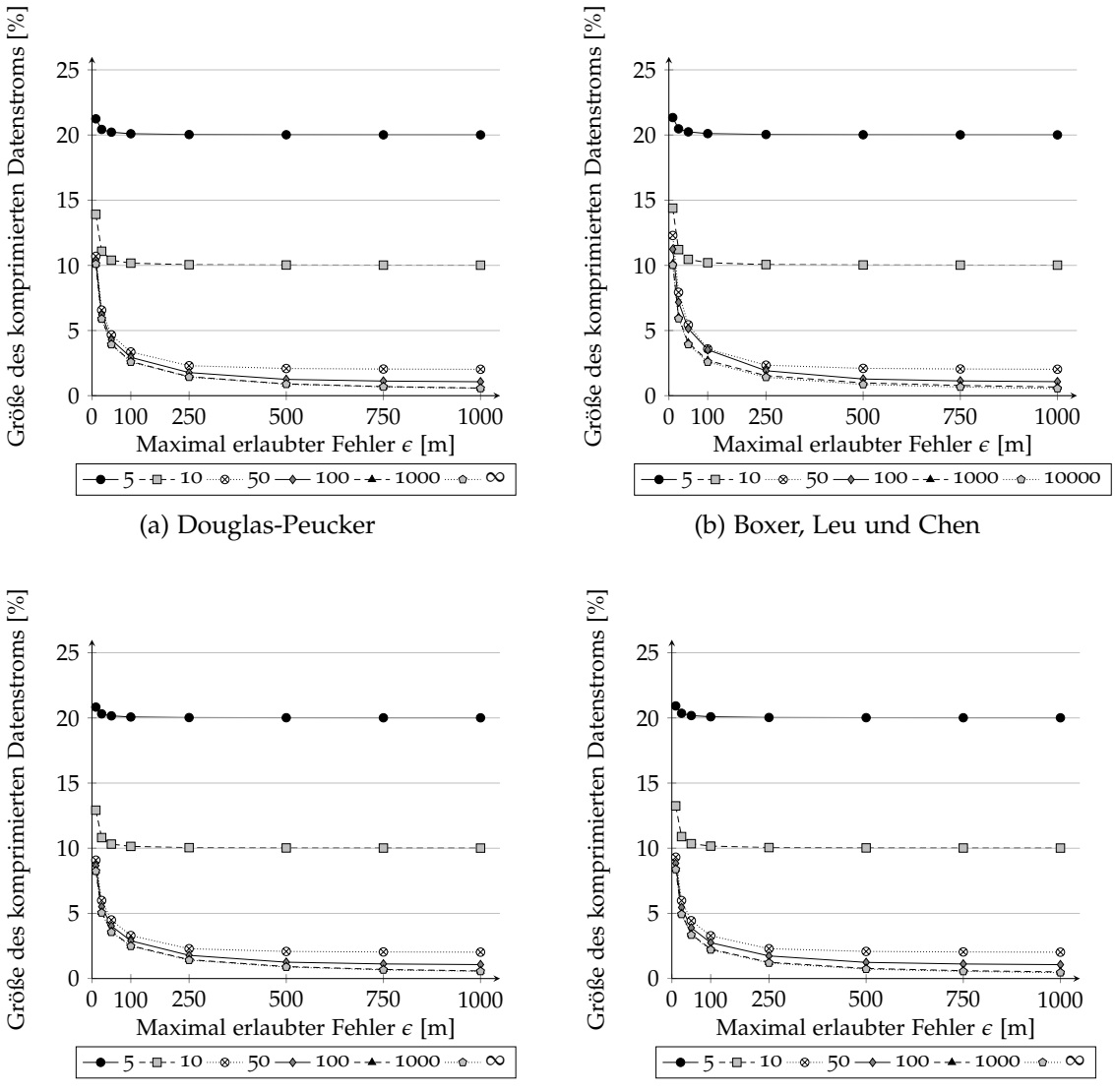
Ist die Fenstergröße größer als der gesamte Pfad, wird die mit dem Algorithmus mögliche optimale Vereinfachung erreicht.

Der maximal erlaubte Fehler ist der zweite Parameter, der Einfluss auf die Größe des vereinfachten Polygonzuges hat. Durch Vergrößerung des maximal erlaubten Fehlers verringert sich die Größe des vereinfachten Polygonzuges zunächst sehr stark und flacht dann ab ca. 250 Metern ab.

Eigenheiten der Verfahren

Die Algorithmen „Erster und letzter Punkt“, „Douglas-Peucker“, „PLAZA“ und „Boxer, Leu und Chen“ verhalten sich bezüglich der Kompressionsrate in Abhängigkeit vom maximal erlaubten Fehler sehr ähnlich (siehe Abbildung 9.1).

Auch für verschiedene Fenstergrößen verhalten sich die Algorithmen sehr ähnlich (siehe Abbildungen 9.2a bis 9.2d auf der nächsten Seite). Man erkennt gut, wie die Fenstergröße bei kleinen Werten (für Fenstergrößen von 5 und 10 Punkten) die Größe der Vereinfachung nach unten beschränkt. Bei größeren Fenstergrößen schwindet der Einfluss der Fenstergröße



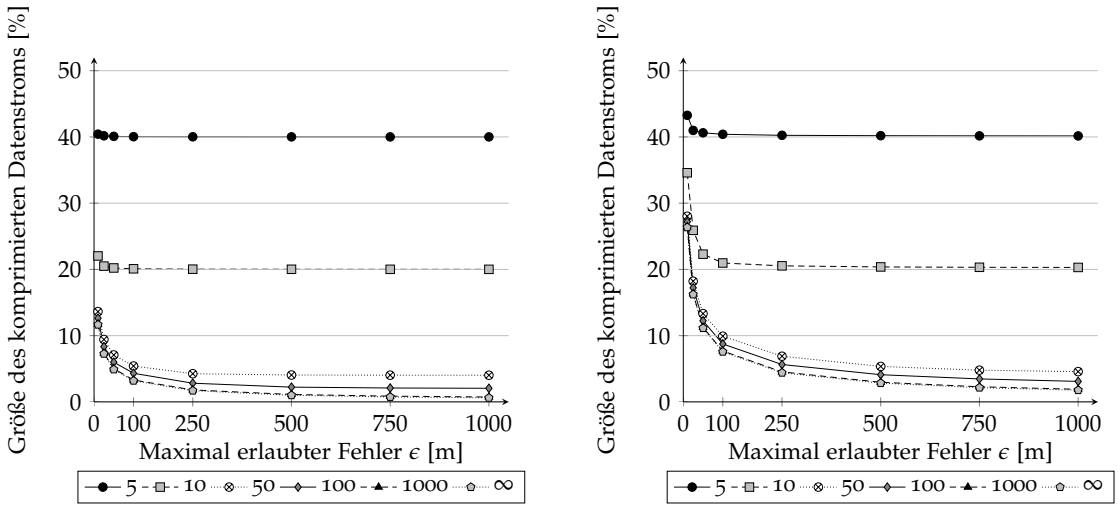
(a) Douglas-Peucker

(b) Boxer, Leu und Chen

(c) PLAZA

(d) Erster und letzter Punkt

Abbildung 9.2.: Durchschnittliche Größe der Vereinfachung (1). Für Fenstergrößen von 5, 10, 50, 100, 1000 und ∞ vielen Punkten.



(a) Lineare Regression (b) Lineare Extrapolation

Abbildung 9.3.: Durchschnittliche Größe der Vereinfachung (2). Für Fenstergrößen von 5, 10, 50, 100, 1000 und ∞ vielen Punkten.

und der maximal erlaubte Fehler ϵ hat größeren Einfluss auf die Größe des vereinfachten Polygonzuges.

Durch die Geradenabschnitte des linearen Regressionsverfahrens kann der Ursprungspfad genauer angenähert werden, als mit den anderen Verfahren, da die lineare Regression Stützpunkte erlaubt, die nicht auf dem Ursprungspfad liegen. Durch diese freie Wahl der Punkte entsteht aber kein zusammenhängender Polygonzug, sondern springende Geradenstücke, die jeweils zwei Stützpunkte benötigen. Daher fällt die Kompression nicht so hoch aus, wie bei den anderen Verfahren (siehe Abbildung 9.1). Erst bei sehr großen Fenstergrößen erreicht die Kompressionsrate der linearen Regression ähnliche Werte wie die der anderen Verfahren (siehe Abbildung 9.3a).

Beim Verfahren „Lineare Extrapolation“ besteht der komprimierte Pfad teilweise aus Punkten, die nicht auf dem Ursprungspolygonzug liegen. Zudem benötigen die Vektoren wie beim Regressionsverfahren jeweils zwei Stützpunkte. Daher fällt die Kompression auch bei diesem Verfahren nicht so hoch aus wie bei den anderen Verfahren (siehe Abbildung 9.1). Es fällt auf, dass die Größe des komprimierten Polygonzuges langsamer kleiner wird als beim linearen Regressionsverfahren (siehe Abbildung 9.3b). Die geringere Kompression erklärt sich dadurch, dass das Verfahren den Polygonzug von „außen“ annähert. Dadurch kann es dem Verlauf des Polygonzuges in Kurven weniger gut folgen, als beispielsweise das Verfahren „Erster und letzter Punkt“, das den Polygonzug von „innen“ annähert oder die „Lineare Regression“ die den Polygonzug von beiden Seiten annähert.

Das Verfahren „Distanzbasiert“ schneidet bezüglich der Kompressionsrate am schlechtesten ab (siehe Abbildung 9.1). Insbesondere bei kleinen Werten für den maximal erlaubten Fehler benötigt das Verfahren sehr viele Punkte. Während alle anderen Verfahren schon bei einer

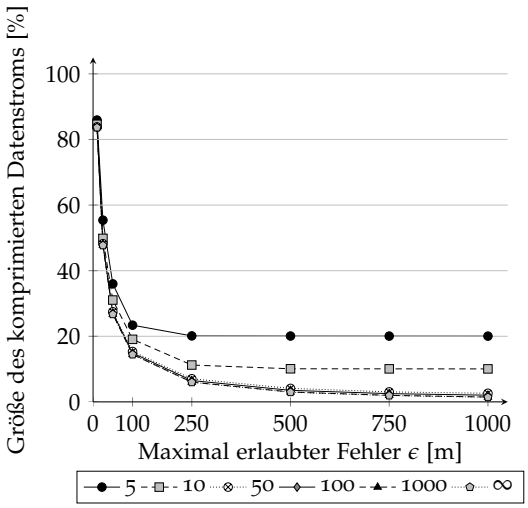


Abbildung 9.4.: Durchschnittliche Größe der Vereinfachung (3): Distanzbasiert. Für Fenstergrößen von 5, 10, 50, 100, 1000 und ∞ vielen Punkten.

Fenstergröße von fünf Punkten und einem maximalen Fehler von 5 Metern eine Kompression auf weniger als die Hälfte der Punkte erreichen, benötigt der distanzbasierte Algorithmus dafür nahezu alle Punkte des Ursprungspolygons. Erst ab einem maximal erlaubtem Fehler > 250 Meter erreicht das Verfahren eine signifikante Kompression (siehe Abbildung 9.4).

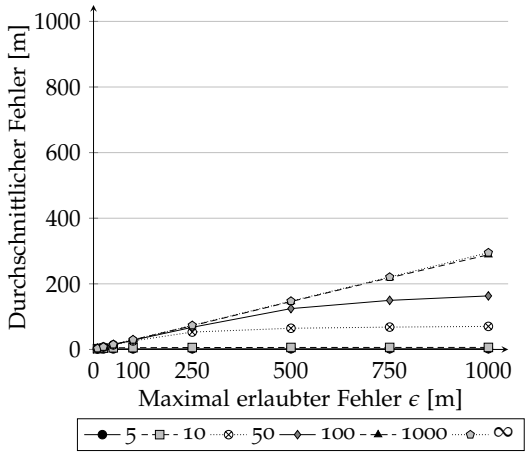
9.1.2. Durchschnittlicher Fehler

Allgemeine Beobachtungen

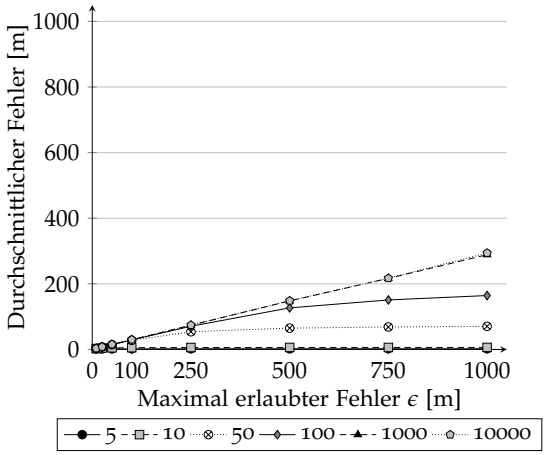
Wenn der Fehler des komprimierten Polygonzuges gegenüber dem ursprünglichen Polygonzug gleichverteilt wäre, müsste der Wert des durchschnittliche Fehlers die Hälfte des maximal erlaubten Fehlers annehmen. In den Abbildungen 9.6a bis 9.6b auf Seite 75 ist zu sehen, dass dies nur für das puffernd-strombasierten Verfahren „Erster und letzter Punkt“ und das verwerfend-strombasierte Verfahren „PLAZA“ annähernd gegeben ist. Bei den anderen Verfahren liegt der durchschnittliche Fehler ca. 40 % niedriger.

Eigenheiten der Verfahren

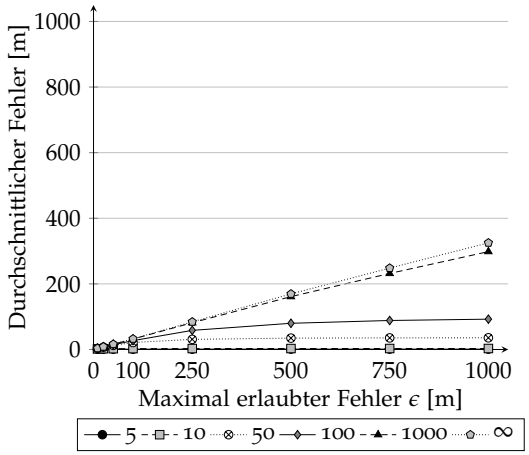
Das lineare Regressionsverfahren (siehe Abbildung 9.5c) erreicht im Vergleich mit den Verfahren „Douglas-Peucker“, „Distanzbasiert“, „Boxer, Leu und Chen“ und „Lineare Extrapolation“ für Fenstergrößen bis 100 Punkte einen deutlich kleineren durchschnittlichen Fehler.



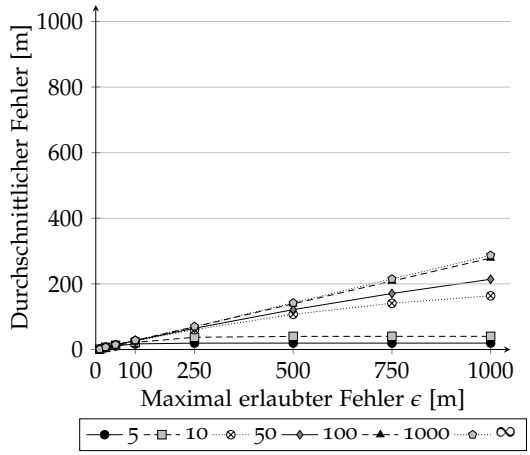
(a) Douglas-Peucker



(b) Boxer, Leu und Chen



(c) Lineare Regression



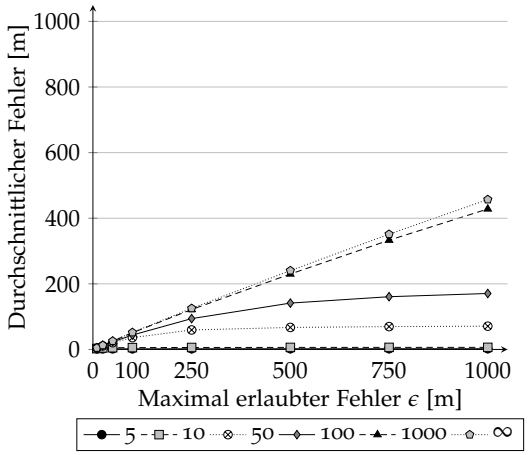
(d) Distanzbasiert

Abbildung 9.5.: Durchschnittlicher Fehler der Vereinfachung (1). Für Fenstergrößen von 5, 10, 50, 100, 1000 und ∞ vielen Punkten.

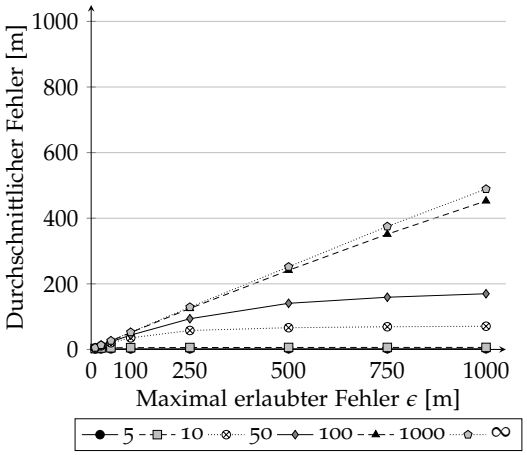
9.1.3. Median der Fehler

Der Median der Fehler liegt für alle Verfahren unterhalb des durchschnittlichen Fehlers. Das bedeutet, dass es im Ursprungspolygon einige Punkte gibt, die sehr weit vom vereinfachten Polygonzug entfernt liegen, der größere Teil der Punkte aber relativ gut vom vereinfachten Polygonzug angenähert wird.

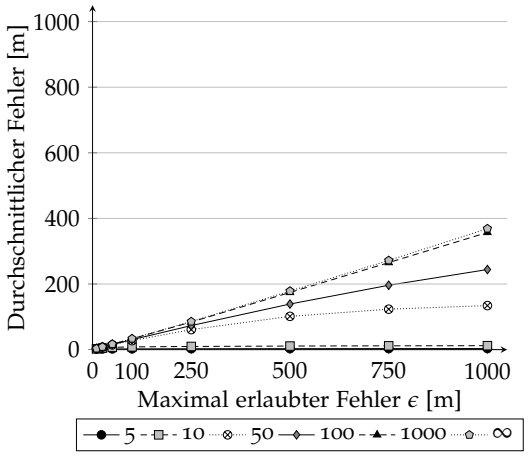
Für die strombasierten Verfahren „PLAZA“ und „Erster und letzter Punkt“ (siehe Abbildungen 9.8a bis 9.8b auf Seite 77) liegt der Median deutlich höher, als bei den anderen Verfahren (siehe Abbildungen 9.7a bis 9.7d auf Seite 76).



(a) PLAZA



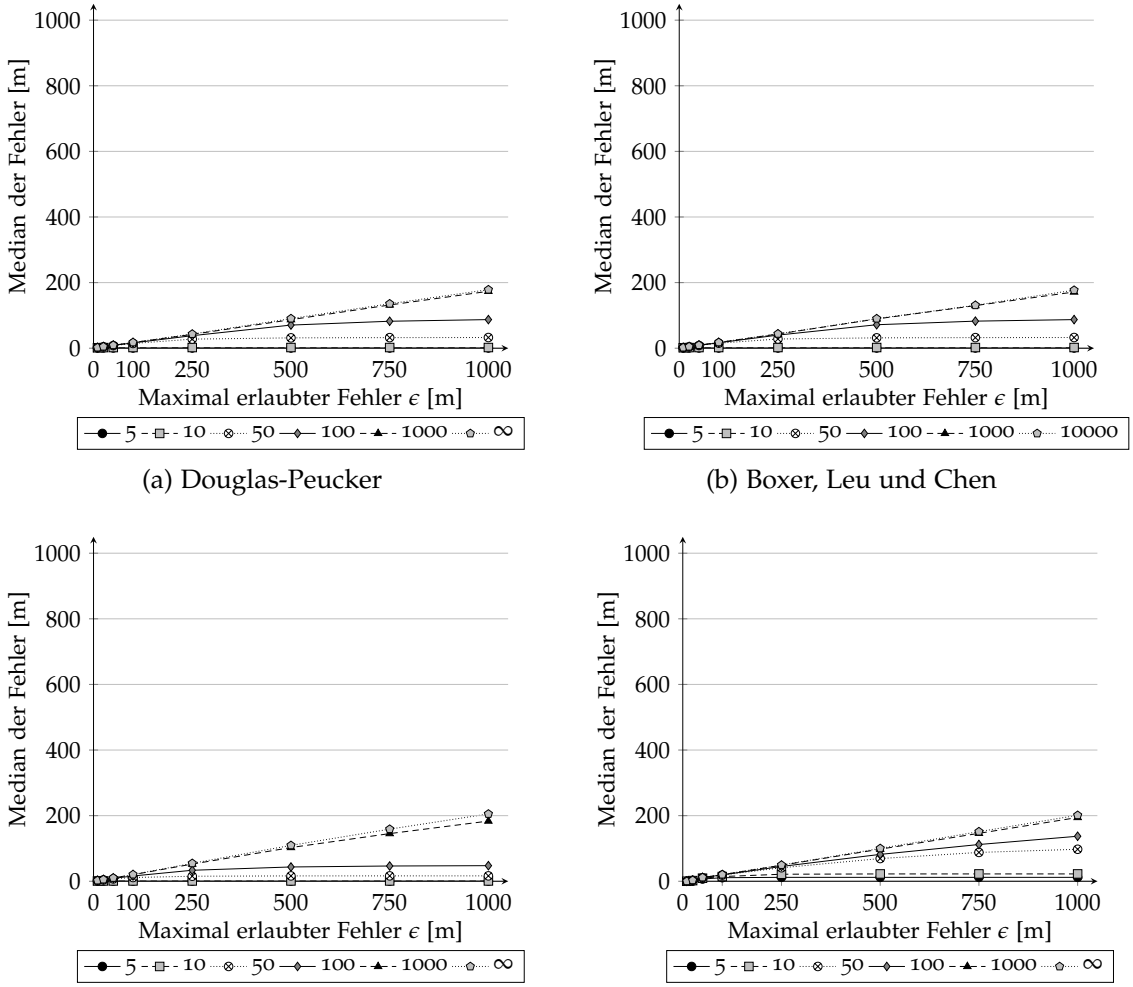
(b) Erster und letzter Punkt



(c) Lineare Extrapolation

Abbildung 9.6.: Durchschnittlicher Fehler der Vereinfachung (2). Für Fenstergrößen von 5, 10, 50, 100, 1000 und ∞ vielen Punkten.

Das ist damit zu erklären, dass strombasierte Verfahren den Polygonzug nicht so genau annähern können wie fensterbasierte Verfahren, da sie nur einzelne Punkte bearbeiten. Die fensterbasierten Verfahren arbeiten dagegen auf Teilpolygonzügen und können diese so komprimieren, dass sie näher am ursprünglichen Polygonzug liegen.



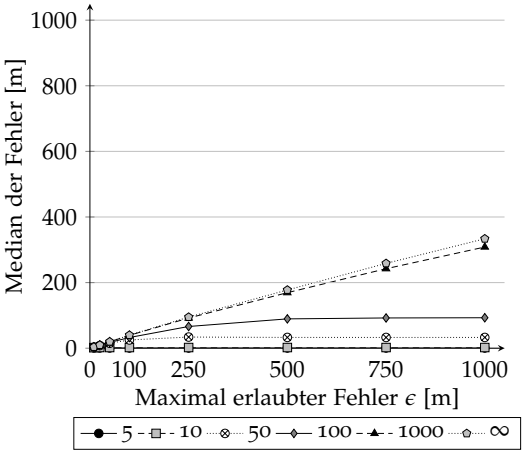
(a) Douglas-Peucker

(b) Boxer, Leu und Chen

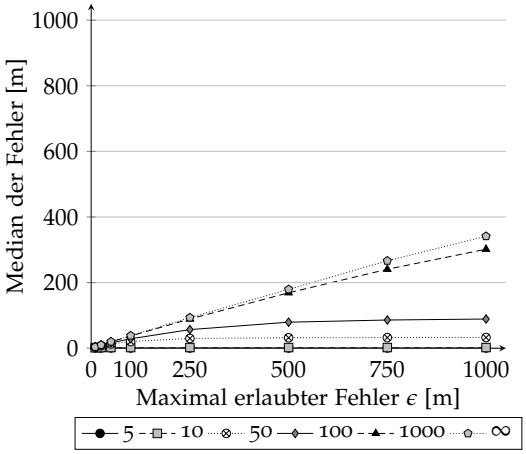
(c) Lineare Regression

(d) Distanzbasiert

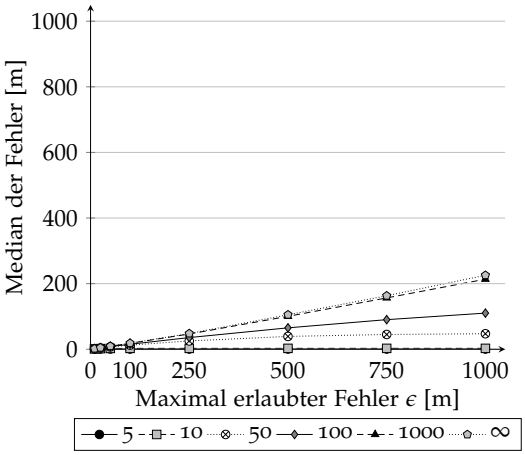
Abbildung 9.7.: Durchschnittlicher Median der Fehler der Vereinfachung (1). Für Fenstergrößen von 5, 10, 50, 100, 1000 und ∞ vielen Punkten.



(a) PLAZA



(b) Erster und letzter Punkt



(c) Lineare Extrapolation

Abbildung 9.8.: Durchschnittlicher Median der Fehler der Vereinfachung (2). Für Fenstergrößen von 5, 10, 50, 100, 1000 und ∞ vielen Punkten.

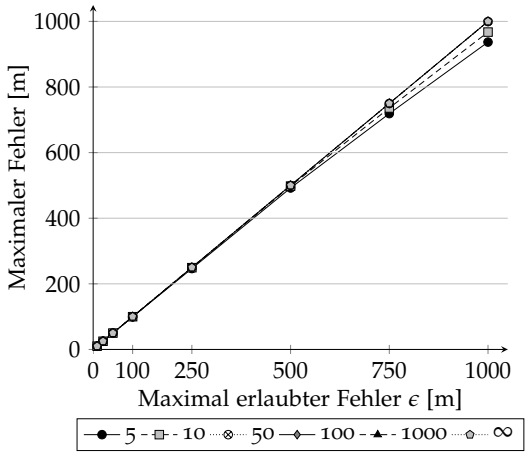


Abbildung 9.9.: Durchschnittlicher maximaler Fehler der Vereinfachung (1): Lineare Extrapolation. Für Fenstergrößen von 5, 10, 50, 100, 1000 und ∞ vielen Punkten.

9.1.4. Maximaler Fehler

Wenn die Verfahren den maximal erlaubten Fehler voll ausnutzen, bildet sich in den Schaubildern eine Diagonale. Durch den Einfluss der Fenstergröße auf die Kompression nähert sich die Kurve aber nur für sehr große Fenstergrößen dieser Diagonale an.

Das einzige Verfahren, bei dem die Diagonale auch bei kleinen Fenstergrößen fast erreicht wird, ist die „Lineare Extrapolation“ (siehe Abbildung 9.9).

9.1.5. Experimentelle Laufzeit

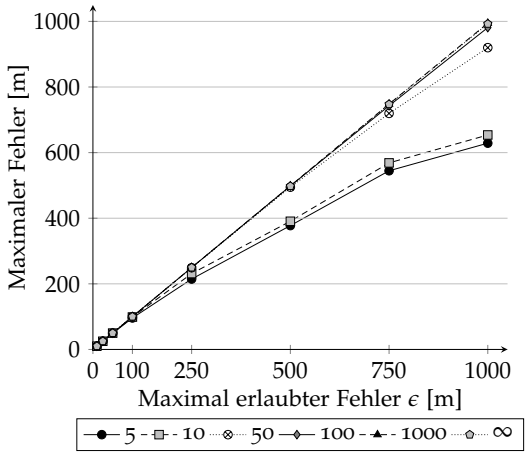
Allgemeine Beobachtungen

Die experimentelle Laufzeit der Verfahren wurde in Millisekunden pro verarbeitetem Punkt gemessen. Dabei ergeben sich für größere Fenstergrößen teilweise Laufzeitunterschiede von mehreren Größenordnungen.

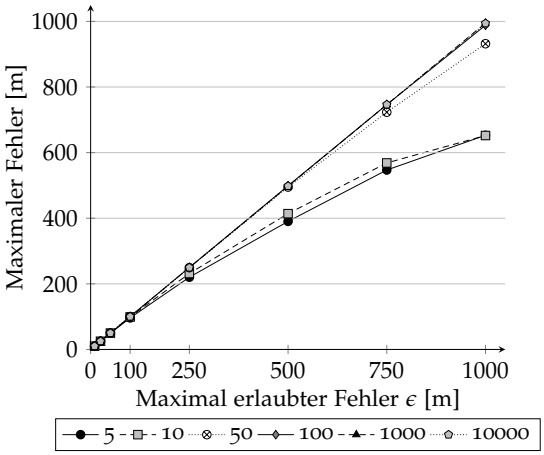
Für kleine Fenstergrößen sind die Unterschiede kleiner, das langsamste Verfahren „Boxer, Leu und Chen“ ist aber immer noch zehnmal langsamer als das schnellste Verfahren (siehe Abbildung 9.11).

Eigenheiten der Verfahren

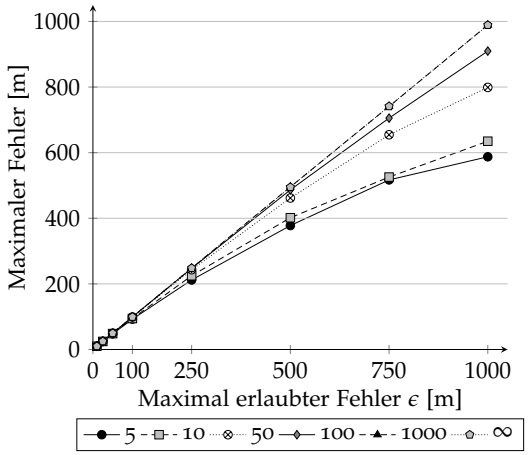
Die Verfahren „Douglas-Peucker“, „Lineare Extrapolation“, „PLAZA“ und „Distanzbasiert“ (siehe Abbildungen 9.12a bis 9.12d auf Seite 81) benötigen bei größer werdendem maximal erlaubtem Fehler immer weniger Zeit für die Vereinfachung. Dabei hat der maximal erlaubte



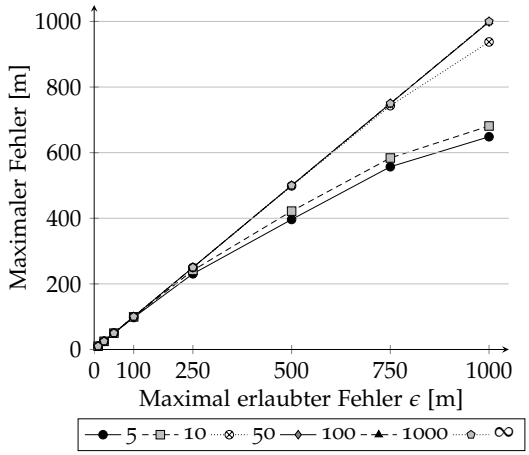
(a) Douglas-Peucker



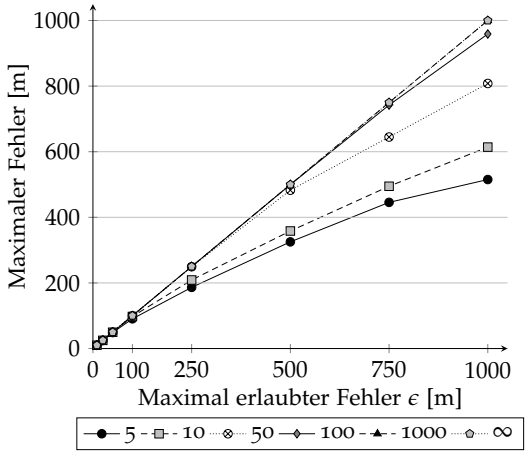
(b) Boxer, Leu und Chen



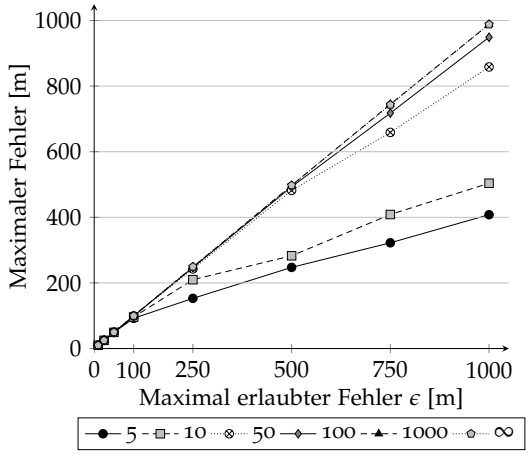
(c) PLAZA



(d) Erster und letzter Punkt



(e) Lineare Regression



(f) Distanzbasiert

Abbildung 9.10.: Durchschnittlicher maximaler Fehler der Vereinfachung (2). Für Fenstergrößen von 5, 10, 50, 100, 1000 und ∞ vielen Punkten.

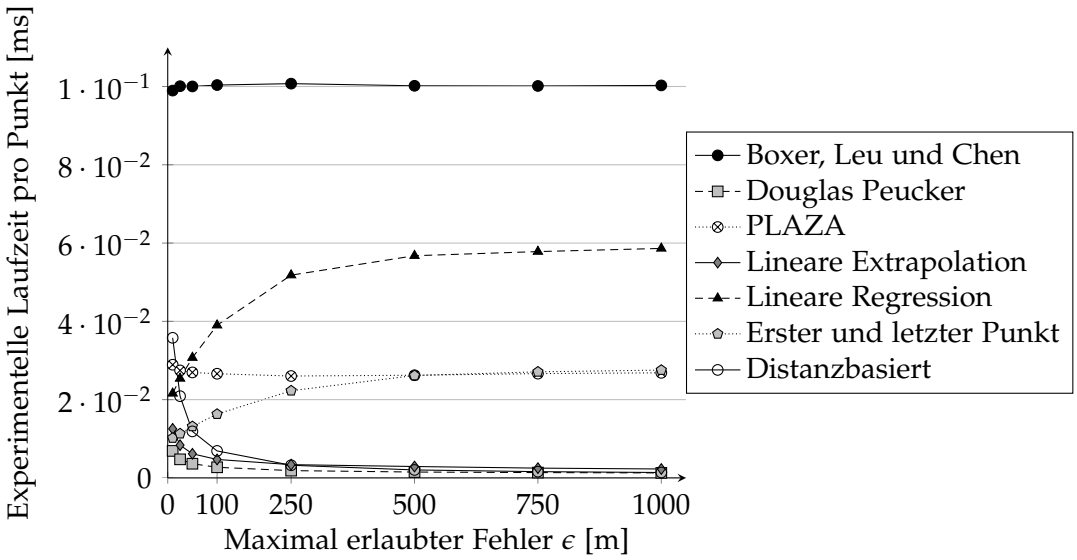


Abbildung 9.11.: Vergleich der durchschnittlichen experimentellen Laufzeit der Verfahren pro Punkt in Millisekunden bei einer Fenstergröße von 100 Punkten.

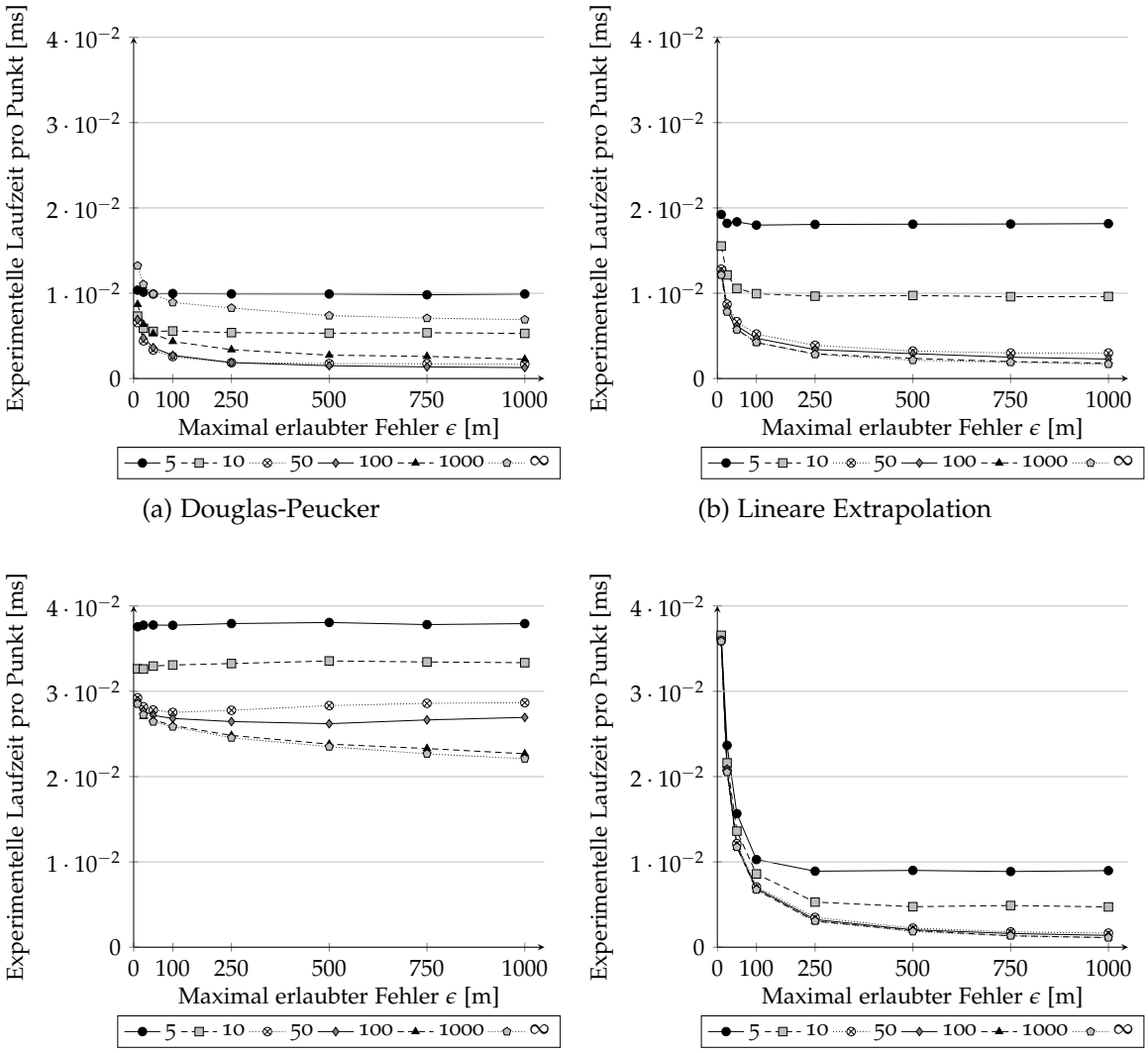
Fehler beim distanzbasierten Algorithmus den größten Einfluss, da bei ihm die Größe des vereinfachten Polygonzuges bei kleinen Werten von ϵ noch sehr groß ist.

Der „Douglas-Peucker“ Algorithmus ist für Fenstergrößen bis 100 Punkte für alle maximal erlaubten Fehler der schnellste Algorithmus. Erst für Fenstergrößen über 100 Punkten wird sein quadratisches Laufzeitverhalten sichtbar (siehe Abbildung 9.12a) und er fällt hinter den distanzbasierten Algorithmus zurück.

Die Laufzeit des „PLAZA“ Verfahrens steigt für Fenstergrößen bis 100 Punkte und für große maximal erlaubte Fehler leicht an. Dieses Verhalten war auch bei mehrfachen Simulation zu beobachten, es ist also nicht auf äußere Einflüsse während der Simulation zurückzuführen, sondern eine Eigenschaft des Verfahrens.

Die Verfahren „Lineare Regression“ und „Erster und letzter Punkt“ haben eine asymptotische Laufzeit von $O(n^2)$. Dadurch steigt bei diesen Verfahren die experimentelle Laufzeit für große Fenstergrößen stark an (siehe Abbildungen 9.13a bis 9.13b auf Seite 82).

Beim Verfahren „Boxer, Leu und Chen“, das ebenfalls eine quadratische Laufzeit aufweist, ist die experimentelle Laufzeit dagegen fast nur von der Fenstergröße abhängig (siehe Abbildung 9.14). Für große Fenstergrößen ist das Verfahren mehrere Größenordnungen langsamer als alle anderen Verfahren. Aus diesem Grund wurde auf eine Simulation mit einer Fenstergröße, die größer als die Anzahl der Punkte des Datensatzes ist, verzichtet.



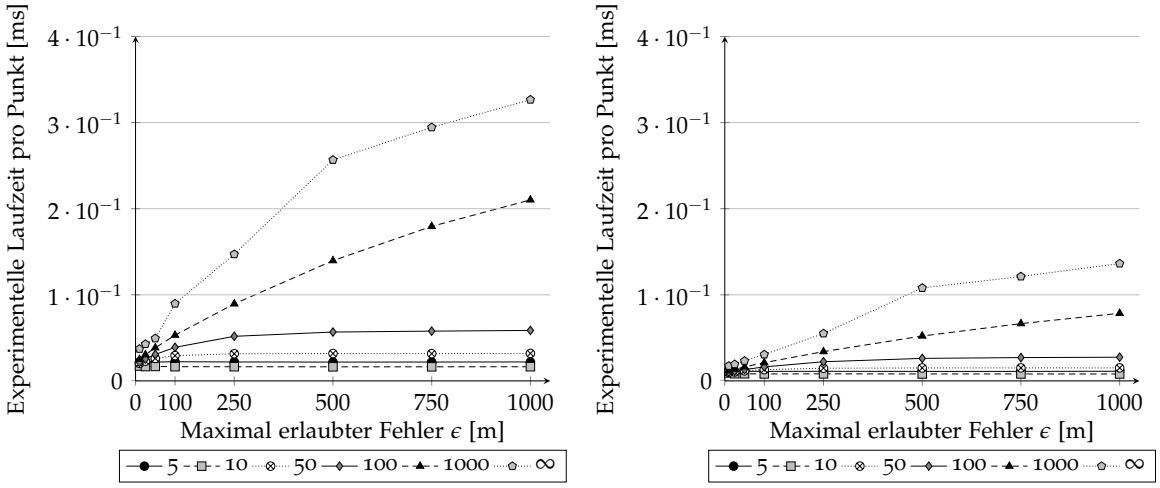
(a) Douglas-Peucker

(b) Lineare Extrapolation

(c) PLAZA

(d) Distanzbasiert

Abbildung 9.12.: Durchschnittliche experimentelle Laufzeit der Verfahren pro Punkt in Millisekunden (1). Für Fenstergrößen von 5, 10, 50, 100, 1000 und ∞ vielen Punkten.



(a) Regression

(b) Erster und letzter Punkt

Abbildung 9.13.: Durchschnittliche experimentelle Laufzeit der Verfahren pro Punkt in Millisekunden (2)

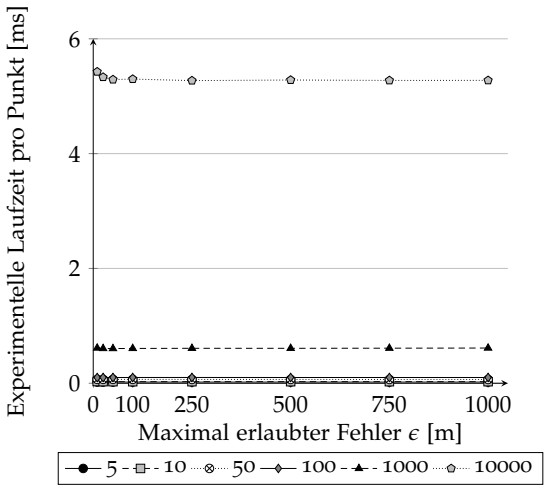


Abbildung 9.14.: Durchschnittliche experimentelle Laufzeit der Verfahren pro Punkt in Millisekunden (3): Boxer, Leu und Chen. Für Fenstergrößen von 5, 10, 50, 100, 1000 und ∞ vielen Punkten.

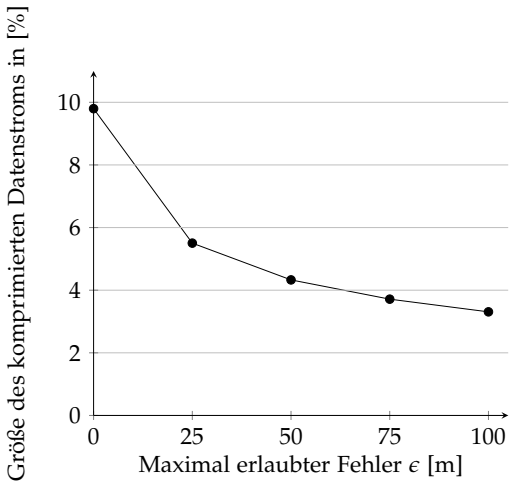


Abbildung 9.15.: Größe des komprimierten Datenstroms nach Kompression mit dem kartenbasierten Verfahren in % des Ursprungsdatenstroms mit einen Suchradius von $\delta = 500$ Metern. Der maximal erlaubte Fehler ϵ wird für den vorgeschaltenen distanzbasierten Algorithmus verwendet.

9.2. Kartenbasierter Algorithmus

Der angepasste kartenbasierte Algorithmus von Jensen und Čivilis wurde sowohl mit als auch ohne vorgeschaltenen distanzbasierten Algorithmus untersucht. Die Verwendung eines maximal erlaubten Fehlers von $\epsilon = 0$ entspricht in den Diagrammen dem Verfahren ohne distanzbasierten Filter.

9.2.1. Datenkompression

Die durch den einfachen Kartenabgleich erhoffte hohe Kompressionsrate konnte nicht erreicht werden. Selbst wenn der Speicherbedarf für die Straßenkarte nicht berücksichtigt wird, benötigt der einfache kartenbasierte Algorithmus ab einem maximal erlaubten Fehler von 50 Metern und bei einer Fenstergröße von 100 Punkten mehr Punkte, als die meisten Verfahren zur Polygonzugvereinfachung.

Durch Vorschalten des distanzbasierten Algorithmus zur Polygonzugvereinfachung kann jedoch die Kompressionsrate erhöht werden. Dabei wird der Positionsdatenstrom durch den distanzbasierten Algorithmus gefiltert und erst danach mit der Karte abgeglichen. Das verkürzt sowohl die Laufzeit, als auch die Genauigkeit des Kartenabgleichs im Kreuzungsbereich.

Da der distanzbasierte Algorithmus im Kreuzungsbereich und an anderen Stellen, an denen sich das mobile Objekt langsam bewegt, einige Punkte aus dem Datenstrom herausfiltert,

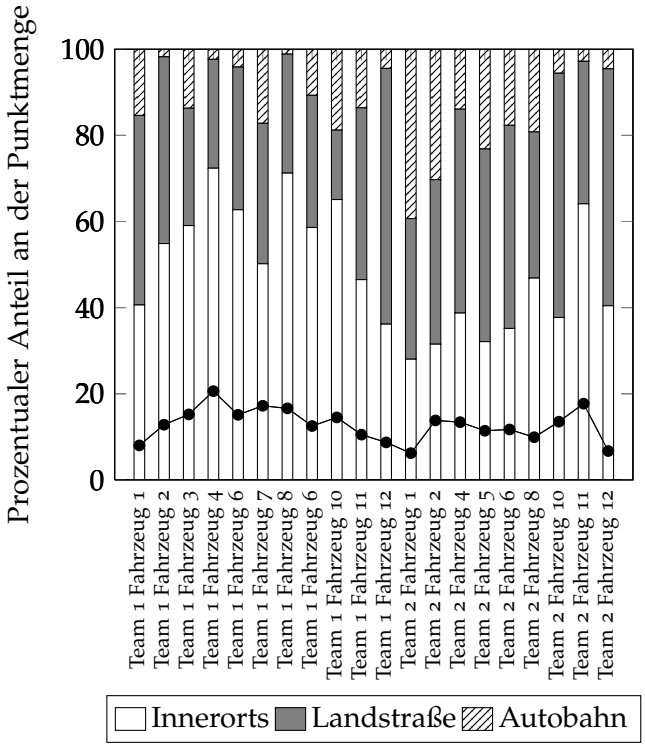


Abbildung 9.16.: Prozentuale Anteile der verschiedenen Straßenarten an den Datensätzen und die prozentuale Größe des komprimierten Datenstroms —●—. Es ist zu sehen, dass die Größe des komprimierten Datenstroms größer ist, wenn der Innenstadtanteil relativ groß ist.

wird die Kompressionsrate durch diesen erhöht. In Abbildung 9.15 ist die Größe des komprimierten Datenstroms für verschiedene Werte von ϵ dargestellt. Der Suchradius δ für den Kartenabgleich betrug 500 Meter.

Die Größe des komprimierten Datenstroms hängt maßgeblich von der Anzahl und der Länge der befahrenen Straßensegmente ab. Auf Autobahnen und Landstraßen sind die Straßensegmente im Allgemeinen länger als in Städten. Dadurch lässt sich der Datenstrom außerhalb von Städten stärker komprimieren (siehe Abbildung 9.16), da hier mehr Punkte auf dieselben Straßensegmente abgeglichen werden und weniger Wechsel des Straßensegments stattfinden.

9.2.2. Experimentelle Laufzeit

Die im Rahmen dieser Arbeit implementierte einfache Kartenverwaltung hat keinen Geo-Index. Zum Kartenabgleich müssen deshalb immer alle Straßen durchlaufen werden, wodurch das Verfahren sehr langsam wird. Das Verfahren lässt sich stark beschleunigen, wenn

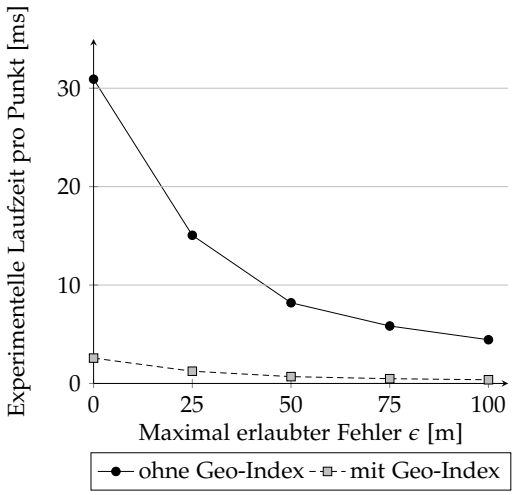


Abbildung 9.17.: Experimentelle Laufzeit des kartenbasierten Verfahrens in Millisekunden pro Punkt des Ursprungsdatenstroms mit einen Suchradius von $\delta = 500$ Metern.

mit einer räumlichen Abfrage die Straßenanzahl reduziert wird, deren Kartenabgleichsgewicht berechnet werden muss.

Da die experimentelle Laufzeit proportional zur Anzahl der abgeglichenen Punkte ist, lässt sie sich senken, indem dem Kartenabgleich der distanzbasierte Algoritmus als Filter vorgeschalten wird, da dieser die Punktanzahl vor dem Kartenabgleich reduziert. In Abbildung 9.17 ist dargestellt, wie sich die experimentelle Laufzeit des Verfahrens in Millisekunden pro Punkt des Ursprungsdatenstroms verhält.

Da die verwendete Komponente zur Kartenverwaltung keinen Geo-Index besitzt wurde zur Bestimmung der experimentellen Laufzeit mit Geo-Index zu einem Trick gegriffen. Da die Laufzeit dadurch verlängert wird, dass alle Straßen der großen Straßenkarte durchsucht werden müssen, wurde ein kleiner Kartenausschnitt daraus ausgewählt (siehe Abbildung 8.2). Mit diesem Ausschnitt wurden alle Punkte mit einem Suchradius von $\delta = 100\,000$ Metern abgeglichen. Der große Suchradius wurde gewählt, damit jeder Punkt auf eines der Segmente abgeglichen werden kann, um realistische Werte für die experimentelle Laufzeit zu erhalten.

Der Kartenausschnitt enthält nur 1,4 % der Straßen der gesamten Karte. Die Anzahl der Straßensegmente ist daher ähnlich eingeschränkt, wie dies durch einen Geo-Index möglich ist.

In Abbildung 9.17 ist zu sehen, wie sich die experimentelle Laufzeit durch diese Reduktion der Straßenanzahl reduziert. Das Verfahren wurde stark beschleunigt, es ist aber auch unter diesen Bedingungen noch mehr als eine Größenordnung langsamer als die Verfahren zur Polygonzugvereinfachung, mit Ausnahme des Verfahrens von Boxer, Leu und Chen.

Kapitel 10

Zusammenfassung und Ausblick

10.1. Algorithmen zur Polygonzugvereinfachung

In dieser Arbeit wurde gezeigt, dass sich Algorithmen zur Polygonzugvereinfachung zur Kompression von Positionsdatenströmen eignen.

Die Polygonzugvereinfachungsverfahren „PLAZA“, „Douglas-Peucker“ und „Boxer, Leu und Chen“ können die zur Simulation verwendeten Positionsdatenströme um bis zu 97 % komprimieren, ohne um mehr als 50 Meter vom ursprünglichen Positionsdatenstrom abzuweichen. Von den bereits von Kaya untersuchten Verfahren erreicht das Verfahren „Erster und letzter Punkt“ als Einziges eine etwas höhere Kompressionsrate, es weicht aber durchschnittlich deutlich weiter vom Ursprungsdatenstrom ab, als das „Douglas-Peucker“ Verfahren und das Verfahren von Boxer, Leu und Chen.

Um bei der Datenstromverarbeitung einen maximalen Zeitabstand zwischen den Punkten des komprimierten Datenstroms zu garantieren, wurde ein Zeitfenster verwendet. Die Anzahl der Punkte, die in ein Zeitfenster fallen, beschränken die Größe des komprimierten Datenstroms nach unten. Die Simulationsergebnisse zeigen, dass die Größe des komprimierten Datenstroms ab einem Fenster, in das mindestens 100 Punkte passen, kaum noch kleiner wird. Eine Fenstergröße von ungefähr 1000 Punkten ergibt das beste Verhältnis aus Kompression, Laufzeit und Aktualisierungsfrequenz.

Beim Vergleich aller Eigenschaften der Verfahren fällt auf, dass das Douglas-Peucker Verfahren den anderen Verfahren überlegen ist. Es weist gute Werte für den durchschnittlichen und den maximalen Fehler auf, der Median der Fehler liegt niedrig und seine experimentelle Laufzeit ist die kürzeste aller untersuchten Verfahren (siehe Tabelle 10.1). Zudem ist es das einzige untersuchte Verfahren, das unter Verwendung der Zeitgleichheitsdistanz bei mehrfacher Vereinfachung fehlerbeschränkt ist. Es hat dadurch den Vorteil, dass zum Einhalten eines maximal erlaubten Fehlers nicht bekannt sein muss, ob der Datenstrom bereits komprimiert wurde oder nicht.

	Größe der Vereinfachung	Experimentelle Laufzeit	Durchschnittlicher Fehler	Maximaler Fehler	Median der Fehler
Lineare Extrapolation	11,25 %	5,76 μs	16,61 m	49,9975 m	9,18 m
Erster und letzter Punkt	**3,40 %**	16,22 μs	26,07 m	**49,9985 m**	19,45 m
Lineare Regression	5,01 %	38,48 μs	15,98 m	49,9981 m	10,04 m
Distanzbasiert	26,75 %	11,71 μs	**14,32 m**	49,7658 m	20,00 m
PLAZA	3,61 %	26,65 μs	25,38 m	49,6613 m	19,55 m
Douglas-Peucker	3,96 %	**5,26 μs**	14,98 m	49,9459 m	**9,06 m**
Boxer, Leu und Chen	4,06 %	604,20 μs	15,27 m	49,9560 m	9,33 m

Tabelle 10.1.: Übersicht über die Simulationsergebnisse der Algorithmen zur Polygonzugvereinfachung. Gemessen bei einem maximal erlaubten Fehler von 50 Metern und einer Fenstergröße von 1000 Punkten.

Ausblick

Da die in dieser Arbeit verwendete Implementierung des Douglas-Peucker-Verfahrens eine worst-case-Komplexität von $O(n^2)$ aufweist, sollte untersucht werden, wie stark sich die experimentelle Laufzeit durch die Verwendung laufzeitoptimierter Implementierungen senken lässt, wie sie Herschberger und Snoeyink in [HS92] vorschlagen.

Das Verfahren von Boxer, Leu und Chen war in der Simulation das mit Abstand langsamste Verfahren zur Polygonzugvereinfachung. Das ist auf die vielen komplexen Datenstrukturen zurückzuführen, die im Algorithmus verwendet werden. Laut [Box+93] lässt sich das Verfahren aber so implementieren, dass seine Zeitkomplexität auf $\Theta(n)$ gesenkt wird. Dadurch steigt die Komplexität der Implementierung aber weiter an. Es bleibt zu untersuchen, ob sich dadurch in der Simulation eine signifikante Verkürzung der Laufzeit ergibt.

10.2. Kartenbasierter Algorithmus

Die Vermutung, ein kartenbasierter Algorithmus könnte höhere Kompressionsraten erreichen, als Verfahren zur Polygonzugvereinfachung, bestätigte sich nicht. Je nach Datensatz schwankt beim kartenbasierten Algorithmus die Größe des komprimierten Datenstroms zwischen 6,2 % und 20,6 % des ursprünglichen Datenstroms.

Die Kompressionsrate hängt beim kartenbasierten Verfahren neben dem Positionsdatenstrom auch von der Karte selbst ab. Enthält diese viele kurze Straßenabschnitte, werden viel mehr Punkte für den komprimierten Datenstrom benötigt, als mit Karten mit vielen langen Straßenabschnitten. Das kartenbasierte Verfahren ist daher eher für Autobahnstrecken, als für Fahrten in Innenstädten geeignet.

Es hat sich gezeigt, dass sich die Kompressionsrate erhöhen lässt, wenn der distanzbasierte Algorithmus vor dem Kartenabgleich ausgeführt wird. Dadurch werden insbesondere in Kreuzungsbereichen, in denen sich die mobilen Objekte langsam bewegen, viele Punkte gefiltert und damit der Abgleich mit fehlerhaften Straßensegmenten vermieden.

Die Simulation hat gezeigt, dass das Verfahren auch bei Verwendung eines Geo-Index um mehr als eine Größenordung langsamer ist als die Verfahren zur Polygonzugvereinfachung.

Neben der schlechten Kompressionsrate und der Laufzeit sprechen einige weitere Argumente gegen das kartenbasierte Verfahren. So sind die Punkte relativ zu Straßensegmenten angegeben und die Position in Geokoordinaten ist daher nicht direkt ablesbar. Außerdem ist es aufwendig, den Pfad aus den Daten zu rekonstruieren. Insgesamt ist zudem unklar, welche Anwendungen Nutzen aus kartenbasierten komprimierten Daten ziehen könnten.

Ausblick

Bei den kartenbasierten Verfahren bleibt zu untersuchen, wie stark sich die Kompressionsrate durch Vorverarbeitung des Kartenmaterials verbessern lässt.

Danksagung

An dieser Stelle möchte ich den folgenden Personen (in alphabetischer Reihenfolge) danken, ohne die meine Arbeit in dieser Form nicht möglich gewesen wäre.

- **Renate Gommeringer**, dafür dass sie diese Arbeit Korrektur gelesen hat und mich mit ihren Anmerkungen unterstützt hat,
- **Dipl.-Inf. Nicola Hönle**, für den Themenvorschlag und die Betreuung der Arbeit,
- **Prof. Christian S. Jensen** von der Universität Aalborg, für seine Unterstützung mit den INFATI-Datensätzen und den dazugehörenden Dokumenten,
- **Wiltrud Kessler**, dafür dass sie diese Arbeit Korrektur gelesen hat,
- **Dipl.-Inf. Carlos Lübbe**, für die Erläuterung des Operatorenkonzepts,
- und **meinen Eltern Andrea und Hans-Peter Reimann**, dafür dass sie diese Arbeit Korrektur gelesen und mir wertvolle Tipps gegeben haben.

Vielen Dank!

Anhang A

Koordinatentransformation der INFATI-Daten

A.1. Vorgehensweise

Die INFATI-Daten liegen in einem sehr ausführlichen Format vor [Jen+04; JS04]. Zudem sind die Koordinatendaten der GPS Pfade in Universalen Transversalen Mercator (UTM) Koordinaten angegeben. Da die verfügbaren Kartendaten des OpenStreetMap Projekts [Ope08] in geodätischen Koordinaten angegeben sind, wurde entschieden, überall geodätische Koordinaten zu verwenden. Deshalb müssen die UTM-Koordinaten der INFATI-Daten zur Weiterverarbeitung in geodätische (Breitengrad, Längengrad, Zeit)-Koordinaten umgewandelt werden.

Beide Datenquellen verwenden das Referenzellipsoid des „World Geodetic System 1984“ (WGS84). Ausführliche Informationen dazu finden sich in [Nat00].

Für diese Umwandlung muss ein mehrstufiger Prozess durchlaufen werden, der im Folgenden erklärt wird. Zuerst werden die INFATI-Daten eingelesen und die für uns relevanten Informationen extrahiert. Dabei werden für die spätere Verarbeitung die Zeit- und die Koordinatendaten in separaten Dateien abgelegt. Nun werden mit einer externen Anwendung die Koordinaten transformiert. Daran anschließend werden die Zeit- und Koordinatendaten wieder zusammengeführt.

A.2. Umwandlung der INFATI-Pfade in (x,y,t)-Koordinaten

Eine Zeile in den INFATI-Datensätzen hat das folgende Format. Über die Datenwerte wurde zur Veranschaulichung jeweils der Spaltenname geschrieben, die Zeilen wurden aufgrund ihrer Länge bei den Punkten umgebrochen.

```
ID     ENTRYID       CARID DRIVERID RDATE  RTIME XCOORD YCOORD  ...
593833 1220101075851   1       0     220101 75851 556318 6321638 ...

... MPX    MPY     SAT HDOP MAXSPD SPD STRTCOD
... 556315 6321638  9   1     50   44   3640
```

Für uns relevant sind die Spalten `RDATE` und `RTIME` die den Zeitpunkt enthalten. `RDATE` enthält das Datum in der Form `TagMonatJahr`. `RTIME` enthält die Uhrzeit in der Form `StundenMinutenSekunden`. In beiden Fällen wurde eine führende Null, beim Tag bzw. bei der Stunde, während des Aufzeichnens weggelassen.

Zudem sind die Spalten `XCOORD` für den Rechtswert und `YCOORD` für den Hochwert der Universal Transverse Mercator (UTM) Koordinaten für uns relevant. Die Region, in der die Messpunkte aufgezeichnet wurden, befindet sich in der UTM-Zone 32V. Eine ausführlichere Beschreibung des Datenformates findet sich in [Jen+04].

Zur Vorbereitung der Umwandlung wird die ursprüngliche Datei eingelesen und es werden zwei Ausgabedateien erzeugt, die die Zeit sowie die Koordinatendaten getrennt enthalten.

Die Datei mit den Koordinatendaten enthält zusätzlich einen Header für das Programm „Geographic Translator 2.4.2“, das weiter unten vorgestellt wird. Die zweite Datei enthält nur die Zeitpunkte als Unix-Timestamp.

A.3. Umwandlung der UTM-Koordinaten in geodätische Koordinaten

Für diesen Unmwandlungsschritt wird der „Geographic Translator 2.4.2“ der National Geospatial-Intelligence Agency der Vereinigten Staaten von Amerika [Nato8] genutzt. Da dieser nur räumliche Koordinaten umwandelt, wird der GPS-Pfad wie oben beschrieben auf zwei Dateien aufgeteilt. Die Datei mit den Koordinaten beginnt mit folgendem Header:

```
COORDINATES: Universal Transverse Mercator (UTM)
DATUM: WGE
# ELLIPSOID: WE
END OF HEADER

# Route
32, N, 556373.0, 6318773.0
32, N, 556374.0, 6318784.0
32, N, 556373.0, 6318795.0
...
```

Mit dem „Geographic Translator“ werden die Koordinaten in dieser Datei in geodätische Koordinaten umgewandelt. Dazu müssen weitere Optionen gesetzt werden. Die geodätischen Einheiten müssen auf Grad ohne weitere Unterteilung in Minuten oder Sekunden, die

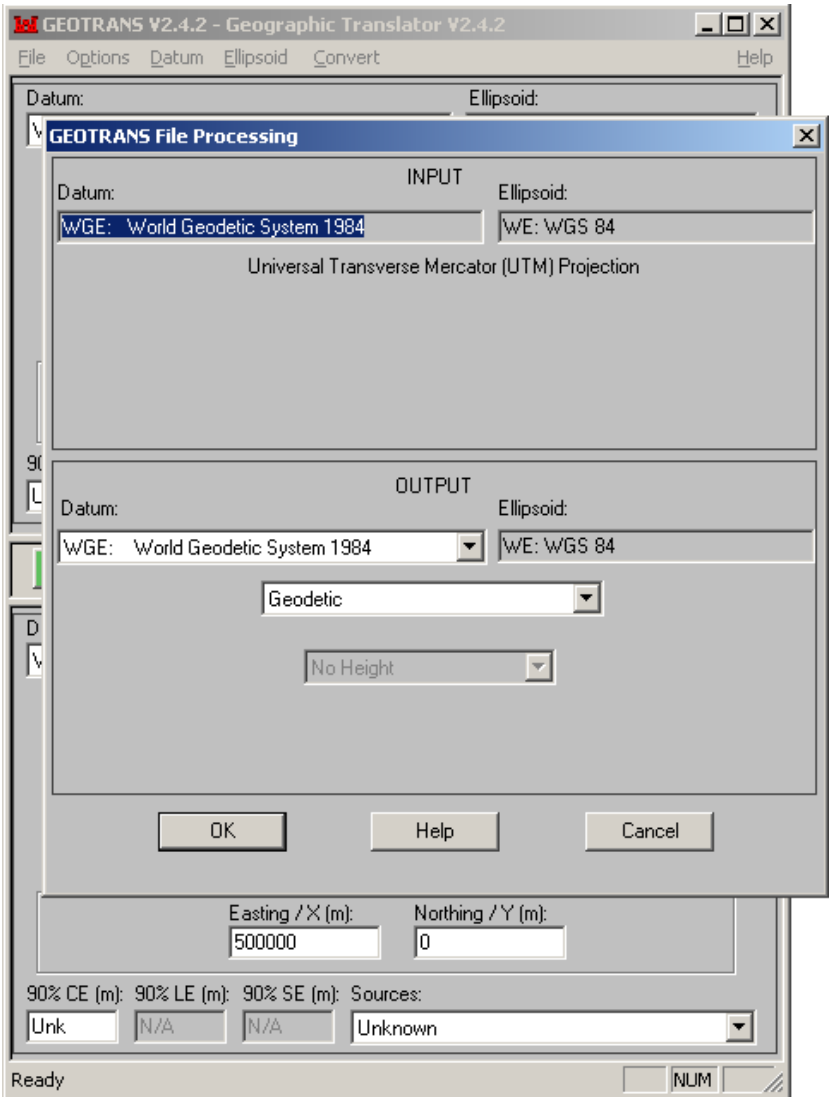


Abbildung A.1.: Koordinatentransformation mit dem „Geographic Translator“

Hemisphärendarstellung auf $+-$ anstelle von Nord/Süd gesetzt werden. Danach wird die Datei geladen und konvertiert (siehe Abbildung A.1).

Zusammenführen der Zeit- und Raumdaten

Nach der Konvertierung der räumlichen Daten werden die getrennten Informationen wieder zusammengeführt. Ergebnis dieses Vorgangs ist eine Datei, die pro Zeile ein Wertetripel (Breitengrad, Längengrad, Zeit) enthält.

```
57.00902,9.92822,977493637
```

```
57.00912,9.92824,977493638
57.00922,9.92823,977493639
57.00933,9.92822,977493640
57.00942,9.92817,977493641
...
```

Diese Daten dienen als Eingabedaten für die Simulation.

Anhang B

Koordinatentransformation innerhalb des Operators

B.1. Gründe für die Umwandlung in ein metrisches System

Durch die elliptische Form der Erde ist ein Längen- oder Breitengrad an verschieden Orten unterschiedlich lang. Ein Längengrad entspricht am Äquator 1855,32 Metern [Hum09] und verkürzt sich dann in Richtung der Pole um den Faktor cos (Breitengrad). Ein Breitengrad bleibt nahezu konstant bei einer Länge von ungefähr 1856 Metern [Hum09]. Das gilt auch für den Bereich um den 57. Breitengrad, in dem sich die Testdaten befinden.

Durch diese veränderlichen Werte für die Längen der Gradmaße ist es schwer, Distanzen zwischen Koordinaten in Metern auszudrücken. Um diese Berechnungen zu vereinfachen werden alle Koordinaten beim Laden in die Vereinfachungskomponente in metrische Koordinaten in der Ebene umgerechnet. Auf diesen Daten können dann insbesondere euklidische Distanzen sehr einfach berechnet werden.

Für die Umrechnung der Längengrade wurde derselbe Wert von 1856 Metern genutzt wie für die Breitengrade, da sich die beiden Werte nur minimal unterscheiden.

B.2. Umwandlung der Koordinaten

B.2.1. Längengrade

Die Längengrade werden beim Import mit ihrer Länge in Metern am Äquator sowie dem Verkürzungsfaktor multipliziert.

$$x_{metrisch} = \cos(\text{Längengrad}) \cdot \text{Längengrad} \cdot 1856$$

B.2.2. Breitengrade

Die Breitengrade werden beim Import mit ihrer Länge in Metern multipliziert.

$$y_{metrisch} = \text{Breitengrad} \cdot 1856$$

Anhang C

Details zu den verwendeten Daten

C.1. Geschwindigkeitsverteilung der INFATI-Datensätze

Die INFATI-Datensätze enthalten, neben der Position und der Zeit, sowohl die gemessene Geschwindigkeit des Fahrzeugs, als auch die geltende Geschwindigkeitsbegrenzung. Die Verteilung dieser beiden Geschwindigkeitswerte wird in den nachfolgenden Diagrammen dargestellt. Die Datensätze sind nicht durchgehend nummeriert, da teilweise Datensätze von Jensen u. a. aus Datenschutzgründen entfernt wurden.

Mit ▯▯ wird die geltende Geschwindigkeitsbegrenzung dargestellt, mit ▮▮ die gemessene Geschwindigkeit. Zu Punkten mit einer Geschwindigkeitsbegrenzung von 0 km/h lagen keine Informationen über eine Geschwindigkeitsbegrenzung vor. Das deutet darauf hin, dass die Punkte von Jensen u. a. nicht auf die Straßenkarte abgeglichen werden konnten.

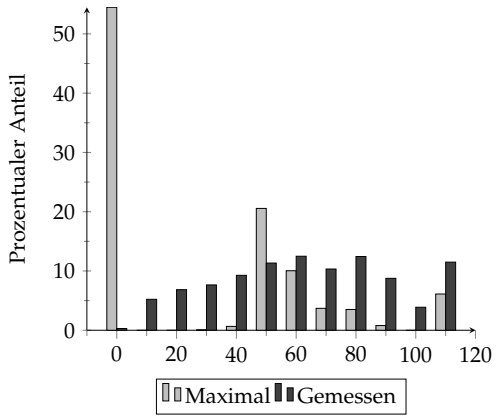


Abbildung C.1.: Charakteristik des Datensatzes Team 1 Fahrzeug 1

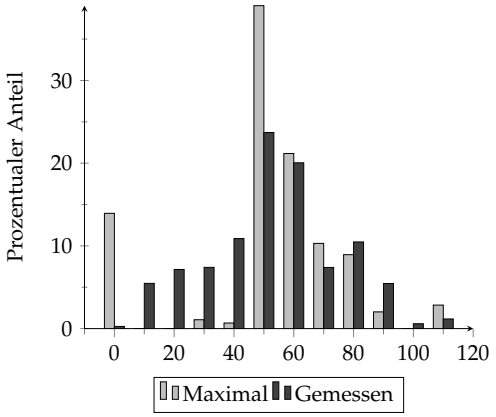


Abbildung C.2.: Charakteristik des Datensatzes Team 1 Fahrzeug 2

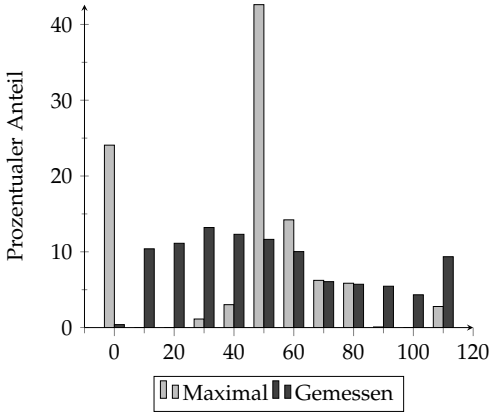


Abbildung C.3.: Charakteristik des Datensatzes Team 1 Fahrzeug 3

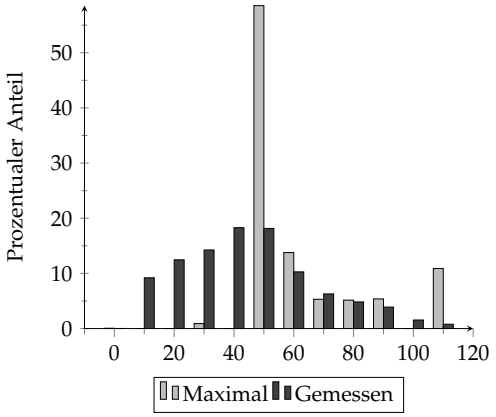


Abbildung C.4.: Charakteristik des Datensatzes Team 1 Fahrzeug 4

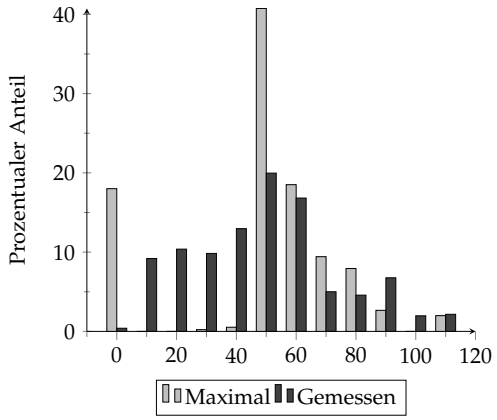


Abbildung C.5.: Charakteristik des Datensatzes Team 1 Fahrzeug 6

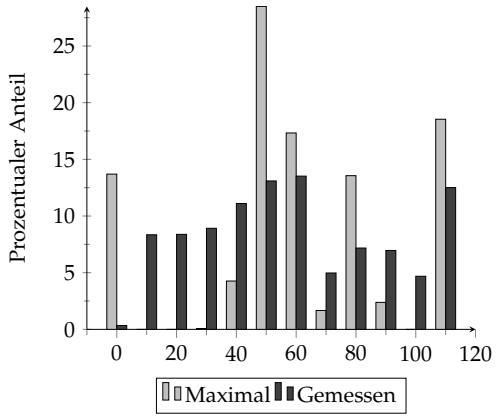


Abbildung C.6.: Charakteristik des Datensatzes Team 1 Fahrzeug 7

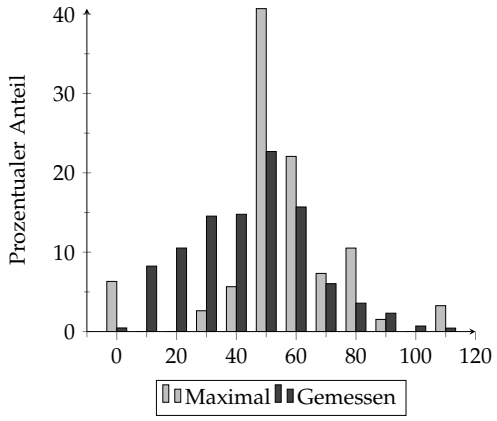


Abbildung C.7.: Charakteristik des Datensatzes Team 1 Fahrzeug 8

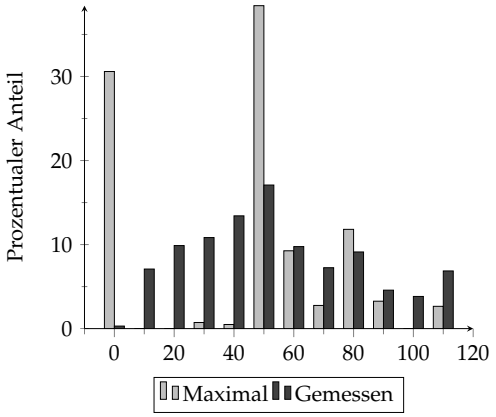


Abbildung C.8.: Charakteristik des Datensatzes Team 1 Fahrzeug 9

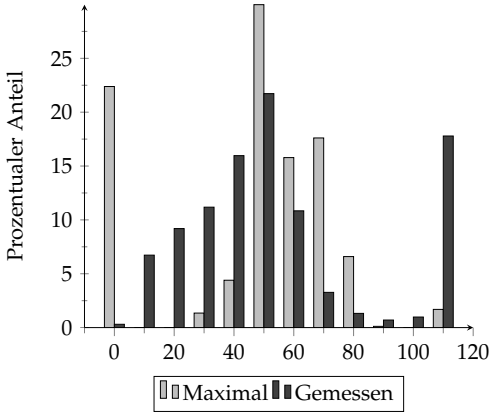


Abbildung C.9.: Charakteristik des Datensatzes Team 1 Fahrzeug 10

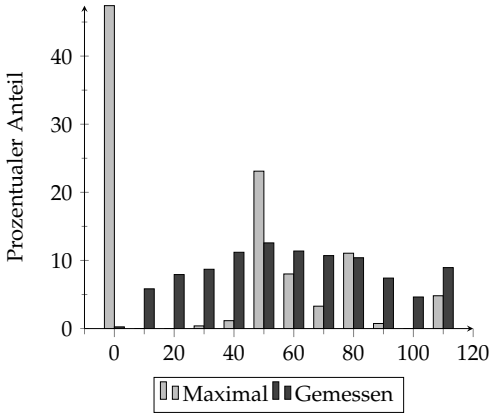


Abbildung C.10.: Charakteristik des Datensatzes Team 1 Fahrzeug 11

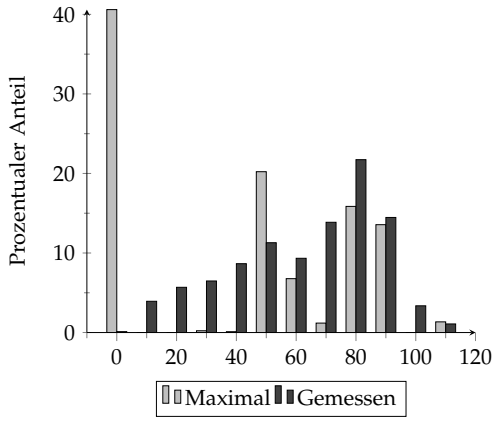


Abbildung C.11.: Charakteristik des Datensatzes Team 1 Fahrzeug 12

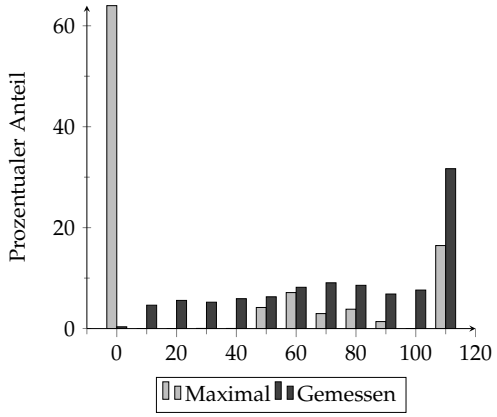


Abbildung C.12.: Charakteristik des Datensatzes Team 2 Fahrzeug 1

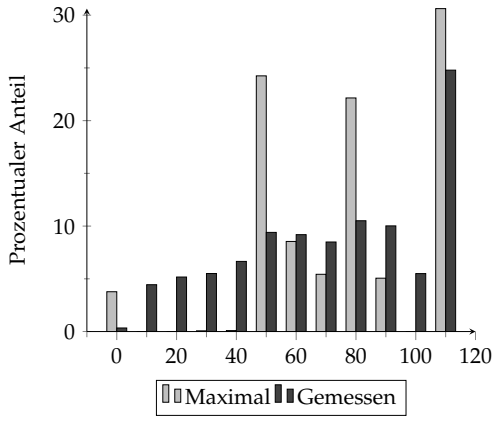


Abbildung C.13.: Charakteristik des Datensatzes Team 2 Fahrzeug 2

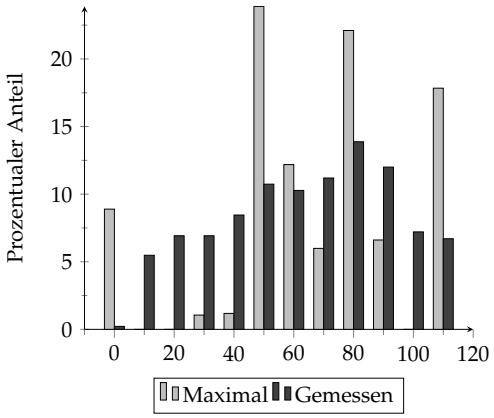


Abbildung C.14.: Charakteristik des Datensatzes Team 2 Fahrzeug 4

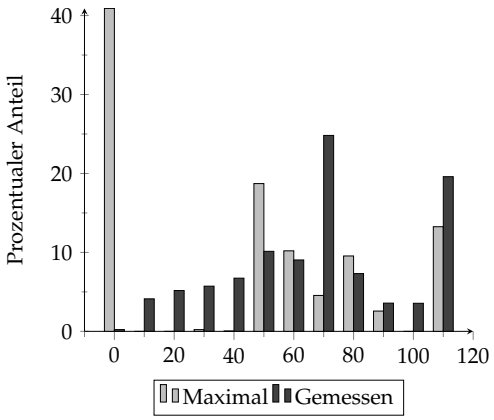


Abbildung C.15.: Charakteristik des Datensatzes Team 2 Fahrzeug 5

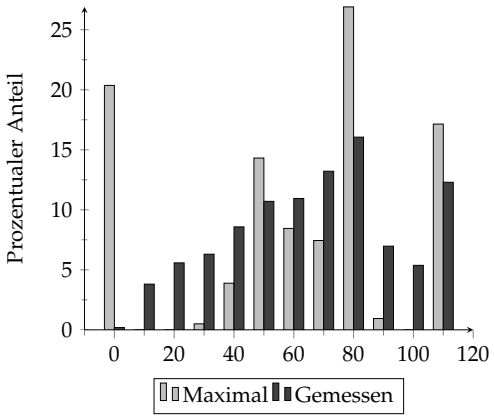


Abbildung C.16.: Charakteristik des Datensatzes Team 2 Fahrzeug 6

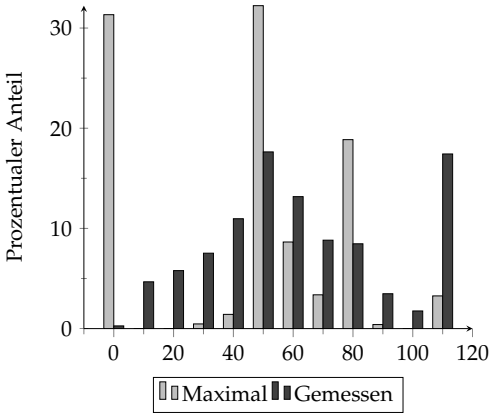


Abbildung C.17.: Charakteristik des Datensatzes Team 2 Fahrzeug 8

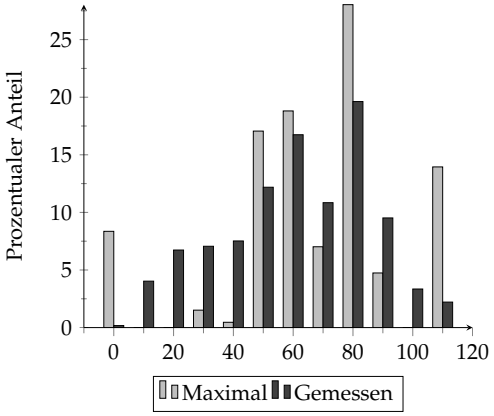


Abbildung C.18.: Charakteristik des Datensatzes Team 2 Fahrzeug 10

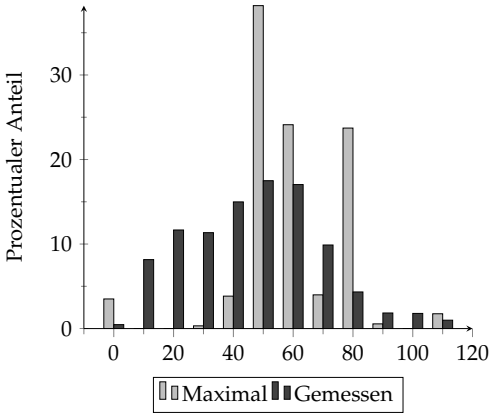


Abbildung C.19.: Charakteristik des Datensatzes Team 2 Fahrzeug 11

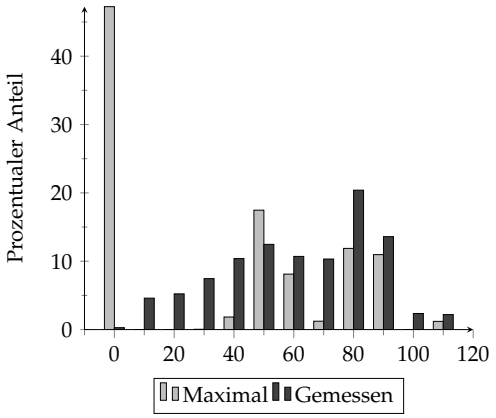


Abbildung C.20.: Charakteristik des Datensatzes Team 2 Fahrzeug 12

Datensatz	Anzahl Positionen	Punkte ohne max. Geschwindigkeit	Durchschnittliche Geschwindigkeit	Anteil Innerorts	Anteil Landstraße	Anteil Autobahn
Team 1 Fahrzeug 1	41355	54,48 %	63,41	40,61 %	44,02 %	15,37 %
Team 1 Fahrzeug 2	79517	13,94 %	52,41	54,88 %	43,37 %	1,75 %
Team 1 Fahrzeug 3	73079	24,07 %	52,68	59,06 %	27,26 %	13,68 %
Team 1 Fahrzeug 4	14228	0,06 %	43,92	72,38 %	25,28 %	2,34 %
Team 1 Fahrzeug 6	27185	18,01 %	48,32	62,73 %	33,15 %	4,12 %
Team 1 Fahrzeug 7	36331	13,70 %	58,26	50,18 %	32,63 %	17,19 %
Team 1 Fahrzeug 8	44261	6,32 %	43,80	71,24 %	27,63 %	1,13 %
Team 1 Fahrzeug 9	87613	30,59 %	53,46	58,61 %	30,70 %	10,69 %
Team 1 Fahrzeug 10	63427	22,38 %	54,98	65,10 %	16,13 %	18,77 %
Team 1 Fahrzeug 11	86547	47,41 %	59,67	46,50 %	39,91 %	13,59 %
Team 1 Fahrzeug 12	117717	40,60 %	62,36	36,16 %	59,40 %	4,44 %
Team 2 Fahrzeug 1	263437	63,99 %	79,87	28,02 %	32,67 %	39,31 %
Team 2 Fahrzeug 2	84834	3,78 %	73,02	31,51 %	38,21 %	30,28 %
Team 2 Fahrzeug 4	124751	8,89 %	62,85	38,74 %	47,35 %	13,91 %
Team 2 Fahrzeug 5	175210	40,91 %	68,45	32,10 %	44,75 %	23,15 %
Team 2 Fahrzeug 6	113623	20,36 %	65,99	35,16 %	47,18 %	17,67 %
Team 2 Fahrzeug 8	162681	31,33 %	63,90	46,86 %	33,95 %	19,19 %
Team 2 Fahrzeug 10	93550	8,36 %	60,21	37,71 %	56,73 %	5,56 %
Team 2 Fahrzeug 11	63474	3,50 %	45,90	64,11 %	33,10 %	2,79 %
Team 2 Fahrzeug 12	117510	47,25 %	60,72	40,43 %	55,03 %	4,55 %

Tabelle C.1.: Übersicht über die Geschwindigkeitsverteilung der INFATI-Datensätze.

C.2. Kartenauschnitt aus OpenStreetMaps

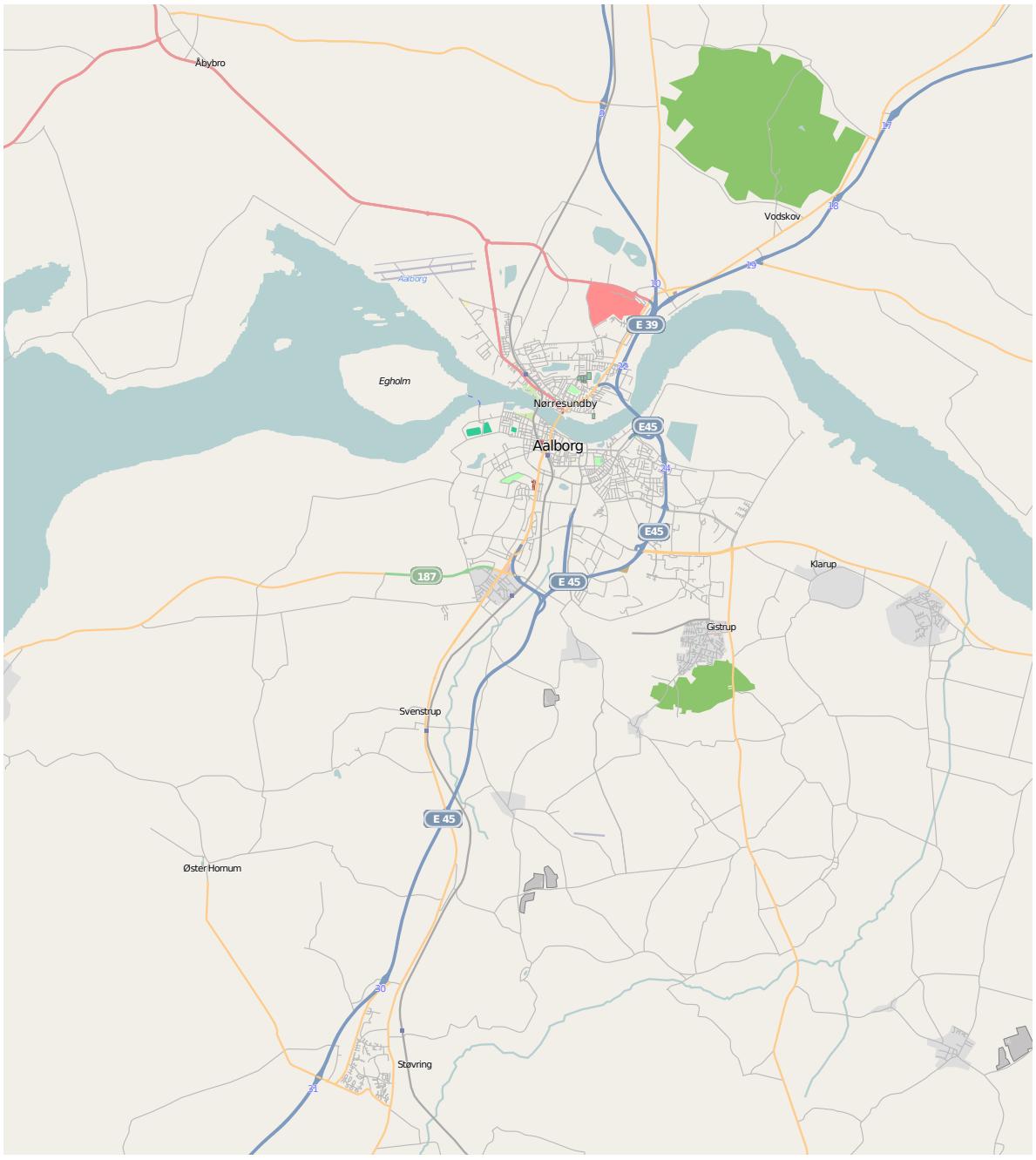


Abbildung C.21.: Kartenausschnitt der Stadt Aalborg (Dänemark) und ihrer Umgebung.

Literaturverzeichnis

[AA+07] Mohammad Ali Abam u. a. „Streaming algorithms for line simplification“. In: *SCG '07: Proceedings of the twenty-third annual symposium on Computational geometry*. Herausgegeben von Jeff Erickson. ACM, 2007, Seiten 175–183. Zitiert auf Seite 33.

[AG96] Helmut Alt und Leonidas J. Guibas. „Discrete geometric shapes: matching, interpolation, and approximation“. In: *Handbook of Computational Geometry*. Amsterdam: Elsevier Science Publishers B.V. North-Holland, 1996, Seiten 121–153. Zitiert auf Seiten 31, 32.

[Aga+05] Pankaj K. Agarwal u. a. „Near-Linear Time Approximation Algorithms for Curve Simplification“. In: *Algorithmica* 42.3 (Juni 2005), Seiten 203–219. Zitiert auf Seite 33.

[AV00] Pankaj K. Agarwal und Kasturi R. Varadarajan. „Efficient Algorithms for Approximating Polygonal Chains“. In: *Discrete & Computational Geometry* 23.2 (2000), Seiten 273–291. Zitiert auf Seite 33.

[BB82] Danna H. Ballard und Christopher M. Brown. *Computer Vision*. 2. Auflage. Englewood Cliffs: Prentice Hall, 1982, Seiten 232–247. Zitiert auf Seite 32.

[Box+93] Laurence Boxer u. a. „Polygonal approximation by boundary reduction“. In: *Pattern Recognition Letters* 14.2 (1993), Seiten 111–119. Zitiert auf Seiten 34, 45, 49, 50, 88.

[CC96] W. S. Chan und F. Chin. „Approximation of Polygonal Curves with Minimum Number of Line Segments or Minimum error“. In: *International Journal of Computational Geometry and Applications* 6.1 (1996), Seiten 59–77. Zitiert auf Seite 33.

[Cip+09] Nazario Cipriani u. a. „Verteilte Datenstromverarbeitung von Sensordaten“. In: *Datenbank-Spektrum* 9.28 (2009), Seiten 37–43. Zitiert auf Seiten 9, 11.

[Čiv+04a] Alminas Čivilis u. a. „Efficient Tracking of Moving Objects with Precision Guarantees“. In: *Annual International Conference on Mobile and Ubiquitous Systems* 0 (2004), Seiten 164–173. Zitiert auf Seite 57.

[Čiv+04b] Alminas Čivilis u. a. *Efficient Tracking of Moving Objects with Precision Guarantees*. Technischer Bericht. Database und Programming Technologies Institute at Aalborg University, 2004. Zitiert auf Seiten 57, 60.

[ČJP05] Alminas Čivilis, Christian S. Jensen und Stardas Pakalnis. „Techniques for Efficient Road-Network-Based Tracking of Moving Objects“. In: *IEEE Transactions on Knowledge and Data Engineering* 17.5 (Mai 2005), Seiten 698–712. Zitiert auf Seiten 54, 57.

[ČJP07] Alminas Čivilis, Christian S. Jensen und Stardas Pakalnis. Kapitel Tracking of Moving Objects with Accuracy Guarantees, Seiten 285–309. In: „Spatial Data on the Web“. Herausgegeben von Alberto Belussi u. a. Springer-Verlag, 2007. Zitiert auf Seite 57.

[CW05] Hu Cao und Ouri Wolfson. „Nonmaterialized Motion Information in Transport Networks“. In: *10th International Conference on Database Theory*. Herausgegeben von Thomas Eiter und Leonid Libkin. Band 3363. Lecture Notes in Computer Science. Springer-Verlag, 2005, Seiten 173–188. Zitiert auf Seite 55.

[CWT06] Hu Cao, Ouri Wolfson und Goce Trajcevski. „Spatio-temporal data reduction with deterministic error bounds“. In: *The International Journal on Very Large Data Bases* 15.3 (September 2006), Seiten 211–228. Zitiert auf Seiten 15, 17, 20–22, 33, 42, 43.

[DP73] David H. Douglas und Thomas K. Peucker. „Algorithms for the reduction of the number of points required to represent a digitized line or its caricature“. In: *Canadian Cartographer* 10.2 (Dezember 1973), Seiten 112–122. Zitiert auf Seiten 33, 41.

[Gre02] Joshua S. Greenfeld. „Matching GPS observations to locations on a digital map.“ In: *81th Annual Meeting of the Transportation Research Board*. 2002. Zitiert auf Seiten 8, 54, 56.

[HG97] Paul S. Heckbert und Michael Garland. *Survey of Polygonal Surface Simplification Algorithms*. Technischer Bericht. Carnegie-Mellon University Pittsburgh School Of Computer Science, 1997. Zitiert auf Seite 32.

[Hön+08] Nicola Hönle u. a. „Preprocessing Position Data of Mobile Objects“. In: *Mobile Data Management*. Herausgegeben von Xiaofeng Meng u. a. IEEE, 2008, Seiten 41–48. Zitiert auf Seite 9.

[HS92] John Hershberger und Jack Snoeyink. „Speeding Up the Douglas-Peucker Line-Simplification Algorithm“. In: *Proceedings of the 5th International Symposium on Spatial Data Handling*. Band 1. 1992, Seiten 134–143. Zitiert auf Seiten 41, 88.

[Hum09] Sigurd Humerfelt. *The Earth according to WGS 84 (every degree of latitude)*. 2009. URL: `http://home.online.no/~sigurdhu/Grid_1deg.htm`. Zitiert auf Seite 97.

[HWL08] Bernhard Hofmann-Wellenhof und Herbert Lichtenegger. *GNSS - Global Navigation Systems*. Herausgegeben von Elmar Wasle. Springer, 2008. Zitiert auf Seite 8.

[II86] Hiroshi Imai und Masao Iri. „An Optimal Algorithm for Approximating a Piecewise Linear Function“. In: *Journal of information processing* 9.3 (1986), Seiten 159–162. Zitiert auf Seiten 31, 33.

[II88] Hiroshi Imai und Masao Iri. „Polygonal Approximations of a Curve – Formulations and Algorithms“. In: *Computational Morphology*. Herausgegeben von G. T. Toussaint. Elsevier Science, 1988, Seiten 71–86. Zitiert auf Seite 33.

[Jen+04] Christian S. Jensen u. a. *The Infati Data*. Technischer Bericht. Database und Programming Technologies Institute at Aalborg University, 2004. Zitiert auf Seiten 3, 10, 63, 93, 94.

[JP07] Christian S. Jensen und Stardas Pakalnis. „TRAX - Real-World Tracking of Moving Objects“. In: *33rd International Conference on Very Large Data Bases*. Herausgegeben von Christoph Koch u. a. ACM, 2007, Seiten 1362–1365. Zitiert auf Seiten 55, 57.

[JS04] Christian S. Jensen und Richard T. Snodgrass. *The INFATI GPS data*. 2004. URL: `http://www.cs.aau.dk/~csj/INFATI/gpsData.zip`. Zitiert auf Seiten 63, 64, 93.

[Kay05] Aydin Kaya. „Puffer-Komponente für dynamische Positionsdaten“. Diplomarbeit. Universität Stuttgart, Fakultät Informatik, Elektrotechnik und Informationstechnik, 2005, Seite 117. Zitiert auf Seiten 3, 8, 10, 13, 23, 27, 33.

[LC88] Jia Guu Leu und Limin Chen. „Polygonal Approximation of 2-D Shapes Through Boundary Merging“. In: *Pattern Recognition Letters* 7.4 (1988), Seiten 231–238. Zitiert auf Seiten 34, 45, 47.

[McM87] Robert B. McMaster. „Automated Line Generalization“. In: *Cartographica* 24.2 (1987), Seiten 74–111. Zitiert auf Seiten 19, 33.

[MPT97] G. Manis, G. Papakonstantinou und P. Tsanakas. „Optimal piecewise linear approximation of digitized curves“. In: *13th International Conference on Digital Signal Processing Proceedings* 2 (Juli 1997), Seiten 1079–1081. Zitiert auf Seite 33.

[MTN08] Koichi Miyashita, Tsutomu Terada und Shojiro Nishio. „A Map Matching Algorithm for Car Navigation Systems that Predict User Destination“. In: *22nd International Conference on Advanced Information Networking and Applications – Workshops*. IEEE Computer Society, 2008, Seiten 1551–1556. Zitiert auf Seite 54.

[Nat00] National Geospatial-Intelligence Agency. *Department of Defense World Geodetic System 1984, Its Definition and Relationships With Local Geodetic Systems*. Technischer Bericht. National Geospatial-Intelligence Agency, 2000. Zitiert auf Seite 93.

[Nat08] National Geospatial-Intelligence Agency. *Geographic Translator 2.4.2*. National Geospatial-Intelligence Agency. Oktober 2008. URL: `http://earth-info.nga.mil/GandG/geotrans/`. Zitiert auf Seite 94.

[Nex09] Nexus. *Sonderforschungsbereich 627: Nexus, Umgebungsmodelle für Mobile Kontextbezogene Systeme*. 2009. URL: `http://nexus.informatik.uni-stuttgart.de`. Zitiert auf Seite 9.

[Ope08] OpenStreetMap Foundation. *OpenStreetMap*. Dezember 2008. URL: `http://openstreetmap.org`. Zitiert auf Seiten 64, 93.

[Pal+04] Themistoklis Palpanas u. a. „Online Amnesic Approximation of Streaming Time Series“. In: *ICDE*. IEEE Computer Society, 2004, Seiten 338–349. Zitiert auf Seiten 20, 28, 33.

[Pfo08] Dieter Pfoser. „Map-Matching“. In: *Encyclopedia of GIS*. Herausgegeben von Shashi Shekhar und Hui Xiong. Springer, 2008, Seiten 632–633. Zitiert auf Seite 53.

[PJ03] Dieter Pfoser und Christian S. Jensen. „Indexing of network constrained moving objects“. In: *Proceedings of the 11th ACM international symposium on Advances in geographic information systems*. ACM, 2003, Seiten 25–32. Zitiert auf Seite 57.

[Rot08] Kurt Rothermel. Kapitel Kontextbezogene Systeme – die Welt im Computer modelliert, Seiten 31–42. In: „Digitale Visionen“. Herausgegeben von Alexander Roßnagel, Tom Sommerlatte und Udo Winand. Springer-Verlag, 2008. Zitiert auf Seite 9.

[SWP08] Emad Soroush, Kui Wu und Jian Pei. „Fast and quality-guaranteed data streaming in resource-constrained sensor networks“. In: *Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing*. ACM, 2008, Seiten 391–400. Zitiert auf Seite 35.

[TJ07] Dalia Tiešyté und Christian S. Jensen. „Challenges in the Tracking and Prediction of Scheduled-Vehicle Journeys“. In: *PerCom Workshops*. IEEE, 2007, Seiten 407–412. Zitiert auf Seiten 55, 60.

[TJ08] Dalia Tiešyté und Christian S. Jensen. „Efficient Cost-Based Tracking of Scheduled Vehicle Journeys“. In: *Mobile Data Management*. Herausgegeben von Xiaofeng Meng u. a. IEEE, 2008, Seiten 9–16. Zitiert auf Seiten 55, 60.

[WBK00] Christopher E. White, David Bernstein und Alain L. Kornhauser. „Some map matching algorithms for personal navigation assistants“. In: *Transportation Research Part C: Emerging Technologies* 8.1-6 (2000), Seiten 91–108. Zitiert auf Seiten 56, 57, 60.

[Whi85] Ellen R. White. „Assessment of Line-Generalization Algorithms Using Characteristic Points“. In: *The American Cartographer* 12.1 (1985), Seiten 17–27. Zitiert auf Seite 33.

Alle URLs wurden zuletzt am 20.04.2009 geprüft.

Erklärung

Hiermit versichere ich, diese Arbeit selbständig verfasst und nur die angegebenen Quellen benutzt zu haben.

(Steffen Reimann)