Qualitative Repräsentation und Generalisierung von Bewegungsverläufen

Klaus Stein

Institut für Informatik
der Technischen Universität München

Qualitative Repräsentation und Generalisierung von Bewegungsverläufen

Klaus Stein

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender:		Univ.-Prof. Dr. F. Matthes
Prüfer der Dissertation:	1.	Univ.-Prof. Dr. Dr. h. c. W. Brauer,
	2.	Univ.-Prof. Chr. Freksa, Ph. D. Universität Bremen

Die Dissertation wurde am 17. 6. 2003 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 26. 11. 2003 angenommen.

Kurzzusammenfassung

Die Verarbeitung von Bewegungsverläufen ist für eine Vielzahl von Anwendungen – beispielsweise die Steuerung eines mobilen autonomen Roboters oder die automatische Generierung von Wegbeschreibungen – von Interesse.

Neben der numerischen Darstellung von Verläufen in Form von Punkt- oder Vektorsequenzen bieten sich – unter anderem zur Benutzerinteraktion, z. B. als Vorstufe der Generierung natürlichsprachlicher Wegbeschreibungen – qualitative Darstellungen an. Die qualitative Repräsentation abstrahiert dabei von exakten numerischen zugunsten symbolischer Richtungs- und Entfernungsangaben (*far north, close east, ...*), um letztlich eine Beschreibung des Bewegungsverlaufes als Abfolge von Grundformen zu erhalten (*large right-turn, small u-turn-left, ...*).

Sowohl numerische als auch qualitative Bewegungsbeschreibungen finden dabei in unterschiedlichen Referenzsystemen statt, wobei vor allem das für die Messung der numerischen bzw. Generierung der qualitativen Darstellung genutzte Referenzsystem große Auswirkungen auf die Art der Akkumulation von Meßfehlern hat.

Im ersten Teil der Arbeit werden Verfahren zur Generierung qualitativer aus numerischen Darstellungen in unterschiedlichen Referenzsystemen sowie zum Rechnen auf qualitativen Repräsentationen vorgestellt.

Bei der Betrachtung der Form einer Bewegung, z. B. zur Erzeugung einer Wegbeschreibung, interessiert häufig der Bewegungsverlauf „im Großen“ (wo ist der Serviceroboter langgefahren), nicht aber die Feinstruktur der Bewegung (Rangierbewegungen, Schlenker, etc). Im Hauptteil der Arbeit werden sowohl neu entwickelte Algorithmen zur Vereinfachung (Generalisierung) von Bewegungsspuren vorgestellt, als auch Generalisierungsalgorithmen aus der Geographie für die Bewegungsverarbeitung modifiziert und erweitert. Ein wichtiger Beitrag sind hierbei inkrementelle Algorithmen, die schon während der laufenden Messung einer Bewegung eine Generalisierung erlauben.

Die vorgestellten Algorithmen werden schließlich hinsichtlich ihrer Eigenschaften untersucht und eingeordnet.

Inhaltsverzeichnis

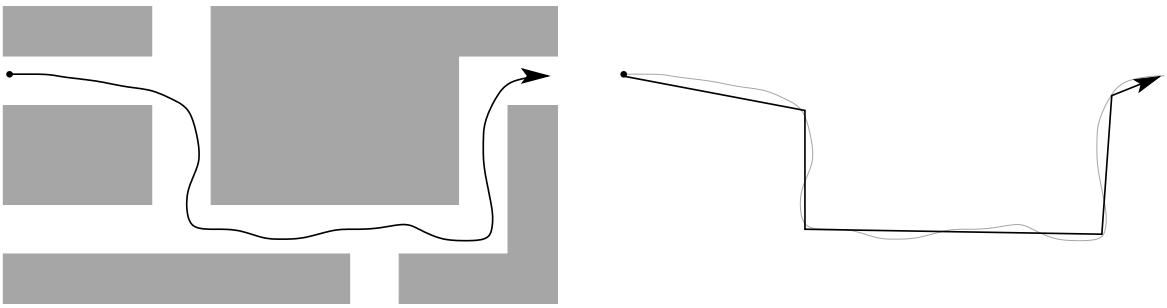
1. Einführung

Der Weg ist das Ziel.
Fernöstliche Weisheit

Die Wahrnehmung und Verarbeitung von Bewegung ist eine der grundlegenden Fähigkeiten von Mensch und Tier. Ohne sie kann weder ein Fußballtorwart einen Ball fangen, noch ein Fuchs einem Hasen hinterherjagen oder ein Autofahrer von der Arbeit nach Hause fahren.

Auch viele technische Anwendungen erfordern die Erkennung und Verarbeitung von Bewegungen in unterschiedlichen Formen. Damit ein Spaceshuttle an eine Raumstation andocken, ein mobiler Roboter anhand einer Wegbeschreibung autark durch ein Gebäude fahren oder die freundliche Stimme eines Autonavigationssystems mir sagen kann, daß ich nun rechts abbiegen muß, müssen technische Systeme in der Lage sein, Bewegungen zu erfassen und zu verarbeiten.

Abbildung 1.1 zeigt links den Verlauf der Bewegung eines mobilen Roboters durch die Gänge eines Bürogebäudes. Während der Roboter durch die Gänge fuhr, wurde seine Position laufend gemessen. Damit liegt der Verlauf der Bewegung nun als Sequenz von (vielleicht 100) Positionsangaben (Eckpunkten) vor. Rechts ist der gleiche Bewegungsverlauf vereinfacht dargestellt. Obwohl die vereinfachte Darstellung mit nur 6 Eckpunkten auskommt, beschreibt sie den Bewegungsverlauf gut genug, um zu erkennen, welche Gänge der Roboter entlanggefahren ist.



```
<medium-distance forward> <close right> <medium-distance left>
<close left> <very-close right>
```

Abbildung 1.1.: Fahrt durch die Gänge eines Bürogebäudes

Beide Darstellungen beschreiben den Bewegungsverlauf als Folge numerischer Koordinaten, eine für die Verarbeitung mittels Computer geeignete Repräsentation. Für die Interaktion mit einem menschlichen Benutzer sind numerische Angaben hingegen häufig unbefriedigend. Beschreibt ein Mensch den Verlauf einer Bewegung, so geschieht dies mittels qualitativer Angaben wie „ein Stück weit geradeaus, dann rechts, ...“. Diese Beschreibung ist nicht nur für den Menschen verständlicher, sie reduziert die Repräsentation auch auf die wesentlichen Informationen.

Daher sind neben einer numerischen Repräsentation auch qualitative Darstellungen von Bewegungsverläufen von Interesse. In Abbildung 1.1 unten ist eine qualitative Beschreibung der Bewegungsspur angegeben, die einer Bewegungsbeschreibung durch den Menschen schon näherkommt.

In dieser Arbeit wird sowohl die Erzeugung und Verarbeitung qualitativer Bewegungsrepräsentationen behandelt als auch unterschiedliche Algorithmen zur Generalisierung (Vereinfachung) von Bewegungsverläufen vorgestellt und untersucht:

- Kapitel 2 liefert eine Einführung in die Repräsentation von Bewegungsverläufen unter Einbeziehung unterschiedlicher Referenzsysteme.
- In Kapitel 3 wird eine Architektur zur qualitativen Repräsentation von Bewegungsverläufen in unterschiedlichen Granularitäten beschrieben. Dabei werden Verfahren zur Erzeugung qualitativer Beschreibungen aus numerischen Bewegungsspuren sowie eine Algebra zum Rechnen auf qualitativen Bewegungssequenzen vorgestellt.
- Kapitel 4 führt die grundlegenden Konzepte der Generalisierung von Bewegungsspuren ein und beschreibt wesentliche Eigenschaften von Generalisierungsalgorithmen.
- In den Kapiteln 5 mit 11 werden nun einzelne Generalisierungsalgorithmen beschrieben. Ein Teil der Algorithmen stammt ursprünglich aus der Geographie und wurde für die Verarbeitung von Bewegungsverläufen angepaßt, andere wurden speziell zur Generalisierung von Bewegungsverläufen neu entwickelt.
- Kapitel 12 beschreibt den Einsatz von Generalisierungsalgorithmen zur Analyse der Feinstruktur von Bewegungsverläufen.
- Kapitel 13 schließlich leistet die Zusammenstellung und Einordnung der vorgestellten Algorithmen anhand ihrer Eigenschaften.

2. Bewegung

Der Duden notiert zu *bewegen* lapidar: *Lage ändern* oder *veranlassen*. Auf die übertragene Bedeutung des Wortes werde ich im Folgenden nicht weiter eingehen. Eine Bewegung ist also eine Veränderung der Lage. In dieser einfachen Beschreibung sind zwei weitere wesentliche Begriffe versteckt: Raum und Zeit.

- Damit ich sagen kann, daß sich etwas in einer bestimmten Lage befindet, muß ein Bezugssystem vorhanden sein. Das kann zum einen die Position eines Körpers in einem Koordinatensystem sein, zum anderen aber auch seine Stellung bezüglich eines anderen Körpers.
- Weiterhin findet die Veränderung in der Zeit statt. Der Begriff Bewegung beschreibt hierbei nicht die Tatsache, daß ein Körper sich *vorher* hier und *nachher* dort befindet, sondern den Vorgang dieser Ortsveränderung selbst.

Bewegung ist der Vorgang, der Ort und Zeit zusammenbringt, oder anders: Zeit wird durch Bewegung wahrnehmbar. Unsere Zeiteinheiten sind grundlegend durch Bewegung definiert, in erster Linie durch eine spezielle Form, der periodischen Bewegung. Dies beginnt bei der Drehung der Erde um sich selbst, die die Zeit in Tage einteilt, geht über das gleichmäßige hin und her des Pendels der Standuhr, das die Sekunden angibt, bis hin zur Schwingung des Caesiumatoms, das die Grundlage der physikalischen Zeitmessung bildet.[1]

Der Bewegungsbegriff ist vielgestaltig. Der Zug, der von Lenggries nach München fährt,[2] bewegt sich, ebenso das besagte Pendel der Standuhr, der sich drehende Kreisel oder die Fahne im Wind. Die beschriebenen Vorgänge unterscheiden sich untereinander sehr stark. Der Zug ändert durch die Bewegung seinen Ort, er befindet sich zunächst in Lenggries, dann in Bad Tölz, in Holzkirchen, und schließlich in München. Das Pendel verläßt seinen Ort im Kasten der Uhr nicht, schwankt lediglich periodisch hin und her. Der Kreisel hingegen bleibt völlig auf der Stelle, er dreht sich in sich selbst, doch seine Position ändert sich nicht. Und die Fahne schließlich bleibt ebenfalls an ihrem Fahnenmast, ändert dabei lediglich ihre Form.

Diesen Vorgängen gemein ist die Tatsache, daß irgend ein Teil des Objekts, das sich bewegt, seine Lage im Raum geändert hat, daß es also einen Zeitpunkt t_1

[1] Eine Sekunde entspricht 9192631770 Schwingungen eines Caesium-133-Atoms.

[2] und in dem ich gerade sitze, während ich diese Zeilen schreibe

gab, an dem er eine andere Position hatte, als zum Zeitpunkt t_0. Die Repräsentation einer Bewegung ist also die Beschreibung der zeitlichen Abfolge von Lageänderungen im Raum. In dieser Arbeit geht es um die Verarbeitung von Bewegungsverläufen, die diese Lageänderungen repräsentieren.

2.1. Repräsentation von Raum und Zeit

> Kaum ein anderer Aspekt der physikalischen Wirklichkeit ist uns Menschen allgegenwärtiger als der räumliche, so daß wir uns sogar komplexe, nicht-räumliche Sachverhalte durch räumliche Analogien erschließen.
>
> *Daniel Hernández*[3]

Es existieren eine ganze Reihe unterschiedliche Vorstellungen von Raum und Zeit, die teilweise kulturabhängig sind. Aus mathematischer Sicht bildet die euklidische Geometrie eine solide Grundlage zur Raumdarstellung,[4] die sich somit auch in der Informatik zur Verarbeitung räumlicher Informationen anbietet. Ungeachtet dessen gibt es vielfältige Ansätze, alternative Raum- und Zeitbeschreibungen so zu formalisieren, daß sie mit Mitteln der Logik und Informatik verarbeitbar sind.

Dabei ist es für gewöhnlich nicht „der Raum“, der beschrieben wird, sondern eine räumliche Konfiguration, d. h. die Lage von Objekten zueinander oder in Bezug auf ein gegebenes Koordinatensystem.[5] In Abschnitt 2.3 werden die unterschiedlichen Referenzsysteme (in Hinsicht auf die Beschreibung von Bewegungsspuren) genauer betrachtet.

Ziel einer Modellierung und Formalisierung räumlichen (und zeitlichen) Wissens ist die Möglichkeit zur Speicherung, Verarbeitung, Wiedergabe und Kommunikation räumlicher (und zeitlicher) Informationen,[6] bei der Entwicklung der Geometrie durch die ägyptischen Landvermesser vor 4000 Jahren nicht anders als bei der Programmierung eines CAD-Systems heute. Die Anforderungen an diese Modellierung ergeben sich aus den Fragen:

- Was für räumliches und zeitliches Wissen liegt in welcher Form vor und wird wie gewonnen?

[3] [Her96]

[4] Die Effekte der allgemeinen Relativitätstheorie, die eine nichteuklidische Geometrie verlangen, treten ebenso wie die der Quantenmechanik in der Praxis nur in Sonderfällen auf.

[5] Die Frage, ob es so etwas wie einen absoluten Raum als eigene Entität (als „etwas für sich“) überhaupt gibt, oder ob Raum nur ein System von Beziehungen ist, beschäftigt die Philosophie schon seit Jahrtausenden. Einen Überblick über die Entwicklung räumlicher Vorstellungen in der Philosophie liefert z. B. [Nun95].

[6] Selbstverständlich ist ein wesentlicher Aspekt der Modellierung von Raum die Forschung über die Struktur des Raumes selbst sowie über die Art der Verarbeitung durch den Menschen. Ich werde mich hier allerdings auf die Anwendung derartiger Forschungsergebnisse beschränken.

- Wie soll dieses Wissen verarbeitet werden?
- Was ist das Ziel der Verarbeitung?
- Wer ist Empfänger dieser Informationen?

2.1.1. Unterschiedliche Repräsentationsformen

Werden räumliche und zeitliche Informationen durch exakte Messungen gewonnen, liegen sie also in numerischer Form vor, ist das Instrumentarium der elementaren Geometrie für ihre Verarbeitung geeignet und kann der Empfänger mit metrischen Daten gut umgehen, so sind diese häufig erste Wahl.

Im alltäglichen Umgang mit räumlichen Informationen liegen diese Voraussetzungen hingegen häufig nicht vor. Menschen denken intuitiv eben nicht in metrischen Systemen, und dieser Tatsache wird unterschiedlich begegnet:

- Aufgrund der großen Beschreibungs- und Verarbeitungsmächtigkeit metrischer Raum- und Zeitmodellierungen wird der Umgang mit dem metrischen System gelehrt, so sollten die meisten Schulabgänger in der Lage sein, die Breite eines Schrankes zu messen oder auszurechnen, wie spät es in zwei Stunden sein wird.

- Metrische Daten werden dem Benutzer nicht in Form von Zahlen sondern graphisch veranschaulicht angeboten, der gezeichnete Grundriß eines Hauses vermittelt dem Betrachter weit mehr als eine Zahlenkolonne, auch wenn ihr Informationsgehalt technisch gesehen identisch ist.[7] Darüberhinaus werden häufig auch Werkzeuge zur Manipulation in dieser graphischen Darstellung bereitgestellt (CAD Systeme). Dies geht bis hin zu einer Repräsentation, die der optischen Raumwahrnehmung durch den Menschen möglichst nahe kommt (dreidimensionale perspektivische Darstellung von Objekten, evtl. sogar mittels 3D-Brille (Datenhelm), durch Datenhandschuh etc. auch die Möglichkeit der Manipulation dieser Objekte durch Agieren im Raum).[8]

- Statt räumliche (und zeitliche) Informationen in einer der menschlichen *Wahrnehmung* nahekommenden Weise zu repräsentieren, bietet es sich ebenso an, den mentalen *Modellen* des Menschen nahekommende Symbolisierungen von

[7] Wird der Grundriß als Vektorgraphik gespeichert, ist er nichts anderes als eine Folge von Zahlen.

[8] Zur Darstellung von Zeiträumen ist eine der natürlichen Wahrnehmung nahekommende Repräsentation dagegen seltener. Zwar sollen in einer Simulation häufig Vorgänge gleichschnell ablaufen wie im Realen, aber dennoch käme kaum jemand auf die Idee, die Tatsache, daß es 6 Stunden dauert, von Hamburg nach München zu fahren, einem anderen dadurch zu vermitteln, daß man ihn 6 Stunden warten läßt (ganz abgesehen davon, daß diese 6 Stunden Warten ganz anders wahrgenommen würden als 6 Stunden Zugfahrt).

räumlichen Konzepten und Beziehungen zwischen Objekten zu nutzen.[9] Das zur Beschreibung räumlicher und zeitlicher Bezüge zur Verfügung stehende Vokabular ist sehr umfangreich und ausdifferenziert[10] und meist qualitativ und vergleichend.[11]

Die Nutzung durch den Menschen ist bei weitem nicht das einzige (und auch nicht unbedingt das wichtigste) Kriterium, neben numerischen auch qualitative Repräsentationen des Raumes zu verwenden.

- Metrische Informationen stehen häufig gar nicht zur Verfügung und müßten umständlich beschafft (errechnet, gemessen) werden.
- Ungenaue Angaben sollen modelliert werden.
- Die räumlichen oder zeitlichen Beziehungen zwischen Objekten sollen herausgearbeitet werden.
- Alles in allem: die exakten metrischen Angaben verdecken in manchen Fällen den Blick auf das Wesentliche.

Das Wissen, daß der Stift sich *in* der Tasche befindet, ist nicht nur *hinreichend*, um zu schließen, daß man die Tasche öffnen muß, um an den Stift zu kommen, sondern es ist darüberhinaus eine wesentlich besser handhabbare Darstellung als die Angabe der metrischen Informationen: Es müßten die Form, die räumliche Ausdehnung (unter Berücksichtigung der Tatsache, daß die Tasche innen hohl ist) und die exakte Position beider Objekte modelliert werden (wozu diese Daten erstmal beschafft werden müssen), um dann auszurechnen, daß kein Weg zwischen meiner Hand und dem Stift existiert, der nicht von der Tasche blockiert wird, um daraus dann zu folgern, daß die Tasche zu öffnen ist.

Die metrische Angabe ist unnötig genau, vor allem aber enthält sie die relevante Information nur sehr indirekt, wogegen sie in der symbolischen Beschreibung offensichtlich ist. Ein wesentlicher Vorteil ist also die Beschränkung auf die und das Herausheben der relevanten Informationen.

Dies gilt ebenso für technische Systeme. Ein bekanntes Beispiel aus der Informatik hierfür sind die Layoutmanager für graphische Benutzeroberflächen, bei denen vom Programmierer angegeben wird, ob bestimmte Bedienelemente nebeneinander, übereinander, gruppiert etc. anzuordnen sind und die Software daraus die Positionierung auf dem Bildschirm berechnet. Die Bedienelemente könnten zwar auch mittels graphischer Tools fest positioniert werden (und dies geschieht auch

[9] Menschen *sprechen* über Raum und *verstehen* einander dabei auch, womit sich hier ein Blick auf die Sprache lohnt.

[10] Man betrachte beispielsweise alleine die Untersuchung der Bedeutungsvielfalt des Verbs „drehen“ in [Hab99].

[11] Die Ausdrucksstärke dieses Vokabulars zeigt sich auch in der Vielzahl übertragener Bedeutungen.

zunehmend), die Nutzung eines Layoutmanagers erlaubt aber dem Benutzer, die Größe des Anwendungsfensters zu ändern, woraufhin die Bedienelemente neu angeordnet werden müssen, um sichtbar zu bleiben. Natürlich *berechnet* der Layoutmanager abhängig von gegebenen Constraints (die aktuelle Fenstergröße, die Größe der einzelnen Bedienelemente, die Beziehungen, in denen sie zueinander stehen etc.) exakte Positionen, die Flexibilität der Darstellung wird aber gerade dadurch erreicht, daß der Nutzer seine Vorgaben in Form qualitativer Angaben gemacht hat.

Diese Modellierungen besitzen ob ihrer Prägnanz also eine große Ausdrucksmächtigkeit, was sie für alle spatiotemporalen Anwendungen interessant macht, in denen die metrische Information nicht das wesentliche Element darstellt, sondern in denen Eigenschaften von Raum (und Zeit) modelliert und verarbeitet werden sollen.

2.1.2. Qualitative Modelle

Quantitative (numerische) und qualitative Repräsentationen des Raumes (und der Zeit) beschreiben Größen wie Entfernung, Ausdehnung, Lage, Fläche, und diese Beschreibung findet durch Vergleich statt. Sie unterscheiden sich dabei in folgenden Punkten:

- Während im quantitativen Fall ein genormtes Maß (z. B. der Urmeter in Paris[12]) und seine Untereinheiten als Vergleichsmaßstab dienen (eine Stange ist 3 Meter lang, wenn das Urmeter drei mal reinpaßt), sind es im qualitativen kontextabhängige Vergleichsgrößen, die dabei auch ständig wechseln können („Auf einer Fläche von der Größe eines Fußballfeldes wirkt das kaum mannshohe Podest eher mickrig“)

- Die quantitative Darstellung nutzt beliebige (auch gebrochenrationale) Vielfache ihrer Vergleichsgröße (ihrer Grundeinheit) und baut sich daraus einen Maßstab.[13] Ein Tisch ist 1.32 m lang, ein Seil 20 m und eine Aschenbahn[14] 400 m. Die qualitative Repräsentation (und insbesondere das räumliche mentale Modell des Menschen) wählt hingegen seine Vergleichsobjekte möglichst so, daß sie ohne Vorfaktoren auskommen. So sind die Aussagen, daß Anna näher bei Bert als bei Carl steht, ein Elefant größer als ein Auto oder eine CD genauso groß ist wie eine DVD sicher qualitativ. Die Aussagen, daß der Tisch sechsmal so lang ist wie ein Frühstücksbrettchen oder der Schrank 9.2 Mal

[12] der mittlerweile allerdings nicht mehr die Norm für die SI Einheit Meter darstellt. Die SI-Einheit Meter ist definiert als die Strecke, die das Licht im Vakuum in 299792458^{-1} s zurücklegt.

[13] Zur Erhöhung der praktischen Handhabbarkeit werden noch passende Untereinheiten eingeführt (Zentimeter, Millimeter, Kilometer, ...)

[14] auf der Innenbahn

so hoch wie eine Blumenvase sind sicher quantitativ, wenn sie sich auch nicht auf das genormte metrische System beziehen. Die Grenze ist hier fließend: ein Buch ist doppelt so dick wie ein anderes, die neue Kanne halb so groß wie die alte.[15]

- Quantitative Aussagen implizieren eine gewisse Exaktheit. Ist der Tisch 1.32 m lang, so ist dies eine auf den Zentimeter genaue Angabe, die maximal eine Ungenauigkeit im Millimeterbereich toleriert. Qualitative Angaben sind nicht per se ungenauer, die Aussage: „beide Tische sind gleich hoch“ beinhaltet sogar eine sehr hohe Genauigkeit, wenn es um die Frage geht, ob man sie zu einem großen Tisch zusammenstellen kann. Den meisten qualitativen Angaben wohnt aber von vornherein eine gewisse Ungenauigkeit inne. Wenn Peter etwas größer ist als Paul und Kuno viel größer ist als Paul, dann enthalten diese Aussagen zwar genug Information, um zu folgern, daß Kuno größer ist als Peter, die genauen Größenverhältnisse bleiben allerdings unbekannt, es ist nicht mal klar, ob Kuno etwas oder viel größer ist als Peter. Dies rührt häufig auch daher, daß das Vergleichsobjekt selbst unbestimmt bleibt: Wie dick ein armdicker Ast ist, hängt vom Arm ab, den sich der Sprecher dabei vorstellt, und für ein Kind ist knietiefer Schnee etwas anderes als für einen Erwachsenen.[16]

Qualitative Modellierungen beschreiben vor allem Relationen zwischen Objekten, angefangen bei Intervallkalkülen,[17] die die zeitliche Relation von Ereignissen („vor“, „während“, „nach“, …) beschreiben, über topologischen Modellierungen wie dem bekannten RCC-8-Calculus,[18] der die Lage zweier räumlicher Regionen zueinander beschreibt: sie „berühren sich“, „überlappen“, „liegen ineinander“ etc. bis zur Einbeziehung von Richtungs- und Entfernungsangaben durch Einführung von Referenzsystemen (siehe auch Kapitel 2.3 und 3.3). Das Referenzsystem kann dabei von den beschriebenen Objekten selbst induziert werden, wie im Fall des Doppelkreuzkalküls von Freksa,[19] in dem die Lage eines Punktes bezüglich zweier Referenzpunkte beschrieben wird, oder extern vorgegeben werden, wie dies mit der Einführung „qualitativer“ Distanzen und Richtungen[20] geschieht. Eine Übersicht über das Gebiet des qualitativen räumlichen Schießens (QRS) findet sich in [Coh97], eine Zusammenstellung zur qualitativen Repräsentation von Raum und Zeit leistet [Sto97], einen knappen Überblick liefert auch [Mus00, S. 11–18].

[15] Interessant ist hierzu auch ein Blick auf die Sprache: der Lastzug ist „so lang wie 7 Autos“ und nicht „siebenmal so lang wie ein Auto“.
[16] „Ein Mann wie ein Baum – sie nannten ihn Bonsai“
[17] [All83, Fre92a], mit Anwendung auf räumliche Konfigurationen: [Fre91, Her91]
[18] [Cla81, RCC92]
[19] [Fre92b]
[20] [Her94, CDFH97, GE]

2.2. Repräsentation von Bewegungen

Ein Bewegungsverlauf ist wie erwähnt die Veränderung der Position eines Objektes[21] im Raum. Befindet sich ein Körper K zum Zeitpunkt t_0 am Ort P und zum Zeitpunkt $t_1 > t_0$ am Ort $Q \neq P$, so hat K sich von P nach Q bewegt. Die Ortsveränderung von P nach Q ist das Ergebnis der Bewegung. Die eigentliche Bewegung ist der Weg von K in der Zeit $[t_0, t_1]$. Für einen punktförmigen Körper beschreibt die Abbildung

$$\begin{aligned} b : [t_0, t_1] &\rightarrow \mathcal{D} \\ b(t) &= v_t \end{aligned}$$

mit $b(t_0) = P$ und $b(t_1) = Q$ den Verlauf einer Bewegung im Raum $\mathcal{D}$ (im zweidimensionalen Fall gilt $\mathcal{D} = \mathcal{R}^2$, im dreidimensionalen $\mathcal{D} = \mathcal{R}^3$): Jedem Zeitpunkt t wird ein Punkt v_t im Raum zugeordnet.[22] Neben dieser aus der Kinematik stammenden Repräsentation eines Bewegungsverlaufes als parametrisierte Kurve existieren eine ganze Reihe weiterer Darstellungen numerischer und qualitativer Art.

2.2.1. Numerische Repräsentation

In der Praxis liegt eine Bewegung nicht als kontinuierlicher Verlauf sondern als diskrete Punktfolge vor. Eine Bewegung wird mit einem bestimmten Zeitraster Δt gemessen, damit ist die Position v des betrachteten Körpers K zu den Zeitpunkten $t_0, t_0 + \Delta t, t_0 + 2\Delta t, \ldots, t_1$ bekannt. Diese gerasterte Messung von Bewegungsverläufen findet dabei nicht nur in technischen Systemen statt, sondern nach heutigem Wissen auch in der Bewegungswahrnehmung des Menschen (siehe auch Kapitel 11).

Damit läßt sich ein Bewegungsverlauf als Folge von Positionen $v_{t_0} v_{t_0+\Delta t} \ldots v_{t_1}$ darstellen. Der Einfachheit halber werden die einzelnen Positionen einfach durchnummeriert, wir bekommen den Polygonzug $p = v_1 v_2 \ldots v_n$, der eine numerische Beschreibung eines Bewegungsverlaufes darstellt, wobei t_0 und Δt als Attribute von p angegeben werden. Nur die Eckpunkte v_i des Polygonzuges repräsentieren dabei Informationen über den Bewegungsverlauf, da ja nur sie gemessen wurden.

Ist die Abtastrate zur Messung des Bewegungsverlaufes hoch genug gewählt (Δt hinreichend klein), so weicht dieser zwischen zwei benachbarten Meßpunkten v_i und v_{i+1} nur wenig von der Strecke $[v_i v_{i+1}]$ ab, so daß der Polygonzug die Form des Bewegungsverlaufes gut wiedergibt. Wie hoch die Abtastrate gewählt werden

[21] oder eines Teils eines Objektes

[22] Die Abbildung ist nicht zwingend umkehrbar, so kann ein Punkt im Raum 2 oder mehr Zeitpunkten zugeordnet sein, wenn beispielsweise eine Schleife gefahren wurde. Sie beschreibt die Bewegung eines Punktes, die räumliche Ausdehnung des bewegten Objektes bleibt unberücksichtigt.

muß, hängt dabei vor allem von den Eigenschaften des betrachteten Körpers ab, d. h. wie schnell dieser beschleunigen, abbremsen und Richtungsänderungen vollziehen kann.

Statt den Bewegungsverlauf als Folge von Punkten anzugeben, läßt er sich auch als Folge von Orts*veränderungen* beschreiben: mit $w_i = v_{i+1} - v_i$ beschreibt die Sequenz $w_1 w_2 \ldots w_{n-1}$ die gleiche Bewegung wie der Polygonzug p oben. Auf die Unterschiede zwischen beiden Darstellungen gehe ich in Abschnitt 2.3 näher ein, beide beschreiben die Form eines Bewegungsverlaufes und werden im Folgenden zur numerischen Bewegungsbeschreibung genutzt.

2.2.2. Qualitative Repräsentation

Wie schon allgemein für die Beschreibung räumlicher Konfigurationen (siehe Abschnitt 2.1) ist auch für die Beschreibung von Bewegungsverläufen für eine Reihe von Aufgaben eine qualitative Repräsentation besser geeignet als eine numerische.[23] Ein Sportreporter käme wohl kaum auf die Idee, die Kür eines Eisläufers durch die Angabe numerischer Koordinaten zu beschreiben und selbst ein Mathematiker wird den Weg zu seinem Büro in Begriffen wie „dann rechts, den Gang entlang und die zweite Tür links“ und nicht numerisch angeben. Die in Abschnitt 2.1 genannten Vorteile einer qualitativen Darstellung gelten auch hier: sie hat keine Probleme mit ungenauen Angaben (die wenigsten Menschen – eingeschlossen Mathematiker – kennen die exakten Koordinaten und Abmessungen der Gänge zu ihrem Büro) und sie ist vielfach sehr effizient (die Angabe „zweite Tür links“ liefert genau die benötigte Information).[24]

Die hier suggerierte starke Trennung zwischen qualitativer und metrischer Darstellung des Raumes existiert allerdings in der Praxis nicht. Selbstverständlich werden Wegbeschreibungen wie „über die Ampel und dann nach *hundert Metern* links ab“ oder Angaben wie „eine *zwanzig Meter* lange Schlange vor der Kinokasse“ im Alltag benutzt und verstanden. Allerdings werden diese metrischen Angaben kontextabhängig interpretiert. Wenn die Abzweigung nach links erst nach 120 Metern oder sogar erst nach 200 Metern kommt, wird die Wegbeschreibung dennoch verstanden, ohne daß dies zu Problemen führen würde. Behaupte ich hingegen: „Ich laufe *hundert Meter* in zehn Sekunden“, so glaubt mir das zum einen natürlich niemand, zum anderen ist hier aus dem Kontext klar, daß die hundert Meter ein ex-

[23] Zur qualitativen Modellierung von Bewegung existieren eine ganze Reihe von Ansätzen. Für eine qualitative Theorie der Bewegung siehe beispielsweise [Gal95, Gal97, Gal93] und [Mul98b, Mul98a], eine qualitative Kinematik liefert [For83]. Eine Zusammenstellung dieser Ansätze findet sich in [Mus00, S. 18–26], ein Überblick über die psychologischen Aspekte in [EMS+98] (siehe aber auch [Joh73, Joh75, Joh76]).

[24] Dies steht nicht im Widerspruch dazu, daß beispielsweise bei einer Wegbeschreibung häufig redundante Angaben gemacht werden, da diese zur Absicherung dienen, falls der Empfänger der Wegbeschreibung an irgendeiner Stelle verläuft.

aktes Maß darstellen. In diesen qualitativen Beschreibungen werden also metrische Angaben gemacht, diese aber kontextabhängig interpretiert und nicht automatisch als absolute Größen im Sinn der geometrischen Darstellung des Raumes angesehen.

Die Beschreibung von Bewegungsverläufen kann dabei in unterschiedlicher Weise erfolgen:

- Eingebunden in eine bestimmte Umgebung, d. h. als Sequenz durchlaufener Regionen: von Region A durch Region B nach Region C usw. („aus der Küche durchs Wohnzimmer rechts am Couchtisch vorbei über die Veranda die Treppe hinunter in den Garten“).
- Durch die *Form* der Bewegung selbst, d. h. als Sequenz von Formelementen wie Strecke, Bogen, Schleife usw. („Auf ein kurzes Stück gerader Fahrt folgte ein langgezogener Bogen nach rechts und nach einer scharfen Linksdrehung hielt er an“).

Der Mensch nutzt beide Möglichkeiten gleichermaßen und wechselt auch in der Beschreibung einer Bewegung zwischen beiden Darstellungen.

Die Beschreibung eines Bewegungsverlaufes als Folge durchfahrener Regionen (erster Fall) wird unter anderem zur Wegplanung für mobile Roboter genutzt: Der Raum wird in (konvexe) Regionen aufgeteilt und der Weg, den der Roboter fahren soll, dann als Sequenz einander benachbarter Regionen angegeben.[25]

In dieser Arbeit (Kapitel 3) werden dagegen Bewegungsverläufe betrachtet, die als Sequenz von Richtungsänderungen oder Formelementen gegeben sind (zweiter Fall). Während eine regionenbasierte Beschreibung eines Bewegungsverlaufes rein durch die Umgebung definiert wird, durch die die Bewegung verläuft, ist eine formenbasierte Beschreibung zunächst umgebungsunabhängig.

Die oben angegebene regionenbasierte Wegbeschreibung „aus der Küche durchs Wohnzimmer rechts am Couchtisch vorbei über die Veranda die Treppe hinunter in den Garten“ liefert mir, wenn ich das Haus nicht kenne, zu dem sie gehört, keine Informationen über die Form des Bewegungsverlaufes, sie ist aber völlig ausreichend, um von der Küche in den Garten zu gelangen. Die formenbasierte Beschreibung hingegen veranschaulicht die Form des Bewegungsverlaufes, ist aber als Wegbeschreibung ungeeignet, da Aussagen wie „lange geradeaus“ einfach zu ungenau sind.

Dies ändert sich bei einer Einbettung dieser Beschreibung in eine konkrete Umgebung, da eine Bewegung durch die Gänge eines Bürogebäudes relativ wenig Freiheitsgrade besitzt. „Lange geradeaus“ ist dann einfach durch die Länge des Ganges beschränkt. Reichert man diese Beschreibung nun noch durch Landmarken (feste Punkte der Umgebung) an: „lange geradeaus und nach der Säule links“, eignet sie sich ebenfalls als Wegbeschreibung.

[25] Zur regionenbasierten Wegplanung für Roboter siehe z. B. [FMB00, HHM99].

In den meisten Fällen besteht eine Wegbeschreibung aus einer Kombination obiger Elemente, wobei auch die Form des Bewegungsverlaufes selbst wieder als Landmarke dient. (beispielsweise ein gebogener Gang).[26]

2.2.3. Dynamik des Bewegungsverlaufes

In obigen Beispielen wird eigentlich keine *Bewegung* beschrieben, sondern der zurückgelegte *Weg*, also die *Spur* der Bewegung. Die Dynamik fehlt dabei fast völlig, lediglich die zeitliche Reihenfolge des Bewegungsverlaufes wird wiedergegeben. Geschwindigkeit und Beschleunigung, also die *Dynamik* der Bewegung, müssen zusätzlich durch andere Begriffe („schnell", „langsam", „flott", „Gas geben", „rennen", „schlendern") ausgedrückt werden. Die Beschreibung des Weges liefert den räumlichen, die Beschreibung der Dynamik den zeitlichen Ablauf einer Bewegung.

In der numerischen Repräsentation eines Bewegungsverlaufes als Punkt- oder Vektorfolge ist die Dynamik implizit enthalten, wenn die einzelnen Punkte mit einem festen Zeitraster Δt gemessen wurden (bzw. die einzelnen Vektoren jeweils die Positionsänderung in einem Zeitintervall der Länge Δt beschreiben).

2.3. Referenzsysteme

Abbildung 2.1 zeigt unterschiedliche Repräsentationen einer als gerichteter Polygonzug vorliegenden Bewegung. Die Repräsentation I (links oben) beschreibt sie als Folge von kartesischen Koordinaten, II zeigt sie als Vektorfolge. In Bild III wird die Richtung jedes Vektors[27] relativ zum Vorgänger angegeben und in Bild IV wird auch die Länge jedes Vektors relativ zu seinem Vorgänger angegeben. Wir bekommen die Sequenzen:

I: $\langle 6,4\rangle\ \langle 4,20\rangle\ \langle 8,26\rangle\ \langle 12,29\rangle\ \langle 20,28\rangle\ \langle 32,12\rangle$
II: $\langle 16.1\,\mathrm{m}, 352.9^\circ\rangle\ \langle 7.2\,\mathrm{m}, 33.7^\circ\rangle\ \langle 5\,\mathrm{m}, 53.1^\circ\rangle\ \langle 8.1\,\mathrm{m}, 97.1^\circ\rangle\ \langle 20\,\mathrm{m}, 143.1^\circ\rangle$
III: $\langle 16.1\,\mathrm{m}, [352.9^\circ]\rangle\ \langle 7.2\,\mathrm{m}, +40.8^\circ\rangle\ \langle 5\,\mathrm{m}, +19.4^\circ\rangle\ \langle 8.1\,\mathrm{m}, +44^\circ\rangle\ \langle 20\,\mathrm{m}, +46^\circ\rangle$
IV: $\langle [16.1\,\mathrm{m}], [352.9^\circ]\rangle\ \langle -8.9\,\mathrm{m}, +40.8^\circ\rangle\ \langle -2.2\,\mathrm{m}, +19.4^\circ\rangle\ \langle +3.1\,\mathrm{m}, +44^\circ\rangle\ \langle +11.9\,\mathrm{m}, +46^\circ\rangle$

2.3.1. Alles das Gleiche?

Mathematisch gesehen besteht der einzige Unterschied zwischen I und II in der Kenntnis des Startpunktes: Repräsentation II ist aus I einfach berechenbar, ist der Startpunkt $\langle 6,4\rangle$ bekannt, so kann umgekehrt auch I aus II berechnet werden. Repräsentation III kann ebenfalls leicht aus II berechnet werden, ist die (absolute)

[26] Eine Untersuchung zu typischen Elementen von Wegbeschreibungen (Studenten beschreiben eine Route auf einem Campus) liefert [TL99].

[27] abgesehen vom ersten

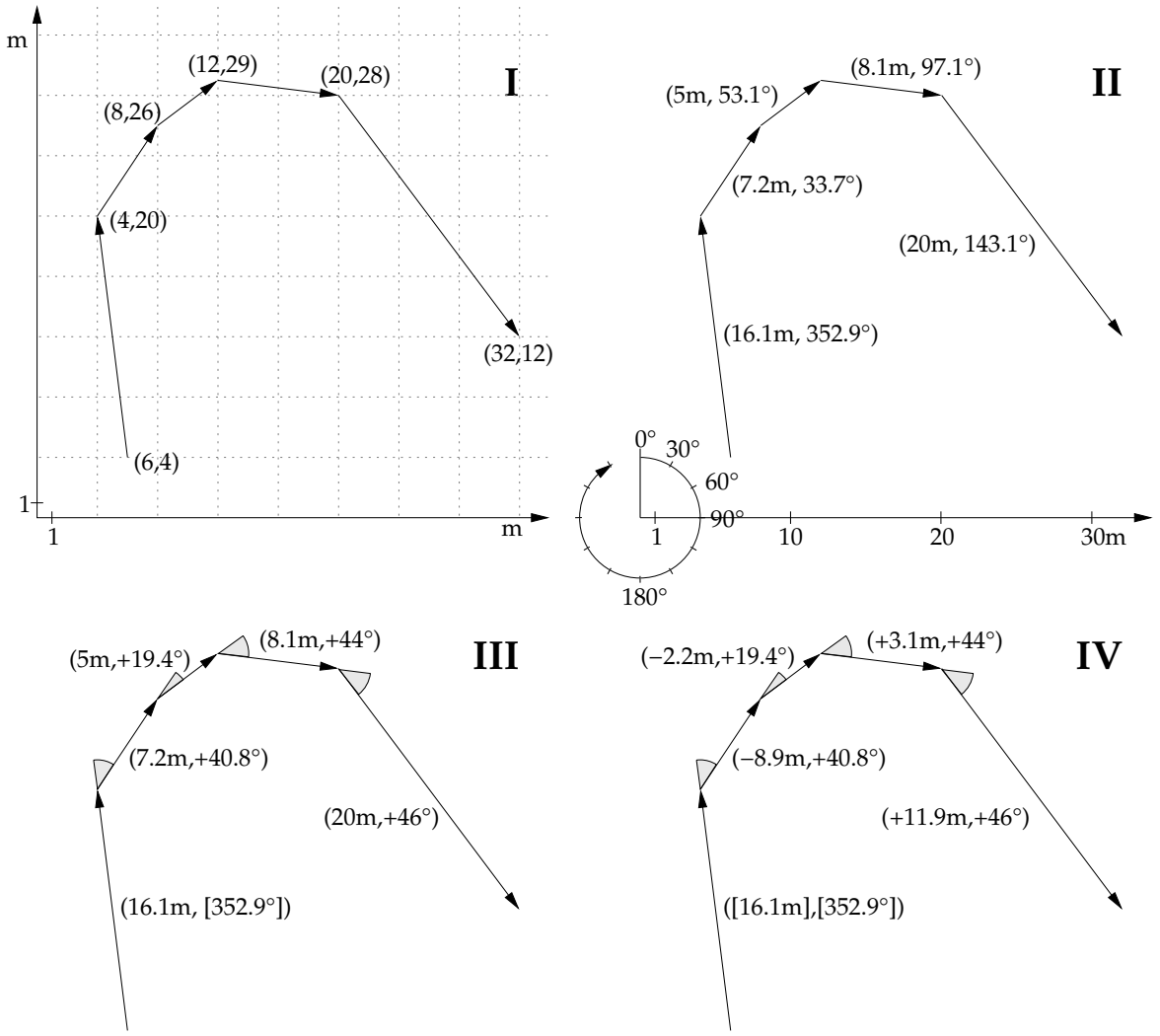


Abbildung 2.1.: Unterschiedliche Darstellungen einer Bewegungsspur

Richtung des ersten Vektors bekannt (in der Graphik in eckigen Klammern angegeben), so läßt auch hier wieder II aus III berechnen.

Die strukturelle Änderung besteht hier jeweils darin, daß zunehmend von Ortsinformation abstrahiert wird und eine Reduktion auf die Form der Bewegung stattfindet. Während in Bild I zu jedem Zeitpunkt die Ortsinformation direkt vorliegt, ist die Trajektorie in Bild II ortsunabhängig bezüglich Translation, nicht aber bezüglich Rotation. Die Trajektorie in Bild III dagegen ist in ihrer Form weiterhin exakt beschrieben, macht aber weder über Ort noch über Richtung irgendwelche Angaben. Im Gegensatz zu Abbildung II sind die Winkelangaben im Grunde nicht mehr einem einzelnen Vektor zuzuordnen sondern geben die Drehung zwischen

zwei Vektoren an. Damit bietet sich hier eine andere Darstellung an:

$$\text{III:}\quad 16.1\,\text{m}\ (+40.8^\circ)\ 7.2\,\text{m}\ (+19.4^\circ)\ 5\,\text{m}\ (+44^\circ)\ 8.1\,\text{m}\ (+46^\circ)\ 20\,\text{m}.$$

Repräsentation IV hingegen scheint nicht befriedigend, obwohl sie bezüglich der Entfernungsangaben in gleicher Weise konstruiert wurde wie III bezüglich der Richtungen. Darstellung III leistet den Formerhalt bei gleichzeitiger Richtungsunabhängigkeit, wogegen Darstellung IV nichts Gleichwertiges (eine „Entfernungsunabhängigkeit“ oder „Größenunabhängigkeit“) besitzt. Darüberhinaus ist die Angabe der Länge des ersten Vektors unbefriedigend, sie paßt nicht ins Schema, kann aber auch nicht einfach weggelassen werden.

In III benötigen sowohl Richtungs- als auch Entfernungsangaben zwar kein externes Koordinatensystem zur Darstellung, sie nutzen aber extern vorgegebene Maße (Grad und Meter). Gibt man nun die Längen der einzelnen Vektoren relativ zueinander an (die Länge des 2. Vektors ist 0.447 Mal die Länge des ersten), so bekommt man eine größenunabhängige (skalierungsinvariante) Darstellung.[28]

Leider ist diese Darstellung noch nicht perfekt, sie macht Probleme bei der Modellierung von Stillstand. Beträgt die Länge eines Vektors 0, kann die seines Nachfolgers nicht mehr als Vielfaches davon angegeben werden. Gibt man hingegen die Länge jedes Vektors nicht in Bezug auf seinen Vorgänger sondern in Bezug auf die Gesamtlänge an, so erhält man ebenfalls eine skalierungsunabhängige Darstellung:

$$\text{V:}\quad 0.285\ (+40.8^\circ)\ 0.128\ (+19.4^\circ)\ 0.089\ (+44^\circ)\ 0.144\ (+46^\circ)\ 0.355.$$

Auch diese Darstellung ist letztlich nicht zufriedenstellend. Sie beschreibt zwar einen gegebenen Polygonzug rotations-, translations- und skalierungsinvariant, allerdings wird durch diesen Polygonzug ja eine Bewegung beschrieben. Da nun aber alle Entfernungsangaben bezüglich der Gesamtlänge erfolgen, führt das Anhängen eines neuen Vektors (das beschriebene Objekt bewegt sich weiter) zur Neuberechnung sämtlicher Entfernungsangaben.

Aus diesem Grund und da in der Praxis die Größe einer Bewegung für den Vergleich zweier Bewegungsspuren relevant ist, wogegen Ort und Richtung nicht immer interessieren, beschränke ich mich im Folgenden auf Darstellung I, II und III.

2.3.2. Größere Unterschiede

Wie oben erwähnt besteht der Unterschied zwischen I und II mathematisch gesehen in der Kenntnis des Startpunktes und zwischen II und III in der Kenntnis der Bewegungsrichtung. Liefert man diese Informationen hinzu, so sind alle Darstellungen ineinander umwandelbar. In der perfekten Welt der Mathematik gilt dies auch, in

[28] Es ist ebenso möglich, die Winkel relativ zum Vorgängerwinkel anzugeben, allerdings ohne direkten praktischen Nutzen.

der Praxis jedoch nicht. Praktisch ist ausschließlich die Transformation I $\Rightarrow$ II $\Rightarrow$ III gestattet.

Dies liegt an der Art und Weise, *wie* eine Bewegung *gemessen* wird:

- Die Messung einer Mausbewegung auf dem Bildschirm erfolgt in einer Art kartesischem Koordinatensystem und ist auf einen Pixel genau. Ein Gefährt, daß mit GPS ausgestattet ist (beispielsweise ein modernes Segelschiff), bestimmt seine Position ebenfalls bezüglich eines externen Koordinatensystems[29] und macht dabei bei jeder Messung einen Meßfehler von einigen (oder auch mehr) Metern. Beide Bewegungen liefern Meßdaten in Form von I.

- Ein autonomer Roboter mit eingebautem Kompaß oder ein Pfadfinder, der sich am Polarstern orientiert kennen beide ihre Bewegungsrichtung auf einige Grad genau, müssen die in eine Richtung zurückgelegte Entfernung hingegen direkt messen oder schätzen, beispielsweise durch Zählen von Radumdrehungen, von Schritten oder Messen der Zeit, die in eine Richtung gelaufen wurde. Dies liefert Meßdaten in Form von II.

- Ist dagegen auch kein Kompaß vorhanden, müssen sowohl Entfernung als auch Richtung relativ bestimmt werden, der Serviceroboter mißt Lenkeinschlag und Radumdrehungen und liefert Meßdaten in Form von III.

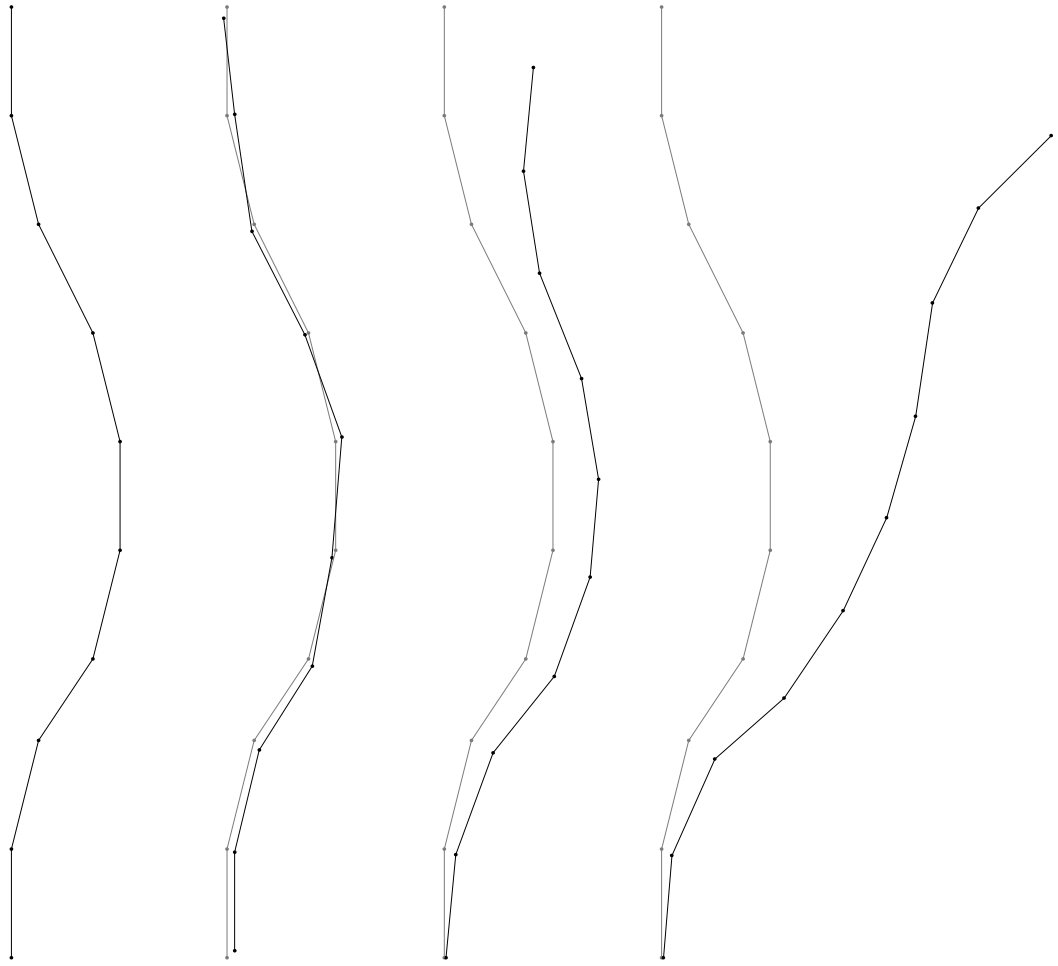
Der wesentliche Punkt, in dem sich die drei Messungen unterscheiden, ist die Akkumulation von Fehlern (Abb. 2.2).

- Im 1. Fall kann zwar eine gewisse Meßungenauigkeit bestehen mag, diese bleibt jedoch lokal begrenzt. Egal, wie lange ich fahre, ich weiß immer bis auf eben diese Meßungenauigkeit, wo ich mich befinde.

- Im 2. Fall beginnen sich Fehler zu akkumulieren, und zwar sowohl Entfernungs- als auch Winkelfehler. Fehler in der Entfernungsmessung addieren sich direkt auf. Wird die Entfernung in jedem Schritt um 10 cm zu kurz gemessen, so sind das nach 10 Schritten schon 1 m, nach 100 Schritten schon 10 m. Die Winkelfehler addieren sich hingegen nicht direkt auf, egal wie lange die Bewegung andauert, die Richtung ist bis auf die Meßungenauigkeit bekannt. Allerdings trägt der Meßfehler bei der Winkelmessung dazu bei, daß die Ortsinformation leicht verfälscht wird. Im Unterschied zum ersten Fall wird die Ortsinformation immer ungenauer, je länger die Bewegung andauert.

- Im 3. Fall schließlich akkumuliert auch der Fehler in der Richtungsbestimmung, d. h. bei einem Meßfehler von $+5^\circ$ ergibt dies nach nur 10 Schritten

[29] Dieses ist bei Bewegungen auf der Erdoberfläche zwar nicht kartesisch, was aber in diesem Zusammenhang irrelevant ist.



Ganz links die Originalbewegung, daneben eine Messung mit lokalem Ortsfehler, an dritter Stelle Messung mit Kompaß (Meßfehler von 5° im Uhrzeigersinn sowie einer Verkürzung in der Entfernungsmessung) und ganz rechts eine rein relative Messung (Meßfehler ebenfalls 5°, diesmal allerdings akkumulierend, und identischer Entfernungsfehler wie bei Kompaß).

Abbildung 2.2.: Unterschiedliche Auswirkungen von Meßfehlern

eine mögliche Abweichung von 45°, die hier auftretenden Verzerrungen können sehr groß werden.

Die Messung wird also zunehmend ungenauer. Der entscheidende Sprung findet dabei von Fall 2 auf Fall 3 statt. Fehlt ein Kompaß, so kann durch die Akkumulation der Winkelfehler eine massive Verformung der Bewegungsspur auftreten, gerade Bewegungen werden zu Kurven usw. Der Unterschied zwischen Fall 1 und Fall 2 ist dagegen zwar vorhanden und auch sichtbar, wirkt sich aber weit weniger stark aus.

Im obigen Beispiel enthält die Winkelbestimmung einen systematischen Fehler von $+5°$, d. h. der Meßfehler wirkt sich immer in gleicher Richtung aus. Ist der Fehler gleichmäßiger verteilt (wird also einmal zur Linken, einmal zur Rechten falsch gemessen), sind die Auswirkungen geringer. Allerdings kommen bei der Messung von Bewegungsspuren solche systematischen Fehler gar nicht so selten vor.

Wie wichtig ein Kompaß sein kann, zeigt aber nicht nur das hier vorgestellte Beispiel:

- Die Natur setzt unter anderem auf absolute Richtungsangaben, wenn die Form einer Bewegungsspur zur Navigation genutzt werden soll, wie das Beispiel der Wüstenameise *Cataglyphis* (siehe Kasten 1) zeigt.
- Fast jeder kennt die Geschichten von einsamen Wanderern in der Wüste, die mangels Kompaß immer im Kreis gelaufen sind.

Die Messungen in unterschiedlichen Referenzsystemen führen also zu sehr unterschiedlichen Ergebnissen. Wichtig ist hierbei, wie eine Bewegungsspur *gemessen* wurde, nicht, wie sie dargestellt wird. Wurde eine Bewegung durch die Bestimmung einer Folge von Positionen gemessen (I), so kann sie in jeder der vorgestellten Repräsentationen dargestellt werden und alle diese Darstellungen können problemlos ineinander überführt werden. Wurden dagegen sowohl Richtung als auch Entfernung ausschließlich relativ gemessen und liegen die Bewegungsdaten somit in Form III vor, so wäre es falsch, sie in eine Koordinatendarstellung umzuwandeln und in dieser Form weiterzuverarbeiten, *ohne* dabei anzugeben, daß sie nicht in dieser gemessen wurde, da diese Darstellung das Vorhandensein von Ortsinformation suggeriert.

Das heißt nicht, daß diese Umwandlung nicht möglich ist, die kartesische Darstellung ist auch häufig Mittel der Wahl, um auf Polygonzügen zu rechnen. Durch die *Repräsentation* einer Bewegung in einer bestimmten Darstellung darf nur nicht die Information verloren gehen, wie sie *gemessen* wurde.

2.3.3. Unterschiedliche Referenzrahmen

Die Notwendigkeit von Referenzsystemen zur Repräsentation von Ortsinformation beschränkt sich nicht auf numerische Beschreibungen. Jede Positionsbeschreibung

Kasten 1: Die Wüstenameise *Cataglyphis*.

Die Natur betreibt einen erheblichen Aufwand, um an absolute Richtungsinformationen zu kommen. Die Wüstenameise *Cataglyphis* entfernt sich auf der Suche nach Nahrung über mehr als 100 m von ihrem Nest, schafft es aber jederzeit, auf direktem Wege zu ihm zurückzukehren. Für die Navigation mittels lokaler Landmarken müßte sie für den Rückweg den gelaufenen Pfad rückwärts durchlaufen, sie läuft aber geradlinig durch für sie unbekanntes Gelände zu ihrem Nest zurück.

Sie betreibt hierzu path integration: durch ständiges Aufsummieren jeder ihrer Bewegungen weiß sie immer, in welcher Richtung und Entfernung von ihrer aktuellen Position sich ihr Nest befindet (ihr „homing vector“). Hierfür ist es wichtig, daß sie ständig über ihre aktuelle Richtung informiert ist. Rüdiger Wehner weist in [Weh99] nach, daß sie sich dazu einer besonderen Art von Kompaß bedient: der Polarisation des Lichts. Der Himmel über der Wüste polarisiert das Sonnenlicht dergestalt, daß ein Blick auf einen beliebigen Ausschnitt des Himmels die exakte Winkelabweichung zur Sonne liefert. Der Aufwand für diesen Kompaß ist enorm: *Cataglyphis* besitzt auf der Retina Rezeptoren zur Bestimmung der Polarisation ultravioletten Lichts. Damit ist sie in der Lage, ihren aktuellen Winkel zur Sonne zu bestimmen. Allerdings verändert die Sonne während die ganze Zeit ihre Position, so daß hier ständig eine Korrekturrechnung (eine Neukalibrierung) stattfinden muß.[a]

Um nun von einem beliebigen Ort zu ihrem Nest zurückzukehren, läuft sie zunächst einfach ihren homing vector ab und beginnt dann eine lokale Suche, wobei sie sich auch lokaler Landmarken bedient.

Obwohl die Ameise ihre Position ausschließlich aufgrund ihrer Eigenbewegung bestimmt, benötigt sie diesen externen Kompaß. Die Abweichung durch Fehler bei der Entfernungsschätzung sind klein genug, um das Nest direkt oder mittels lokaler Suche zu finden, würden sich dagegen Winkelfehler aufsummieren, wäre die Abweichung zu groß, um das Nest noch finden zu können.

[a] Diese Korrektur ist alles andere als trivial, da sich ja nicht nur die Himmelsrichtung, sondern auch die Höhe über dem Horizont ändert, was das Bild der Polarisierung auf der Retina beeinflußt.

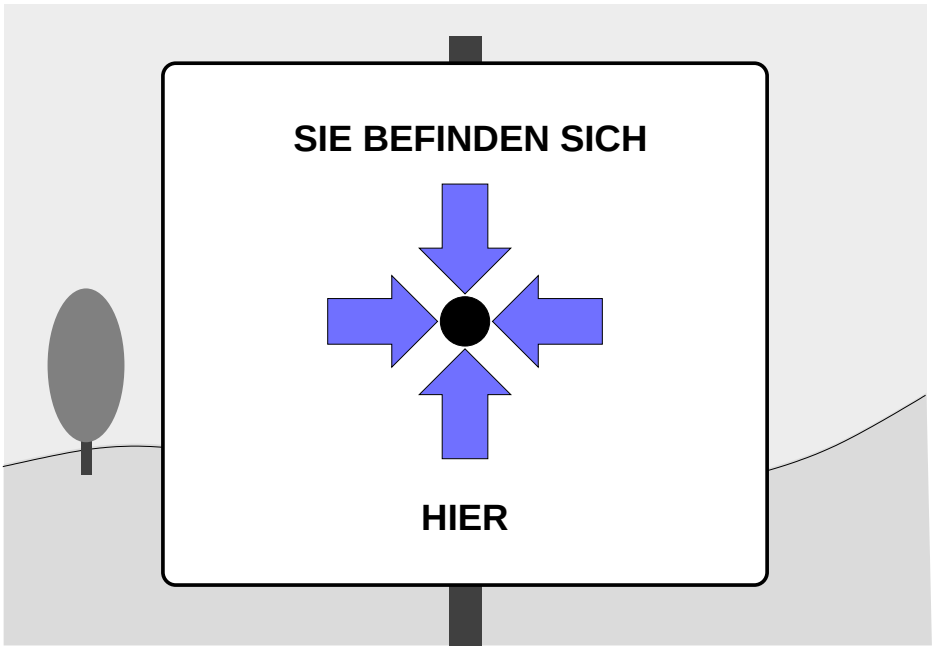


Abbildung 2.3.: „You are here!“

benötigt irgendeine Art von Bezugspunkt, um Informationswert zu besitzen, wie Abbildung 2.3 sehr schön deutlich macht. Ein derartiges Schild mag mich auf einer Wanderung beruhigen (da ich mich anscheinend noch in einer zivilisierten Gegend aufhalte, in der Menschen Schilder aufstellen) oder verwirren, es gibt mir zumindest keine Auskunft darüber, wo denn nun „hier“ ist.

Die Raumkognition kennt eine ganze Reihe unterschiedlicher Referenzsysteme und eine noch größere Anzahl unterschiedliche Bezeichnungen, die durch die Fachwissenschaften von den Neurowissenschaften über die Geographie und Linguistik bis hin zur Informatik nicht nur fachübergreifend sondern auch schon innerhalb einer Disziplin unterschiedlich genutzt werden. Ein kurzer Überblick findet sich in [Mus00], eine ausführlichere Darstellung in [Lev96].

Gerade die natürliche Sprache arbeitet mit einer ganzen Reihe unterschiedlicher Referenzsysteme, zwischen denen kontextabhängig hin- und hergesprungen wird, was auch zu Uneindeutigkeiten führen kann: Die Bedeutung der Aussage „Der Ball liegt vor dem Eimer“ ist klar, ich sehe einen Eimer und zwischen mir und und dem Eimer liegt der Ball. Sage ich hingegen: „Der Ball liegt vor dem Auto“, ist die Sache unklarer. Meine ich damit, daß – wie beim Eimer – der Ball zwischen mir und dem Auto liegt, oder meine ich damit, daß der Ball vor der vorderen Stoßstange des Autos liegt? Habe ich Auto und Ball in meinem Blickfeld, sind beide Interpretationen möglich, bin ich dagegen vielleicht gerade im Haus und beschreibe, wo der Ball in der Garage zu finden ist, eher die zweite. Die Aussage: „Der Ball liegt rechts vorne im Auto“ hingegen ist wieder klar, hier ist eindeutig der Beifah-

rersitz (inklusive Fußraum) des Autos gemeint.[30]

Die Ambiguität im obigen Beispiel rührt daher, daß nicht sofort klar ist, ob sich die Angabe „vorne“ auf den Referenzrahmen des Sprechers oder auf den Referenzrahmen des Autos bezieht. Beim Eimer tritt das Problem hingegen nicht auf, da er kein explizites „vorne“ besitzt und somit keinen eigenen Referenzrahmen aufspannen kann. Bezieht sich die Angabe dagegen auf einen Gegenstand innerhalb des Autos, ist dessen Referenzrahmen so präsent, daß die Beschreibung wieder eindeutig wird.[31]

In der Beschreibung einer räumlichen Konfiguration liegen also mehrere Referenzsysteme gleichzeitig vor:

- absolute Referenzsysteme wie die Längen- und Breitengrade und die Himmelsrichtungen, die für den gesamten betrachteten Raum global Gültigkeit besitzen,[32]
- das Referenzsystem des Beobachters,
- das Referenzsystem des beobachteten Objektes,
- das Referenzsystem eines Referenzobjektes, bezüglich dessen die Position eines Objektes bestimmt wird.
- räumlich begrenzte Referenzsysteme, innerhalb derer sich das beobachtete Objekt befindet wie das Innere des Autos. Jedes räumlich begrenzte Referenzsystem kann als absolutes Referenzsystem benutzt werden, wenn es alle für die Anwendung interessanten Vorgänge umfaßt, aus Sicht der Anwendung also global gültig ist.

Für die Beschreibung von Bewegungen sind für uns vor allem die folgenden drei Referenzsysteme interessant:[33]

allozentrisch heißt im Folgenden ein globales Referenzsystem, in dem die komplette Bewegung beschrieben wird. Dies umfaßt Darstellung I und II.

[30] Interessanterweise käme kaum jemand auf die Idee, den Ball hier ganz vorne rechts unter der Motorhaube zu vermuten, der mögliche Bereich wird hier automatisch auf den Fahrgastraum beschränkt, da Bälle sich selten unter Motorhauben befinden. Befindet sich hingegen der Ölmeßstab „rechts vorne“, so würden die wenigsten Menschen ihn im Fahrgastraum suchen.

[31] Das kann sich wieder ändern, sobald jemand *im* Auto sitzt und damit wieder seinen Referenzrahmen mitbringt usw.

[32] Die Diskussion, ob so etwas wie „absoluter Raum“ und damit ein ausgezeichnetes Referenzsystem existiert, spare ich mir an dieser Stelle, es reicht, wenn das Referenzsystem bezüglich der jeweiligen Anwendung global gültig ist.

[33] Die Begriffe sind, wie oben erwähnt, nicht eindeutig bestimmt sondern werden in unterschiedlichen Bedeutungen benutzt, daher definiere ich sie hier für diese Arbeit.

intrinsisch heißt ein räumlich begrenztes Referenzsystem, das benutzt wird, um einen Abschnitt der Bewegung zu beschreiben. Führt beispielsweise eine Besichtigungstour durch ein Gebäude, so wird die Bewegung innerhalb des Gebäudes häufig in einem völlig anderen Referenzsystem beschrieben als außerhalb.

egozentrisch heißt das Referenzsystem, das durch die Bewegung selbst definiert wird. Ein Objekt bewegt sich zu einem bestimmten Zeitpunkt in eine bestimmte Richtung und dreht sich nun nach rechts oder links, die Beschreibung erfolgt also aus sich des sich bewegenden Objekts selbst. Die Besonderheit ist hier, daß sich das Referenzsystem mit der Bewegung mitdreht und sich somit dauernd ändert. Dies entspricht Darstellung III.

In vielen Fällen induziert das bewegte Objekt selbst einen intrinsischen Referenzrahmen, z. B. das Auto, das ein „vorne“, „hinten“, „links“ und „rechts“ besitzt. Dieser intrinsische Referenzrahmen entspricht häufig dem egozentrischen Referenzrahmen, der durch die Bewegung selbst induziert wird (aber nicht immer, das Auto kann auch rückwärts fahren). Ich spreche von einem *egozentrischen* Referenzrahmen, wenn eine Richtung relativ zur aktuellen Bewegungsrichtung angegeben wird, andernfalls vom *intrinsischen* Referenzrahmen des bewegten Objektes.

Die hier gewählte Einteilung für unterschiedliche Referenzsysteme fokussiert hier (allozentrisch vs. egozentrisch) auf die Bewegungs*richtung*. Wie oben gezeigt existieren zwar auch Unterschiede im Meßfehler von Darstellung I („GPS“) zu Darstellung II („Kompaß“), diese sind allerdings zum einen weniger dramatisch, zum anderen ist in der Praxis der Meßfehler bei der Bestimmung der Entfernung weit geringer als bei der Bestimmung der Bewegungsrichtung.[34]

Die Einbeziehung von intrinsischen Referenzrahmen macht hingegen keine weiteren Probleme. Solange die Lage des den intrinsischen Referenzrahmen aufspannenden Referenzobjektes bezüglich eines allozentrischen Bezugssystems bekannt ist, können die Darstellungen leicht ineinander umgerechnet werden.

Neben den drei oben genannten Bezugssystemen besitzt ein viertes interessante Eigenschaften für die Beschreibung von Bewegungsabläufen: das beobachterzentrierte (deiktische). In der überwiegenden Mehrzahl der Fälle unterscheidet sich das deiktische nicht von einem intrinsischen (oder auch allozentrischen) Bezugssystem. Dies gilt immer dann, wenn der Beobachter den Bewegungsverlauf in Draufsicht verfolgt. Beschreibe ich die Bewegung des Mauszeigers auf dem Bildschirm vor mir mit Begriffen wie „hoch“, „runter“, „links“, „rechts“, so ist es sicher legitim, hier von einem deiktischen Bezugssystem zu reden, es unterscheidet sich nur nicht wesentlich von dem intrinsischen des Bildschirms. Gleiches gilt, wenn ich von der Tribüne aus ein Handballspiel beobachte. Obwohl meine Sicht auf das

[34] Dies zeigen sowohl die Erfahrungen des Bremer Rolland-Projektes als auch eigene Experimente mit einer Pioneer 2-Plattform sowie LEGO-Mindstorms Robotern.

Spielfeld perspektivisch verzerrt ist, wird sich die Beschreibung der Bewegung eines Spielers aus meiner Sicht nicht wesentlich von der Beschreibung im Referenzsystem des Handballfeldes unterscheiden (sieht man davon ab, daß ich vom „linken“ und vom „rechten“ Tor sprechen kann).

Die Situation ändert sich aber in dem Moment gewaltig, in dem der Beobachter sich *innerhalb* des Gebietes befindet, in dem die Bewegung stattfindet. Das nun aufgespannte Referenzsystem hat eine ungewöhnliche Struktur: Jemand „kommt auf mich zu“, „geht von mir weg“, „an mir vorbei“, „um mich herum“ usw. Abbildung 2.4 illustriert dies. In der Beschreibung von Bewegungsverläufen überlagern sich nun ein rechtwinkliges und ein polares Koordinatensystem, die beide den Beobachter als Referenzpunkt haben, in komplexer Weise.

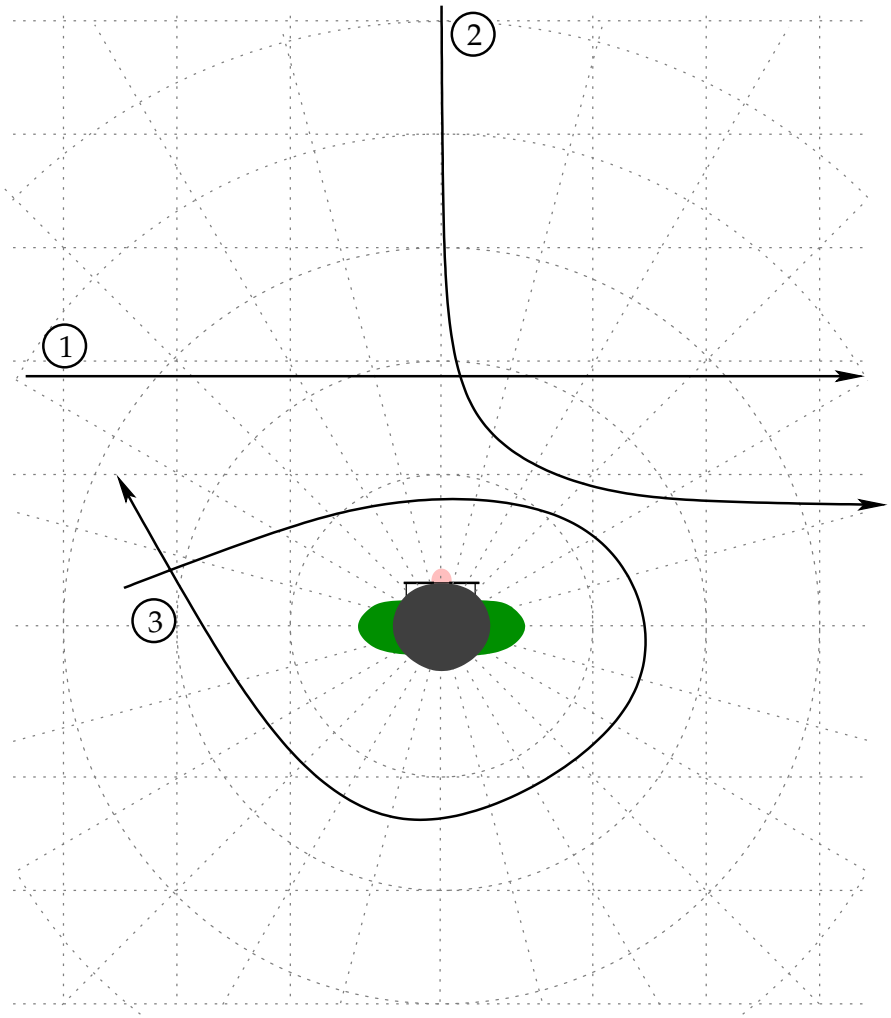
Fährt ein Fahrzeug „von links nach rechts vor mir vorbei“, so bewegt es sich geradlinig und dabei rechtwinklig zu meiner Blickrichtung.[35] Fährt es „frontal“, „von halbrechts“, „von rechts“, … „auf mich zu“, so bewegt es sich geradlinig auf einer Geraden, die durch die Position des Beobachters führt. Die Sache ist allerdings noch komplizierter. Befindet sich jemand links vom Beobachter und bewegt sich auf diesen zu, so „kommt er von links auf mich zu“, aber keineswegs geht er „nach rechts“. Dies liegt daran, daß er sich auf einer Geraden bewegt, die durch den Beobachter hindurchführt. Die gleiche Bewegung auf einer dazu parallelen Geraden, die deutlich vor oder hinter dem Beobachter vorbeiführt wäre dagegen mit „von links nach rechts“ korrekt beschrieben.

Befindet er sich aber links vom Beobachter, so meint „nach rechts“ etwas anderes: eine Bewegung in den Bereich „vorne“, wobei die Entfernung zum Beobachter ungefähr gleich bleibt, also eine Bewegung auf einem Kreisbogen.

Dies alles zeigt, daß eine deiktische Bewegungsbeschreibung in diesem Spezialfall durch die Überlagerung der Referenzsysteme nicht einfach zu modellieren ist. Die hier geschilderte Problematik tritt vor allem in der Benutzerinteraktion auf. Hinzu kommen nun noch die technischen Probleme der perspektivischen Verzerrung, die bei einem Beobachter inmitten des Geschehens besonders stark sind. So ist die Bestimmung der zurückgelegten Distanz aufgrund der Verkürzung durch die Perspektive bei einer senkrecht zur Blickrichtung stattfindenden Bewegung weit einfacher, als bei einer näherkommenden oder sich entfernenden.[36] Bei der Richtung ist es hingegen umgekehrt, eine Bewegung direkt auf den Beobachter zu ist sehr eindeutig, hier werden auch kleine Winkelabweichungen deutlich. Bei schräg zum Beobachter verlaufenden Bewegungen hingegen ist der Winkelfehler weit größer.

[35] Aspekte wie das Drehen des Kopfes, wodurch das durch die Körperstellung induzierte Referenzsystem nicht mehr mit dem durch die Kopfstellung induzierten deckungsgleich ist, betrachte ich an dieser Stelle nicht weiter.

[36] Erfolgt die Bewegung nun noch um den Beobachter herum, muß dieser sich drehen, um sie weiter verfolgen zu können, was die Sache weiter kompliziert, da sich jetzt das komplette Bezugssystem mitdreht.



1: „Er fährt von links nach rechts vor mir vorbei“
2: „Er fährt frontal auf mich zu, biegt dann nach rechts[a] ab und entfernt sich wieder“
3: „Er umkreist mich im Uhrzeigersinn“

[a] dieses „rechts“ ist uneindeutig, da nicht klar ist, ob es aus Sicht des Beobachters oder aus Sicht des Fahrzeugs gemeint ist.

Abbildung 2.4.: Beobachterzentrierte (deiktische) Bewegungsbeschreibung

3. Qualitative Bewegungsrepräsentation

Wie schon in Kapitel 2 beschrieben, ist neben der numerischen Repräsentation von Bewegungsverläufen häufig auch eine qualitative Repräsentation gewünscht, z. B. zur Generierung von Routenbeschreibungen oder zur automatischen Verarbeitung von durch den Menschen gegebenen Wegbeschreibungen. Dieses Kapitel stellt ein Instrumentarium zur qualitativen Beschreibung von Bewegungsverläufen in unterschiedlichen Granularitäten mit besonderem Fokus auf der *Form* der Bewegung vor.

Die im folgenden beschriebene Architektur wurde im Rahmen des DFG-Projektes „Qualitative Repräsentation von Bewegungsverläufen“ [1] von Alexandra Musto[2] entwickelt und von mir ergänzt und erweitert. Die Darstellung repräsentiert die Bewegung eines punktförmigen Objekts, abstrahiert also von Form und Ausdehnung des bewegten Körpers.

3.1. Eine Zwei-Schichten-Architektur zur Bewegungsverarbeitung

Abbildung 3.1 zeigt eine Zwei-Schichten-Architektur zur qualitativen Bewegungsbeschreibung. Die hier vorgestellte Architektur basiert auf der Grundidee, eine Bewegung als Sequenz elementarer „Basisbewegungen“ zu beschreiben. Damit definiert die Wahl dieser „Basisbewegungen“ in starkem Maße die Art der Beschreibung, ihre Konkatenation liefert die Beschreibung der Gesamtbewegung.

Vektorielle Repräsentation: Auf der vektoriellen Ebene besteht eine Basisbewegung aus der Angabe einer (qualitativen) *Richtung*, in der die Bewegung erfolgt und einer (qualitativen) *Distanz*, die in dieser Richtung zurückgelegt wurde, beide Komponenten zusammengefaßt bilden einen *Qualitative Motion Vector* (QMV), eine Sequenz von QMVs beschreibt die Bewegung.

[1] Projekt BR609/9-1 bis 9-3 im DFG-Schwerpunktprogramm „Raumkognition“: „Qualitative Repräsentation von Bewegungsverläufen: Kognitive und psychophysische Grundlagen“

[2] Siehe [Mus00]

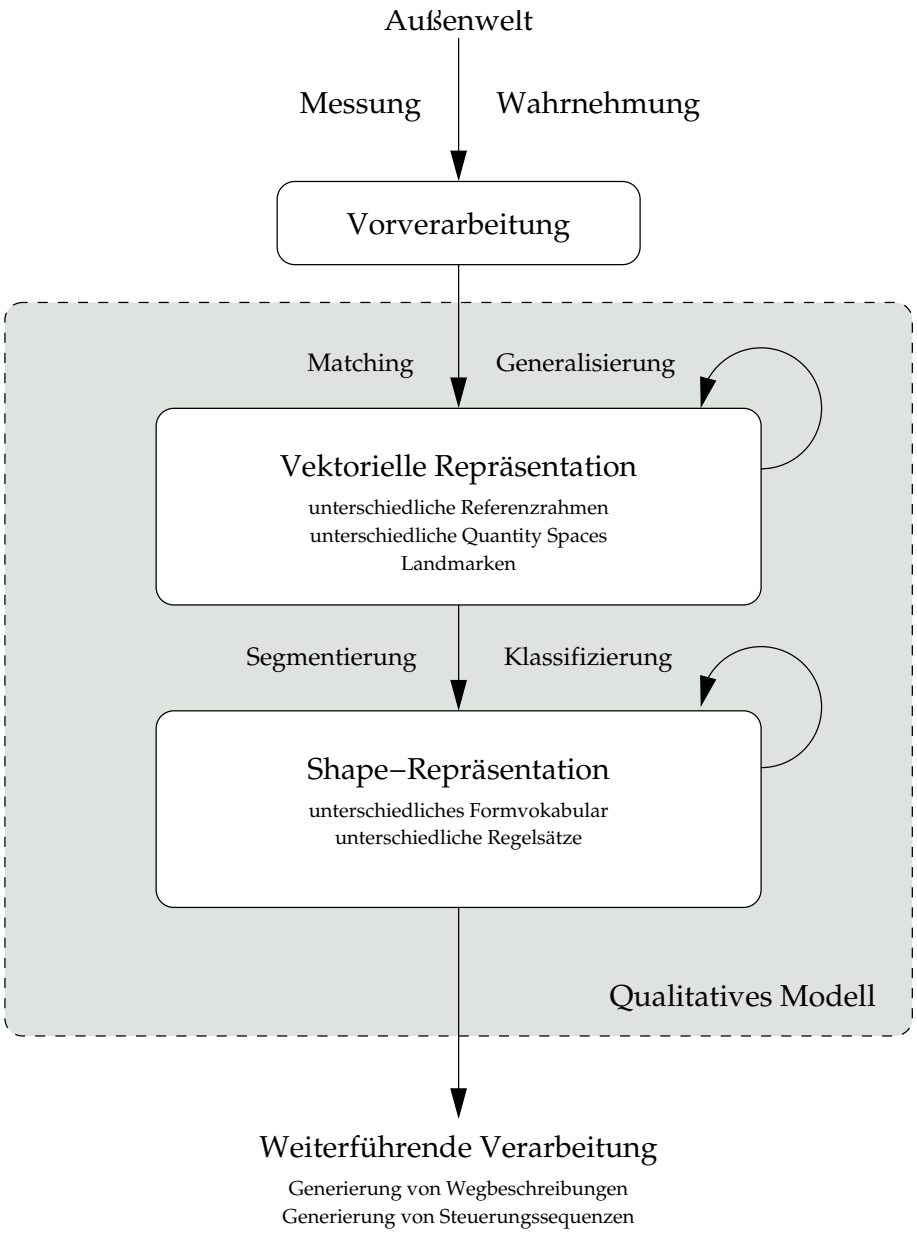


Abbildung 3.1.: Zwei-Schichten-Architektur zur qualitativen Repräsentation von Bewegungsverläufen

Shape-Repräsentation: Auf der propositionalen Ebene ist ein *Shape*, also eine bestimmte Grundform wie „Rechtskurve“, Grundelement der Beschreibung und eine Sequenz dieser Shapes beschreibt die Bewegung. Die einzelnen Shapes werden hierbei durch Attribute genauer spezifiziert: „langgezogene Rechtskurve“.

Während die Beschreibung in der vektoriellen Schicht also eher feingranular ist und die Bewegung hier als Folge von Ortsveränderungen repräsentiert wird, setzt sich die Beschreibung in der propositionalen Schicht aus einer Abfolge von Formen oder Figuren zusammen, die passend zum betrachteten Kontext gewählt werden (im Straßenverkehr sind dies andere als beim Eislauf) und in diesem eine klare Semantik besitzen.

Die vektorielle Schicht ist damit nahe an der technischen Messung von Bewegungsabläufen oder auch ihrer Wahrnehmung durch das visuelle System, wogegen die propositionale Schicht eher der sprachlichen Beschreibung von Bewegungsabläufen nahekommt.

Die dargestellte Architektur leistet dabei zweierlei.

- Zum einen ist sie eine mögliche qualitative Beschreibung für die kognitive und psychophysische Verarbeitung der visuellen Bewegungswahrnehmung des Menschen. Die Außenwelt wird durch das visuelle System wahrgenommen, dabei werden Bewegungen detektiert, vom visuellen System vorverarbeitet und liegen dann als Sequenz von Bewegungsvektoren vor. Diese Sequenz wird weiterverarbeitet, indem Formelemente identifiziert werden und die Bewegung schließlich in einer propositionalen Beschreibung vorliegt, die dann beispielsweise kommuniziert werden kann.

  Damit liefert die Architektur ein Schema des Ablaufs von der Wahrnehmung einer Bewegung (z. B. der Kür eines Eisläufers) bis zu ihrer verbalen Beschreibung („Eine große Schleife gefolgt von einer Pirouette und einer scharfen Kehre“).

  Dies soll nicht heißen, daß sie den Anspruch erhebt, psychophysische Vorgänge korrekt zu modellieren. Es eignet sich allerdings zur Modellierung von Teilaspekten. Ich werde in dieser Arbeit auf die psychophysische Seite nicht weiter eingehen. Der kognitionswissenschaftliche Hintergrund der Architektur ist in [Mus00] beschrieben, Experimente und Ergebnisse zur Psychophysik der Bewegungswahrnehmung finden sich unter anderem in den Arbeiten unserer Projektpartner sowie gemeinsamen Veröffentlichungen.[3]

- Zum anderen liefert sie die Architektur zur qualitativen Bewegungsverarbeitung in technischen Systemen. Eine Bewegung wird gemessen (z. B. mißt ein

[3] [EMS+97, EZM+98, EMZ+98, EMS+98, ES99, ESB+00, ME98, SZB+98, MSE+00, RSB+, SZ95]. Für eine knappe Beschreibung eines psychophysischen Modells zur Bewegungsverarbeitung siehe Abschnitt 11.1.

autonomer Roboter seine Eigenbewegung oder aus einer Videosequenz werden mittels Bildverarbeitung Bewegungen extrahiert), und soweit vorverarbeitet, daß sie als Punktfolge oder Vektorfolge vorliegt. Diese Sequenz wird dann je nach Bedarf geglättet und vereinfacht (Generalisierung) und schließlich in eine qualitative Darstellung (QMV-Sequenz) umgewandelt. Durch Segmentierung dieser Sequenz und Klassifizierung der einzelnen Segmente in Shapes wird schließlich eine propositionale Darstellung gewonnen, die schließlich zur Generierung von Wegbeschreibungen oder Ähnlichem genutzt werden kann.

Dabei sind in allen Schritten weitere Operatoren wie die Umwandlung in unterschiedliche Referenzsysteme oder der Vergleich zweier Bewegungsspuren definiert.

Sowohl kognitiv als auch in technischen Systemen besteht ein Unterschied in der Wahrnehmung/Messung von Eigenbewegungen[4] im Gegensatz zu beobachteten Bewegungen.[5] Diesem Umstand wird durch Modellierung der Bewegung in unterschiedlichen Referenzsystemen Rechnung getragen.

3.2. Qualitative vektorielle Repräsentation

Basis der vektoriellen Repräsentation ist die QMV-Sequenz. Ein QMV (Qualitative Motion Vector) beschreibt eine relative Positionsänderung eines Objektes. Hierfür sind drei Informationen relevant:

- Wie weit hat sich das Objekt bewegt (Distanz d).
- In welche Richtung hat sich das Objekt bewegt (Winkel α).
- Wie lange hat es dazu gebraucht (Zeit Δt).

Wählt man Δt hinreichend klein, so erhält man eine fast beliebig exakte Repräsentation des Bewegungsverlaufes. Der Motion Vektor (MV) $\langle d, \alpha, \Delta t \rangle$ ist daher ein geeignetes Grundelement zur Beschreibung von Bewegungssequenzen

$$\langle d_1, \alpha_1, \Delta t_1 \rangle \langle d_2, \alpha_2, \Delta t_2 \rangle \langle d_3, \alpha_3, \Delta t_3 \rangle \langle d_4, \alpha_4, \Delta t_4 \rangle \langle d_5, \alpha_5, \Delta t_5 \rangle \ldots$$

Durch die Entfernung d und die dafür benötigte Zeit Δt ist die Geschwindigkeit kodiert, der Vergleich aufeinanderfolgender MV liefert die Beschleunigung.

[4] Wahrnehmung der Bewegung durch optischen Fluß, fühlen der Fliehkräfte, Messung der direkten Umgebung mittels Ultraschall, der Anzahl Raddrehungen, des Lenkeinschlags, das bewegte Objekt trägt ein Referenzsystem in sich, daß sich mit ihm bewegt, ...

[5] (Mehr oder weniger perspektivisch verzerrte) Draufsicht auf die Bewegung, die Bewegung findet in einem von außen vorgegebenen Referenzsystem statt.

Bezeichner	Intervall	Belegung
`zero`	$[0, d_0]$	$[0\,\text{m}, 0\,\text{m}]$
`very-close`	$(d_0, d_1]$	$(0\,\text{m}, 4\,\text{m}]$
`close`	$(d_1, d_2]$	$(4\,\text{m}, 16\,\text{m}]$
`medium-distance`	$(d_2, d_3]$	$(16\,\text{m}, 64\,\text{m}]$
`far`	$(d_3, d_4]$	$(64\,\text{m}, 256\,\text{m}]$
`very-far`	(d_4, ∞)	$(256\,\text{m}, \infty)$

Tabelle 3.1.: Mögliche Instantiierung von symbolischen Entfernungsangaben

Wie schon in Kapitel 2 erwähnt, gilt sowohl für die menschliche Wahrnehmung als auch für viele technische Anwendungen, daß Abläufe mit einer festen Framerate gemessen werden, d. h. Δt ist konstant. In so einem Fall ist ein MV von $\langle d, \alpha \rangle$ hinreichend und Δt wird als Attribut der Gesamtsequenz angegeben. Im Folgenden wird Δt meist nicht explizit mit angegeben.

3.2.1. Quantity Spaces

Werden nun Entfernungen und Winkel nicht in numerischer Form sondern durch symbolische Bezeichner angegeben, so erhält man eine qualitative Darstellung, den QMV, z. B. `<far north>` oder `<close west>`. Hierbei ist ein QMV Repräsentant einer ganzen Schar von Bewegungen. `far` beschreibt nicht exakt eine bestimmte Distanz von vielleicht 100 m, sondern ein ganzes Intervall von z. B. 64 m bis 256 m. Ebensowenig bezeichnet die Angabe `north` einen exakten Winkel, auch sie repräsentiert ein Intervall. Jeder QMV besitzt somit einen Akzeptanzbereich von Richtungen und Entfernungen, die auf ihn abgebildet werden.[6]

Ist nun eine numerische MV-Sequenz

$$\langle 10\,\text{m}, 0^\circ \rangle \, \langle 50\,\text{m}, 10^\circ \rangle \, \langle 130\,\text{m}, 76^\circ \rangle \, \langle 100\,\text{m}, 104^\circ \rangle \, \langle 60\,\text{m}, 99^\circ \rangle \ldots$$

gegeben,[7] so kann diese einfach in eine QMV-Sequenz

```
<close north> <medium-distance north> <far east>
<far east> <medium-distance east>
```

[6] Diese qualitative Aufteilung der Richtungs- und Entfernungsdomäne stammt aus dem Gebiet des qualitativen räumlichen Schließens, siehe z. B. [Her94, CDFH97]. Hernández (Mitantragsteller des Raumkognitionsprojektes an der TUM) und Jungert führen in [JH96] eine erste Version des QMV ein, der dort allerdings noch etwas anders aussieht.

[7] Sie liegt als Ergebnis einer Messung in dieser Form vor oder läßt sich aus einem Polygonzug einfach bestimmen.

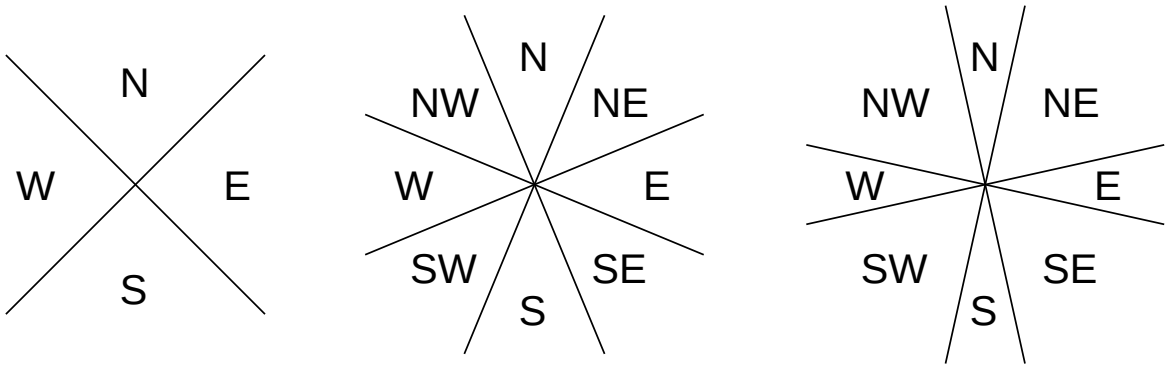


Abbildung 3.2.: Qualitative Richtungen

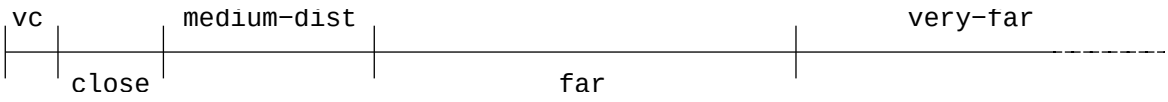


Abbildung 3.3.: Qualitative Distanzen

umgewandelt werden.[8] Wichtig hierfür ist lediglich, daß für jede mögliche Entfernung und jede mögliche Richtung eine eindeutige Zuordnung zu einem symbolischen Bezeichner existiert.[9]

Tabelle 3.1 zeigt ein Beispiel für eine konkrete Belegung mit sechs symbolischen Entfernungsangaben, Abbildung 3.3 liefert eine graphische Veranschaulichung. Abbildung 3.2 zeigt mögliche Belegungen für qualitative Richtungen und Abbildung 3.5 zeigt eine Zusammenführung beider Angaben. Jedes Segment in dieser Darstellung entspricht hierbei genau einem QMV.

Für eine gegebene Entfernungsdomäne $\mathfrak{E}$ und Richtungsdomäne $\mathfrak{R}$:

$$\mathfrak{E} = \{\texttt{zero}, \texttt{very-close}, \texttt{close}, \texttt{medium-distance}, \texttt{far}, \texttt{very-far}\},$$
$$\mathfrak{R} = \{\texttt{zero}, \texttt{north}, \texttt{east}, \texttt{south}, \texttt{west}\}$$

ist ein QMV ein Paar $\langle E, R\rangle$ mit $E \in \mathfrak{E}$ und $R \in \mathfrak{R}$. Aus Konsistenzgründen existiert die Richtung `zero`, um Stillstand zu modellieren (dies ist nicht zwingend notwendig, da bei einer Entfernung von `zero` die Richtungsangabe irrelevant ist).

$\mathfrak{E}$ und $\mathfrak{R}$ sind dabei nicht vom System vorgegeben, sondern können anwendungsabhängig gewählt werden. Eine Einteilung in vier Richtungen ist für viele Anwendungen zu ungenau, die Erweiterung auf 8 Richtungen ist kanonisch:

$$\mathfrak{R}' = \{\texttt{zero}, \texttt{north}, \texttt{ne}, \texttt{east}, \texttt{se}, \texttt{south}, \texttt{sw}, \texttt{west}, \texttt{nw}\}.$$

[8] Unter der Annahme, daß 0° in Richtung Norden und 90° in Richtung Osten zeigt

[9] Ich gehe hier davon aus, daß der Bewegungsverlauf als numerische MV-Sequenz gegeben ist.

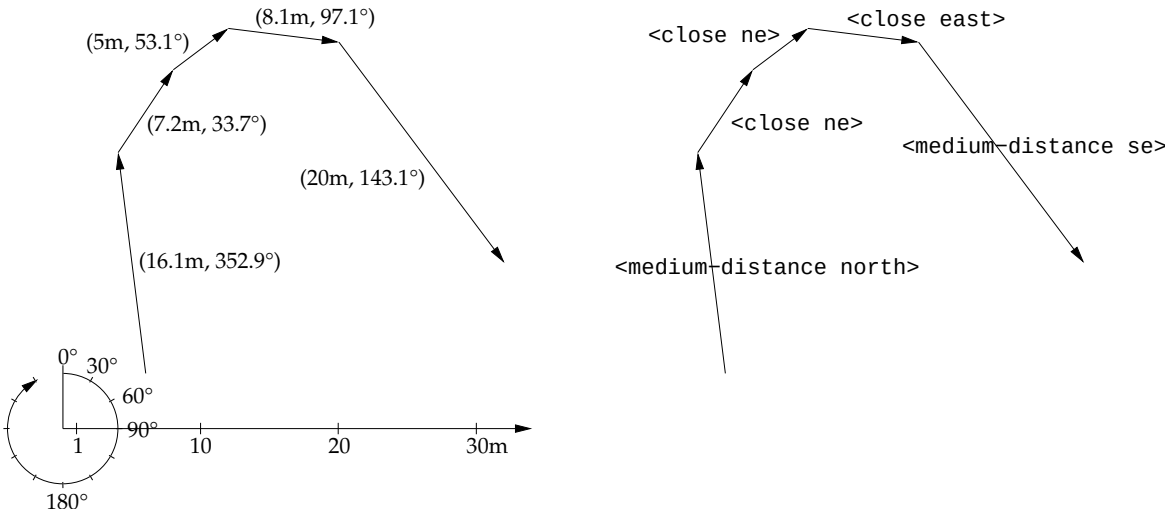


Abbildung 3.4.: Abbildung einer numerischen MV-Sequenz in eine QMV-Sequenz

Abbildung 3.4 greift die Beispielssequenz aus Abbildung 2.1 nochmals auf und zeigt ihre Umwandlung in eine QMV-Sequenz mit 8 Richtungen.

Die Bezeichner `north`, `east`, ... werden im Folgenden für alle allozentrischen qualitativen Richtungsangaben benutzt. Sie sind nicht zwingend identisch mit den Himmelsrichtungen eines absoluten geozentrischen Koordinatensystems,[10] sondern dienen als Bezeichner der Richtungen eines beliebigen extern vorgegebenen Koordinatensystems: bei der Beschreibung einer Mausbewegung auf dem Bildschirm bezeichnet `north` keine Bewegung in Richtung des geographischen Nordpols sondern einfach eine Bewegung nach oben („Landkartennorden“). Die Wahl fiel auf Himmelsrichtungen als Bezeichner, weil diese gut eingängig sind.[11]

3.2.2. Anforderungen an die Akzeptanzbereiche

Die qualitative Beschreibung einer Bewegung erfolgt also durch die Abbildung aller möglichen Entfernungen und Richtungen auf eine (kleine) endliche Zahl von symbolischen Bezeichnern (eine Diskretisierung des Raumes). Die Zuordnung ist durch die Angabe von Intervallen definiert und sollte einige Bedingungen erfüllen:

- Sie muß den ganzen zu beschreibenden Raum abdecken: Kann eine numerische Entfernungs- oder Richtungsangabe nicht auf einen QMV abgebildet werden, so ist die entsprechende Bewegung nicht qualitativ darstellbar.

[10] insbesondere nicht in Hinblick auf die Verzerrungen nahe den Polen

[11] Begriffe wie „rechts“ und „links“ verbieten sich als allozentrische Bezeichner, weil sie auch egozentrisch benutzt werden, die Beschreibung der Mausbewegung kann also nicht mit „oben“, „links“, „unten“, „rechts“ erfolgen und die Wahl völlig neuer Bezeichner kostet Lernaufwand, daher bieten sich die Himmelsrichtungen an.

- Die Akzeptanzbereiche sollten zusammenhängend sein.
- Die Intervalle sollten einander nicht überlappen, da sonst die Zuordnung eines MV zu einem QMV nicht mehr eindeutig ist. Dies ist technisch zwar leicht umzusetzen, überzeugt aber nicht sofort. Die scharfen Ränder der Bereiche scheinen willkürlich, eine Bewegung unter einem Winkel von 47° führt ebensogut nach `north` wie nach `east`, was eine Überlappung beider Bereiche nahelegt.
  - Durch Wahl einer feineren Diskretisierung können solche Effekte abgeschwächt werden. Statt die Bereiche `north` und `east` überlappen zu lassen, wird der Bereich dieser Überlappung mit `ne` bezeichnet und `north` und `east` sind nun die Benennungen der kleiner gewordenen Intervalle. Somit wird nun[12] in 8 statt 4 Richtungen diskretisiert und 47° liefert damit die Richtung `ne`. Die harten Grenzen bleiben hierbei allerdings bestehen und nun ist eben strittig, ob eine Bewegung in Richtung `north` oder `ne` verlief. Außerdem führt die Wahl einer zu feinen Aufgliederung zu einer Rückkehr zur numerischen Repräsentation: es macht keinen Unterschied, ob ich mit 360 symbolischen Richtungsbezeichnern oder mit numerischen Winkeln in einer Genauigkeit von 1° rechne.
  - Die Wahl einer Fuzzy-Modellierung weicht die harten Grenzen auf, indem jeder MV nicht eindeutig *einem* Segment zugeordnet wird, sondern der Grad der Zugehörigkeit zu den unterschiedlichen Segmenten bestimmt wird. Eine Bewegung in Richtung 90° wird dabei beispielsweise mit Zugehörigkeit 1 dem Segment `east` zugeordnet, wogegen eine Bewegung in Richtung 47° zu 0.51 dem Segment `east` und zu 0.49 dem Segment `north` zugeordnet wird.

    Die genaue Modellierung einer Fuzzy-Repräsentation von Bewegungen mittels FMVs (Fuzzy Motion Vectors), des Rechnens auf FMV-Sequenzen und die Umwandlung von FMV in QMV und umgekehrt findet sich in [Mus00].

- Auf den Entfernungen existiert eine Ordnung, d. h. `very-far` $>$ `far` $> \ldots$, die durch die Ordnung auf den numerischen Entfernungen induziert ist.

  Die numerische Entfernung 10 m entspricht der qualitativen Entfernung `close`, 20 m der Entfernung `medium-distance`. Da 20 m $>$ 10 m, gilt somit `medium-distance` $>$ `close`. Diese Ordnung muß konsistent sein, d. h. für alle numerischen Entfernungen $d_{\texttt{med}}$ des Intervalls `medium-distance` und $d_{\texttt{close}}$ des Intervalls `close` muß gelten $d_{\texttt{med}} > d_{\texttt{close}}$, analog für alle anderen Entfernungen (die Entfernungsintervalle dürfen dabei allerdings richtungsabhängig sein, siehe unten).

[12] bei analogem Vorgehen für die anderen Richtungen

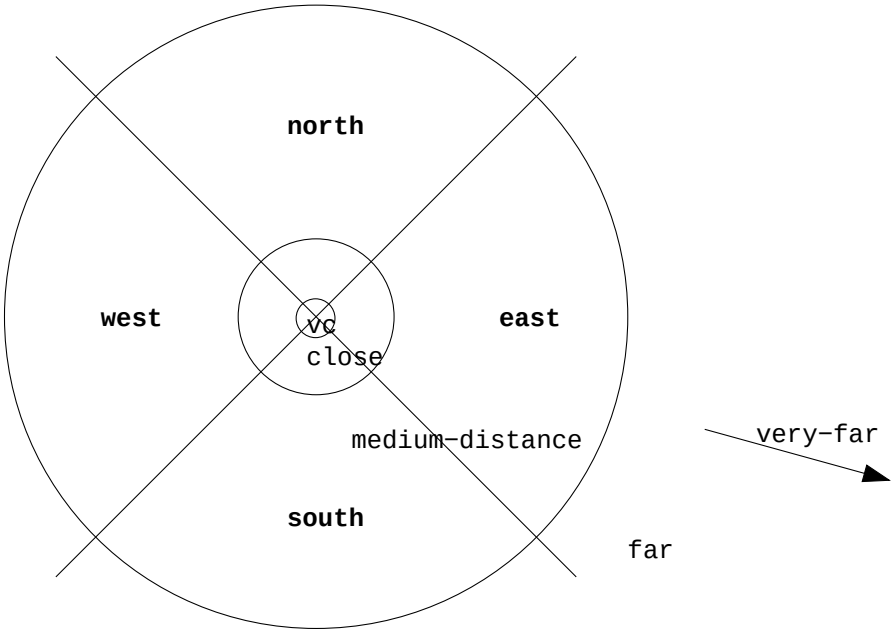


Abbildung 3.5.: Akzeptanzbereiche für Richtung und Entfernung

- Die Grenzen der Entfernungsintervalle werden kontextabhängig bestimmt. Die Beschreibung der Bewegung eines Mauszeigers auf dem Bildschirm ist durch die Bildschirmgröße (beispielsweise 1600 × 1200 Pixel) beschränkt, somit sollte der Bezeichner für die größte Distanz (`very-far`) in dieser Größenordnung liegen und beispielsweise einer Entfernung ab 800 oder 1000 Pixeln entsprechen. Für die Bewegung eines Serviceroboters in einem Fabrikgelände dagegen liegt die größte Entfernung vielleicht bei 2000 m, was eine Wahl von `very-far` ab 1000 m nahelegt.
- Ebenso wird die Anzahl unterschiedlicher Richtungen und Entfernungen anwendungsabhängig gewählt.

Weiterhin gilt:

- Richtungs- und Entfernungsintervalle müssen nicht zwingend unabhängig sein, statt der runden Akzeptanzbereiche aus Abbildung 3.5 (euklidische Norm) sind genauso quadratische Akzeptanzbereiche möglich (Maximumsnorm, Abbildung 3.6). Dies widerspricht nicht der Forderung nach einer konsistenten Ordnung auf der Entfernungsdomäne, das Entfernungsmaß ist hier lediglich winkelabhängig.[13]
- Die Richtungssegmente müssen nicht zwingend gleich groß gewählt werden, wie schon Abbildung 3.2 (rechts) zeigt, sie müssen nicht einmal symmetrisch

[13] Umgekehrt sind auch entfernungsabhängige Winkel denkbar.

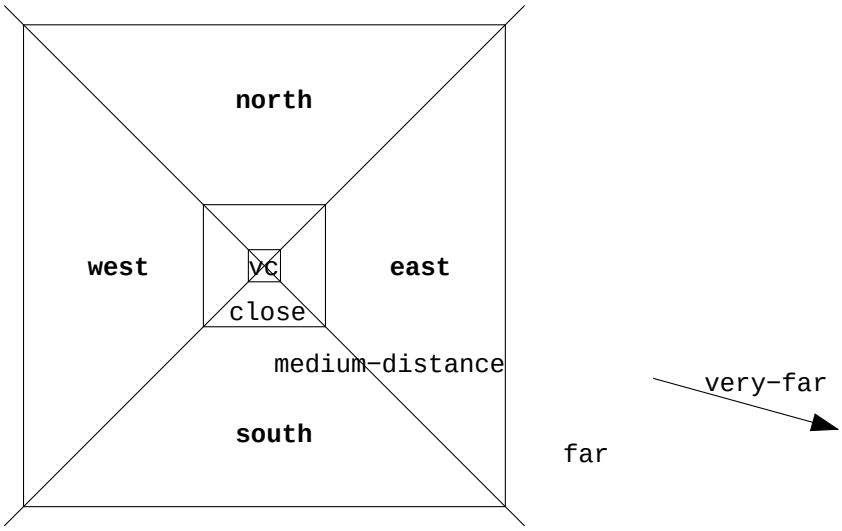


Abbildung 3.6.: Quadratische Akzeptanzbereiche für Richtung und Entfernung

sein. Allerdings ist die Wahl einer geraden Anzahl gleichgroßer (und zueinander punktsymmetrischer) Richtungssegmente von Vorteil, wenn es darum geht, Rechenoperatoren auf QMVs zu definieren (Siehe Abschnitt 3.4.2).

Mittels der nach diesen Kriterien gewählten Akzeptanzbereiche kann nun eine endliche Anzahl unterschiedlicher QMV gebildet werden können. Durch Aneinanderhängen dieser einzelnen QMV können beliebige Bewegungsspuren beschrieben werden. Aus praktischen Gründen bietet sich an, in solchen Sequenzen gleiche aufeinanderfolgende QMV durch Angabe eines Index zusammenzufassen, aus

```
<close north> <close north> <close north> <far west>
<far west> <medium-distance west> <close south>
```

wird also

```
<close north>3 <far west>2 <medium-distance west>1
<close south>1.
```

3.3. QMV-Sequenzen in unterschiedlichen Referenzrahmen

Wie schon in Abschnitt 2.3 gezeigt, können Bewegungen in sehr unterschiedlichen Referenzrahmen (allozentrisch und egozentrisch) dargestellt werden. Darüberhinaus besteht die Möglichkeit, während einer Bewegung, die durch unterschiedliche Regionen verläuft, zwischen verschiedenen intrinsischen Referenzrahmen dieser Regionen zu wechseln.

∘	north	east	south	west
north	forward	right	backward	left
east	left	forward	right	backward
south	backward	left	forward	right
west	right	backward	left	forward

Tabelle 3.2.: Umwandlung allozentrischer in egozentrische Richtungen

Die QMV-Darstellung ist in der Lage, sowohl egozentrisch als auch allozentrisch gemessene Bewegungsspuren zu repräsentieren, wobei beide Darstellungen, wie in den folgenden Abschnitten beschrieben, in unterschiedlicher Weise aus den numerischen Daten generiert werden.

3.3.1. Allozentrische QMV

Alle bisherigen Beispiele von QMV-Sequenzen arbeiten auf einem allozentrischen, also extern vorgegebenen Referenzrahmen. Ihre Erzeugung aus numerischen Daten, die Wahl der Akzeptanzbereiche etc. wurde schon ausführlich dargestellt.

Die allozentrische QMV-Repräsentation ist die geeignete Darstellung für alle allozentrisch vorliegenden numerischen Bewegungssequenzen und ebenso für die Repräsentation innerhalb intrinsischer Referenzrahmen. Hier muß lediglich (beispielsweise durch Wahl unterschiedlicher Bezeichner) sichergestellt werden, daß immer klar ist, welches intrinsische Referenzsystem für welches Teilstück der Bewegung als Bezugssystem dient.

3.3.2. Egozentrische QMV

Für die Darstellung einer Bewegung als egozentrische QMV-Sequenz ist eine andere Richtungsdomäne

$$\mathfrak{R} = \{\texttt{zero}, \texttt{forward}, \texttt{backward}, \texttt{left}, \texttt{right}\}$$

nötig.[14] Eine egozentrische QMV-Sequenz wird nun entweder aus einer allozentrischen QMV-Sequenz oder direkt aus einer numerischen Bewegungsspur erzeugt, wie in den folgenden Abschnitten beschrieben.

Erzeugung aus einer allozentrischen QMV-Sequenz

Im egozentrischen Referenzsystem wird die Richtung jedes QMV abhängig von seinem Vorgänger bestimmt, damit wird z. B. die allozentrische QMV-Sequenz

[14] Hier für vier Richtungen, eine Erweiterung auf acht oder mehr Richtungen ist problemlos möglich.

```
<close north> <medium-distance north> <close east>
<close north> <close west> <medium-distance south>
```

als egozentrische QMV-Sequenz

```
<close forward> <medium-distance forward>
<close right> <close left> <close left>
<medium-distance left>
```

ausgedrückt.

Der Regelsatz für diese Umwandlung ist trivial: Zeigen aufeinanderfolgende (allozentrische) QMV in die gleiche Richtung, so ist die (egozentrische) Richtung des zweiten bezüglich des ersten `forward`, zeigen sie in entgegengesetzte Richtungen, dann `backward` usw. Tabelle 3.2 zeigt die vollständige Verknüpfungsmatrix.[15] Die Richtung `zero` fehlt hier. Der Nullvektor ist ein Sonderfall, hier muß die Richtung bezüglich des vorletzten Vektors angegeben werden. Diese Sonderbehandlung ist ein Grund, sowohl allo- als auch egozentrisch ohne eine explizite Richtung `zero` zu arbeiten und einem Nullvektor immer die Richtung seines Vorgängers zuzuweisen. Der erste Vektor einer egozentrischen QMV-Sequenz zeigt immer in Richtung `forward`.

Damit die Abbildung einer allozentrischen auf eine egozentrische QMV-Sequenz funktioniert, ist wesentlich, daß alle allozentrischen Richtungssektoren gleich groß sind. Abbildung 3.2 rechts zeigt eine Richtungsdomäne, in der die „diagonalen“ Richtungen `ne`, `nw`, `se` und `sw` einen größeren Sektor umfassen als die Richtungen `north`, `south`, `east` und `west`. Diese Einteilung kann sowohl sprachlich als auch wahrnehmungspsychologisch[16] begründet sein, sie macht jedoch technische Probleme. Wird die mittels der Richtungsdomäne Abb. 3.2 rechts erzeugte allozentrische QMV-Sequenz

```
<close north> <close ne> <close ne>
```

in die egozentrische Sequenz

```
<close forward> <close forward-right> <close forward>
```

umgewandelt, so umfaßt das erste `forward` (erster QMV) einen Winkel von 30°, da er aus der allozentrischen Richtung `north` erzeugt wurde, die einen Winkel von

[15] Das Konstruktionsprinzip sollte klar sein, eine Matrix für 8 Richtungen kann analog konstruiert werden.

[16] Wird eine Bewegungsspur in „Draufsicht“ beschrieben, z. B. die Bewegung des Mauszeigers auf dem Bildschirm, so ist bekannt, daß Abweichungen bezüglich der horizontalen und vertikalen Achse weit feiner differenziert werden als diagonale Bewegungen („oblique effect“, siehe [GBH98]), gleiches findet sich in der Sprache: Abweichungen größer 15° von der horizontalen oder vertikalen Achse werden nicht mehr mit Bezeichnern wie „oben“ oder „rechts“ sondern mit zusammengesetzten Bezeichnern wie „rechts oben“ benannt [ZSB+98].

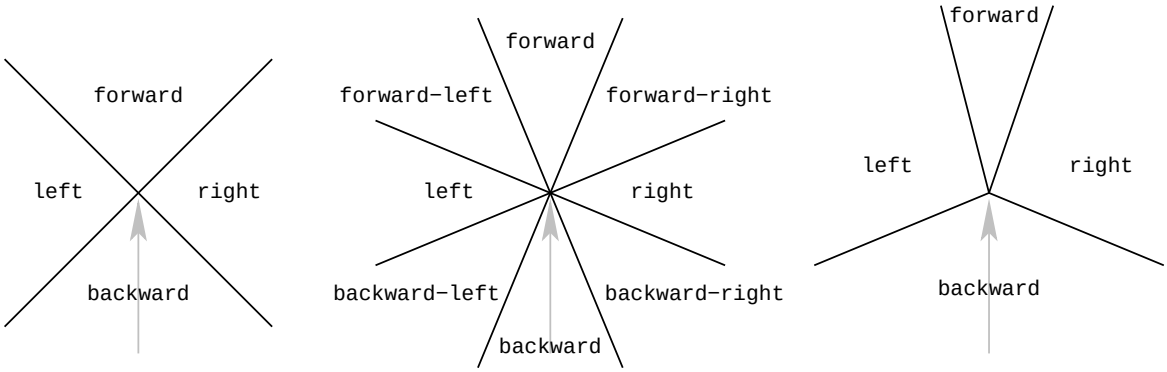


Abbildung 3.7.: Egozentrische qualitative Richtungen

30° umfaßt. Das zweite `forward` (dritter QMV) dagegen umfaßt einen Winkel von 60°, da er aus der allozentrischen Richtung `ne` erzeugt wurde, die einen Winkel von 60° umfaßt. In der egozentrischen Sequenz ist dieser Unterschied jedoch nicht mehr erkennbar. Die Sequenz ist damit nicht mehr rotationsinvariant. Nur zu allozentrischen Richtungsdomänen, in denen alle Richtungssektoren gleichgroß sind, lassen sich zugehörige egozentrische Richtungsdomänen angeben (Abb. 3.2 links wird auf Abb. 3.7 links abgebildet, Abb. 3.2 Mitte auf Abb. 3.7 Mitte). Auch in diesem Fall ist die egozentrische Darstellung nicht völlig rotationsinvariant, da die Rotation hier nur in diskreten Schritten (Winkel des Einzelsektors) erfolgen darf.

Egozentrische „Messung“

Statt eine egozentrische QMV-Sequenz aus einer allozentrischen zu gewinnen, kann sie natürlich auch direkt aus einer numerisch vorliegenden Bewegungsspur erzeugt werden. Abbildung 3.7 zeigt mögliche Akzeptanzbereiche für eine egozentrische Messung. Während bei der Erzeugung einer allozentrischen QMV-Sequenz aus einer numerischen Beschreibung das Matching der Bewegungsrichtung bezüglich der allozentrischen (numerischen) Richtung des einzelnen MV erfolgt, geschieht dies bei der direkten Erzeugung einer egozentrischen QMV-Sequenz anhand des relativen Winkels. Anschaulich gesagt werden die Richtungskreuze im allozentrischen Fall global gleichausgerichtet auf die Eckpunkte der Sequenz gesetzt, während sie im egozentrischen Fall jeweils senkrecht auf dem Vorgängervektor stehen, wie Abbildung 3.8 zeigt.

Wir bekommen hier (Abb. 3.8 links) die allozentrische QMV-Sequenz

```
<close north> <close north> <close east> <close north>
<close east> <close south>,
```

die nach Tabelle 3.2 umgewandelt die egozentrische QMV-Sequenz

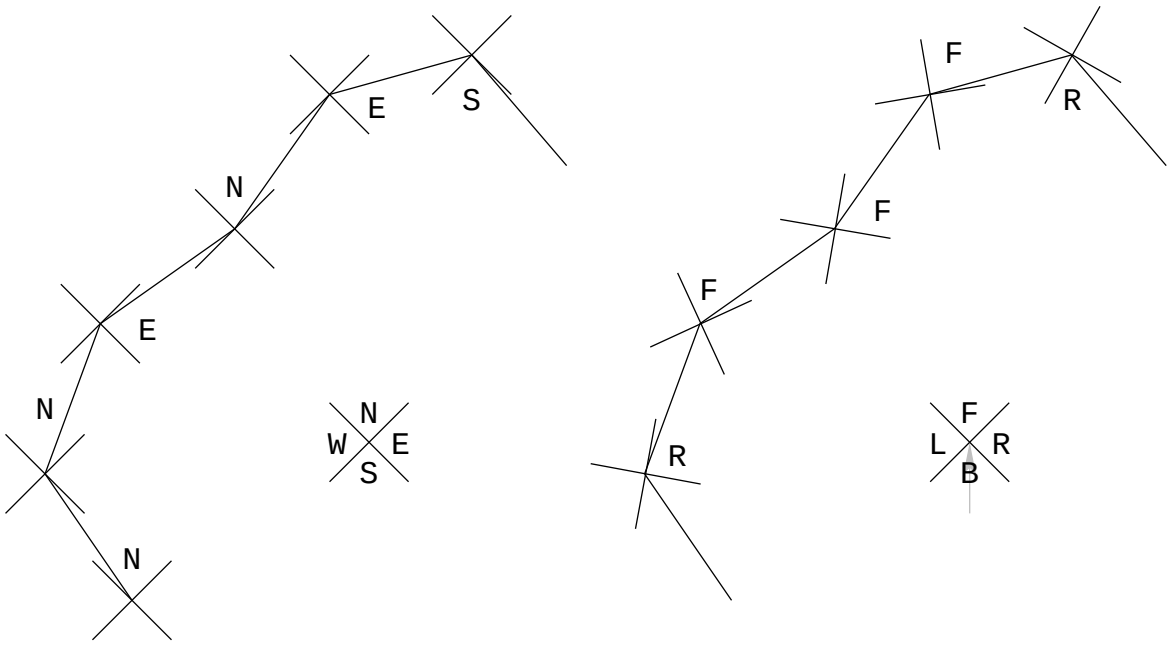


Abbildung 3.8.: Allozentrische und egozentrische Messung

```
<close forward> <close forward> <close right>
<close left> <close right> <close right>
```

liefert. Diese unterscheidet sich allerdings bedeutend von der aus der numerischen Darstellung erzeugten (Abb. 3.8 rechts) egozentrischen QMV-Sequenz

```
<close forward> <close right> <close forward>
<close forward> <close forward> <close right>.
```

Die durch direkte Messung gewonnene Sequenz gibt die Form der Bewegung hier besser wieder, als die über den Umweg der allozentrischen Darstellung errechnete.

Probleme bei egozentrischer „Messung“

Leider ist das Ergebnis einer direkten Generierung einer egozentrischen QMV-Sequenz häufig weit schlechter, wie Abbildung 3.9 an einem Extrembeispiel zeigt: Die hier gezeigte MV-Sequenz beschreibt eine Kreisbewegung. Jeder Vektor ist dabei gegenüber seinem Vorgänger um 30° nach rechts verdreht, wodurch bei einer egozentrischen Messung mit 4 (gleichgroßen) Richtungssektoren ausschließlich die Richtung `forward` zum Tragen kommt (der Richtungssektor `forward` umfaßt hier den Bereich von $-45°$ bis $+45°$), wir bekommen

```
<close forward>12,
```

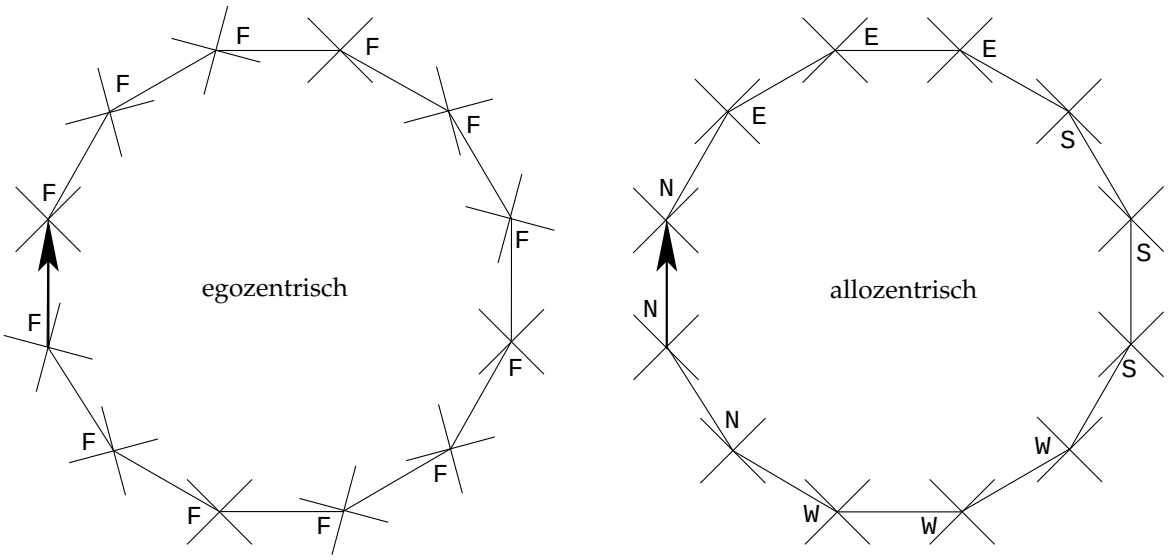


Abbildung 3.9.: Egozentrische und allozentrische Messung einer Kreisbewegung

eine QMV-Sequenz, die eine gerade Strecke beschreibt. Diese Darstellung ist selbstverständlich unbefriedigend. Die allozentrische QMV-Sequenz

```
<close north>2 <close east>3 <close south>3
<close west>3 <close north>1
```

gibt den Bewegungsverlauf hier weit besser wieder, und auch die aus ihr gewonnene egozentrische QMV-Sequenz

```
<close forward>2 <close right>1 <close forward>2
<close right>1 <close forward>2 <close right>1
<close forward>2 <close right>1
```

beschreibt eine – wenn auch sehr eckige – Kreisbewegung. Daß hier eine gleichmäßige Kreisbewegung stattfindet, ist sowohl allozentrisch als auch egozentrisch mit nur vier unterscheidbaren Richtungen selbstverständlich nicht darstellbar.

Gesucht ist also eine Darstellung von Bewegungsverläufen als egozentrische QMV-Sequenz, die die beschriebenen Probleme vermeidet. Die folgenden Abschnitte beschreiben unterschiedliche Vorgehensweisen zur Erzeugung einer egozentrischen QMV-Sequenz aus einer numerisch gegebenen Bewegungsspur sowie die dabei auftretenden Probleme und möglichen Lösungsansätze.

„Messung“ auf Umwegen

Liegt eine Bewegung als allozentrisch gemessener numerischer Polygonzug vor, und ist eine qualitative egozentrische Darstellung gewünscht, bietet sich der Umweg über die qualitative allozentrische Darstellung an. Das Ergebnis ist zwar nicht

unbedingt optimal, wie das Beispiel aus Abbildung 3.8 gezeigt hat, extreme Verfälschungen können dabei aber nicht auftreten.

Liegt die Bewegung hingegen als egozentrisch gemessene MV-Sequenz vor, so sieht die Sache etwas anders aus. Eine egozentrisch gemessene Sequenz enthält (siehe Abschnitt 2.3.2) von sich aus schon Winkelfehler. Wird eine solche MV-Sequenz nun in ein Koordinatensystem eingebettet und in eine allozentrische QMV-Sequenz umgewandelt, so führt dies zu Fehlern. Bei einem angenommenen Winkelfehler von 5° kann die Bewegung sich innerhalb von 20 MV um über 90° verdrehen, so daß bei einer längeren Bewegung der Bezeichner `north` im sechzigsten QMV nicht mehr die gleiche Richtung beschreibt, wie im ersten. Dies bedeutet aber lediglich, daß es falsch wäre, aus einer egozentrisch gemessenen MV-Sequenz eine allozentrische QMV-Sequenz zu erzeugen und diese als Endergebnis zu präsentieren. Wird die allozentrische QMV-Sequenz dagegen lediglich als Zwischenschritt genutzt, um daraus dann schließlich die egozentrische QMV-Sequenz zu erzeugen, so stellt dies kein Problem dar, die egozentrische Darstellung impliziert ja schon die Möglichkeit von Winkelfehlern.

Die so gewonnene egozentrische Darstellung beschreibt die gemessene Bewegung, besitzt aber einen Schönheitsfehler: Während in einer egozentrisch gemessenen Bewegung die Richtung eines Vektors nur von der Richtung seines Vorgängers abhängt, was auch für die qualitative egozentrische Darstellung gelten sollte, so kommt durch den allozentrischen Zwischenschritt ein Koordinatensystem ins Spiel, und die Ausrichtung dieses Koordinatensystems hat entscheidenden Einfluß auf die resultierende QMV-Sequenz.[17]

Direktes Vorgehen

Gesucht ist also eine Methode, die aus einer egozentrischen MV-Sequenz direkt eine egozentrische QMV-Sequenz erzeugt und dabei die Probleme aus obigem Extrembeispiel vermeidet. Wichtig für die folgenden Überlegungen ist, daß eine egozentrische Darstellung[18] (ob numerisch oder qualitativ) sicher nicht geeignet ist, um Aussagen über die globale Position des sich bewegenden Objektes zu tätigen: nach einer Bewegung von mehreren hundert MV ist durch die Akkumulation von Winkel- und Entfernungsfehlern keine globale Aussage über die aktuelle Richtung bezüglich der Startrichtung und zur aktuellen Entfernung vom Startpunkt möglich, dies kann somit auch nicht Ziel einer qualitativen egozentrischen Darstellung sein. Ziel ist vielmehr eine möglichst gute Wiedergabe der *Form* der Bewegung als Abfolge von Rechts- und Linkskurven und geraden Teilstücken.

[17] Wäre die Bewegung aus Abb. 3.8 in einem um 45° verdrehten allozentrischen Koordinatensystem gemessen worden, so würde ihre allozentrische QMV-Beschreibung nicht diese ständigen Wechsel zwischen `north` und `east`, sondern vier aufeinanderfolgende gleichgerichtete QMVs enthalten.

[18] oder genauer: eine fehlerbehaftete egozentrische Messung, siehe auch Kapitel 2.3.2

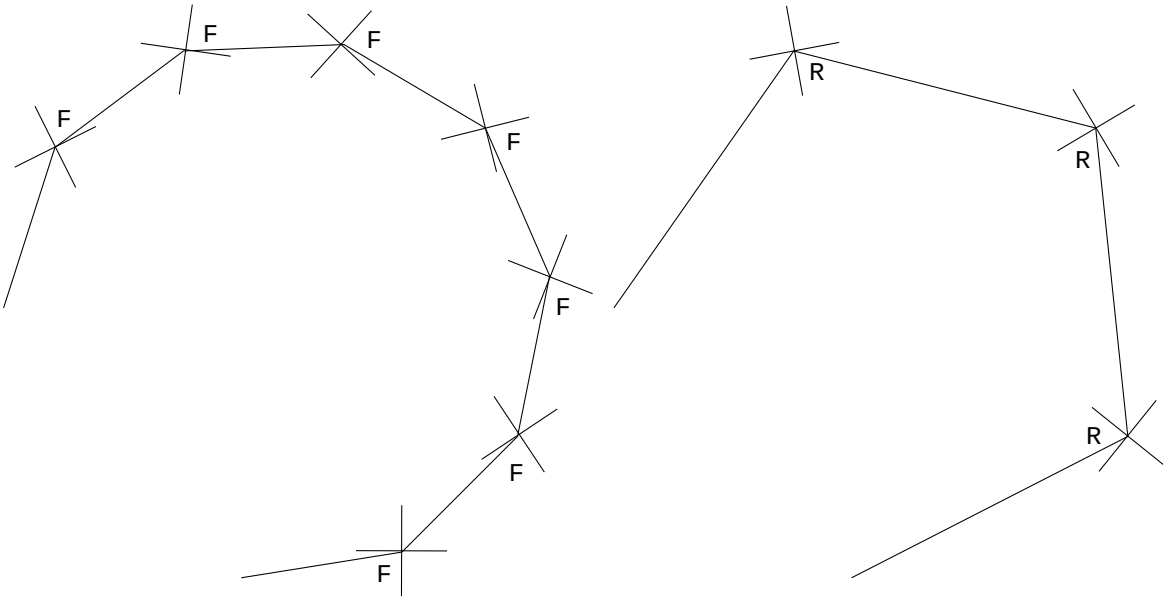


Abbildung 3.10.: Probleme durch Erhöhung der Meßgenauigkeit

Einen großen Einfluß auf die möglichen „Verzerrungen“ hat hierbei die Anzahl der unterscheidbaren Richtungen, je mehr Richtungen zur Verfügung stehen, desto besser werden Kurven erkannt. Zu viele Richtungen würden jedoch den qualitativen Ansatz schnell ad absurdum führen, da – abgesehen von der Wahl symbolischer Bezeichner für die einzelnen Richtungen – fast kein Unterschied mehr zur Darstellung durch numerische Winkel besteht (bei 30 oder 40 unterscheidbaren Richtungen möchte ich nicht mehr unbedingt von einer „qualitativen“ Darstellung sprechen).

Eine Erhöhung der Anzahl unterscheidbarer Richtungen wird auch einem weiteren Phänomen nicht gerecht, der Abhängigkeit des Fehlers von Meßgenauigkeit und Geschwindigkeit. Abbildung 3.10 zeigt die Messung einer Kreisbewegung, wobei die Bewegung in der Abbildung links mit einem feineren Zeitraster (doppelte Frequenz) gemessen wurde. In der Darstellung als egozentrische QMV-Sequenz zeigt sich, daß die genauere Darstellung

`<close forward>`^{8}

die Bewegung schlechter beschreibt als die gröbere Darstellung

`<medium-distance forward> <medium-distance right>`^{3}.

Wird die MV-Sequenz mit einem festen Zeitraster gemessen, so erzeugt sowohl eine höhere Frequenz als auch eine langsamere Bewegung diesen Effekt, die Erkennung von Kurven ist hier also geschwindigkeitsabhängig.

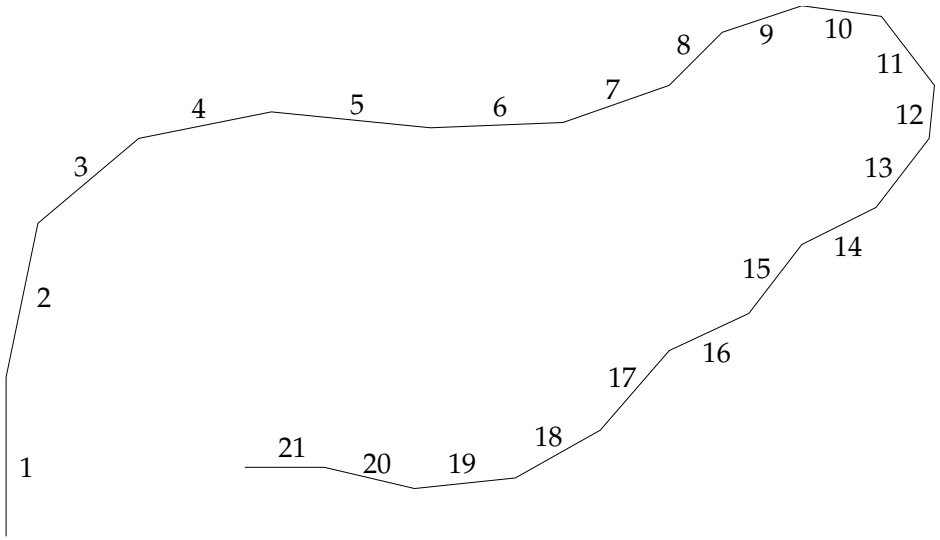


Abbildung 3.11.: Egozentrische MV-Sequenz

Musto schlägt daher vor, hier von einer geschwindigkeits- (und frequenz-) auf eine distanzabhängige Darstellung zu wechseln, indem aufeinanderfolgende MVs so lange zusammengefaßt werden, bis die insgesamt zurückgelegte Entfernung einer festgelegten Distanz entspricht. Dies entspricht der Vorverarbeitung der numerischen MV-Sequenz mittels der in Kapitel 9.2 beschriebenen Σ-Generalisierung. Grundsätzlich leistet jede generalisierende Vorverarbeitung hier das Gewünschte, da annähernd geradlinige Teilstücke zu einer einzelnen geraden Strecke generalisiert und Kurven im Mittel schärfer werden. Die so gewonnene egozentrische QMV-Sequenz repräsentiert dann allerdings nicht mehr jeden numerischen MV durch einen QMV, jeder QMV ist vielmehr Repräsentant einer Folge zusammengefaßter MV.

Zur direkten frequenzunabhängigen egozentrischen QMV-Darstellung einer MV-Sequenz, bei der jeder numerische MV durch genau einen egozentrischen QMV repräsentiert wird, ist ein anderer Ansatz nötig.

Abbildung 3.11 zeigt die (egozentrische) MV-Sequenz

$0.90\,\mathrm{m}\ (11.7^\circ)\ 0.89\,\mathrm{m}\ (38.2^\circ)\ 0.75\,\mathrm{m}\ (28.8^\circ)\ 0.76\,\mathrm{m}\ (17.0^\circ)\ 0.90\,\mathrm{m}\ (-8.0^\circ)\ 0.75\,\mathrm{m}$
$(-17.0^\circ)\ 0.64\,\mathrm{m}\ (-25.7^\circ)\ 0.42\,\mathrm{m}\ (26.6^\circ)\ 0.47\,\mathrm{m}\ (26.0^\circ)\ 0.45\,\mathrm{m}\ (44.8^\circ)\ 0.49\,\mathrm{m}$
$(-136.7^\circ)\ 0.30\,\mathrm{m}\ (31.9^\circ)\ 0.49\,\mathrm{m}\ (25.9^\circ)\ 0.47\,\mathrm{m}\ (-25.9^\circ)\ 0.49\,\mathrm{m}\ (27.4^\circ)\ 0.50\,\mathrm{m}$
$(-24.1^\circ)\ 0.60\,\mathrm{m}\ (19.7^\circ)\ 0.55\,\mathrm{m}\ (23.3^\circ)\ 0.57\,\mathrm{m}\ (19.2^\circ)\ 0.52\,\mathrm{m}\ (-13.2^\circ)\ 0.45\,\mathrm{m}$

Eine allozentrische Messung liefert[19]

N N E E E E E N E E S S S W S W S W W W W,

[19]Die Darstellung beschränkt sich auf die Angabe der Richtung (Entfernungen werden der Übersicht halber nicht mit angegeben), wobei vier Richtungen unterschieden werden. Aufgrund der

als egozentrische Sequenz also

F F R F F F F L R F R F F R L R L R F F F

mit dem schon beschriebenen Problem der ständigen Richtungswechsel im Teilstück 13 mit 18. Die QMV-Sequenz enthält hier eine Art Zickzackbewegung (R L R L R), obgleich die numerischen MV in diesem Teilstück die fast gleiche Richtung besitzen (die Schwankungen in der Richtung sind klein), womit die Folge F F F F F dieses Teilstück besser repräsentieren würde.[20]

Der Effekt tritt auch nur aufgrund der diagonalen Lage (um $225° = 180° + 45°$) der MV in diesem Teilstück auf, die gleiche Bewegungsspur, in einem etwas verdrehten Koordinatensystem gemessen, würde eine andere QMV-Sequenz (ohne die R L R L R Folge im Teilstück 13 mit 18) liefern.

Eine direkte egozentrische Messung hingegen liefert einfach

F F F F F F F F F F F F F F F F F F F F F,

oder kürzer: `forward` 21, da ausnahmslos alle MV weniger als 45° gegenüber ihrem Vorgänger verdreht sind, auch diese Darstellung überzeugt nicht.

Eine gute egozentrische Darstellung sollte sowohl die Rechtskurve am Anfang (Vektoren 1 mit 5) als auch die 180°-Kehre nach rechts (9 mit 13) erkennen, ohne das folgende Teilstück (13 mit 18) als Zickzack[21] zu repräsentieren. Das bedeutet nicht, daß in einer egozentrischen QMV-Sequenz keine solchen Zickzack-Folgen vorkommen dürfen. Liegt in der numerischen MV-Sequenz eine deutliche Zickzackbewegung vor (aufeinanderfolgende Vektoren weichen in ihrer Richtung im Wechsel rechts- und linksseitig deutlich voneinander ab), so soll dies natürlich auch in der QMV-Sequenz als Zickzack repräsentiert werden. Schwankt die Richtung aufeinanderfolgender numerischen MV hingegen nur leicht (wie eben im Teilstück 13 mit 18), so stellt eine Folge von `forward`-QMV eine bessere Repräsentation dar.

Ein Verfahren zur Erzeugung einer egozentrischen QMV-Sequenz aus einer numerischen Bewegungsspur sollte also folgende Eigenschaften aufweisen:

- Kurven sollen erkannt werden, auch wenn sie langsam durchfahren werden.
- Annähernd geradlinige Teilstücke sollen als Folge von `forward`-Vektoren modelliert werden.
- Die Erkennung sollte weitgehend rotationsinvariant sein, also nicht von der Lage der MV-Sequenz in einem bestimmten Koordinatensystem abhängen.

Länge der Sequenz werden die Bezeichner `forward`, `left`, `right`, `backward` zu F, L, R und B (bzw. `north`, `west`, `east`, `south` zu N, W, E und S) abgekürzt.

[20] Diese Schwankungen haben nicht unbedingt irgendetwas mit Meßfehlern bei der Messung der numerischen MV-Sequenz zu tun. Das beobachtete Objekt kann sich durchaus in dieser Form bewegt haben.

[21] ständiger Wechsel zwischen zwei benachbarten Richtungen

Auf einer gegebenen egozentrischen MV-Sequenz ist das Erkennen von Kurven im Grunde nicht weiter schwer. Der Winkel zwischen zwei MV ist (abgesehen von einer gewissen Meßungenauigkeit) bekannt,[22] addiert man nun diese relativen Winkel aufeinanderfolgender MV und wird der resultierende Winkel irgendwann zu groß, so muß eine Kurve gefahren worden sein. Ein annähernd geradlinig gefahrenes Streckenstück zeichnet sich hingegen dadurch aus, daß die (vorzeichenbehaftete) Summe dieser Winkel um 0° schwankt.

Gesucht ist nun ein Verfahren, daß die oben aufgestellten Forderungen erfüllt. Hierzu werden im Folgenden unterschiedliche Ansätze vorgestellt, die obige Forderungen unterschiedlich gut erfüllen:

versteckt allozentrisch: Die Drehwinkel α_i werden so lange aufsummiert, bis ihre Summe $\alpha = \sum \alpha_i > 45^\circ$ (oder $\alpha < -45^\circ$) wird (in unserer Beispielsequenz: $\alpha = 11.7^\circ + 38.2^\circ = 49.9^\circ$), nun wird eine Rechtskurve (Linkskurve) erkannt und α entsprechend korrigiert $\alpha \leftarrow \alpha - 90^\circ$ usw. Dieses Vorgehen funktioniert zwar, liefert aber nichts substantiell Neues, es entspricht genau dem Umweg über die allozentrische QMV-Darstellung mit `north` in Richtung des ersten MV, liefert also wieder den nicht erwünschten Zickzack.

Der einzige „Vorteil“ gegenüber dem Zwischenschritt über eine allozentrische QMV-Sequenz besteht darin, daß es bezogen auf die Gesamtsequenz rotationsinvariant ist, da die Richtung der Folgevektoren bezüglich des ersten MV und nicht bezüglich eines externen Koordinatensystems bestimmt wird. Dies löst aber die anderen oben dargestellten Probleme nicht.

Zurücksetzen von α: Statt nach Erkennen einer Kurve den Richtungswinkel α zu korrigieren, könnte man ihn auch einfach wieder auf 0 setzen, was der Vorstellung entspricht, daß die Kurve ja erkannt wurde und es danach potentiell wieder gerade weitergeht. Dies führt aber umgehend dazu, daß Kurven nicht mehr richtig repräsentiert werden, da die Erkennung ja schon nach jeweils 45° einsetzt, wie auch die Sequenz

`F F R F R F F L F R F R F R F F F F F F F`

zeigt: Die Kurve im Teilstück 2 mit 5 wird durch `F R F R` repräsentiert, was eigentlich eine 180°-Kurve nach rechts beschreibt. Andererseits kann der Akzeptanzbereich für `forward` nicht einfach von $[-45^\circ, 45^\circ]$ auf $[-90^\circ, 90^\circ]$ erweitert werden, um diesem Übersteuern zu begegnen, da sonst plötzlich die komplette vordere Halbebene als `forward` klassifiziert würde.

„Geraderücken“ von α: Obiges Beispiel hat gezeigt, daß das Zurücksetzen von α auf 0° zu radikal ist. Dies legt die Idee nahe, dieses Zurücksetzen langsamer

[22] Diese Meßungenauigkeit kann nun wieder abhängig von der Frequenz sein, mit der gemessen wird, dies kann aber berücksichtigt werden.

zu machen und über mehrere Vektoren zu verteilen. Hierzu wird ein „Begradigungswinkel“ $\breve{\alpha}$ festgelegt (z. B. $\breve{\alpha} = 5°$) und in jedem Schritt so von der aktuellen Winkelsumme abgezogen, daß die Bewegung „gerader“ wird: Damit berechnet sich die „geradegerückte“ Winkelsumme der ersten k Drehwinkel α_i zu: $\alpha = \text{straighten}(k, \breve{\alpha})$ mit

$$\text{straighten}(k, \breve{\alpha}) = \begin{cases} \text{skew}(\alpha_1, \breve{\alpha}) & \text{für } k = 1 \\ \text{skew}(\alpha_k + \text{straighten}(k-1, \breve{\alpha}), \breve{\alpha}) & \text{für } k > 1 \end{cases}$$

und

$$\text{skew}(\alpha, \breve{\alpha}) = \begin{cases} 0° & \text{für } |\alpha| \leq \breve{\alpha} \\ \alpha - \breve{\alpha} & \text{für } \alpha > \breve{\alpha} \\ \alpha + \breve{\alpha} & \text{für } \alpha < -\breve{\alpha} \end{cases}$$

Anschaulich gesagt wird der Betrag der Winkelsumme α in jedem Schritt um $\breve{\alpha}$ verkleinert (sofern er noch größer als $\breve{\alpha}$ ist, andernfalls eben auf $0°$ gesetzt). Wird die Winkelsumme α trotz dieser Abschwächung groß genug, daß der Sektor `forward` verlassen wird (im Beispiel $|\alpha| > 45°$), so wird eine Kurve erkannt und α angepaßt: für $\alpha > 45° : \alpha \leftarrow \alpha - (90° - 2\breve{\alpha})$ und für $\alpha < -45° : \alpha \leftarrow \alpha + (90° - 2\breve{\alpha})$.[23] Findet hier eine Wende statt (der Bereich `forward` wird nicht nach `left` oder `right`, sondern direkt nach `backward` verlassen), erfolgt die Anpassung dementsprechend durch $\alpha \leftarrow \alpha - 180°$ für $\alpha > 0$ und $\alpha \leftarrow \alpha + 180°$ für $\alpha < 0$.

Dieses Vorgehen hat zur Folge, daß im Mittel Kurven schwerer erkannt werden, da die Winkelsumme in jedem Schritt um den Winkel $\breve{\alpha}$ abgeschwächt wird. Durch diese Abschwächung wird aber andererseits erreicht, daß diagonale Sequenzen (Vektoren 13 mit 18) wie erwünscht als Folge von `forward`-QMVs erkannt werden.

Das beschriebene Verfahren erzeugt (mit $\breve{\alpha} = 5°$) die Sequenz

F F F R F F F F F F R F F R F F F F F F F,

die den Bewegungsverlauf wie gewünscht wiedergibt. Wählt man $\breve{\alpha} = 2°$, so liefert dies die Sequenz

F F R F F F F F F F R F F R F F F F F F F.

Durch den kleineren Korrekturwinkel wird die erste Kurve schneller erkannt, ohne daß der begradigende Effekt für die „Treppe“ (Vektoren 13 mit 18) verloren geht ($\breve{\alpha} = 1°$ ist allerdings nicht mehr ausreichend).

[23] Durch den Korrekturfaktor $80° = 90° - 2\breve{\alpha}$ wird die zuletzt erfolgte Begradigung zurückgenommen, da ja nun in die andere Richtung begradigt werden muß.

Zwischen Kurven und Treppen besteht hierbei ein Zusammenhang: Ohne „Geraderücken“ tritt der Treppeneffekt genau dann auf, wenn an einer Stelle des Bewegungsverlaufes die Winkelsumme α sehr groß ist (z. B. $\alpha = 40°$ oder $\alpha = -38°$) und dann mehrere leicht im Zickzack gegeneinander verdrehte Vektoren folgen, am Beispiel: $\alpha = 40°$, die nächsten beiden Vektoren haben den Drehwinkel $\alpha_k = 8°$, damit $\alpha \leftarrow \alpha + \alpha_k = 48°$, also wird eine Rechtskurve erkannt und $\alpha \leftarrow \alpha - 90° = -42°$ korrigiert, nun folgt eine leichte Abweichung in die andere Richtung: $\alpha_{k+1} = -4°$, damit ist im nächsten Schritt $\alpha \leftarrow \alpha + \alpha_{k+1} = -46°$, es wird eine Linkskurve erkannt und $\alpha \leftarrow \alpha + 90° = 44°$ korrigiert usw.

Ob eine (leichte) Zickzackbewegung diese Probleme macht, ist also abhängig vom aktuellen Wert der Winkelsumme α *vor* dem entsprechenden Teilstück, wenn sie nahe einer Grenze liegt, tritt der beschriebene Effekt auf.

Während eine Kurve durchfahren wird, wirkt das „Geraderücken“ um $\check{\alpha}$ lediglich abschwächend, Kurven werden also schwerer erkannt, die Winkelsumme α nach Durchlaufen der Kurve wird dabei aber nicht unbedingt positiv beeinflußt: nach Durchfahren einer (aus 6 Vektoren bestehenden) Kurve von insgesamt 130° beträgt die Winkelsumme ohne Begradigung $\alpha = 40°$, nach Durchfahren einer (ebenfalls aus 6 Vektoren bestehenden) Kurve von 140° mit einer Begradigung um $\check{\alpha} = 2°$ ebenso. Die Begradigung hat also Einfluß auf das Ergebnis (das ist klar), garantiert aber nicht, daß der Wert von α nach der Kurve „besser“ (also kleiner) ist.

Interessant ist nun vor allem, wie es *nach* der Kurve weitergeht: folgt nun ein geraderes Stück, so wird α in jedem Schritt um $\check{\alpha}$ verkleinert, bis α schließlich wieder um 0° schwankt, und nun stören kleine Zickzackbewegungen nicht mehr. Das heißt aber auch: Je dichter so ein kritischer Bereich auf eine Kurve folgt, desto eher bereitet er Probleme. Wählt man $\check{\alpha}$ zu groß, so werden Kurven zu schlecht erkannt, wählt man $\check{\alpha}$ zu klein, so dauert das „Geraderücken“ zu lange.

In der vorgestellten Form ist das Verfahren allerdings noch nicht unabhängig von Geschwindigkeit bzw. Meßgenauigkeit. Der Korrekturwinkel $\check{\alpha}$ wird auf ein Streckenstück mit k Vektoren $k - 1$ Mal angewandt. Wurde das gleiche Teilstück halb so schnell durchfahren oder doppelt so genau gemessen (also $2k$ Vektoren), $2k - 1$ Mal. Wird also sehr genau gemessen oder eine Kurve sehr langsam gefahren, kann sie nicht erkannt werden, da $\check{\alpha}$ zu stark begradigt.

Um ein geschwindigkeitsunabhängiges Verfahren zu bekommen, muß man lediglich $\check{\alpha}$ abhängig von der Vektorlänge wählen: die Sequenz

$$2\,\mathrm{m}\ (22°)\ 2\,\mathrm{m}\ (17°)\ 2\,\mathrm{m}\ (30°)\ \ldots$$

kann genauso gut in der Form

$$1\,\mathrm{m}\ (0^\circ)\ 1\,\mathrm{m}\ (22^\circ)\ 1\,\mathrm{m}\ (0^\circ)\ 1\,\mathrm{m}\ (17^\circ)\ 1\,\mathrm{m}\ (0^\circ)\ 1\,\mathrm{m}\ (30^\circ)\ \ldots$$

geschrieben werden. Da nun doppelt so viele Drehwinkel auftreten, kommt auch die Begradigung um $\breve{\alpha}$ doppelt so oft zur Anwendung. Verallgemeinert ist das gleichbedeutend damit, die Größe des Begradigungswinkels $\breve{\alpha}_{a_k,a_{k+1}}$ zwischen zwei Vektoren a_k und a_{k+1} direkt proportional zur Vektorlänge $|a_k| + |a_{k+1}|$ zu wählen, womit das Verfahren geschwindigkeitsunabhängig ist.[24]

Glättung der Bewegungsspur: Statt zu versuchen, leichte Richtungsschwankungen in der numerischen MV-Sequenz (Teilstück 13 mit 18) bei der Umwandlung in eine egozentrische QMV-Sequenz zu unterdrücken, kann auch zunächst die numerische Bewegungsspur so bearbeitet werden, daß in ihr keine solchen Effekte mehr auftreten: sie wird „geglättet“. Bei einer Glättung werden leichte Schwankungen im Bewegungsverlauf ausgeglichen, im Unterschied zur vorne erwähnten Generalisierung wird dabei die Anzahl der MV nicht verringert, sie bleibt konstant.

Aus dieser geglätteten Bewegungsspur kann die QMV-Sequenz dann problemlos gewonnen werden, ohne daß störende Effekte auftreten. Das Sliding-Window Verfahren (Kapitel 11.2) leistet so eine Glättung und macht die Bewegungsspur insgesamt runder, so daß die beschriebenen Zickzack-Effekte abgeschwächt werden. Allerdings stellt es nicht sicher, daß die Treppeneffekte völlig verschwinden, es macht sie nur unwahrscheinlicher. Noch bessere Ergebnisse liefert das in Kapitel 10.3 beschriebene Verfahren, bei dem alle Schwankungen unterhalb eines bestimmten Winkels unterdrückt werden, es arbeitet jedoch nicht inkrementell, eignet sich also nicht zur direkten Weiterverarbeitung der Kurve „on the fly“.

Im Ergebnis erfüllen also nur die letzten beiden Verfahren („Geraderücken“ von α und Glättung der Bewegungsspur) die oben aufgestellten Forderungen und sind damit zur Erzeugung egozentrischer QMV-Sequenzen aus numerischen MV-Sequenzen geeignet. Die Glättung der Bewegungsspur kann darüberhinaus auch ein sinnvoller Schritt bei der Erzeugung allozentrischer QMV-Sequenzen sein, da auch bei diesen die beschriebenen Zickzackeffekte auftreten können.[25]

[24] Das stimmt nicht ganz, da die oben beschriebene Sequenz bei genauerer Messung natürlich nicht aus einer Folge von jeweils 2 gleichgerichteten aufeinanderfolgenden Vektoren bestehen wird. Der hierbei auftretende Fehler ist allerdings minimal.

[25] wobei dieser Effekt im allozentrischen Fall weniger stört, da hier klar ist, daß mit einem extern vorgegebenen Koordinatensystem gearbeitet wird.

3.4. Rechnen auf QMV-Sequenzen

Im letzten Kapitel ist die Repräsentation von QMV-Sequenzen in unterschiedlichen Referenzrahmen und ihre Generierung aus numerischen Bewegungsdaten beschrieben. Die qualiative *Darstellung* einer Bewegungsspur allein ist in den meisten Fällen unzureichend, man möchte auf QMV-Sequenzen *rechnen* können.

Ein QMV besteht wie vorne beschrieben aus einer symbolischen Entfernungs- und Richtungsangabe `<E,R>` mit[26]

$$E \in \mathfrak{E} = \{\texttt{zero}, \texttt{very-close}, \texttt{close}, \texttt{medium-distance}, \texttt{far}, \texttt{very-far}\},$$
$$R \in \mathfrak{R} = \{\texttt{zero}, \texttt{north}, \texttt{east}, \texttt{south}, \texttt{west}\}.$$

Jeder dieser Bezeichner steht dabei für ein bestimmtes Intervall, jede Entfernung von 4 bis 16 m wird durch den Bezeichner `close` beschrieben usw.

Ist nun eine QMV-Sequenz wie `<close forward> <far forward>` oder `<close forward>`^{20} gegeben, lautet die Frage: was bedeutet das eigentlich, *wie weit* ist er denn nun gefahren? Mit anderen Worten: wie addiert man `close` und `far`?

3.4.1. Symbolisches Rechnen

In [CDFH97] beschreiben Clementini, Di Felice und Hernández die symbolische Komposition von qualitativen Entfernungen anhand von Kompositionsmatrizen. Darauf aufbauend definiere ich im Folgenden eine Addition auf QMV.

Ich betrachte zunächst eine Richtung, d. h. eine Dimension des Raumes (wie z. B. eine Eisenbahn auf einem Gleis ohne Weichen) und erweitere die Betrachtung schließlich auf beliebige Richtungen: Sind die Entfernungen wie vorne verlangt angeordnet, also

$$\texttt{zero} \leq \texttt{very-close} \leq \texttt{close} \leq \texttt{medium-distance} \leq \texttt{far} \leq \texttt{very-far},$$

so lassen sich schon erste Schlüsse ziehen:

$$\forall x, y \in \mathfrak{E}: \quad x + y \geq x, \quad x + y \geq y.$$

Am Beispiel:

$$\left.\begin{array}{rcl} \texttt{very-close} + \texttt{Y} & \geq & \texttt{very-close} \\ \texttt{X} + \texttt{close} & \geq & \texttt{close} \end{array}\right\} \Rightarrow \texttt{very-close} + \texttt{close} \geq \texttt{close}$$
$$\Longrightarrow \texttt{very-close} + \texttt{close} \in \{\texttt{close}, \texttt{medium-distance}, \texttt{far}, \texttt{very-far}\}.$$

[26] Ich behalte im Folgenden für die Beispiele die schon vorne eingeführte Richtungs- und Entfernungsdomäne bei, ebenso die in Tabelle 3.1 aufgeführten Intervalle. Eine Aufteilung in mehr (oder weniger) Richtungen und Entfernungen sowie eine Änderung der Intervallgrenzen ist jederzeit möglich.

Dieses Ergebnis ist sehr ungenau, was daran liegt, daß über die Eigenschaften der einzelnen Intervalle weiter nichts bekannt ist. [CDFH97] fordern daher, daß die Intervallängen monoton steigen:

$$|\underbrace{(d_0,d_1]}_{\texttt{vc}}| \leq |\underbrace{(d_1,d_2]}_{\texttt{close}}| \leq |\underbrace{(d_2,d_3]}_{\texttt{meddist}}| \leq |\underbrace{(d_3,d_4]}_{\texttt{far}}| \leq |\underbrace{(d_4,d_5]}_{\texttt{very-far}}|,$$

damit gilt $\texttt{very-close} + \texttt{very-close} \leq \texttt{close}$

und somit $\texttt{very-close} + \texttt{very-close} \in \{\texttt{very-close}, \texttt{close}\}$,

analog für alle anderen Entfernungen.[27]

Die Addition zweier Entfernungen liefert im Ergebnis also eine Menge von Entfernungen. Für die Verarbeitung von QMV-Sequenzen benötigen wir die Addition von mehr als zwei Entfernungen, man muß also mit dem Ergebnis der Addition weiterrechnen können. Wir erweitern sie daher von einer Abbildung zweier Entfernungen auf eine Menge von Entfernungen

$$\begin{array}{rcl} + : \mathfrak{E} \times \mathfrak{E} & \rightarrow & \mathcal{P}(\mathfrak{E}), \\ x + y & \rightarrow & E \end{array}$$

zu einer Abbildung von Mengen von Entfernungen auf Mengen von Entfernungen:

$$\begin{array}{rcl} \oplus : \mathcal{P}(\mathfrak{E}) \times \mathcal{P}(\mathfrak{E}) & \rightarrow & \mathcal{P}(\mathfrak{E}), \\ A \oplus B & \rightarrow & \bigcup\limits_{\substack{x \in A \\ y \in B}} \{x + y\} \end{array}$$

Die Addition $x + y$ zweier einzelner Entfernungen $x, y \in \mathfrak{E}$ wird dabei als Addition $\{x\} \oplus \{y\}$ zweier einelementiger Mengen modelliert. Da $\oplus$ auf $\mathcal{P}(\mathfrak{E})$ abgeschlossen, assoziativ und kommutativ ist, ist $\langle \mathcal{P}(\mathfrak{E}), \oplus \rangle$ ein abelscher Monoid mit neutralem Element $\{\texttt{zero}\}$.

Auch bei der Beschränkung auf eine räumliche Dimension sind Bewegungen in 2 Richtungen möglich (die Eisenbahn kann auf ihrem Gleis vor und zurück fahren). Durch Erweiterung der Entfernungsdomäne auf

$$\begin{aligned} \mathfrak{E}_{\pm} = \{&-\texttt{very-far}, -\texttt{far}, -\texttt{medium-distance}, -\texttt{close}, -\texttt{very-close}, \texttt{zero}, \\ &\texttt{very-close}, \texttt{close}, \texttt{medium-distance}, \texttt{far}, \texttt{very-far}\} \end{aligned}$$

bekommen wir die Subtraktion von Entfernungen. Die Intervalle für die negativen Entfernungen werden dabei direkt aus ihren positiven Pedanten erzeugt ($-\texttt{close}$ entspricht dem Intervall $[-d_2, -d_1)$). Damit gelten die oben genannten Monotonieforderungen jeweils nur für die Teilmenge der positiven oder (mit umgedrehtem Vorzeichen) der negativen Intervalle.

[27] In obiger Arbeit wird schließlich noch Summenmonotonie gefordert, d. h. jedes Intervall ist mindestens so groß wie die Summe der Intervallängen seiner Vorgänger ($\forall i : d_i \geq 2d_{i-1}$ leistet dies).

Das negative Vorzeichen entspricht dabei einer Umdrehung der Bewegungsrichtung, es gilt also

$$\texttt{<far north>} = \texttt{<-far south>},$$

analog für alle anderen Entfernungen und für `east` und `west`.

$\langle \mathcal{P}(\mathfrak{E}_{\pm}), \oplus \rangle$ ist *keine* Gruppe, da die meisten Elemente kein Inverses besitzen, so gilt z. B.

$$\{\texttt{close}\} \oplus \{-\texttt{close}\} = \{-\texttt{close}, -\texttt{very-close}, \texttt{zero}, \texttt{very-close}, \texttt{close}\}.$$

Im Allgemeinen liefert die Addition zweier Entfernungsmengen eine Entfernungsmenge mit mehr Elementen zurück, wie das Beispiel zeigt. Dies muß nicht so sein, so ist

$$\{\texttt{very-close}, \texttt{close}, \texttt{medium-distance}\} \oplus \{\texttt{very-far}\} = \{\texttt{very-far}\},$$

doch sind dies Ausnahmen. In Extremfällen wird die Entfernung völlig unbestimmt, so gilt[28]

$$\{\texttt{very-far}\} \oplus \{-\texttt{very-far}\} = \mathfrak{E}_{\pm}.$$

Um den möglicherweise stark unterschiedlichen Intervallgrößen Rechnung zu tragen, führen [CDFH97] ein Absorptionsgesetz ein: Werden eine sehr große und eine sehr kleine Entfernung addiert, so absorbiert die große Entfernung die kleine schlicht: $\texttt{far} + \texttt{very-close} = \texttt{far}$. Die Anwendung dieser Regel zerstört jedoch die Assoziativität von $\oplus$:

$$\begin{aligned}
&((\{\texttt{far}\} \oplus \{\texttt{very-close}\}) \oplus \{\texttt{very-close}\}) \oplus \{\texttt{very-close}\} = \\
&= (\{\texttt{far}\} \oplus \{\texttt{very-close}\}) \oplus \{\texttt{very-close}\} = \\
&= \{\texttt{far}\} \oplus \{\texttt{very-close}\} = \\
&= \{\texttt{far}\},
\end{aligned}$$

$$\begin{aligned}
&\{\texttt{far}\} \oplus (\{\texttt{very-close}\} \oplus (\{\texttt{very-close}\} \oplus \{\texttt{very-close}\})) = \\
&= \{\texttt{far}\} \oplus (\{\texttt{very-close}\} \oplus \{\texttt{very-close}, \texttt{close}\}) = \\
&= \{\texttt{far}\} \oplus \{\texttt{very-close}, \texttt{close}, \texttt{medium-distance}\} = \\
&= \{\texttt{far}, \texttt{very-far}\}.
\end{aligned}$$

Die hier beschriebene Addition auf Entfernungen ($\oplus$) liefert geeignete Ergebnisse bei der Komposition weniger Entfernungen ([CDFH97] gehen auf die Addition von mehr als zwei Distanzen gar nicht ein). Dies ist für die Beschreibung vieler räumlicher Konfigurationen hinreichend, zum einen sind hier meist

[28] unter der Annahme der Summenmonotonie für die Teilmenge der positiven bzw. negativen Intervalle

nur wenige Entfernungen zu addieren, zum anderen (und das ist der wichtigere Punkt) wird der Unbestimmtheit von Entfernungsangaben hier durch externe Constraints begegnet: Wenn das Bücherregal an der Wand steht, dicht rechts daneben (`very-close`) der Gummibaum und weit rechts von diesem (`far`) die Kommode, dann muß dieses ganze Szenario auch noch in das Zimmer passen, die Räumlichkeiten sind also begrenzt, und die Gegenstände selbst haben zusätzlich auch eine Ausdehnung. Die beschriebene Konfiguration läßt also aufgrund der externen Constraints (Wandlänge, Breite von Bücherregal, Gummibaum und Kommode) wenig Unklarheiten, unabhängig davon, wie genau oder ungenau die Addition von `far` und `very-close` erfolgt.

Für die Verarbeitung von QMV-Sequenzen gilt dies hingegen nicht. Sie zeichnen sich dadurch aus, daß sie aus einer Vielzahl qualitativer Richtungs- und Entfernungsangaben bestehen, die nicht durch irgendwelche Constraints stark eingeschränkt sind (sofern sie Bewegungen auf freiem Feld beschreiben, eine Bewegung durch die Gänge eines Bürogebäudes beispielsweise ist weit stärker eingeschränkt). Um z. B. die zurückgelegte Entfernung zu bestimmen, können hier Summen mit hundert und mehr Summanden auftreten. Das Absorptionsgesetz liefert hier schnell falsche Ergebnisse: während $\texttt{far} - \texttt{very-close} = \texttt{far}$ noch einsichtig und $\texttt{far} - \texttt{very-close}^3 = \texttt{far}$ noch vertretbar ist, ist $\texttt{far} - \texttt{very-close}^{50} = \texttt{far}$ schlicht falsch. Dies ist ein für QMV-Sequenzen durchaus relevanter Fall, die Sequenz `<far north> <very-close south>`^{50} beschreibt eine schnelle Fahrt nach Norden gefolgt von einem langsamen Rückweg nach Süden.

Doch auch ohne Absorption ist die symbolische Addition auf Bewegungen nicht brauchbar, da die Menge möglicher Entfernungen wie oben gezeigt schnell gegen $\mathfrak{E}_{\pm}$ strebt.

Die symbolische Addition ist für mehr als zwei Summanden schlechter, als sie sein müßte, wie folgende Überlegung zeigt: nach Definition von $\oplus$ gilt

$$\begin{aligned}&\{\texttt{close}\} \oplus \{\texttt{close}\} \oplus \{\texttt{close}\} = \\ &= \{\texttt{medium-distance}, \texttt{close}\} \oplus \{\texttt{close}\} = \\ &= \{\texttt{far}, \texttt{medium-distance}, \texttt{close}\}.\end{aligned}$$

Gegen den ersten Schritt ($\{\texttt{close}\} \oplus \{\texttt{close}\} = \{\texttt{medium-distance}, \texttt{close}\}$) ist nichts einzuwenden, das Ergebnis der Berechnung ist aber unnötig ungenau: Drei Mal `close` ergibt nicht unbedingt `far`. `close` entspricht dem Intervall $(d_1, d_2]$, `far` dem Intervall $(d_3, d_4]$. $\texttt{close}^3$ entspricht damit dem Intervall $(3d_1, 3d_2]$. Wenn $3d_2 \leq d_3$, so gilt:

$$\{\texttt{close}\} \oplus \{\texttt{close}\} \oplus \{\texttt{close}\} = \{\texttt{medium-distance}, \texttt{close}\}.$$

Die Ergebnismenge wurde oben unnötig groß, weil vom ersten auf den zweiten Schritt die Information verloren ging, *wie weit* zwei Mal `close` in das Intervall

`medium-distance` hineinragt, ob also ein drittes `close` ausreicht, um über die obere Intervallgrenze von `medium-distance` zu kommen. Obwohl die einzelnen Schritte in sich korrekt sind, ist das Gesamtergebnis nicht zufriedenstellend.

Statt also die Summe über n Summanden iterativ auf der Addition von zwei Summanden aufzubauen, ist es besser, mittels Intervallarithmetik direkt eine Summenfunktion für n Summanden zu definieren. Seien l_X und r_X die linke und rechte Intervallgrenze des zur Entfernung $X \in \mathfrak{E}$ gehörenden Intervalls I_X. Ich definiere[29]

$$\begin{aligned} \sum_n : \mathfrak{E}^n &\rightarrow \mathcal{P}(\mathfrak{E}) \\ \sum_{i=1}^{n} X_i &\rightarrow \{E \in \mathfrak{E} \mid (I_E \cap (\sum_{i=1}^{n} l_i, \sum_{i=1}^{n} r_i)) \neq \emptyset\}. \end{aligned}$$

Die Erweiterung auf Mengen erfolgt analog zu oben:

$$\begin{aligned} \sum : (\mathcal{P}(\mathfrak{E}))^n &\rightarrow \mathcal{P}(\mathfrak{E}) \\ \sum_{i=1}^{n} M_i &\rightarrow \{E \in \mathfrak{E} \mid (I_E \cap \bigcup_{X_i \in M_i} \{(\sum_{i=1}^{n} l_i, \sum_{i=1}^{n} r_i)\}) \neq \emptyset\}. \end{aligned}$$

Dies in dieser Form durchzurechnen, würde zu einer kombinatorischen Explosion führen, ist aber in der Praxis nicht nötig: Jede durch Addition von zwei Entfernungen entstandene Menge von Entfernungen läßt sich durch ein einziges Intervall darstellen, da alle enthaltenen Entfernungen benachbart sind. Damit gilt für $M \subset \mathfrak{E} : I_M = \bigcup_{X_i \in M} I_{X_i}$. Hierbei gilt, daß man möglichst durchgängig mit der Intervalldarstellung rechnet und erst am Ende zur symbolischen Darstellung zurückkehrt, da hier Informationsverlust auftritt.

Leider ist auch die Intervallarithmetik für die Verarbeitung von QMV-Sequenzen noch zu unbestimmt, und dies, obwohl sich die Betrachtungen bislang auf eine Dimension des Raumes beschränkt haben. Nimmt man nun noch eine 2. Dimension hinzu (Bewegungen in der Ebene), zieht also unterschiedliche Richtungen in Betracht, so tritt neben die diskutierte Unbestimmtheit in der Entfernung auch noch die in der Richtung. Die Sequenz `<close north> <close south>` addiert liefert nicht nur die möglichen QMV `<close north>`, `<very-close north>`, `<zero zero>`, `<very-close south>` und `<close south>`, sondern darüberhinaus noch `<medium-distance west>`, `<close west>`, `<very-close west>`, `<very-close east>`, `<close east>` und `<medium-distance east>`, am Beispiel (Intervallgrenzen nach Tabelle 3.1):

$$\underbrace{\texttt{<close north>}}_{\langle 15\,\mathrm{m}, 40^\circ\rangle} + \underbrace{\texttt{<close south>}}_{\langle 14\,\mathrm{m}, 132^\circ\rangle} = \underbrace{\texttt{<medium-distance east>}}_{\langle 19\,\mathrm{m}, 3.27^\circ\rangle}$$

[29] Ich schneide hier mit dem beidseitig offenen Intervall. Dies führt dazu, daß lediglich berührende Intervalle nicht erfaßt werden, was meines Erachtens sinnvoll ist.

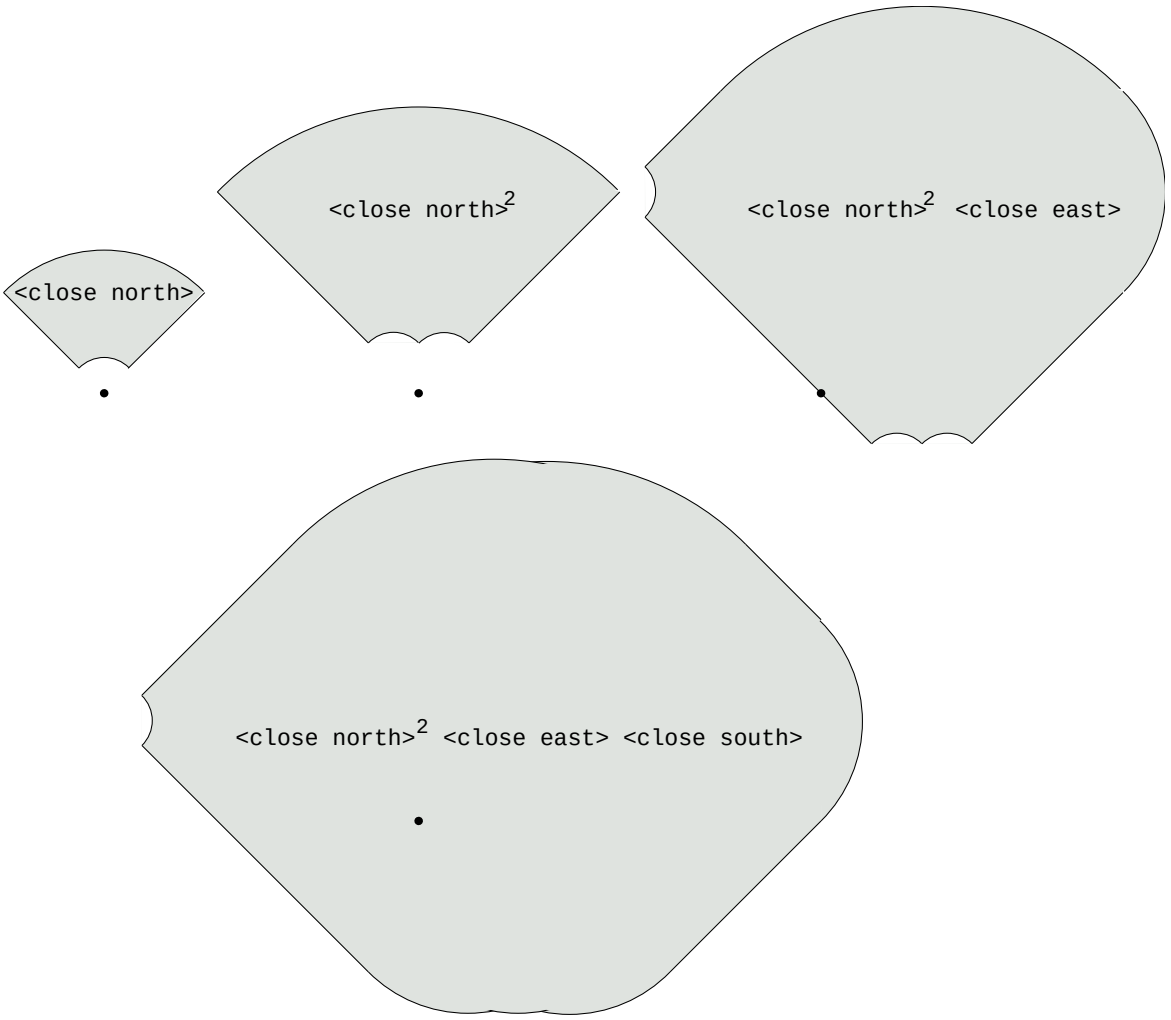


Abbildung 3.12.: Darstellung der möglichen Zielbereiche einer QMV-Sequenz

Auf den ersten Blick verwundert, daß hier in `east-west`-Richtung eine größere Entfernung (`medium-distance`) auftreten kann, als in `north-south`-Richtung überhaupt gefahren wurde (`close`). Dadurch, daß bei nur vier Richtungen jede Richtung einen Winkel von 90° umfaßt und somit nach rechts und links Abweichungen um bis zu 45° möglich sind, kann diese seitliche Abweichung so groß werden. Bei einer Richtungsdomäne mit acht Richtungen kann dieser Effekt nicht mehr auftreten, hier beträgt die maximale seitliche Abweichung 0.76 der einfachen Entfernung.

Wie schnell das Rechnen auf QMV-Sequenzen in der Ebene mittels Intervallarithmetik zu fast völligem Informationsverlust führt, zeigt Abbildung 3.12 am Beispiel der kurzen Sequenz

`<close north>2 <close east> <close south>.`

Die grauen Flächen zeigen die mögliche Endposition nach dem ersten, zweiten, dritten und vierten Schritt.

Schon nach nur vier Schritten kann sich das beschriebene Objekt irgendwo in einer großen Wolke um den Startpunkt aufhalten. Was dies für eine QMV-Sequenz mit mehreren hundert QMV bedeutet, sollte klar sein.

3.4.2. QMV-Algebra

Exaktes Rechnen auf QMV-Sequenzen führt also schnell zu großer Unbestimmtheit und ist damit in der Praxis nicht brauchbar. Die im Folgenden vorgestellte Algebra[30] macht daher einige einschränkende Annahmen, liefert dafür aber in der Praxis brauchbare Ergebnisse.

Wahl der Intervallgrenzen

Die erste Einschränkung betrifft die Wahl der Intervallgrenzen für die Entfernungsdomäne. Wählt man diese geschickt, so lassen sich unterschiedliche qualitative Entfernungsangaben ineinander umrechnen. Die Entfernung `close` beschreibt das Akzeptanzintervall $(d_1, d_2]$ (Tabelle 3.1), die Entfernung `medium-distance` das Intervall $(d_2, d_3]$. Wählt man nun die Intervallgrenzen dergestalt, daß für ein $n > 1$ gilt: $d_2 = nd_1$ und $d_3 = nd_2$, so gilt für jede Entfernung $d \in (d_1, d_2] : nd \in (d_2, d_3]$, und damit $\texttt{close}^n = \texttt{medium-distance}$.

Dies funktioniert für alle Intervalle bis auf das erste. `zero` modelliert Stillstand, spannt also kein Intervall auf, daher besitzt `very-close` 0 als untere Intervallgrenze.[31] Damit besitzt `very-close` das Akzeptanzintervall $(0, d_1]$, hier gilt also nicht $nd_0 = d_1$. In der Praxis ist der hierdurch entstehende kleine Fehler im Verhältnis zur Unbestimmtheit durch die qualitative Darstellung verschwindend gering. Das letzte Intervall macht hingegen keine Probleme. Es ist zwar in Tabelle 3.1 mit (d_4, ∞) angegeben, in der Praxis ist die maximale Entfernung aber durch äußere Umstände beschränkt.

Ausgehend von einem „virtuellen“ d_0 als Startwert und einem festgelegten Faktor n bestimmen sich die Intervallgrenzen also wie folgt:

$$d_1 = nd_0, \quad d_2 = nd_1 = n^2 d_0, \quad d_3 = nd_2 = n^3 d_0, \quad d_4 = nd_3 = n^4 d_0.$$

Damit lassen sich alle (positiven) Entfernungen ineinander umrechnen. Es gilt: $\texttt{far} + \texttt{close} = \texttt{close}(n^2) + \texttt{close} = \texttt{close}(n^2 + 1)$.

Um „schöne“ Vielfache zu erhalten, sollte $n \in \mathcal{N}$ gewählt werden, für $n \geq 2$ ist die vorne erwähnte Forderung der Summenmonotonie erfüllt (jedes Intervall ist

[30] Erstmal von mir beschrieben in [Ste98].

[31] außer man möchte modellieren, daß sehr langsame Bewegungen nicht wahrgenommen werden, in diesem Fall beschreibt d_0 die Wahrnehmungsschwelle.

größer als die Summe seiner Vorgänger), die Beispielwerte in Tabelle 3.1 wurden mit $n = 4$ (und $d_0 = 1\,\mathrm{m}$) bestimmt.

Umgekehrt läßt sich aus einem gegebenen Vielfachen auch die zugehörige Entfernung bestimmen. Sei die Entfernung `close`^{k} gegeben. Die durch diese Angabe repräsentierten numerischen Entfernungen liegen im Intervall $(kd_1, kd_2]$. k läßt sich in der Form $k = an^b$ (mit $a < n$) schreiben. Für $a = 1$ ist die resultierende Entfernung eindeutig: für $k = n$ ist dies genau `medium-distance`, für $k = n^2$ `far` und für $k = n^3$ `very-far`. Für $a > 1$ überlappt das resultierende Intervall mit den Akzeptanzintervallen zweier benachbarter Entfernungen. Es liegt nahe, nun die Entfernung zu wählen, mit deren Akzeptanzbereich die Überlappung größer ist. Das Intervall $(kd_1, kd_2] = (an^b d_1, an^{b+1} d_1] =$ mit $a > 1$ überlappt für $b = 0$ mit den Akzeptanzintervallen von `close` $(d_1, nd_1]$ und `medium-distance` $(nd_1, n^2 d_1]$, für $b = 1$ mit `medium-distance` und `far` usw., allgemein: mit $(n^b d_1, n^{b+1} d_1]$ und $(n^{b+1} d_1, n^{b+2} d_1]$, wobei die Überlappung mit dem ersten Intervall $(an^b d_1, n^{b+1} d_1]$ und mit dem zweiten Intervall $(n^{b+1} d_1, an^{b+1} d_1]$ beträgt.

Für $a = \frac{2n}{n+1}$ sind beide Intervalle gleich groß. Für eine Entfernung $x \in \mathfrak{E}$ gilt damit:[32]

$$x^k = \begin{cases} \texttt{zero} & \text{für} \quad k = 0 \\ x & \text{für} \quad k \in \left[1, \frac{2n}{n+1}\right) \\ \mathrm{succ}(x) & \text{für} \quad k \in \left[\frac{2n}{n+1}, \frac{2n^2}{n+1}\right) \\ \mathrm{succ}(\mathrm{succ}(x)) & \text{für} \quad k \in \left[\frac{2n^2}{n+1}, \frac{2n^3}{n+1}\right) \\ \ldots \end{cases}$$

Diese Umwandlung ist in den meisten Fällen fehlerbehaftet und beinhaltet Informationsverlust (da ja eins von zwei möglichen Intervallen ausgewählt wird). Daher sollte möglichst durchgängig mit Vielfachen der kleinsten qualitativen Einheit gerechnet und erst am Ende aller Berechnungen die Rücktransformation durchgeführt werden.

Gleichheitsaxiom

Obige Betrachtungen beschränken sich auf die Umrechnung der unterschiedlichen qualitativen Entfernungen ineinander, wobei alle Entfernungen gleichgerichtet sind, die Betrachtung bleibt auf $\mathfrak{E}$ beschränkt. Die Erweiterung auf $\mathfrak{E}_\pm$ fordert die Subtraktion von Richtungen, mit anderen Worten: welche Entfernung zu seinem Startpunkt hat ein Fahrzeug nach `<far north> <far south>`? Der Versuch exakter Rechnung mittels Intervallarithmetik führt, wie in Kapitel 3.4.1 gezeigt, schnell zu völliger Unbestimmtheit und ist damit ebenso exakt wie unbrauchbar. Ich definiere

[32] wobei $\mathrm{succ}(x)$ die nächst größere Entfernung liefert.

$$\texttt{very-close} - \texttt{very-close} = \texttt{zero} \qquad \text{(Gleichheitsaxiom)}$$

Für alle anderen Richtungen $x \in \mathfrak{E}_{\pm}$ gilt damit ebenfalls: $x - x = \texttt{zero}$. Weiterhin gilt:

$$\texttt{<very-close north>} \oplus \texttt{<very-close south>} = \texttt{<zero zero>},$$

analog für `east` und `west`. Insbesondere wird also hier angenommen, daß eine Bewegung in `north-south`-Richtung keine `east`- oder `west`-Verschiebung zur Folge hat und umgekehrt.

Diese Annahmen stellen selbstverständlich eine Verfälschung dar. Es ist äußerst unwahrscheinlich, daß die Bewegung

```
<close north>6 <close east>10 <close north>5
<close west>3 <close south>11 <close west>7
```

wieder genau an ihrem Startpunkt endet, obgleich sowohl die Addition aller QMV in `north-south`- wie in `east-west`-Richtung im Ergebnis `<zero zero>` liefert. Dennoch sind diese Annahmen notwendig und sinnvoll. Notwendig, weil der Versuch exakten Rechnens wie gezeigt schnell zu völliger Unbestimmtheit führt, sinnvoll, weil das Ergebnis richtig interpretiert brauchbar ist. Die Addition einer ganzen Reihe qualitativer Entfernungen kann prinzipbedingt keine exakte Gesamtentfernung liefern, das Ergebnis darf demzufolge auch nicht als exakte Angabe interpretiert werden. Der Versuch, durch Entfernungsaddition auf einer QMV-Sequenz die aktuelle Position eines Fahrzeugs zu bestimmen, ist zum Scheitern verurteilt.[33] Die Bestimmung exakter Positionen und Entfernungen ist aber sicher auch nicht Ziel qualitativer Arithmetik, dazu ist die qualitative Darstellung einfach zu ungenau.

3.4.3. QMV-Vektorraum

Vier Richtungen

Die oben eingeführten Rechenoperationen behandeln QMV in `north-south`- und `east-west`-Richtung getrennt voneinander, diese Richtungen sind voneinander unabhängig. Wählt man nun

$$e^1 = \texttt{<very-close east>} \quad \text{und} \quad e^2 = \texttt{<very-close north>}$$

als Basis, so erhält man einen Vektorraum $\mathfrak{V}$, in dem jede Summe beliebiger QMV als Vektor darstellbar ist. Jede Entfernung wird als Vielfaches von `very-close` ausgedrückt, die Richtungen durch $\texttt{west} = -\texttt{east}$ und $\texttt{south} = -\texttt{north}$, wobei gilt:

$$\forall x \in \mathfrak{R}: \quad \texttt{<very-close } x\texttt{>}^{-k} = \texttt{<very-close } -x\texttt{>}^{k}.$$

Grundsätzlich spricht nichts gegen $k \in \mathcal{R}$, auch wenn man häufig $k \in \mathcal{Z}$ (also ganzzahlige Vielfache) wählen wird.

[33] man könnte mittels Intervallarithmetik lediglich das Gebiet bestimmen, in dem man suchen muß.

Acht und mehr Richtungen

Besteht die Richtungsdomäne aus mehr als vier Richtungen, so ist die lineare Unabhängigkeit nicht mehr gegeben. Wie oben werden zwei (optimalerweise zueinander senkrechte) Richtungen als Basis gewählt und die dazwischenliegenden Richtungen als Kombination selbiger ausgedrückt, so gilt für acht Richtungen mit gleichgroßen Akzeptanzbereichen (nach Pythagoras):

$$\texttt{<very-close ne>}^{k} = \texttt{<very-close north>}^{k'} \oplus \texttt{<very-close east>}^{k'}$$

mit $k' = \frac{k}{\sqrt{2}}$. Hier ist die Wahl von $k \in \mathcal{R}$ auf jeden Fall sinnvoll.

Umwandlung in QMV

Das Ergebnis einer Rechnung im QMV-Vektorraum ist in der Regel kein QMV. Die Addition der QMV (mit $n = 4$)

$$\begin{aligned}
\texttt{<very-close south>}^{4} &\equiv \begin{pmatrix} 0 \\ -4 \end{pmatrix} \\
\texttt{<very-close east>}^{5} &\equiv \begin{pmatrix} 5 \\ 0 \end{pmatrix} \\
\texttt{<close north>}^{3} &\equiv \begin{pmatrix} 0 \\ 12 \end{pmatrix} \\
\texttt{<very-close west>}^{2} &\equiv \begin{pmatrix} -2 \\ 0 \end{pmatrix} \\
\begin{pmatrix} 0 \\ -4 \end{pmatrix} + \begin{pmatrix} 5 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 12 \end{pmatrix} + \begin{pmatrix} -2 \\ 0 \end{pmatrix} &= \begin{pmatrix} 3 \\ 9 \end{pmatrix}
\end{aligned}$$

liefert im Ergebnis den Vektor

$$\begin{pmatrix} 3 \\ 9 \end{pmatrix} \equiv \texttt{<very-close north>}^{9} \oplus \texttt{<very-close east>}^{3}.$$

Um im Ergebnis wieder einen QMV zu erhalten, kommen die vorne vorgestellten Akzeptanzbereiche zum Einsatz. Das Verhältnis der Komponenten des Vektors liefert wie im Numerischen seine Richtung (den Winkel), zu welcher die zugehörige qualitative Richtung bestimmt wird. Die Länge des Vektors (in Vielfachen von `very-close`) berechnet sich wie oben durch $\sqrt{3^2 + 9^2}$, die Bestimmung der zugehörigen qualitativen Entfernung wurde oben schon beschrieben.

3.4.4. Graphische Darstellung

Die Darstellung einer QMV-Sequenz in Textform ist unübersichtlich und vermittelt dem Leser vor allem bei langen Sequenzen keine gute Vorstellung über die Form

der beschriebenen Bewegung: Sie ist unanschaulich. Für die graphische Darstellung von QMV-Sequenzen bietet sich der oben beschriebene QMV-Vektorraum an. `north-south` und `east-west` liefern die Achsen eines Koordinatensystems mit `very-close` als Einheit. Eine in dieses Koordinatensystem eingezeichnete QMV-Sequenz zeigt dem Betrachter auf einen Blick die Form der Bewegung.

Eine QMV-Sequenz resultiert vielfach aus einer numerisch gemessenen Bewegungsspur. Um diesen numerischen Polygonzug mit der zugehörigen QMV-Sequenz zu vergleichen, möchte man beide in einem gemeinsamen Koordinatensystem darstellen, hierzu sind numerische Repräsentanten der einzelnen QMV nötig. Es liegt nahe, diese jeweils aus der Mitte des zugehörigen Akzeptanzbereiches zu wählen, damit zeigt ein Repräsentant der Richtung `north` in Richtung $0°$, `east` in Richtung $90°$ usw. Analog wird die Entfernung `close` (mit zugehörigem Intervall $(d_1, d_2] = (d_1, nd_1]$) durch die numerische Distanz $\frac{d_1+d_2}{2} = \frac{n+1}{2}d_1$ repräsentiert usw. Dabei gilt auch für den numerischen Repräsentanten einer Entfernung:

$$\texttt{close}^n \quad \equiv \quad n\,\frac{n+1}{2}\,d_1 = \frac{n+1}{2}\,d_2 \quad \equiv \quad \texttt{medium-distance}.$$

Dies gilt allgemein für alle Entfernungen. Als Repräsentant des kleinsten Intervalls `very-close` wird selbstverständlich $\frac{n+1}{2}d_0$ mit dem schon oben eingeführten „virtuellen“ d_0 gewählt, damit obige Beziehung auch hier gültig ist, auch wenn das Akzeptanzintervall zu `very-close` $(0, d_1]$ umfaßt. Die größte Entfernung `very-far` besitzt analog dazu den numerischen Repräsentanten $\frac{n+1}{2}d_4$.

Aufgrund dieser Beziehung ist es möglich, direkt auf den numerischen Repräsentanten einer QMV-Sequenz zu arbeiten, und diese damit graphisch in einem Koordinatensystem darzustellen. Abbildung 3.13 zeigt die graphische Darstellung des schon von vorne (Abb. 3.4 und 2.1) bekannten Bewegungsverlaufes und der zugehörigen QMV-Sequenz. Die Form des Bewegungsverlaufes wird durch die QMV-Sequenz korrekt wiedergegeben, nicht aber die Positionen der Eckpunkte. Die graphische Repräsentation der QMV-Sequenz ist insgesamt wesentlich größer als der numerische Originalpolygonzug. Dies liegt vor allem daran, daß der erste und letzte Vektor gerade so eben in das Intervall `medium-distance` fallen, aber an dessen unterem Ende liegen.

3.4.5. Anwendung der Algebra

Die vorgestellte QMV-Algebra ist ein mächtiges Werkzeug zur Verarbeitung qualitativer Bewegungssequenzen. Sie erlaubt die Anwendung des bekannten Instrumentariums der analytischen Geometrie, wodurch eine große Anzahl Algorithmen verfügbar werden, so z. B. die im zweiten Teil der Arbeit vorgestellten Generalisierungsalgorithmen.

Das Rechnen in der QMV-Algebra ist aufgrund des Gleichheitsaxioms[34] feh-

[34] und aufgrund der Rückwandlung von Vektoren des QMV-Vektorraumes in QMV

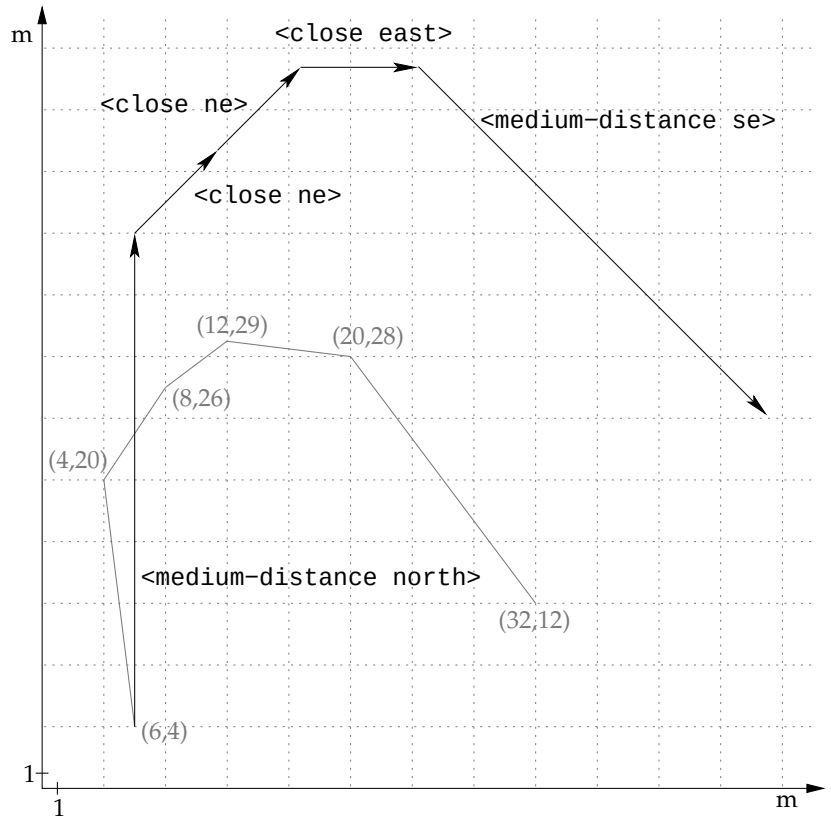


Abbildung 3.13.: Graphische Repräsentation einer QMV-Sequenz

lerbehaftet und ungenau. Das bedeutet nicht, daß sie keine brauchbaren Ergebnisse liefert sondern lediglich, daß diese Fehler bei der Interpretation der Ergebnisse berücksichtigt werden müssen. Wie schon oben erwähnt, kann die Addition der QMV einer mehrere Hundert Vektoren langen QMV-Sequenz sicher keine verläßliche Auskunft über die Entfernung (und Richtung) vom Start- zum Endpunkt der Bewegung geben.

Ziel des Rechnens im QMV-Vektorraum kann also nicht sein, Aussagen über die globale Position eines bewegten Objektes zu machen. Eine Reihe von Algorithmen zur Bewegungsverarbeitung, z. B. die in Kapitel 4 ff vorgestellten Generalisierungsalgorithmen, arbeiten zwar auf der kompletten Bewegungssequenz, ändern diese aber nur im Kleinen (so wird bei der Generalisierung der Bewegungsverlauf geglättet, wobei die Form im Großen erhalten bleibt). Werden diese Algorithmen nun auf (im QMV-Vektorraum repräsentierte) QMV-Sequenzen angewandt, so kann dies zwar zu leichten Verzerrungen führen, die Grundform bleibt jedoch erhalten. Dies liegt daran, daß die *Veränderungen* durch die Verarbeitung im Kleinen stattfinden, womit die gemachten Annahmen zulässig bleiben.

Wichtig für das Rechnen im QMV-Vektorraum ist also immer, zu prüfen, wie

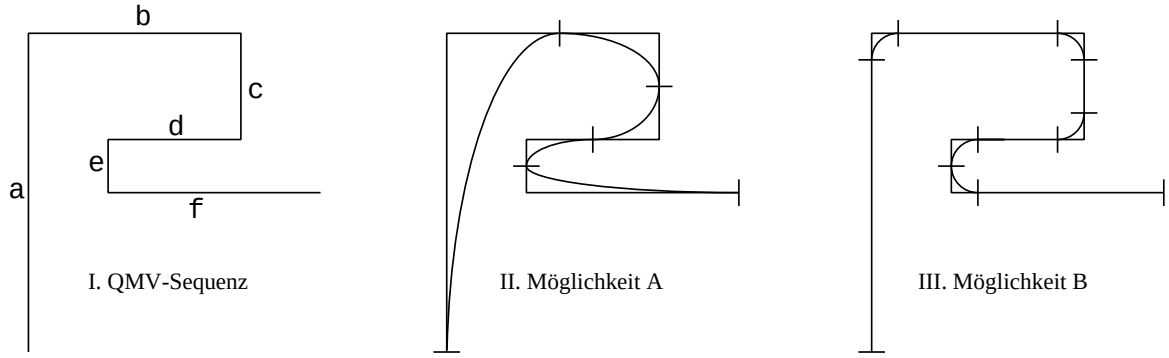


Abbildung 3.14.: Ansätze zur Segmentierung

sich die entstehenden Fehler auswirken und ob das Ergebnis für die konkrete Anwendung brauchbar ist, mit anderen Worten: wie aussagekräftig die QMV-Sequenz ist, die man im Ergebnis erhält.

Liegen numerische Daten vor, wurde die QMV-Sequenz also aus eine numerischen MV-Sequenz erzeugt und soll nun weiterverarbeitet (z. B. generalisiert) werden, so empfiehlt es sich eigentlich immer, möglichst alle Berechnungen direkt auf der numerischen MV-Sequenz durchzuführen und erst das Ergebnis in eine QMV-Sequenz umzuwandeln, da die Fehler im Numerischen wesentlich kleiner sind.

3.5. Shape-Repräsentation

Die zweite Schicht der vorne eingeführten Architektur basiert auf einer propositionalen Darstellung einer Bewegung, eine Bewegung wird hier als Folge von Formen wie `straight-line`, `left-turn`, `right-turn`, `u-turn`, ... beschrieben, die mit Größenangaben (und optional Geschwindigkeitsangaben) attributiert werden. Im Gegensatz zur QMV-Darstellung, bei der jeder numerische MV auf einen QMV abgebildet wird, entspricht eine Grundform hier einer ganzen Reihe von MV.

In einer gegebenen Bewegungsspur werden nun zunächst diese Grundformen identifiziert, die Bewegungsspur muß also so segmentiert werden, daß jedes Segment genau einer dieser Grundformen entspricht. Im zweiten Schritt werden dann passende Grundformen zu komplexeren Formen zusammengebaut.

3.5.1. Segmentierung auf QMV-Sequenzen

Abbildung 3.14 links zeigt die graphische Darstellung der QMV-Sequenz

```
<very-close north>24 <very-close east>16
<very-close south>8 <very-close west>10
<very-close south>4 <very-close east>16.
```

Diese QMV-Sequenz soll nun so segmentiert werden, daß sie durch die Grundformen `straight-line`, `left-turn`, `right-turn` und `direct-u-turn` beschrieben werden kann.

Die einzelnen Formen lassen sich hierbei an den Richtungsänderungen zwischen den einzelnen Teilstücken festmachen: Von a nach b findet ein `right-turn` statt, von b nach c und c nach d ebenso, von d nach e und e nach f schließlich ein `left-turn`. Damit ist jedes Teilstück (bis auf das erste und letzte) Teil von zwei Grundformen (und eventuell noch eines Streckenstücks zwischen diesen).

Da hier nur vier unterscheidbare Richtungen zur Verfügung stehen, kann aus der QMV-Sequenz nicht abgelesen werden, wie groß beispielsweise die Kurve von Teilstück a nach Teilstück b ist, ob also `<very-close north>`^{24} `<very-close east>`^{16} durch `<medium right-turn>` (im Fall einer sehr „runden“ Bewegung) oder besser durch `<medium straight-line> <small right-turn> <medium straight-line>` (in Fall einer eher eckigen Bewegung) beschrieben werden soll.

In Möglichkeit A (Abb. 3.14) werden alle Kurven möglichst groß gemacht, indem jede Teilstrecke in der Mitte geteilt wird, wodurch keine `straight-line` auftreten, in Möglichkeit B hingegen sind alle Kurven gleich groß, wobei der Kurvenradius von der Länge der kürzesten Teilstrecke bestimmt wird.

Beide Ansätze überzeugen nicht. In Möglichkeit A treten stark verzerrte Grundformen auf, die Sequenz

```
<medium right-turn> <small right-turn>
<small right-turn> <small left-turn>
<medium left-turn>
```

gibt zwar die Abfolge von Rechts- und Linkskurven korrekt wieder, nicht aber ihre Anordnung. Die Sequenz

```
<medium straight-line> <very-small right-turn>
<medium straight-line> <very-small right-turn>
<small straight-line> <very-small right-turn>
<small straight-line> <very-small left-turn>
<very-small left-turn> <medium straight-line>
```

aus Möglichkeit B hingegen enthält zu viele `straight-line`. Eine propositionale Beschreibung nach Möglichkeit B unterscheidet sich kaum von der zugehörigen QMV-Darstellung und stellt gegenüber dieser keinen Mehrwert dar.

Der folgende Algorithmus[35] wählt die einzelnen Kurven möglichst groß, ohne sie dabei zu verzerren. Abbildung 3.15 zeigt nochmals die graphische Darstellung der QMV-Sequenz. Die Strecken a mit f haben die Längen $|a| = 24$, $|b| = 16$, $|c| = 8$, $|d| = 10$, $|e| = 4$ und $|f| = 16$, wie man der oben angegebenen QMV-Sequenz

[35] erstmals beschrieben in [Ste98]

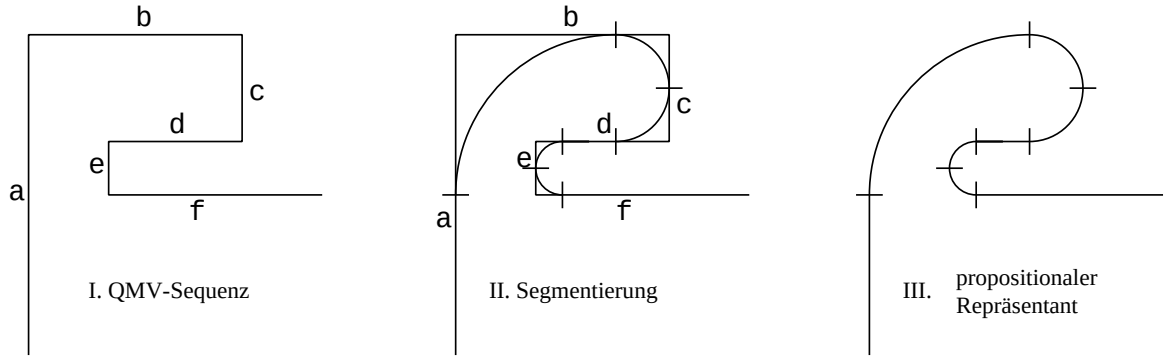


Abbildung 3.15.: Segmentierung einer QMV-Sequenz

entnehmen kann.[36] Jede Strecke b mit e ist Teil zweier Kurven, lediglich Anfangs- und Endstück haben nur Anteil an einer Kurve.

Nun werden die Strecken ihrer Länge nach sortiert, wobei alle Strecken, die an zwei Kurven beteiligt sind, nur mit halber Länge eingehen, da sie später geteilt werden. In unserem Beispiel erhalten wir (e, c, d, b, f, a).

Wir beginnen mit der kürzesten Strecke e. Sie ist Teil zweier Kurven, wird also einfach in der Mitte geteilt. Dadurch werden die Strecken d und f um den entsprechenden Betrag ($\frac{|e|}{2} = 2$) gekürzt ($|d_{neu}| = 8$, $|f_{neu}| = 14$). Nun werden die geänderten Strecken mit ihrer neuen Länge einsortiert. d ist nunmehr nur noch Teil einer Kurve (zwischen c und d), da der Anteil für die Kurve zwischen d und e ja schon abgezogen wurde, wird also mit seiner vollen (neuen) Länge einsortiert, e und f entfallen aus der Sortierung, sie wurden „verbraucht“. Wir erhalten (c, b, d, a).

c ist Teil zweier Kurven, wird also ebenfalls in der Mitte geteilt, b und d werden entsprechend um $\frac{|c|}{2} = 4$ gekürzt ($|b_{neu}| = 12$, $|d_{neu}| = 4$). b wird nun mit voller (neuer) Länge neu einsortiert, c und d fallen weg, wir erhalten (b, a).

b ist nun nur noch Teil einer Kurve, diese erhält also die volle Länge von b (dies klappt aufgrund der Sortierung immer). a wird entsprechend gekürzt ($|a_{neu}| = 12$), fällt ebenso wie b weg, wir sind fertig.

Es bleiben die Streckenstücke

$$|a| = 12, \quad |b| = 0, \quad |c| = 0, \quad |d| = 4, \quad |e| = 0, \quad |f| = 14$$

und die oben erzeugten Kurven

$$\begin{array}{llllll} \mathcal{K}_{ab} & = & \{\texttt{right}, 12\}, & \mathcal{K}_{bc} & = & \{\texttt{right}, 4\}, \quad \mathcal{K}_{cd} = \{\texttt{right}, 4\}, \\ \mathcal{K}_{de} & = & \{\texttt{left}, 2\}, & \mathcal{K}_{ef} & = & \{\texttt{left}, 2\}. \end{array}$$

Der Zusammenbau dieser Teilstücke liefert

[36] Die Einheit ist immer `very-close` und wird in Zukunft weggelassen.

Bezeichner	Intervall	Beispiel
`very-small`	$[0, t_1]$	$[0, 2]$
`small`	$(t_1, t_2]$	$(2, 10]$
`medium`	$(t_2, t_3]$	$(10, 25]$
`large`	$(t_3, t_4]$	$(25, 60]$
`very-large`	(t_4, ∞)	$(60, \infty)$

Tabelle 3.3.: Umrechnung von Vielfachen von `very-close` in propositionale Entfernungen

```
<12 straight-line> <12 right-turn> <4 right-turn>
<4 right-turn> <4 straight-line> <2 left-turn>
<2 left-turn> <14 straight-line>.
```

Hier wird die Größe der einzelnen Formen noch in Vielfachen von `very-close` angegeben. Um wieder zu einer rein qualitativen Darstellung zu gelangen, werden diese Größenangaben mittels einer geeigneten Tabelle (z. B. Tabelle 3.3) auf symbolische Größenangaben abgebildet, wir erhalten

```
<medium straight-line> <medium right-turn>
<small right-turn> <small right-turn>
<small straight-line> <very-small left-turn>
<very-small left-turn> <medium straight-line>
```

als propositionale Darstellung der Bewegung. Will man mit dieser propositionalen Darstellung weiterarbeiten, so empfiehlt es sich, zunächst die numerischen Größenangaben beizubehalten und diese Umwandlung erst nach Ende der Berechnung durchzuführen.

Treten in der QMV-Sequenz direkte 180°-Kehren auf, so werden diese genauso behandelt wie Rechts- oder Linkskurven, d. h. ihre Größe wird nach gleicher Formel berechnet, und sie werden durch `direct-u-turn` beschrieben.

Ein leicht modifizierter Segmentierungsalgorithmus (ebenfalls von mir in [Ste98] beschrieben) leistet eine inkrementelle Segmentierung zur Weiterverarbeitung der Bewegung „on the fly“.

3.5.2. Klassifizierung

Nach der Segmentierung einer Bewegungsspur in eine Reihe von Grundformen werden diese basalen Formen (Shapes) nun zu komplexeren Formen zusammengefaßt, so wird aus zwei ungefähr gleich großen `right-turn` ein `right-u-turn` usw. Die Komposition komplexerer Shapes aus wenigen Grundformen auf Grundlage einer beliebig wählbaren Regelbasis wird in [Mus00] ausführlich beschrieben. Ich gehe an dieser Stelle nicht weiter darauf ein.

4. Generalisierung

Liegt die Beschreibung einer Bewegung wie in Kapitel 2 beschrieben als Polygonzug vor, beispielsweise als Ergebnis einer Messung, so ist die so gegebene Kurve für die weitere Bearbeitung häufig viel zu detailliert und komplex.

Soll beispielsweise die Fahrt einer Radfahrerin durch eine Stadt beschrieben werden, so interessiert häufig nur, welche Radwege und Straßen sie durchfuhr, nicht aber, ob sie dabei viel hin- und herwackelte, oder vielleicht hier und da einem Gullideckel auswich. Hier interessiert die Grobstruktur der Bewegung, also die Form der Kurve im Großen. Will ich hingegen herausfinden, ob sie gut und flüssig fuhr, oder sich sehr unsicher (oder auch alkoholisiert) in Schlangenlinien fortbewegte, interessieren mich die durchfahrenen Straßen weniger, hier interessiert vielmehr gerade die Bewegung im Kleinen.

In beiden Fällen müssen hierzu die relevanten Informationen aus der Bewegungsspur extrahiert werden, d. h. es ist ein (einfacherer) Polygonzug gesucht, der nur noch die für eine bestimmte Fragestellung interessanten Informationen enthält. Algorithmen, die eine gegebene Kurve (einen Polygonzug) vereinfachen, sind sowohl in der Computergraphik als auch in der Geographie/Kartographie unter Begriffen wie „generalization“, „line generalization“ oder „line simplification“ bekannt.[1] In der Computergraphik müssen beispielsweise die in einem eingescannten Bild gefundenen Kanten geglättet werden. In der Kartographie liegt das Hauptaugenmerk auf – heutzutage auch aus Scans von Zeichnungen oder Satellitenbildern extrahierten, früher aber vielfach von Hand eingetasteten – Küstenlinien, Flußläufen und auch Landesgrenzen. Schon 1972 wurde einer der zumindest in der Geographie heute noch grundlegenden Generalisierungsalgorithmen (siehe Kapitel 5) von Douglas und Peucker in der Geographie und (soweit aus den Unterlagen ersichtlich) unabhängig davon und zeitgleich dazu von Ramer in der Computergraphik entwickelt, und in beiden Disziplinen war dieses Problem auch lange vorher schon Gegenstand der Forschung.

Dieses Kapitel liefert eine Einführung in die Generalisierung von Polygonzü-

[1] Sowohl das englische „generalization“ als auch das deutsche „Generalisierung“ bedeuten „Verallgemeinerung“, sind demzufolge also als Benennung für Verfahren zur Vereinfachung von Polygonzügen nicht optimal gewählt. Zumindest der englische Begriff hat sich allerdings in der Literatur in dieser Form durchgesetzt. Da mir keine griffigere deutsche Bezeichnung bekannt ist, bleibe ich hier bei „Generalisierung“.

gen mit Schwerpunkt auf die Anforderungen für die Vereinfachung von Bewegungsspuren. In den nächsten Kapiteln werden dann einzelne Generalisierungsalgorithmen vorgestellt, wobei einige aus der Geographie stammen[2] und zur Verarbeitung von Bewegungsspuren angepaßt und erweitert werden, wogegen andere speziell zur Bewegungsverarbeitung neu entwickelt wurden. In Kapitel 12 werden dann Generalisierungen variabler Stärke betrachtet und zur Trennung von Grob- und Feinstruktur einer Bewegungsspur genutzt. Kapitel 13 schließlich vergleicht die beschriebenen Algorithmen hinsichtlich ihrer Eigenschaften.

4.1. Grundlagen

Robert Mc Master zählt in seinem Überblick über Generalisierungsalgorithmen in der Kartographie [McM87] zu den vier wesentlichen Komponenten der Generalisierung Vereinfachung, Glättung, Verschiebung und Anreicherung (simplification, smoothing, displacement, and enhancement).

Vereinfachung beschreibt das Weglassen „unwichtiger“ oder „überflüssiger“ Punkte einer Kurve, die Originalkurve wird durch eine „einfachere“ Version ersetzt.

Glättung (smoothing) vereinfacht die Originalkurve nicht unbedingt, sondern nivelliert kleine Unebenheiten, macht die Kurve runder.

Verschiebung von Teilen einer Kurve ist nötig, um z. B. zu vermeiden, daß zwei dicht nebeneinanderliegende Kurven durch Generalisierung oder einfach dadurch, daß sie in einer geringeren Auflösung dicker gezeichnet werden, miteinander verschmelzen oder sich überlagern. So sollte beispielweise auch auf einer Weltkarte das Mittelmeer bei Gibraltar noch mit dem Atlantik verbunden sein. Hierzu werden die spanische[3] und die marokkanische Küstenlinie ein klein wenig auseinandergeschoben, so daß sie nicht miteinander verschmelzen.

Anreicherung ist das Hinzufügen von Details zu einer bereits vereinfachten Kurve, vor allem zur Erstellung „natürlich“ wirkender Landkarten: Eine Karte von Norwegen in einem Maßstab von 1:10 000 000 kann selbstverständlich nicht all die vielen tausend Fjorde wiedergeben. Eine schlichte Vereinfachung

[2] Neben den hier vorgestellten existieren noch eine ganze Reihe weiterer Generalisierungsalgorithmen (siehe z. B. den ausführlichen, allerdings schon etwas veralteten Überblick in [McM87]). Viele dieser Algorithmen sind überholt, da andere Algorithmen bessere Ergebnisse liefern oder aufgrund ihrer Eigenschaften für die Verarbeitung von Bewegungsspuren ungeeignet.

[3] an dieser Stelle eigentlich britische

dieser Küstenlinie würde jedoch vielleicht einfach zu einer leicht geschwungenen, einer wellenförmigen oder auch zu einer zickzackartigen Kurve führen, die eben nicht der für Norwegen charakteristischen Form entspricht. Aus diesem Grund sieht man auf solchen Karten zwar nicht jeden Fjord, aber doch eine Küstenlinie, die aus einer Reihe von Fjorden besteht (die in genau dieser Form in der Natur eben nicht vorkommen, einige Fjorde sind vergrößert, andere weggelassen).

Diese Arbeit beschäftigt sich mit der Verarbeitung von Bewegungsspuren, die Anforderungen an Generalisierungsalgorithmen unterscheiden sich daher etwas von denen der Kartographie. Eine Kernaufgabe ist auch hier die Vereinfachung von Kurven. Wie in Kapitel 2 gezeigt, können Bewegungsverläufe als Sequenz von Positionsangaben (als Polygonzug) oder als Sequenz von Bewegungsvektoren (als MV-Sequenz) repräsentiert werden.

Die in den folgenden Kapiteln vorgestellten Algorithmen arbeiten auf Polygonzügen, wobei die Verarbeitung translations- und rotationsinvariant ist.[4] Wandelt man nun einen als MV-Sequenz gegebenen Bewegungsverlauf mit beliebig gewähltem Startpunkt in einen Polygonzug um (siehe Abschnitt 2.3), so kann dieser mit jedem der vorgestellten Algorithmen verarbeitet und das Ergebnis der Verarbeitung wieder in eine MV-Sequenz umgewandelt werden. Auch wenn dies nicht bei jedem Algorithmus explizit erwähnt ist, sind damit alle Algorithmen auch auf in Form von MV-Sequenzen gegebene Bewegungsverläufe anwendbar.[5]

Darüberhinaus können auch QMV-Sequenzen (Kapitel 3) mit Hilfe des QMV-Vektorraumes (Abschnitt 3.4.3) als Polygonzüge dargestellt werden, womit auch auf ihnen alle vorgestellten Algorithmen Anwendung finden können.

Ich verstehe im Folgenden unter Generalisierung g einer Bewegungsspur (eines Polygonzuges) die Abbildung eines Polygonzuges p auf einen *einfacheren* aber zu p *ähnlichen* Polygonzug q:

$$\begin{aligned} g : \mathcal{P} &\rightarrow \mathcal{P}, \qquad \mathcal{P} = \{p \mid p \text{ Polygonzug}\} \\ g(p) &= q. \end{aligned}$$

Der Begriff der Generalisierung hängt also elementar an der Frage, wann ein Polygonzug q *einfacher* ist als ein Polygonzug p, sowie an der Definition der *Ähnlichkeit* zweier Polygonzüge:

Definition 1 *Ein Polygonzug q ist* einfacher *als ein Polygonzug p, wenn q weniger Eckpunkte besitzt als p.*

[4] Das heißt, daß die Generalisierung eines Bewegungsverlaufes unabhängig von seiner Lage im Raum immer gleich verläuft, die Lage des generalisierten Polygonzuges ist dann aber selbstverständlich von der Lage des Originalpolygonzuges abhängig.

[5] Einige Algorithmen können sogar direkt auf MV-Sequenzen arbeiten, was unter Umständen auch Optimierungen ermöglicht. Dies ist dann jeweils angegeben.

Definition 2 *p und q sind einander* ähnlich, *wenn sie „im Großen" die gleiche Bewegung (die gleiche Grundform) beschreiben und sich nur „im Kleinen" unterscheiden.*

Die Definition von Ähnlichkeit befriedigt nicht, da sie kein formales Kriterium liefert, um zu testen, ob zwei Polygonzüge einander ähnlich sind und *wie* ähnlich sie einander sind. Abhilfe schafft die Angabe eines konkreten Ähnlichkeitsmaßes, also eines Kriteriums, das zwei Polygonzüge erfüllen müssen, um einander „ähnlich" zu sein:

Definition 3 *Ein Polygonzug q ist einem Polygonzug p* hinreichend ähnlich *($p \approxeq_M q$), wenn er bezüglich p ein gegebenes Ähnlichkeitsmaß M erfüllt.*[6]

Das gewählte Maß ist dabei optimalerweise symmetrisch, d. h. wenn q bezüglich eines Maßes M zu p ähnlich ist ($p \approxeq_M q$), dann auch umgekehrt ($q \approxeq_M p$).

Definition 4 *Ein Algorithmus g erfüllt ein (symmetrisches) Ähnlichkeitsmaß M, wenn für alle Polygonzüge p gilt: $p \approxeq_M g(p)$.*

Definition 5 *Ein Algorithmus g_a ist (bezüglich eines Ähnlichkeitsmaßes M) für einen bestimmten Polygonzug p* besser *als ein Algorithmus g_b, wenn beide Algorithmen M erfüllen und $g_a(p)$* einfacher *ist als $g_b(p)$.*

g_a ist (bezüglich M) besser *als g_b, wenn dies für alle Polygonzüge p gilt.*

So einfach und klar diese Definition des *besseren* Algorithmus auf den ersten Blick scheint, so schwierig ist die Einschätzung in der Praxis. Zum einen basieren unterschiedliche Generalisierungsalgorithmen häufig auf unterschiedlichen Ähnlichkeitsmaßen, zum anderen sollen häufig weitere Nebenbedingungen erfüllt werden.

4.2. Inkrementelle Generalisierung

Soll eine Bewegungsspur nicht erst im Nachhinein sondern schon während der laufenden Bewegung analysiert und weiterverarbeitet werden, beispielsweise um aufgrund der zurückgelegten Bewegung eine Entscheidung für das Abbiegen an der nächsten Kreuzung fällen zu können, so sind Algorithmen nötig, die einen Polygonzug sequentiell abarbeiten. Ein *inkrementeller* Generalisierungsalgorithmus liest eine Bewegungsspur p Punkt für Punkt ein und liefert die Generalisierung Stück für Stück zurück. Er läßt sich somit als Filter in einem Datenstrom einsetzen, der die generalisierte Spur noch während der Bewegung an nachgestellte Funktionen zur Weiterverarbeitung übergibt.

[6] Ein Ähnlichkeitsmaß kann dabei z. B. sein, daß beide Polygonzüge nicht weiter als eine bestimmte Distanz voneinander abweichen, daß die Fläche zwischen den beiden Kurven unterhalb einer gegebenen Grenze bleibt etc. In den folgenden Kapiteln werden an konkreten Algorithmen unterschiedliche Ähnlichkeitsmaße vorgestellt.

Definition 6 *Eine Generalisierung g heißt* inkrementell, *wenn für alle Polygonzüge $p, p' \in \mathcal{P}$ mit $p = v_1 v_2 \dots v_n$ und $p' = v_1 v_2 \dots v_n v_{n+1} \dots v_{n+k}$ gilt:*

$$\begin{array}{rclcl} g(p) & = & g(v_1 v_2 \dots v_n) & = & w_1 w_2 \dots w_{m-1} w_m, \\ g(p') & = & g(v_1 v_2 \dots v_n v_{n+1} \dots v_{n+k}) & = & w_1 w_2 \dots w_{m-1} w'_m w'_{m+1} \dots w'_{m+l}, \end{array}$$

Ist p Anfang von p', so ist $g(p)$ ohne seinen letzten Punkt Anfang von $g(p')$.

Definition 7 *Eine Generalisierung g heißt* historiefrei *inkrementell, wenn darüberhinaus für jeden Polygonzug $p = v_1 v_2 \dots v_n$ mit $g(p) = w_1 w_2 \dots w_m$ für alle Teilpolygonzüge $p_a = v_1 v_2 \dots v_t$ und $p_b = v_t v_{t+1} \dots v_n$ gilt:*

$$g(p_a) = w_1 w_2 \dots w_l \quad \Longrightarrow \quad g(p_b) = w_l w_{l+1} \dots w_m.$$

Nur die Punkte im aktuell betrachteten Segment gehen in die Generalisierung ein, bereits abgeschlossene Segmente haben keinen Einfluß auf das aktuell bearbeitete Teilstück.

Historiefreie inkrementelle Generalisierungen besitzen über die Länge der Spur (genauer: die Anzahl der generalisierten Segmente) immer lineare Komplexität. Unabhängig davon, wie komplex die Berechnung eines einzelnen Segments ist, steigt die Komplexität der Verarbeitung linear in der Anzahl der Segmente, da die Segmente voneinander unabhängig sind.

4.3. Eckentreue Generalisierung

Definition 8 *Ein* eckentreuer *Algorithmus bildet einen Polygonzug p so auf einen Polygonzug q ab, daß alle Ecken von q auch Ecken von p sind:*

$$p = v_1 v_2 \dots v_n, \quad q = w_1 w_2 \dots w_m : \qquad \forall w_i : w_i \in \{v_1, v_2, \dots, v_n\}.$$

Je nach Sichtweise werden also bestimmte „wichtige“ Eckpunkte aus p ausgewählt, aus denen dann der Polygonzug q gebildet wird bzw. so lange „unwichtige“ Punkte aus p gestrichen, bis nur noch die „wichtigen“ Punkte (und damit die Generalisierung q) übrigbleiben.

4.4. Korridor-Generalisierungen

Korridor-Generalisierungen basieren auf der Idee, um den gegebenen Polygonzug p einen Korridor K (einer festen Breite b) zu legen, in dem p vollständig liegt. Abbildung 4.1 zeigt dies am Beispiel.

Eine wesentliche Ausprägung der Korridor-Generalisierungen sind Generalisierungen mit ε-Ähnlichkeitsmaß.

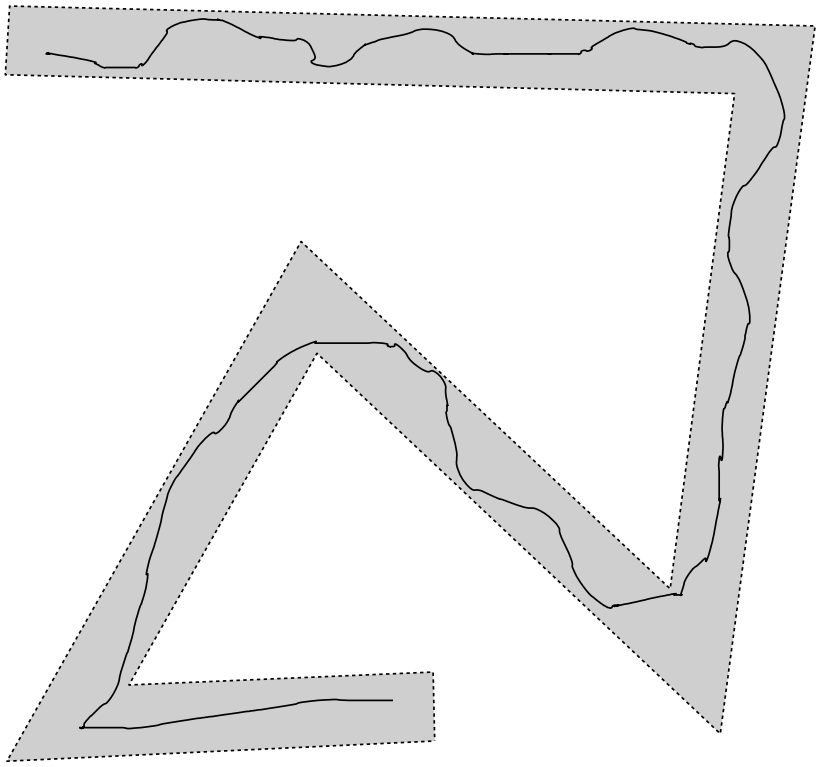


Abbildung 4.1.: Korridor-Generalisierung

Definition 9 *ε-Kriterium: Eine ε-Generalisierung ist eine Abbildung*

$$\begin{aligned} g_\varepsilon : \mathcal{P} &\rightarrow \mathcal{P}, \qquad \mathcal{P} = \{p \mid p \text{ Polygonzug}\} \\ g_\varepsilon(p) &= q, \end{aligned}$$

so daß p und q nirgends einen größeren Abstand als ε voneinander haben.

Legt man nun um q einen Korridor K der Breite $2\,\varepsilon$, so liegt p vollständig in diesem Korridor.

Die Umkehrung ist nicht hinreichend: Abb. 4.2 links zeigt eine Bewegungsspur p, die in einem $b = 2\,\varepsilon$ breiten Korridor K liegt. Legt man nun mittig in diesen Korridor einen Polygonzug q (gestrichelte Linie), so erfüllt q die ε-Bedingung bezüglich p nicht zwingend, wie Abb. 4.2 rechts zeigt: p ragt aus der ε-Umgebung um q heraus.

Weiterhin sieht man in Abb. 4.3, daß sich gemeinhin aus einem gegebenen Korridor nicht eindeutig ein zugehöriger Polygonzug ablesen läßt, und daß eine Korridorumgebung die Form einer Bewegung nicht unbedingt gut wiedergibt. Der Korridor gibt zwar an, *wo*, also an welchen Orten im Raum sich die Originalkurve befindet, nicht aber, in welcher Reihenfolge diese Orte durchlaufen werden: Im oberen Korridorsegment zeigt die Originalkurve eine Schleife, die selbstverständlich

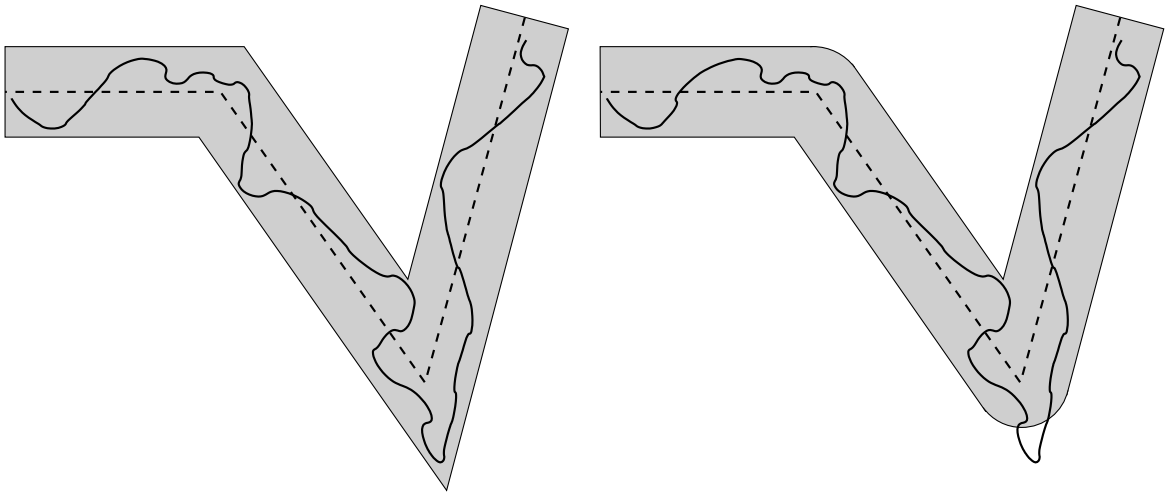


Abbildung 4.2.: Korridor und ε-Abstand

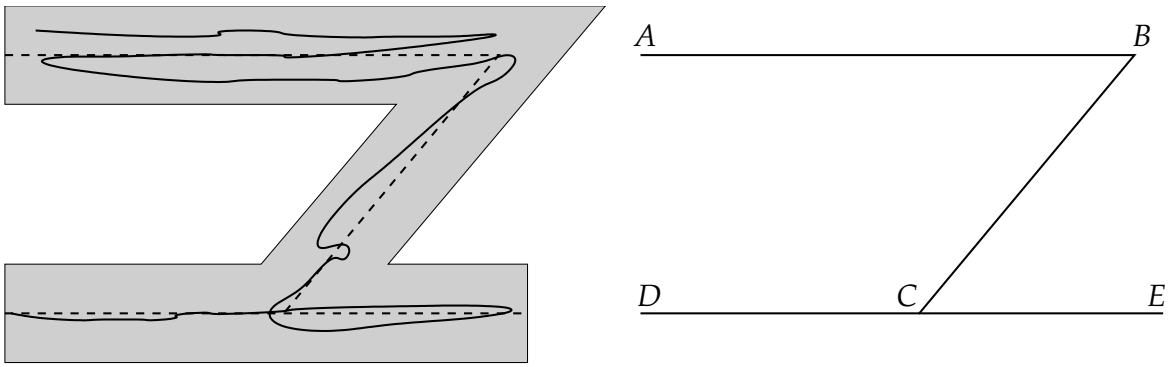


Abbildung 4.3.: Korridor vs. Polygonzug

vom Korridor überhaupt nicht wiedergegeben wird, wogegen unten zwei Segmente T-förmig aufeinanderstoßen, wodurch ein Gebilde entsteht (Abb. 4.3 rechts), das keinen Polygonzug darstellt.

Durch Verbinden der Punkte C und D oder C und E bekommt man die Polygonzüge $ABCDE$ oder $ABCED$, wobei ohne Zusatzinformationen aus dem Originalpolygonzug nicht klar ist, welcher die Situation besser wiedergibt (in unserem Fall wäre aufgrund der Schleife im Segment AB der Polygonzug $ABABCED$ korrekt).

5. Divide-and-Conquer-Generalisierung

Die hier beschriebene Generalisierung wurde annähernd gleichzeitig und soweit mir bekannt unabhängig voneinander sowohl von Urs Ramer[1] zur Bildverarbeitung als auch von David Douglas und Thomas Peucker[2] zur Vereinfachung geographischer Konturen wie Küstenlinien entwickelt. Erland Jungert und Daniel Hernández beschreiben in [JH96] einen sehr ähnlichen Algorithmus zur Generalisierung von Bewegungsspuren.[3] Beide Algorithmen besitzen die in Abschnitt 4.4 angedeuteten Probleme mit Rückläufigkeiten und Schleifen (siehe Abb. 4.3), und wurden daher von mir modifiziert.[4]

5.1. Ramer und Douglas/Peucker

Die Generalisierung basiert auf der vorne (Abschnitt 4.4) erwähnten ε-Bedingung: Finde einen Polygonzug q so, daß er zum Originalpolygonzug p nirgends einen größeren Abstand als ε besitzt. Die zugrundeliegende Idee ist einfach. Durch Anfangs- und Endpunkt des Originalpolygonzuges $p = v_1 v_2 \dots v_n$ wird eine Gerade $g = \overline{v_1 v_n}$ gelegt. Nun wird der am weitesten von der Geraden g entfernt liegende Punkt v_m bestimmt.[5] Ist $\mathrm{dist}(g, v_m) < \varepsilon$, liegen also alle Punkte weniger als ε von $g = \overline{v_1 v_n}$ entfernt, so ist $v_1 v_n$ die gesuchte Generalisierung von p.

Andernfalls wird $p = v_1 v_2 \dots v_n$ in die beiden Teilstücke $p_l = v_1 v_2 \dots v_m$ und $p_r = v_m v_{m+1} \dots v_n$ zerlegt und auf diesen das Verfahren erneut angewandt (Algorithmus 1). Wir haben es hier also mit einem klassischen Divide-and-Conquer-Verfahren zu tun.

Abbildung 5.1 zeigt den Ablauf des Verfahrens: Zunächst wird durch die End-

[1] [Ram72]
[2] [DP73], eine Implementierung in FORTRAN findet sich in [WW88].
[3] Dieser Algorithmus wurde ursprünglich von Persson und Jungert [PJ92] entwickelt und in obiger Quelle auf Bewegungen angewandt.
[4] Siehe auch [Ste98].
[5] Liegt dabei mehr als ein Punkt gleich weit entfernt, wird ein beliebiger ausgewählt.

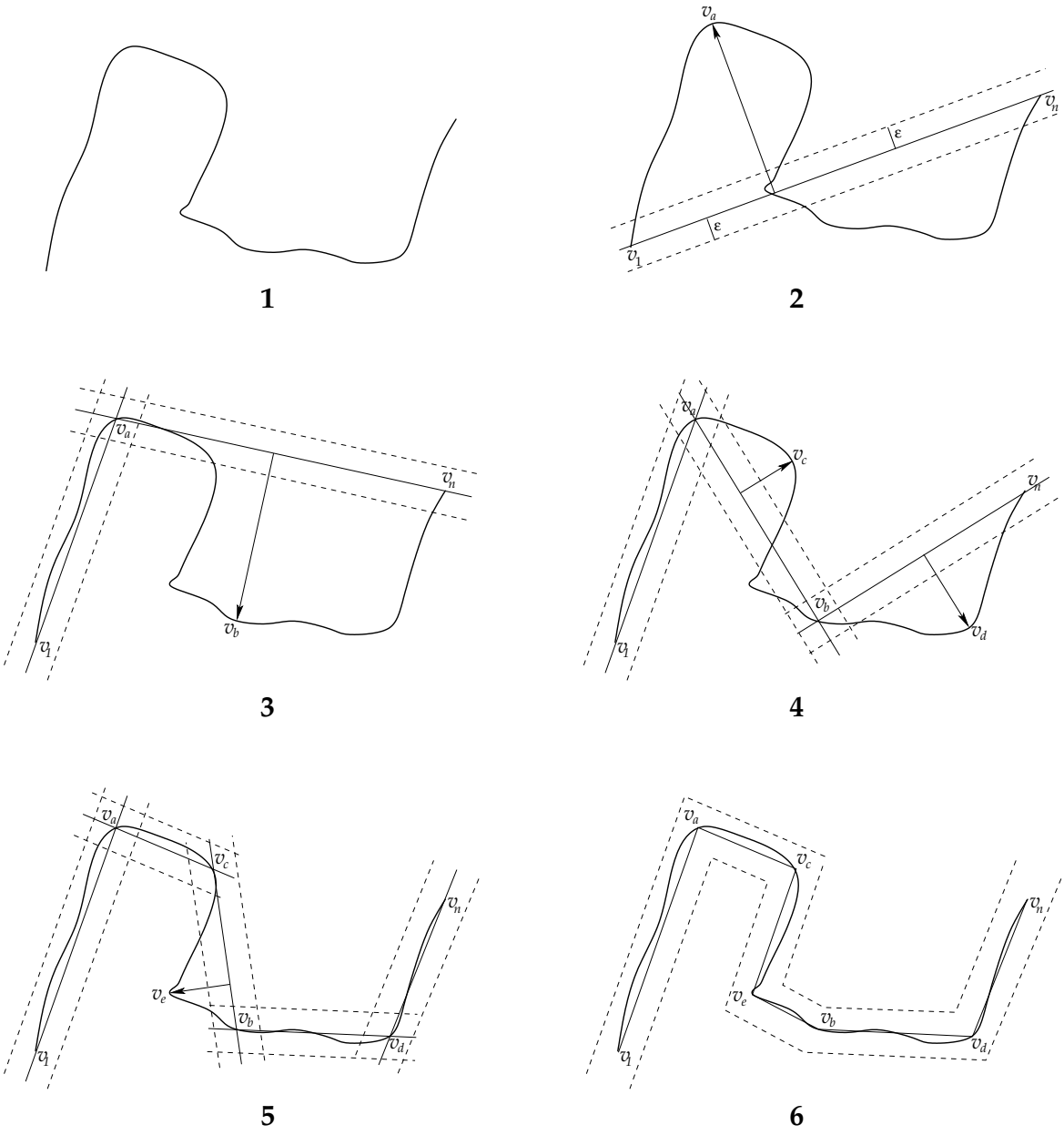


Abbildung 5.1.: Douglas/Peucker-Generalisierung

Algorithmus 1: Douglas/Peucker-Generalisierung

```
1:  function findmaxdist = (Gerade g, Polygonzug p = v_1 ... v_n) → Index, Distanz
    :
2:  begin
3:     i ← 0;                          ▷ Initialisierung
4:     d ← 0;
5:     for all v_k do                  ▷ Wir testen alle Punkte
6:        d' ← |dist(g, v_k)|;         ▷ Distanz zur Geraden
7:        if (d' > d) then             ▷ Punkt mit größerem Abstand gefunden
8:           i ← k;                    ▷ Update
9:           d ← d';
10:       end if
11:    end for
12:    return i, d;                    ▷ Punkt und Entfernung zurückliefern
13: end

14: function dpgen = (Polygonzug p = v_1 ... v_n, Distanz ε) → Polygonzug :
15: begin
16:    m, d ← findmaxdist(v_1v_n, p);  ▷ Den zu v_1v_n entferntesten Punkt finden
17:    if (d > ε) then
18:       return dpgen(v_1 ... v_m, ε) ∘ dpgen(v_m ... v_n, ε);  ▷ Divide and Conquer
19:    else
20:       return v_1v_n;               ▷ Alle Punke liegen im ε-Korridor um v_1v_n
21:    end if
22: end
```

punkte der Originalkurve[6] (v_1 und v_n) eine Gerade $g = \overline{v_1v_n}$ gelegt. Bild 2 zeigt, daß der am weitesten entfernt liegende Punkt v_a nicht im ε-Korridor um g liegt. Daher wird der Algorithmus nun rekursiv auf die Teilstücke $v_1 \dots v_a$ und $v_a \dots v_n$ angewandt (Bild 3). Im ersten Teilstück liegt die Originalkurve vollständig im ε-Korridor um $\overline{v_1v_a}$. Für dieses Segment terminiert der Algorithmus somit und liefert die Strecke $[v_1v_a]$ zurück. Im Teilstück $v_a \dots v_n$ liegt der am weitesten von $\overline{v_av_n}$ entfernte Punkt v_b nicht im zugehörigen ε-Korridor. Daher wird der Algorithmus nun wiederum auf die Segmente $v_a \dots v_b$ und $v_b \dots v_n$ angewandt. Im ersten Segment liegt nun v_c, im zweiten v_d am weitesten (und weiter als ε) von der zugehörigen Geraden entfernt (Bild 4). Bild 5 zeigt, daß nun die Teilstücke $v_a \dots v_c$, $v_b \dots v_d$ und $v_d \dots v_n$ die ε-Bedingung erfüllen, lediglich in $v_c \dots v_b$ muß ein weiteres mal iteriert werden, was schließlich zum Polygonzug $v_1v_av_cv_ev_bv_dv_n$ führt (Bild 6). Im Grunde

[6] Ich spreche im Folgenden von einer Original„kurve“, auch wenn diese als ein Polygonzug vorliegt.

ist es hierbei sogar unwichtig, ob die Originalkurve als Polygonzug oder in anderer Form (z. B. als Spur einer mathematischen Funktion) gegeben ist, solange sich der zu beliebigen Geraden entfernteste Punkt bestimmen läßt.

Wie man leicht sieht, ist obiger Algorithmus nicht auf Polygonzüge im zweidimensionalen Raum beschränkt, die Bestimmung des Abstands eines Punktes zu einer Geraden ist auch im höherdimensionalen Raum in konstanter Zeit möglich (genauer: der Aufwand zur Entfernungsberechnung steigt mit der Dimension des Raumes, geht aber lediglich als konstanter Faktor in die Laufzeit des Algorithmus ein).

5.2. Persson/Jungert/Hernández

Jungert und Hernández schlagen für den zweidimensionalen[7] Fall in [JH96] speziell zur Verarbeitung von Bewegungsdaten einen (zur Douglas/Peucker-Generalisierung) sehr ähnlichen Algorithmus vor.[8] Wie gehabt, wird durch Anfangs- (v_1) und Endpunkt (v_n) des Originalpolygonzuges $p = v_1 v_2 \dots v_n$ eine Gerade $g = \overline{v_1 v_n}$ gelegt. Im zweidimensionalen teilt eine Gerade die Ebene in zwei Hälften. Nun wird der auf der linken und auf der rechten Seite am weitesten entfernt liegende Punkt (v_l und v_r) bestimmt. Ist der Abstand beider Punkte zu g kleinergleich ε, so terminiert der Algorithmus wie gehabt. Liegt einer der beiden Punkte (oBdA: v_l) weiter als ε von g entfernt, so werden wie schon bei der Douglas/Peucker-Generalisierung die Teilstücke $v_1 \dots v_l$ und $v_l \dots v_n$ rekursiv betrachtet. Liegen beide Punkte weiter als ε von g entfernt, so wird die Kurve dreigeteilt und die drei Teilstücke (unter der Annahme, daß v_l vor v_r liegt: $v_1 \dots v_l$, $v_l \dots v_r$ und $v_r \dots v_n$) rekursiv abgearbeitet.(Algorithmus 2).

Abbildung 5.2 zeigt den Ablauf des Algorithmus auf der selben Kurve, auf der vorne (Abb. 5.1) die Douglas/Peucker-Generalisierung gezeigt wurde. Die bezüglich der Geraden $g = \overline{v_1 v_n}$ links und rechts am weitesten entfernt liegenden Punkte v_{l_a} und v_{r_a} werden gefunden (Bild 2) und liegen beide nicht im ε-Korridor um g. Folglich wird rekursiv auf den drei Teilstücken $v_1 \dots v_{l_a}$, $v_{l_a} \dots v_{r_a}$ und $v_{r_a} \dots v_n$ weitergearbeitet. Bild 3 zeigt, daß im ersten und dritten Teilstück die ε-Bedingung erfüllt ist, im zweiten Teilstück sind v_{l_b} und v_{r_b} die bezüglich $\overline{v_{l_a} v_{r_a}}$ maximal entfernten Punkte. Nochmalige rekursive Anwendung des Verfahrens liefert schließlich (Bild 4) den generalisierten Polygonzug $v_1 v_{l_a} v_{l_b} v_{r_b} v_{r_a} v_n$.

Obgleich sich beide Algorithmen sehr ähneln, liefern sie, wie man hier

[7] Obgleich sie sich mit Bewegung von Flugzeugen in der Flugüberwachung beschäftigen, beschränken sie sich auf die Generalisierung in der Ebene. Flugbewegungen werden in der zivilen Luftfahrt als zweidimensionale Bewegungen in übereinandergeschichteten Ebenen betrachtet, zwischen denen gewechselt werden kann.

[8] Aus meinen Unterlagen geht leider nicht hervor, ob ihnen der Algorithmus von Douglas und Peucker bekannt war, zumindest wird [DP73] nicht in [JH96] zitiert.

Algorithmus 2: Persson/Jungert/Hernández-Generalisierung

```
1:  function findmaxdist = (Gerade g, Polygonzug p = v_1 ... v_n)
                                  → Index, Distanz, Index, Distanz:
2:  begin
3:     i_l ← 0;                          ▷ Initialisierung
4:     d_l ← 0;
5:     i_r ← 0;
6:     d_r ← 0;
7:     for all v_k do                    ▷ Wir testen alle Punkte
8:        d' ← dist(g, v_k);             ▷ vorzeichenbehaftete Distanz zur Geraden
                                         ▷ für Punkte links von g ist d' also negativ
9:        if (d' < d_l) then             ▷ links Punkt mit größerem Abstand gefunden
10:          i_l ← k;                    ▷ Update
11:          d_l ← d';
12:       else if (d' > d_r) then        ▷ rechts Punkt mit größerem Abstand gefunden
13:          i_r ← k;                    ▷ Update
14:          d_r ← d';
15:       end if
16:    end for
17:    return i_l, d_l, i_r, d_r;        ▷ Punkt und Entfernung zurückliefern
18: end

19: function jhgen = (Polygonzug p = v_1 ... v_n, Distanz ε) → Polygonzug :
20: begin
21:    a, d_a, b, d_b ← findmaxdist(v_1v_n, p);   ▷ Die zu v_1v_n entferntesten Punkte finden
22:    if a > b then                     ▷ aufsteigend sortieren
23:       (a, b) ← (b, a)                ▷ swap(a, b)
24:       (d_a, d_b) ← (d_b, d_a)        ▷ Distanzen ebenfalls vertauschen
25:    end if
26:    if (|d_a| > ε) then
27:       if (|d_b| > ε) then            ▷ beide Punkte
28:          return jhgen(v_1 ... v_a, ε) ∘ jhgen(v_a ... v_b, ε) ∘ jhgen(v_b ... v_n, ε);
29:       else                           ▷ nur der eine
30:          return jhgen(v_1 ... v_a, ε) ∘ jhgen(v_a ... v_n, ε);
31:       end if
32:    else if (|d_b| > ε) then          ▷ nur der andere
33:       return jhgen(v_1 ... v_b, ε) ∘ jhgen(v_b ... v_n, ε);
34:    else
35:       return v_1v_n;                 ▷ Alle Punke liegen im ε-Korridor um v_1v_n
36:    end if
37: end
```

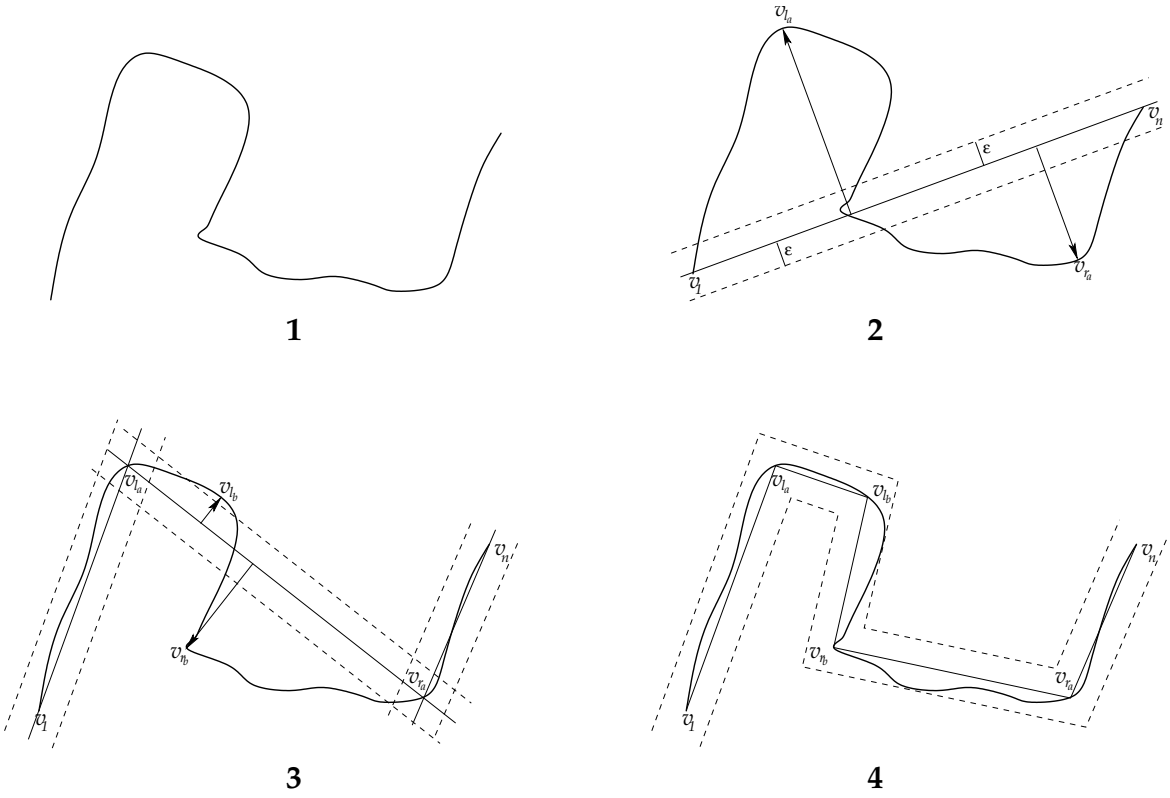


Abbildung 5.2.: Persson/Jungert/Hernández-Generalisierung

sieht, unterschiedliche Generalisierungen. Der zweite Algorithmus läuft nicht nur schneller, er liefert in diesem Fall auch die „bessere“ Generalisierung. Dies muß selbstverständlich nicht so sein.

5.3. Schleifenerkennung

In der in Alg. 1 und 2 angegebenen Form besitzen beide Algorithmen die in 4.4 angedeuteten Probleme mit Rückläufigkeiten und Schleifen, wie Abbildung 5.3 zeigt: Da in beiden Fällen der Abstand zur Geraden $g = \overline{v_1 v_n}$ bestimmt wird, liegt die gesamte Kurve innerhalb des ε-Korridors um g, obgleich die Generalisierung $v_1 v_n$ die Originalkurve nicht gut beschreibt. Hierbei treten zwei Fälle auf:

Externe Schleife: In I. sind Teile der Originalkurve (links von v_1) weiter als ε von der Strecke $[v_1 v_n]$ entfernt.

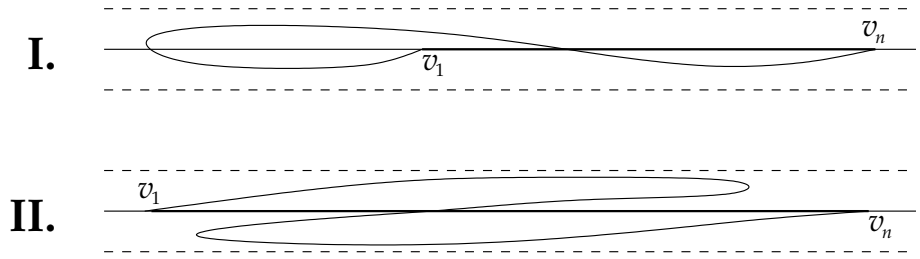


Abbildung 5.3.: Generalisierungsfehler

Interne Schleife: In II. liegt zwar die gesamte Kurve näher als ε an $[v_1 v_n]$, die Schleife in der Originalkurve bleibt dabei aber völlig unbeachtet und wird weggeneralisiert.

Um den ersten Fall in den Griff zu kriegen, ist lediglich eine kleine Modifikation der angegebenen Algorithmen notwendig: Statt die Distanz $d' = |\text{dist}(g, v_k)|$ des Punktes v_k zur Geraden $g = \overline{v_a v_b}$ zu berechnen, wird einfach die Distanz $d' = |\text{dist}(s, v_k)|$ zur Strecke $s = [v_a v_b]$ berechnet.[9]

Fall II. ist etwas schwerer in den Griff zu bekommen. Der beschriebene Fehler kann nur auftreten, wenn die Originalkurve bezüglich s Wendepunkte besitzt. Wendepunkt bedeutet in diesem Zusammenhang: Durchläuft man s und die Originalkurve beide in Richtung v_a nach v_b (im Beispiel von Abb. 5.3 II von v_1 nach v_n), so schließen der Richtungsvektor eines Punktes v_k der Kurve und der Richtungsvektor $v_b - v_a$ der Strecke s normalerweise einen Winkel kleiner 90° ein. Wird die Kurve rückläufig, so wird dieser Winkel größer 90°. Hieran ist also der Beginn einer Schleife erkennbar.

Eine (interne) Schleife zeichnet sich also dadurch aus, daß mindestens ein Wendepunkt existiert. Wird die Strecke s nun an diesem Wendepunkt w verkürzt, indem das Lot von w auf s gefällt wird, und $s' = [w_s v_e]$ (mit w_s Fußpunkt des Lotes) als neue Referenzstrecke bestimmt (Abb. 5.4 II), so liegt bezüglich s' nun eine externe Schleife vor, die wie oben erkannt wird.

Ein wesentlicher Nachteil dieser Implementierung ist ihre Asymmetrie. Sowohl die Douglas/Peucker- als auch die Persson/Jungert/Hernández-Generalisierung liefern beim Durchlaufen der Kurve von v_1 nach v_n die gleiche Generalisierung, wie beim Rückwärtsdurchlaufen von v_n nach v_1. Mit der genannten Korrektur ist diese Symmetrie zerstört.

In Abbildung 5.5 oben wird die Originalkurve von A nach B durchlaufen, unten in Gegenrichtung von B nach A. Das erste Bild (ganz links) zeigt eine Schleife in der Originalkurve, die vom unmodifizierten Douglas/Peucker-Algorithmus (ebenso wie vom Persson/Jungert/Hernández-Algorithmus) nicht erkannt würde.

[9] Die in [WW88] angegebene FORTRAN-Implementierung der Douglas/Peucker-Generalisierung tut dies, allerdings ohne daß es im Text nochmals erwähnt wird.

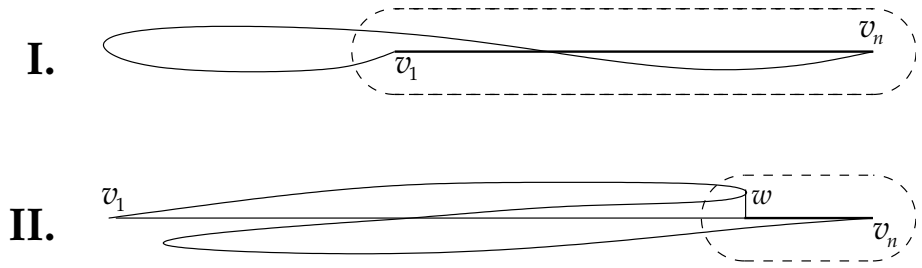


Abbildung 5.4.: Korrektur der Generalisierungsfehler

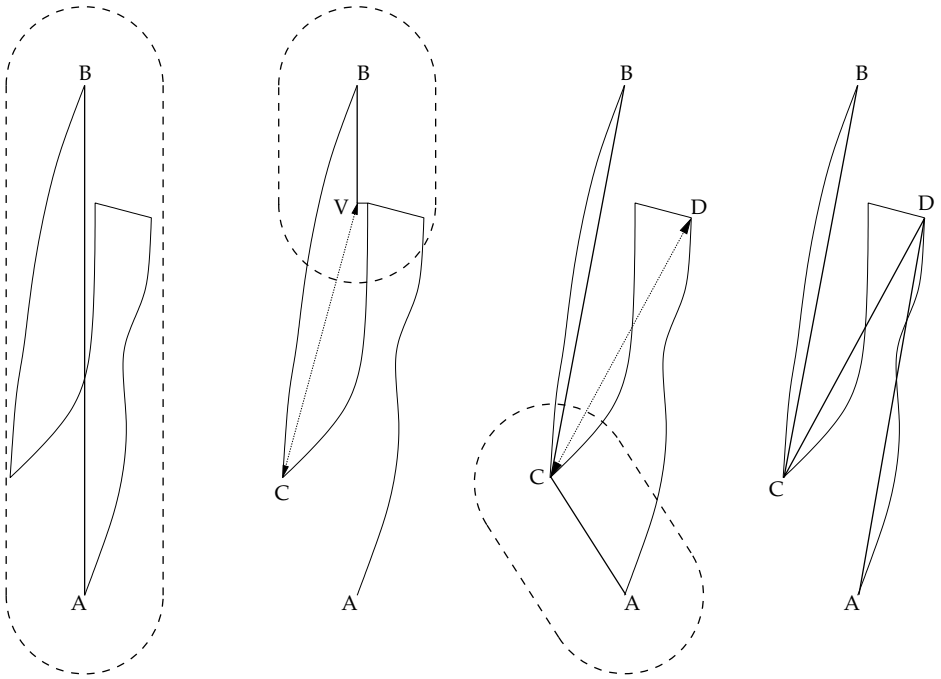
Hier greift nun die gerade beschriebene Korrektur. Bild zwei (oben) zeigt, daß beim Durchlaufen der Kurve von A aus ein Wendepunkt gefunden wird, der die Referenzstrecke von [AB] auf [VB] verkürzt. Für die zwischen dem Wendepunkt und B liegenden Punkte der Originalkurve wird also nun nicht der Abstand zu [AB] sondern zu [VB] bestimmt. Der am weitesten entfernte Punkt ist C. Da dessen Abstand größer als ε ist, wird der Algorithmus wie bekannt mit den Teilsegmenten AC und CB rekursiv aufzurufen. Im Teilstück CB ist nichts mehr zu tun, alle Punkte der Kurve liegen in der entsprechenden ε-Umgebung. Im Teilstück AC hingegen liegt D weiter als ε (und am weitesten) von [AC] entfernt. Dies führt (Bild ganz rechts) zur Generalisierung ADCB.

Die untere Bildreihe zeigt den Durchlauf von B nach A. Hier wird nun zunächst der andere Wendepunkt der Kurve gefunden, und die Referenzstrecke zu dessen Fußpunkt W auf [WA] verkürzt. Von [WA] liegt nun E am weitesten (und weiter als ε) entfernt. Folglich wird wie gehabt rekursiv mit den Teilstücken BE und EA weitergemacht. Teilstück EA ist fertig bearbeitet, alle Punkte liegen im zugehörigen ε-Korridor. Im Segment BE hingegen liegt der Punkt C am weitesten (und weiter als ε) von [BE] entfernt. Somit führt die Generalisierung in dieser Richtung zum Polygonzug BCEA, oder umgedreht: AECB. Beide Generalisierungen unterscheiden sich deutlich im Punkt D bzw. E, wie man in der Abbildung ganz rechts sieht.

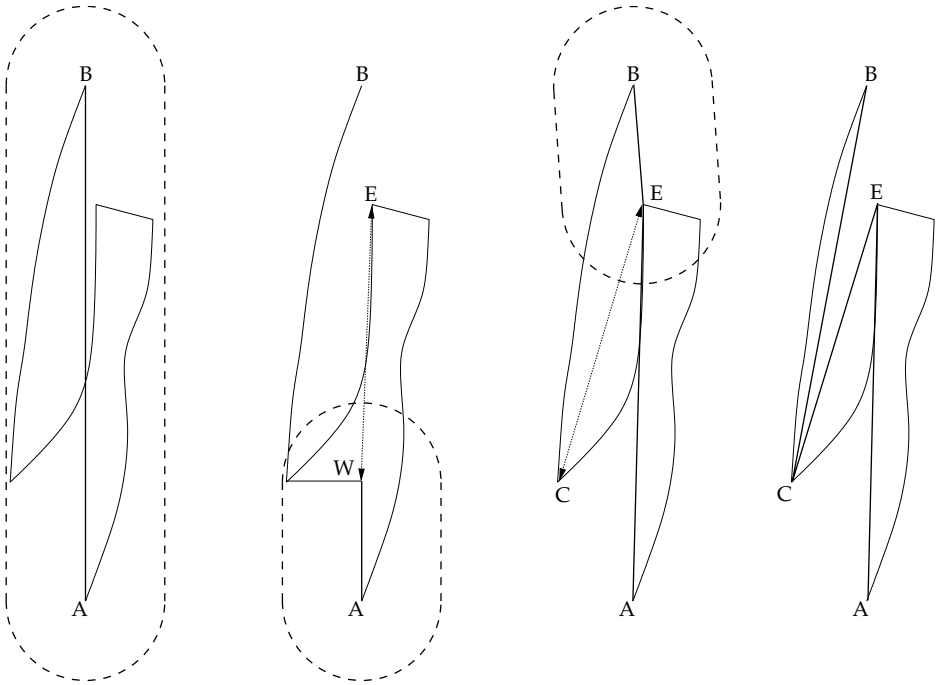
Für die Anwendung der Algorithmen auf Bewegungsspuren ist diese Asymmetrie keine wesentliche Einschränkung, da diese zeitlich gerichtet sind. Für Küstenlinien und andere Polygonzüge ist dies nicht der Fall, sie sind für gewöhnlich nicht gerichtet. Es sollte also keinen Unterschied machen, ob ich die norwegischen Fjorde von Bergen nach Narvik oder von Narvik nach Bergen entlanggehe.[10]

Allerdings sind die Unterschiede der Generalisierung in beiden Richtungen in der Praxis eher gering, auch in dem hier gezeigten konstruierten Beispiel wird beim Durchlaufen in beiden Richtungen der selbe Punkt C gefunden, dies in unterschiedlichen Schritten. Damit sich überhaupt Unterschiede ergeben, müssen sich

[10]Die Kartographie kommt allerdings bislang sehr gut ohne diese Schleifenerkennung aus. Sehr enge Schleifen sind auf Karten aufgrund der Strichdicke nicht mehr darstellbar.



Durchlaufen der Kurve von A nach B…



…und von B nach A.

Abbildung 5.5.: Asymmetrie durch Schleifen

der von [AC] und der von [AW] am weitesten entfernte Punkt unterscheiden, was in der Praxis bedeutet, daß ein Punkt D am weitesten von C und ein anderer Punkt E am weitesten von W entfernt liegt, wobei W den Fußpunkt des Lotes von C auf [AB] darstellt. Dies heißt, daß W somit zum einen weniger als ε von C entfernt liegt und zum anderen [WC] rechtwinklig zu [AB] liegt. Bedenkt man weiterhin, daß ein zu C bzw. W weiter als ε entfernt liegender Punkt sich ja weiterhin im ε-Korridor um [AB] befinden muß, wenn es sich um eine so enge Schleife handelt, daß sie vom unmodifizierten Algorithmus nicht erkannt worden wäre, so ist die Wahrscheinlichkeit doch sehr hoch, daß hier automatisch der gegenüberliegende Wendepunkt (oder zumindest ein direkt benachbart liegender Punkt) gefunden wird.

Natürlich läßt sich so auch ein echt symmetrisches Verfahren angeben. Jedes Segment enthält eine gerade Anzahl von Wendepunkten, hierbei dreht sich die Laufrichtung der Kurve an der einen Hälfte ihrer Wendepunkte bezüglich der Referenzgeraden $g = \overline{AB}$ nach hinten (vh), an der anderen Hälfte wieder nach vorne (hv) (wobei ich A bzw B als Wendepunkt bezeichne, wenn die Originalkurve an diesen Stellen rückläufig ist). Liegen nun die beiden äußersten Wendepunkte (W^1_{vh} und W^1_{hv}) zu weit auseinander ($|W^1_{vh}W^1_{hv}| > \varepsilon'$), so wird an ihnen segmentiert und der Algorithmus mit den Teilstücken AW^1_{vh}, $W^1_{vh}W^1_{hv}$ und $W^1_{hv}B$ erneut aufgerufen. Durch die Wahl von ε' wird gesteuert, ab welcher Größe Schleifen erkannt werden. Es bietet sich an, $\varepsilon' = \varepsilon$ zu setzen. Unschön ist lediglich, daß ε grundsätzlich mit dem Abstand eines Punktes zur Referenzgeraden verglichen wird. Wenn nun W^i_{vh} und W^i_{hv} beide auf der gleichen Seite von $\overline{AB}$ liegen (im Extremfall liegt $\overline{W^i_{vh}W^i_{hv}}$ parallel zu $\overline{AB}$), so hat der Abstand $d = W^i_{vh}W^i_{hv}$ für das Aussehen der Kurve eine völlig andere Bedeutung, als wenn W^i_{vh} und W^i_{hv} auf unterschiedlichen Seiten (und zudem im Extremfall $\overline{W^i_{vh}W^i_{hv}}$ annähernd rechtwinklig zu $\overline{AB}$) liegen. Abbildung 5.6 illustriert dies.

Die Kurve in Bild I besitzt keine Wendepunkte, alle Punkte befinden sich im ε-Korridor, die Generalisierung ist fertig. Die Kurve in Bild II sieht nur wenig anders aus, sie besitzt jedoch bezüglich der Referenzgeraden zwei Wendepunkte, die einen Abstand von knapp 2ε zueinander haben. Mit der Wahl $\varepsilon' = \varepsilon$ würde nun plötzlich eine Schleife erkannt. Um diesen Sprung zu vermeiden, liegt es nahe, $\varepsilon' = 2\varepsilon$ zu wählen, was allerdings dazu führt, daß die Schleife in Bild III nicht mehr erkannt wird, obwohl die asymmetrische Form des Algorithmus dies problemlos tun würde. Es gibt eine Reihe von Möglichkeiten, diese Problematik zu umgehen, beispielsweise indem nicht der Abstand der beiden Wendepunkte sondern der Abstand der zugehörigen Fußpunkte auf der Referenzgeraden genommen wird (hierbei $\varepsilon' = \varepsilon$).

Ein wesentlicher Unterschied zum oben angegebenen asymmetrischen Verfahren besteht darin, daß der asymmetrische Algorithmus von vorneherein an jedem rückläufigen Wendepunkt (vh) die Referenzstrecke verkürzt, wohingegen der Ansatz der Abstandsmessung zwischen Wendepunkten erst im letzten Schritt eingesetzt werden darf, wenn schon alle Punkte im ε-Korridor liegen, da er sonst die

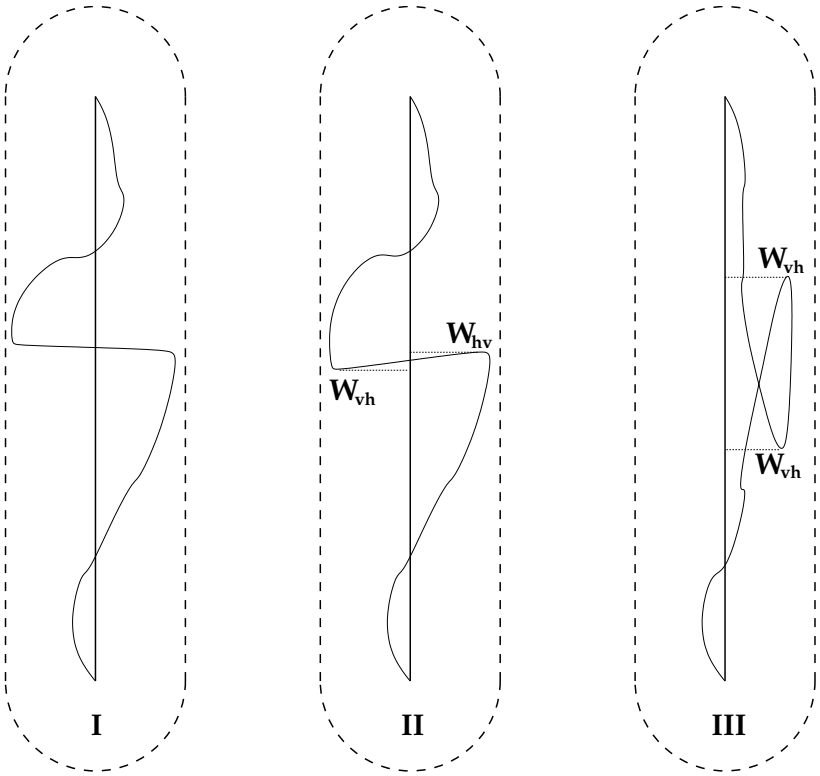


Abbildung 5.6.: Wendepunkte

„normale“ Generalisierung verfälscht.

Die symmetrische Erweiterung funktioniert also zwar, ist allerdings wesentlich weniger robust, man muß ein zusätzliches ε' bestimmen und in der konkreten Anwendung prüfen, welche Parameter sinnvoll sind und wie die Abstände am besten gemessen werden. Da das einfache und robuste asymmetrische Verfahren für die Verarbeitung von Bewegungsspuren sehr gut arbeitet, möchte ich diesen Exkurs hier schließen.

Unabhängig davon, ob man sich für die symmetrische oder asymmetrische Schleifenerkennung entscheidet, können für die Persson/Jungert/Hernández-Generalisierung unerwünschte Ergebnisse auftreten. Bestimmt man die vorzeichenbehaftete Distanz eines Punktes zu einer Geraden $g = \overline{v_a v_b}$, so treten keine Probleme auf, ein Punkt v_r mit $d_r = \text{dist}(g, v_r) > \varepsilon$ und ein Punkt v_l mit $d_l = \text{dist}(g, v_l) < -\varepsilon$ besitzen einen Abstand von mehr als 2ε zueinander ($|v_r v_l| > 2\varepsilon$).

Mißt man nun aber die Entfernung zur Strecke $s = [v_a v_b]$, so ist die Situation nicht mehr so einfach. Liegt für einen Punkt v_k der Fußpunkt l_k seines Lotes auf g in s ($l_k \in [v_a v_b]$), so ist die Sache klar, v_k liegt „neben“ s und hat zu s einen positiven oder negativen Abstand. Liegt v_k hingegen „vor“ v_a oder „hinter“ v_b ($l_k \notin [v_a v_b]$), so bestimmt sich $\text{dist}(s, v_k)$ zu $|v_a v_k|$ (bzw. $|v_b v_k|$).

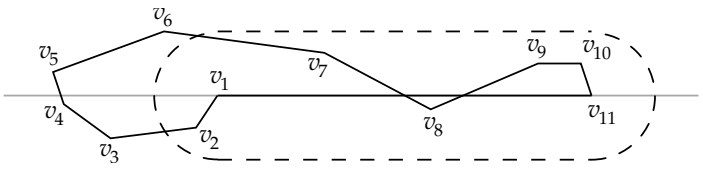


Abbildung 5.7.: Probleme mit Schleifen bei der Persson/Jungert/Hernández-Generalisierung

Natürlich ließe sich auch hier wieder ein „positiver“ bzw. „negativer“ Abstand definieren (für v_k rechts bzw links von g), dies kann jedoch zum in Abbildung 5.7 gezeigten Problem führen: Bei der Generalisierung des gezeigten Polygonzuges haben v_4 auf der einen Seite von $g = \overline{v_1 v_{11}}$ und v_5 auf der anderen Seite von g den maximalen „positiven“ und „negativen“ Abstand zu $s = [v_1 v_{11}]$, was zur Generalisierung $v_1 v_4 v_5 v_{11}$ führt, kein zufriedenstellendes Resultat.

Der Ansatz, einen vorzeichenbehafteten Abstand zu einer Strecke s zu definieren, muß hier scheitern. Während ein Punkt v_k bezüglich einer Geraden g entweder „rechts“ ($\text{dist}(g, v_r) \geq 0$)[11] oder „links“ ($\text{dist}(g, v_r) < 0$) von ihr liegt, so sind bezüglich einer Strecke $s = [v_a v_b]$ 4 unterschiedliche Lagen möglich: „rechts“ und „links“, wenn v_k „neben“ s liegt ($l_k \in [v_a v_b]$), „vor“ s, wenn l_k vor v_a liegt ($v_a \in [l_k v_b]$ und „hinter“ s, wenn l_k hinter v_b liegt ($v_b \in [v_a l_k]$ (Abb. 5.8 links). Dies liefert auch die Lösung des gezeigten Problems. Statt nur $\max_i\{\text{dist}(g, v_i)\}$ und $\min_i\{\text{dist}(g, v_i)\}$, also den rechts und links am weitesten entfernt liegenden Punkt zu bestimmen, werden der am weitesten „vor“, „links“, „hinter“ und „rechts“ liegende Punkt bestimmt. Der am weitesten entfernt liegende Punkt ist auf alle Fälle Teilungspunkt (wenn er weiter als ε entfernt liegt). Als zweiter Teilungspunkt wird nun der am weitesten entfernt liegende Punkt des gegenüberliegenden Bereiches gewählt (wenn dieser ebenfalls weiter als ε entfernt liegt), am Beispiel: liegt der am weitesten von s entfernte Punkt „vor“ s (und weiter als ε entfernt), so wird an ihm und an dem am weitesten „hinter“ s liegenden Punkt (sofern dieser ebenfalls weiter als ε von s entfernt liegt) geteilt, nicht aber an dem am weitesten „links“ oder „rechts“ liegenden Punkt.

Hierdurch wird sichergestellt, daß nicht zwei nebeneinanderliegende Punkte als Teilungspunkte gewählt werden, da zwei einander gegenüberliegende Bereiche mindestens 2ε voneinander entfernt liegen. Wählt man die in Abb. 5.8 rechts gezeigten Bereiche, vergrößert man die Wahrscheinlichkeit dafür, daß ein Punkt „rechts“ oder „links“ von s liegt, was wiederum die Wahrscheinlichkeit erhöht, zwei gültige Teilungspunkte zu bekommen, damit die Persson/Jungert/Hernández-Generalisierung ihren Vorteil gegenüber der Douglas/Peucker-Generalisierung ausspielen kann, die links gezeigten Bereiche stellen dagegen eine bessere Trennung sicher und versprechen somit bessere Generalisierungen.

[11] wir nehmen die 0 der Einfachheit halber hier hinzu

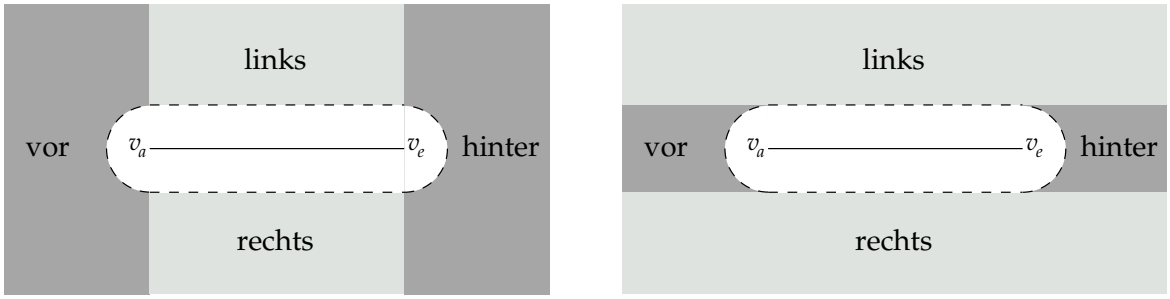


Abbildung 5.8.: Mögliche Lage eines Punktes bezüglich einer Strecke $s = [v_a v_b]$

5.4. Komplexität und Optimierungen

Beide Algorithmen haben (wie die meisten Divide-and-Conquer Verfahren) im Mittel eine (Zeit-)Komplexität von $O(n \log n)$ und im schlechtesten Fall eine Komplexität von $O(n^2)$. Abbildung 5.9 zeigt eine Kurve, bei der beide Algorithmen ein quadratisches Laufzeitverhalten zeigen, da $n + (n-1) + (n-2) \ldots = 15 + 14 + 13 + \ldots$ bzw. $n + (n-2) + (n-3) \ldots = 15 + 13 + 12 + \ldots$ Punkte betrachtet werden müssen. Links oben ist das Originalpolygon mit den Eckpunkten v_1 bis v_{15} abgebildet, in der rechten Spalte die ersten vier Schritte der Douglas/Peucker-Generalisierung und in der linken Spalte die ersten drei Schritte der Persson/Jungert/Hernández-Generalisierung. Letztere kann hier lediglich im ersten Schritt den Vorteil ausspielen, die Kurve in drei statt zwei Teile teilen zu können. Danach laufen beide Algorithmen exakt gleich ab.

Weiterhin ist für die Komplexitätsabschätzung das Verhältnis zwischen der Gesamtlänge (genauer: der Anzahl der Punkte) der Originalkurve zur Anzahl der Punkte pro Segment wichtig, das natürlich zum einen von der Form der Kurve und zum anderen von der Wahl des ε abhängt. Hierzu ist wichtig zu betrachten, wie (im besten Fall) $O(n \log n)$ zustandekommt. Da sich beide Algorithmen hier strukturell nicht unterscheiden, betrachte ich den Douglas/Peucker-Algorithmus: Im ersten Schritt werden all n Punkte einmal durchlaufen, um den zur Startgeraden $g = \overline{v_1 v_n}$ maximal entfernten Punkt zu finden. Dann wird die Kurve an diesem Punkt geteilt (optimalerweise ist dies die Mitte), und nun also zweimal je $n/2$ Punkte durchlaufen, dann viermal $n/4$ Punkte usw. Dies führt zur oben angegebenen Komplexität von $n \log_2 n$ (bzw. für den Persson/Jungert/Hernández-Algorithmus $n \log_3 n$ im optimalen Fall), wie beispielsweise auch bei Quicksort. Im Unterschied zu Quicksort terminieren diese Algorithmen hier nicht erst, wenn in einem Teilstück nur noch 2 Punkte vorhanden sind, sondern bereits, sobald alle Punkte eines Teilstücks im zugehörigen ε-Korridor liegen. Bestehen nun diese Teilstücke im Mittel aus k Punkten, so werden nicht $\log_2 n$ sondern lediglich $\log_2(n/k)$ Rekursionen benötigt, insgesamt also $n \log_2(n/k)$, da ja weiterhin in jedem Schritt alle n Punkte betrachtet werden müssen.

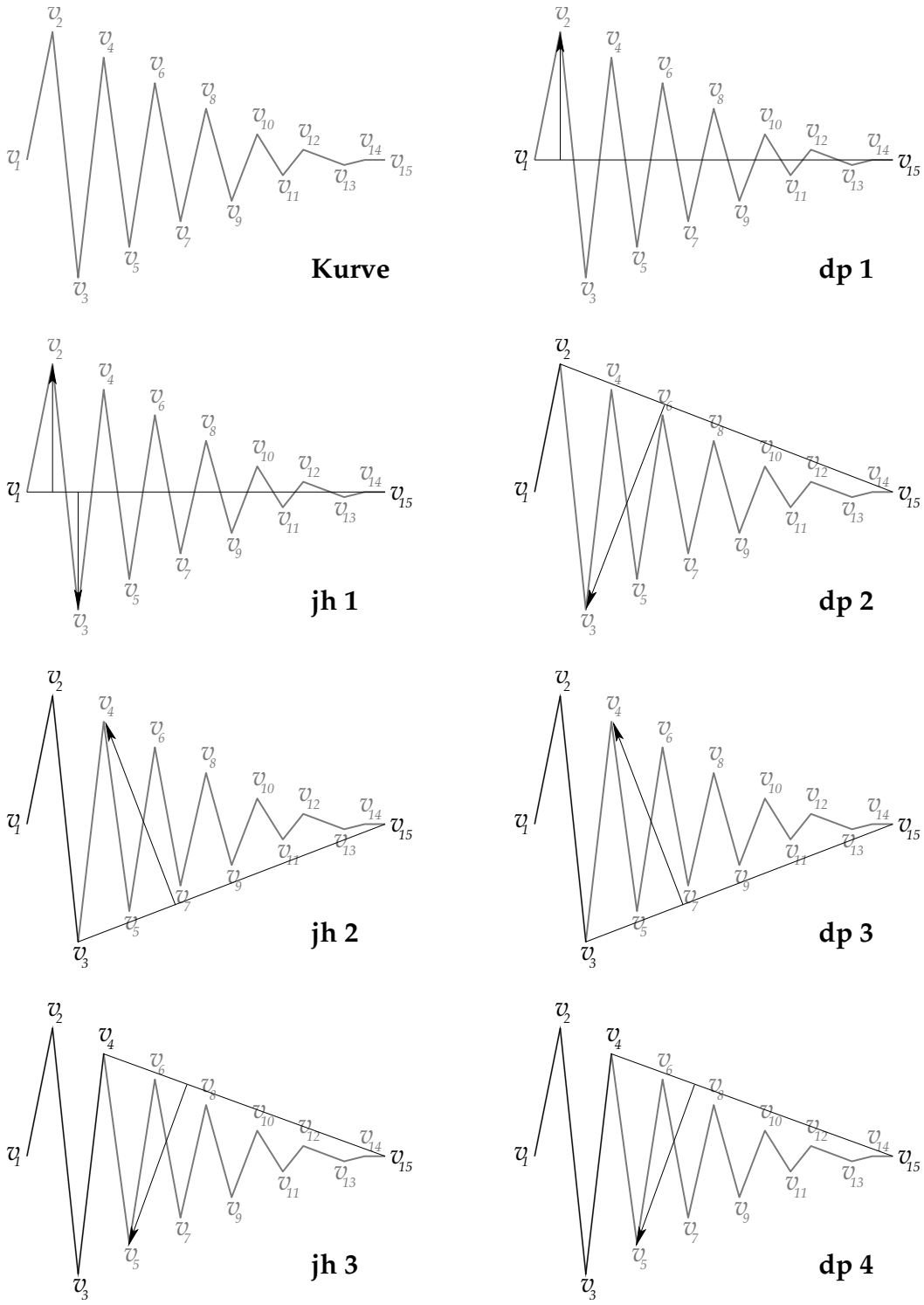


Abbildung 5.9.: Worst Case

Was bedeutet dies nun konkret? Für eine doppelt so lange Kurve ($2n$ Punkte) benötigt der Algorithmus $(2n)\log_2((2n)/k)$ Schritte, also weiterhin $O(n \log n)$. Ist die Kurve hingegen doppelt so genau, besitzt also jedes Teilstück $2k$ Punkte, so benötigt er $(2n)\log_2((2n)/(2k)) = (2n)\log_2(n/k)$ Schritte. In diesem Fall bleibt also der logarithmische Teil konstant, somit $O(n)$. Alle Parameter, die also (bei gleichbleibender Anzahl von Punkten pro Segment) die Anzahl der Segmente erhöhen, wie Verlängerung der Kurve oder stärkere Krümmung im Großen, verändern die Komplexität folglich mit $O(n \log n)$, wogegen Parameter, die die Anzahl der Punkte pro Segment ändern, wie die Größe von ε, die Feinstruktur der Kurve oder die Auflösung[12] der Originalkurve, nur mit $O(n)$ eingehen.

Da die Douglas/Peucker-Generalisierung wie oben erwähnt einer der meistverwandten Algorithmen in der Geographie ist, existieren zu ihm eine Reihe von Optimierungen. So beschreiben beispielsweise John Hershberger und Jack Snoeyink in [HS92, HS97] einen Algorithmus, der mit erhöhtem Overhead, dafür allerdings in garantierten $O(n \log n)$ auch im worst case die gleiche Generalisierung wie Douglas/Peucker liefert. Das dort beschriebene Verfahren sollte auch problemlos auf die Persson/Jungert/Hernández-Generalisierung übertragbar sein. Daß ich an dieser Stelle nicht weiter darauf eingehe, liegt daran, daß es nur auf Polygonzügen korrekt arbeitet, die sich nicht selbst schneiden. Diese Forderung, die in der Geographie – bei Küstenlinien, Flußläufen oder Landesgrenzen – meist problemlos erfüllbar ist, stellt hier eine wesentliche Einschränkung dar. Betrachtet man beispielsweise die Bewegung eines Fahrzeugs, das rangieren muß, um durch eine Engstelle zu kommen, so wird diese Rangierbewegung zu mehrfachen Überschneidungen der Bewegungsspur mit sich selbst führen. Weiterhin kann dieser Algorithmus seine Stärke gegenüber dem Douglas/Peucker-Algorithmus erst bei sehr hohen Punktzahlen (ab 10000 Punkten) ausspielen.

Die Douglas/Peucker- (und ebenso die Persson/Jungert/Hernández-) Generalisierung selbst läßt sich durch geschickte Programmierung ebenfalls optimieren (wobei die Komplexität gleichbleibt). Und wie in den meisten Fällen geht es auch hier um die Optimierung der innersten Schleife, und dort um die Berechnung von $d' = |\mathrm{dist}(g, v_k)|$. g ist eine Schleifeninvariante, d. h. für den zweidimensionalen Fall, die Berechnung des (auf Länge 1 normierten) Normalenvektors n^0 zu $g = \overline{v_1 v_n}$ läßt sich aus der Schleife ziehen, womit $d' = |(v_k - v_1) \circ n^0| = |(x_{v_k} - x_{v_1})x_{n^0} + (y_{v_k} - y_{v_1})y_{n^0}|$ lediglich 3 Additionen/Subtraktionen und 2 Multiplikationen enthält. Ohne Betragsfunktion erhält man eine vorzeichenbehaftete Distanz, was für den Persson/Jungert/Hernández-Algorithmus sehr schön ist, wo ja für beide Seiten der maximal entfernte Punkt gesucht wird.

Seltsamerweise ist der in [WW88][13] angegebene FORTRAN-Code für Dou-

[12] also wie eng die Punkte der Originalkurve liegen

[13] Die Originalimplementierung von Douglas und Peucker liegt mir leider nicht vor, da sie den Algorithmus in ihren Veröffentlichungen nur beschreiben, jedoch nicht angeben. Sie schreiben

glas/Peucker nicht nur reichlich unlesbar (FORTRAN ist zur Programmierung rekursiver Algorithmen wie man hier sieht sehr schlecht geeignet), sondern zudem noch ineffizient programmiert: Zunächst wird die Position des Fußpunktes des Lotes von v_k auf g bestimmt und dann die Entfernung von v_k zu seinem Fußpunkt berechnet. Insgesamt benötigt er 20 Additionen/Subtraktionen, 10 Multiplikationen und 1 Division, um das Quadrat der Entfernung von v_k zu g zu bestimmen. Immerhin kann das (teure) Wurzelziehen in dieser innersten Schleife vermieden werden, da aufgrund der Monotonie von $x \to x^2$ der Punkt mit maximaler quadrierter Distanz gesucht werden kann. Allerdings wird dann aus diesem[14] doch die Wurzel gezogen, um d' mit ε zu vergleichen, anstatt, wie auch Hershberger und Snoeyink vorschlagen, am Anfang einmal ε^2 zu berechnen, und dann jeweils d'^2 mit ε^2 zu vergleichen, was alles Wurzelziehen vermeidet.

Doch auch mit dieser Optimierung bleibt diese Implementierung weit schlechter als die von mir angegebene mit Hilfe des Normalenvektors, und dies selbst dann, wenn man berücksichtigt, daß zur Berechnung von $d' = |\text{dist}(s, v_k)|$ noch zusätzlich 5 Additionen/Subtraktionen und 3 Multiplikationen nötig sind (insgesamt also 8 Additionen/Subtraktionen und 5 Multiplikationen). Für die Erkennung von Schleifen innerhalb der ε-Umgebung von s kommen für jeden Wendepunkt jeweils noch die Kosten für die Verschiebung des Start- bzw. Endpunktes von s hinzu (wobei diese Schleifenerkennung wie erwähnt in der Originalimplementierung gar nicht erfolgt). Richtungsvektor r_s und Normalenvektor n_0 müssen hierbei nicht neu berechnet werden.

Für den zweidimensionalen Fall schlägt [Ram72] eine noch weitergehende Optimierung vor, die jedoch nur anwendbar ist, wenn man auf jegliche Schleifenerkennung verzichten kann: Sei $g = \overline{v_1 v_n}$ eine nicht senkrechte Gerade, so kann ich g in der Form $y = mx + t$ schreiben. Für einen beliebigen Punkt $a = (x_a, y_a)^T$ ist nun $|mx_a + t - y_a|$ der y-Abstand von a zu g. Der Punkt mit dem maximalen y-Abstand zu g besitzt auch den maximalen kartesischen Abstand zu g. Somit läßt sich mit 1 Addition, 1 Subtraktion und 1 Multiplikation der zu g maximal entfernte Punkt finden (m und t müssen für jedes Segment lediglich einmal bestimmt werden). Da dies nur für nicht senkrechte Geraden geht, und für große m die Genauigkeit sinkt, wird bei Geraden mit mehr als 45° Steigung einfach der x-Abstand berechnet.

5.5. Generalisierung ohne Stack

Neben dem oben angegebenen rekursiven Verfahren geben Douglas und Peucker in [DP73] ein weiteres Verfahren an, das auf den gleichen Prinzipien beruht. Um einen Polygonzug $p = v_1 v_2 \dots v_n$ zu generalisieren, wird wie gehabt

lediglich in [DP73], daß sie ihn in FORTRAN IV und zusätzlich eine rekursive Version in ALGOL W programmiert haben.

[14] jetzt immerhin nur für einen Punkt des Segments

Algorithmus 3: Stacklose Douglas/Peucker Generalisierung

```
 1: function dpnostack = (Polygonzug p = v_1 ... v_n, Distanz ε) → Polygonzug :
 2: begin
 3:   m ← n;
 4:   repeat
 5:     l ← m;
 6:     m, d ← findmaxdist(v_1v_l, v_1v_2 ... v_l);   ▷ Entferntesten Punkt finden
 7:   until (d ≤ ε)
 8:   if (l < n) then
 9:     return v_1 ∘ dpnostack(v_l ... v_n, ε);   ▷ Nächstes Teilstück
10:   else
11:     return n;                                  ▷ fertig
12:   end if
13: end
```

der zu $[v_1v_n]$ am weitesten entfernt liegende Punkt v_{k_1} gesucht. Liegt dieser weiter als ε entfernt, wird der Polygonzug auf $v_1v_2 \ldots v_{k_1}$ verkürzt, der zu $[v_1v_{k_1}]$ am weitesten entfernt liegende Punkt v_{k_2} bestimmt, wieder überprüft, ob dieser weiter als ε entfernt liegt, usw., bis schließlich für einen Punkt v_{k_m} das gesamte Teilstück $v_1v_2 \ldots v_{k_m}$ in der ε-Umgebung zu $[v_1v_{k_m}]$ liegt. Dies ist Abbruchbedingung, $[v_1v_{k_m}]$ ist die Generalisierung des ersten Segments und der Algorithmus arbeitet auf dem Teilstück $v_{k_m}v_{k_m+1} \ldots v_n$ weiter (Algorithmus 3).

Der erste so gefundene Punkt v_{k_m} ist mit dem ersten durch den rekursiven Algorithmus (Alg. 1: dpgen) gefundenen Punkt identisch, die weiteren Punkte können sich jedoch unterscheiden. Strukturell bietet der stackfreie Algorithmus keine Vorteile gegenüber der rekursiven Version, er liefert keine bessere Generalisierung und besitzt eine wesentlich schlechtere Laufzeit, da ja große Teile des Polygonzuges für jeden Punkt erneut durchlaufen werden müssen.

Ich gebe die stackfreie Variante (Alg. 3: dpnostack) an dieser Stelle auch nur an, weil sie zeigt, wie sehr die vorhandenen Werkzeuge das algorithmische Denken prägen: Das Paper von Douglas und Peucker stammt aus dem Jahr 1973, und die Algorithmen wurden zunächst in FORTRAN IV implementiert, das Rekursion nicht unterstützt. Die FORTRAN-Implementierung von Algorithmus 1 (dpgen) ist daher zwangsläufig auch nicht rekursiv, sondern arbeitet den Polygonzug in verschachtelten WHILE-Schleifen ab, wobei der Status der einzelnen Punkte in einem großen Array (10000 Punkte) von Hand verwaltet wird.

Dieses nicht-rekursive Herangehen zeigt sich auch in den Beschreibungen des (rekursiven) Algorithmus dpgen in der Arbeit[15] von Douglas und Peucker, der

[15] [DP73]

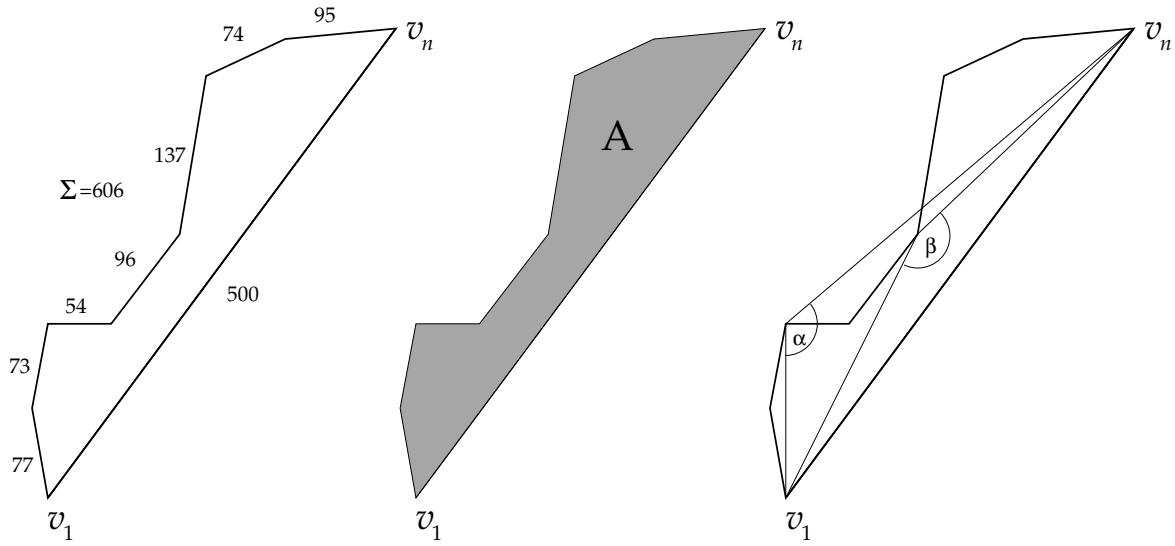


Abbildung 5.10.: Weitere Ähnlichkeitsmaße

nicht als Divide-and-Conquer-Verfahren sondern als sequentielles Verfahren mit Zwischenspeicher beschrieben wird. Unter diesen Voraussetzungen wird klar, wieso Douglas und Peucker den Algorithmus dpnostack überhaupt angeben.

5.6. Andere Abstandsmaße

Die Douglas/Peucker-Generalisierung besitzt eine sehr einfache Grundstruktur:

- Zunächst wird geprüft, ob der Polygonzug $p = v_1 v_2 \dots v_n$ bezüglich eines bestimmten Ähnlichkeitsmaßes (dem maximalen Abstand ε) der Strecke $s = [v_1 v_n]$ ähnlich genug ist. Ist dies der Fall, terminiert die Verarbeitung.
- Andernfalls wird p an einer ausgewählten Stelle v_k (dem maximal entfernten Punkt) geteilt und der Algorithmus auf den Teilstücken $p_l = v_1 v_2 \dots v_k$ und $p_r = v_k v_{k+1} \dots v_n$ rekursiv aufgerufen.

Wie man sieht, ist diese Grundstruktur vom gewählten Ähnlichkeitsmaß unabhängig. Sowohl die Entscheidung, ob p hinreichend ähnlich zu s ist (die Verarbeitung terminiert), als auch die Entscheidung, an welchem v_k andernfalls geteilt wird, kann grundsätzlich nach beliebigen Kriterien getroffen werden, ohne dadurch an der Struktur des Algorithmus irgendetwas zu ändern[16] (allenfalls in der Behand-

[16]Solange als Teilungspunkt v_k ein zwischen Start und Endpunkt liegender Punkt gewählt wird, und die Bearbeitung spätestens abgebrochen wird, wenn in einem Segment nur noch 2 Punkte vorhanden sind, terminiert der Algorithmus sicher, da die Segmente in jedem Schritt kürzer werden. Die Frage dabei ist lediglich, welche Kriterien hier sinnvoll sind, strukturell gesehen könnte man beides auch auswürfeln.

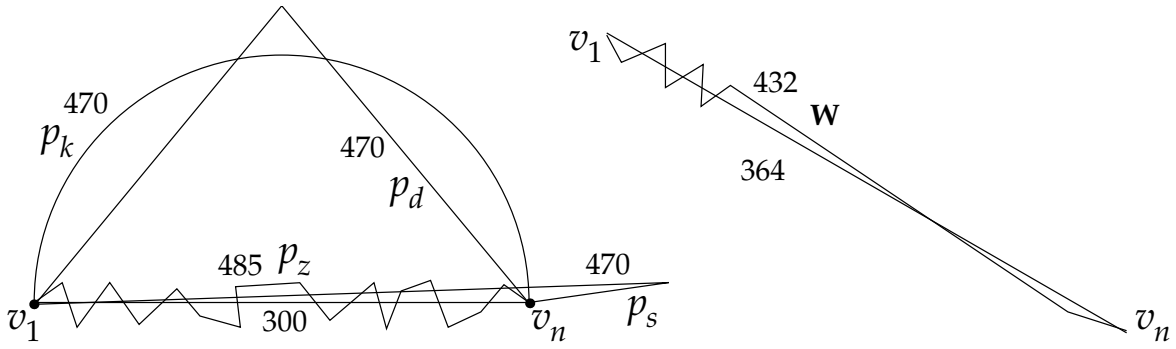


Abbildung 5.11.: Längenmessung

lung von Schleifen könnten Änderungen erforderlich werden).

Statt also den Abstand $d = |\text{dist}(s, v_k)|$ eines Punktes v_k zur Strecke $s = [v_1 v_n]$ zu bestimmen und dann zu prüfen, ob dieser Abstand kleinergleich ε ist (das ε-Ähnlichkeitsmaß), kann hier ein beliebiges anderes Maß eingesetzt werden. Abbildung 5.10 zeigt drei Maße am Beispiel (Polygonzug $p = v_1 v_2 \dots v_n$, Strecke $s = [v_1 v_n]$): links das Längenverhältnis, in der Mitte die Fläche zwischen den Kurven und rechts das Winkelmaß. Diese drei und darüber hinaus noch weitere Maße werden in den folgenden Abschnitten vorgestellt.

5.6.1. Längenverhältnis

War die Bewegung von v_1 nach v_n nicht sehr geradlinig, läuft z. B. ein junger Hund dauernd um sein Frauchen herum, während die beiden sich von v_1 nach v_n bewegen, so läuft er dabei ein Mehrfaches der direkten Strecke von v_1 nach v_n. Bewegt sich jemand hingegen in einem leichten Bogen von v_1 nach v_n oder geht nur an einer Stelle um ein Hindernis herum, so erhöht sich die zurückgelegte Entfernung dadurch nur leicht. Das Verhältnis $\frac{|p|}{|s|}$ der Länge $|p| = \sum_{i=1}^{n-1} |v_i v_{i+1}|$ des Polygonzuges p zur Strecke s ist also ein Maß für die Geradlinigkeit der Bewegung (in Abbildung 5.10 ganz links: Der Originalpolygonzug p besitzt eine Länge[17] von 606, die Strecke $s = [v_1 v_n]$ eine Länge von 500, das Längenverhältnis beträgt also $\frac{|p|}{|s|} = \frac{606}{500} = 1.2121$), je größer das Längenverhältnis ist, desto mehr „Umweg“, den die Bewegungsspur auf dem Weg von v_1 nach v_n beschreibt. Es gibt jedoch nicht wieder, *wie* dieser Umweg aussieht.

Genau darin liegt auch die Schwäche dieses Maßes. Die Feinstruktur der Bewegung hat hier wesentlich mehr Einfluß als die Form der Kurve im Großen. Abbildung 5.11 links zeigt den Weg von v_1 nach v_n in Form einer Zickzacklinie (p_z), eines Dreiecks (p_d), Kreisbogens/Halbkreises (p_k) und einer Schleife (p_s). Das Verhältnis

[17] Ich gebe Längen etc. hier ohne Einheiten an, da es nur auf das relative Verhältnis ankommt.

$\frac{|p|}{|s|}$ ist hierbei für p_d, p_k und p_s identisch, wogegen $\frac{|p_z|}{|s|}$ schlechter ist als diese.

Auf den ersten Blick scheint dies also zumindest ein gutes Maß dafür zu sein, wie verwinkelt und verschnörkelt eine Bewegung ist. Allerdings kann es nicht zwischen einem großen Umweg (z. B. dem Kreisbogen p_k) und einem Zickzackkurs p_z unterscheiden. Es gibt lediglich an, das Wievielfache der direkten Entfernung von v_1 nach v_n zurückgelegt wurde. Etwas interessanter wird es in Kombination mit dem ε-Korridor: Liegt eine Bewegungsspur $\tilde{p}$ von v_1 nach v_n vollständig in der ε-Umgebung zu $s = [v_1 v_n]$ (inklusive Schleifenerkennung) und ist zudem $\frac{|\tilde{p}|}{s}$ sehr groß (z. B. $\frac{|\tilde{p}|}{|s|} > 2$), so liegt eine im Großen geradlinige und im Kleinen sehr „kurvige“ Bewegung vor.

Abbildung 5.11 rechts zeigt, daß auch dies nur unvollkommen gelingt: der Algorithmus hätte hier keine Möglichkeit, zwischen dem gezackten und dem geradlinigen Teil der Bewegung zu unterscheiden, da das Verhältnis der Gesamtlängen als Maß genommen wird ($432/364 \approx 1.2$). Dieses Maß macht also zwar eine Aussage über die Feinstruktur der Bewegung, ist aber (zumindest angewandt in der Douglas/Peucker-Generalisierung) nicht geeignet, nach dieser zu differenzieren.

Ist jedoch aus anderen Gründen klar, daß z. B. solche Zickzackbewegungen gar nicht auftreten können, oder ist man wirklich nur an der Länge des Umwegs interessiert, unabhängig davon, wie diese zustande kommt, so mag dies ein geeignetes Maß sein. Hier ist dann noch zu klären, an welcher Stelle p geteilt wird. Es bieten sich hier zwei Punkte zur Teilung an:

- Mitte des Polygonzuges p: wähle den Punkt v_k, für den die Teilpolygonzüge $v_1 v_2 \ldots v_k$ und $v_k v_{k+1} \ldots v_n$ ungefähr gleich lang sind:

$$\sum_{i=1}^{k-1} |v_i v_{i+1}| \approx \sum_{i=k}^{n-1} |v_i v_{i+1}|.$$

- Mitte der Strecke s: wähle den Punkt v_k, für den der Fußpunkt w_k seines Lotes auf s die Strecke s möglichst mittig teilt:[18]

$$|v_1 w_k| \approx |w_k v_n| \quad \text{mit } w_k \text{ Fußpunkt von } v_k \text{ auf } s.$$

5.6.2. Fläche

Das nächste Maß (Abb. 5.10 Mitte) für den Unterschied zwischen dem Polygonzug p und der Strecke s ist die Fläche unter der Kurve, also die Fläche zwischen p und s (in Relation zu $|s|$).[19] Auch dieses Maß ist für die Verarbeitung von Bewegungsspuren wenig geeignet, wie Abbildung 5.12 (links) zeigt: Die Fläche zwischen p_z und

[18] Im Fall von Schleifen in p können hier mehrere Kandidaten auftreten.

[19] siehe aber auch Abschnitt 5.6.6

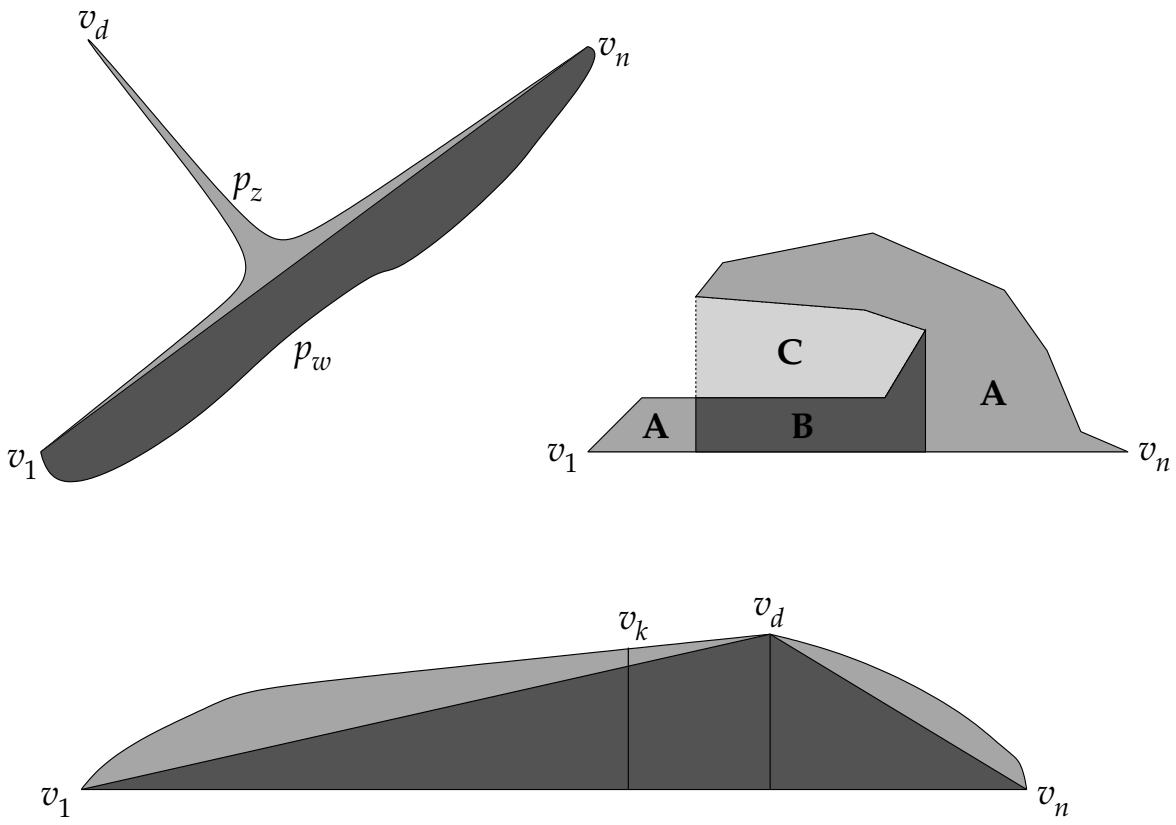


Abbildung 5.12.: Flächenmessung

s ist wesentlich kleiner als zwischen p_w und s, obgleich v_d sehr weit von s entfernt liegt. Dies liegt daran, daß beim Abstecher nach v_d Hin- und Rückweg fast identisch sind, so daß die Fläche zwischen ihnen klein bleibt. Die Entfernung $d = |\text{dist}(s, v_d)|$ wirkt sich hier nur wenig aus, da Fläche ein zweidimensionales Maß ist, und die Breite hier sehr gering ist.

In der Kartographie ist das häufig ein gewollter Effekt, da sie ja auch zweidimensionale Objekte abbildet. Hier sollen schmale Flußmündungen oder kleine Buchten etc. häufig weggeneralisiert werden. In der Verarbeitung von Bewegungen interessiert normalerweise, *wo* jemand entlanggegangen ist, und nicht, wie groß die Fläche ist, die er dabei umlief.

Darüberhinaus ist zu entscheiden, *welche* Fläche gemessen wird. Der Flächeninhalt des Polygons in Abb. 5.12 rechts beträgt A + B. Addiert man jedoch die Beträge der Flächen zwischen s und den einzelnen Teilstücken $[v_i v_{i+1}]$ ($i \in \{1, 2, \ldots, n-1\}$) des Originalpolygonzuges auf, so erhält man A + 3B + 2C. Hierdurch haben Schleifen eine stärkere Wirkung und werden somit besser erkannt.

Zur Teilung von p bei einer Generalisierung mittels Flächenmaß bieten sich

zwei Punkte an. Zum einen die Halbierung der Fläche: Gesucht ist ein Punkt v_k mit w_k Fußpunkt des Lotes von v_k auf s, so daß die Fläche des Polygons $[v_1v_2 \dots v_kw_kv_1]$ annähernd gleich der Fläche des Polygons $[v_kv_{k+1} \dots v_nw_kv_k]$ ist. Leider hält diese Teilung nicht, was sie verspricht. Zwar wird die Fläche zwischen p und s schon halbiert, in der nächsten Rekursionsstufe interessiert s aber nicht mehr, nun wird mit $p' = v_1v_2 \dots v_k$ auf $s' = [v_1v_k]$ und $p'' = v_kv_{k+1} \dots v_n$ auf $s'' = [v_kv_n]$ weitergearbeitet, und die Fläche zwischen s' und p' ist sicher nicht die Hälfte der Fläche zwischen s und p.

Sinnvoller wäre also, einen Punkt v_p so zu bestimmen, daß die Summe der Flächen zwischen $v_1v_2 \dots v_p$ und $[v_1v_p]$ und zwischen $v_pv_{p+1} \dots v_n$ und $[v_pv_n]$ minimal wird. Hierzu müßten allerdings alle n Punkte durchprobiert und die jeweiligen Flächen berechnet werden, was zusätzliche Kosten verursacht und die Komplexität erhöht, da zur Bestimmung der Fläche zwischen $v_1v_2 \dots v_p$ und $[v_1v_p]$ wieder p Punkte betrachtet werden müssen.[20]

Für den am weitesten von $g = \overline{v_1v_n}$ entfernt liegenden Punkt v_d gilt, daß die Fläche des Dreiecks $\triangle v_1v_dv_n$ maximal ist (die Fläche aller anderen Dreiecke $\triangle v_1v_iv_n$ mit $\text{dist}(g, v_i) < \text{dist}(g, v_d)$ ist geringer). Für konvexe Polygone $v_1v_2 \dots v_nv_1$ ist damit bei einer Teilung im Punkt v_d die Fläche der resultierenden Teilstücke minimal (Abb. 5.12 unten), und daher v_d der optimale Punkt zur Teilung. Zwar gilt dies nicht allgemein (z. B. in Abb. 5.12 links oben), die Teilung an p_k stellt dennoch eine gute Heuristik dar.

5.6.3. Winkel

Der Winkel γ_p, unter dem ein Punkt v_p bezüglich $s = [v_1v_n]$ erscheint (Abb. 5.10 rechts oben), ist ein weiteres Maß für den Unterschied zwischen p und s. Zu jedem Punkt v_i wird der Winkel $\gamma_i = \angle v_1v_iv_n$ bestimmt. Sind alle γ_i stumpfer als ein gegebener Winkel $\gamma^{\min}$ (das Ähnlichkeitsmaß), so ist s eine Generalisierung von p, andernfalls muß unterteilt werden. Als Teilungspunkt bietet sich hier der Punkt v_k mit minimalem Winkel $\gamma_k = \angle v_1v_kv_n$ an.

Wie das ε-Ähnlichkeitsmaß liefert auch $\gamma^{\min}$ einen Akzeptanzbereich um s, in dem alle Punkte v_i liegen müssen (siehe Abbildung 5.13), eine Linse aus zwei Kreisbögen mit den Spitzen v_1 und v_n. Den Beweis, daß alle Punkte v_i mit glei-

[20] Um dies zu vermeiden, läßt sich die Fläche auch aus der Differenz der schon aus der Berechnung der Gesamtfläche bekannten Fläche $A_{1,p} = A(v_1v_2 \dots v_pw_pv_1)$ (w_p Fußpunkt des Lotes zu v_p) und des Dreiecks $\triangle v_1v_pw_p$ bestimmen. Hierzu müssen alle Schnittpunkte von $[v_1v_p]$ mit $v_1v_2 \dots v_p$ bestimmt werden, was beschleunigt werden kann, wenn p wenig Wendepunkte besitzt, da jedes in eine Richtung gebogene Teilstück $v_fv_{f+1} \dots v_g$ (d. h. $\forall i \in \{f, f+1, \dots g-2\}$: $\angle v_iv_{i+1}v_{i+2} \leq 180°$ (nach rechts gebogen, analog ($\geq 180°$) nach links gebogen)) mit v_1v_p maximal zwei Schnittpunkte besitzt. Weitere Optimierungen sind durch die Berechnung von konvexen Hüllen über p möglich, eine Erhöhung der Komplexität ist aber im Allgemeinen nicht vermeidbar.

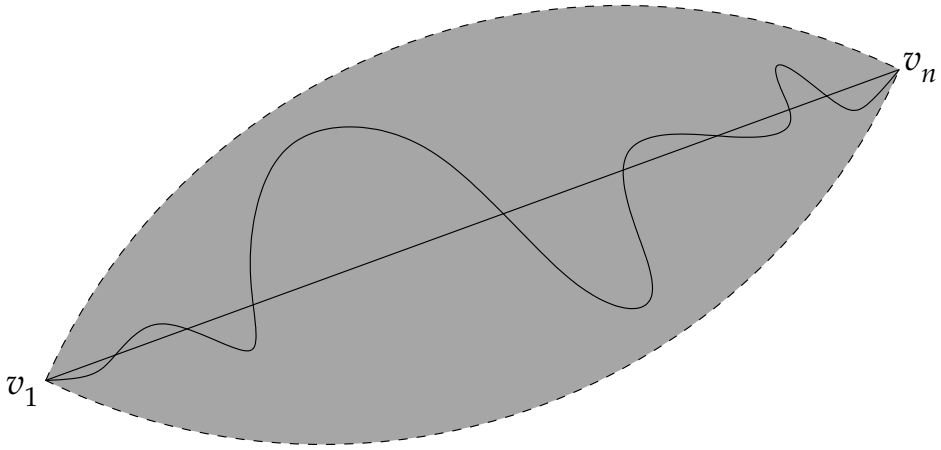


Abbildung 5.13.: Akzeptanzbereich für $\gamma^{\min} = 130^\circ$

chem $\gamma_i = \angle v_1 v_i v_n$ auf einem Kreisbogen durch v_1 und v_n liegen, liefere ich in Abbildung 5.14. Dieser Akzeptanzbereich macht für die Verarbeitung von Bewegungsbahnen wieder mehr Sinn. Interpretiert man den Polygonzug $p = v_1 v_2 \dots v_n$ als Aussage darüber, daß sich das bewegte Objekt zunächst am Ort v_1, dann am Ort v_2, usw. befunden hat, aber unbekannt ist, wie es sich von v_1 nach v_2 bewegt hat – normalerweise wird dies nicht exakt die Strecke $s_{1,2} = [v_1 v_2]$ sein, sondern eher etwas leicht gebogenes – so nimmt man lediglich an, daß die Bewegung zwischen v_1 und v_2 nicht zu sehr von $s_{1,2}$ abweicht (andernfalls wäre die Abtastrate zu gering). Generalisiert man nun mit diesem Winkelmaß, so erhält man einen generalisierten Polygonzug $q = w_1 w_2 \dots w_m$, für den wiederum gilt, daß die Bewegung zwischen w_k und w_{k+1} nicht zu weit von $[w_k w_{k+1}]$ abweicht.

Der größte Nachteil der Linsenform besteht darin, daß an den Rändern v_1 und v_n selbst minimale Bewegungen in die falsche Richtung dazu führen, daß die Spur nicht mehr im Akzeptanzbereich liegt. Daher schlage ich ein weiteres Maß vor.

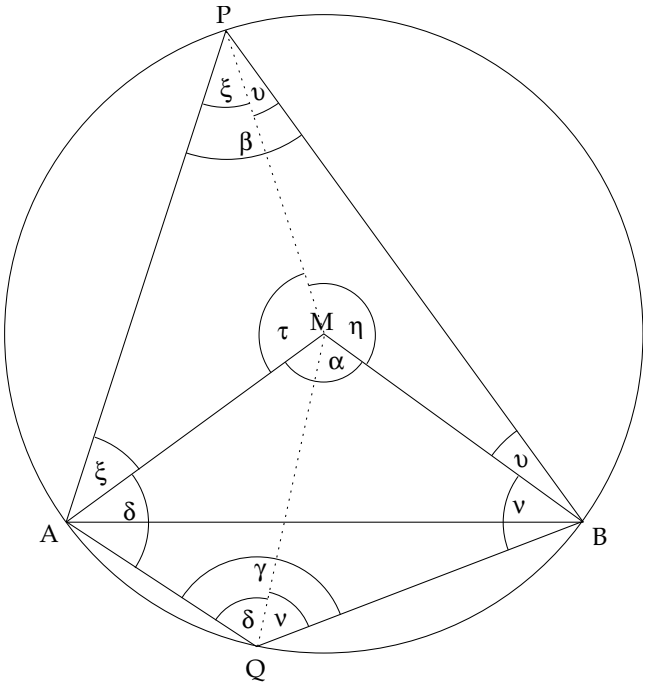
5.6.4. Ellipsen

Nimmt man für jeden Punkt v_k nicht den Winkel $\gamma_k = \angle v_1 v_k v_n$, sondern die Summe der Strecken $[v_1 v_k]$ und $[v_k v_n]$ ($l_k = |v_1 v_k| + |v_k v_n|$), so erhält man ebenfalls einen Akzeptanzbereich: Für einen gegebenen Schwellwert $l^{\max}$ liegen alle Punkte v_i mit $|v_1 v_i| + |v_i v_n| \leq l^{\max}$ in einer Ellipse mit den Brennpunkten v_1 und v_n (Abb. 5.15). Die Ellipse besitzt eine ähnliche Form wie die Linse des Winkelmaßes, besitzt aber gegenüber dieser den Vorteil, daß sie an Start- und Endpunkt nicht so spitz zuläuft, kleine Abweichungen der Spur wirken sich hier also nicht so schnell aus.

Die Form der Ellipse hängt dabei vom Verhältnis von l zur Länge der Strecke s ab. Alle Ellipsen mit gleichem Verhältnis $\frac{l}{|s|}$ sind einander ähnlich, die Angabe eines

Winkel im Kreis

Bezüglich einer gegebenen Strecke $s = [AB]$ liegen alle Punkte P mit gleichem Winkel $\angle APB$ auf einem Kreisbogen (erweiterter Satz von Thales).



Satz 1 *Für jede Sehne $s = [AB]$ eines Kreises mit Mittelpunkt M gilt: für alle Punkte P, die auf dem Kreisbogen von A nach B auf der gleichen Seite von $\overline{AB}$ liegen, ist der Winkel $\angle APB$ gleich.*

Beweis.
Der Punkt P kann dabei auf der gleichen (Fall I, in der Graphik: Winkel β) oder entgegengesetzten Seite (Fall II, in der Graphik: Punkt Q, Winkel γ) von $\overline{AB}$ liegen wie M.
Zunächst zeige ich (Fall I) $\alpha = 2\beta$:

- Sei $\eta \leq 180^\circ$ und $\tau \leq 180^\circ$ (d. h. M liegt im Dreieck $\triangle APB$):

$$2\xi + \tau = 180^\circ, \quad 2\upsilon + \eta = 180^\circ, \quad \tau + \eta + \alpha = 360^\circ, \quad \beta = \xi + \upsilon,$$
$$\text{somit} \qquad \alpha = 360^\circ - \tau - \eta = 360^\circ - 180^\circ + 2\xi - 180^\circ + 2\upsilon = 2\beta$$

- Für seitlich liegende P ($\eta > 180^\circ$ oder $\tau > 180^\circ$) läuft der Beweis analog.

Weiterhin gilt (Fall II) $\gamma = 180^\circ - \alpha/2$:

$$\gamma = \delta + \nu, \quad 360^\circ = \alpha + \delta + \gamma + \nu \quad \Longrightarrow \quad 360^\circ - \alpha = 2\gamma \qquad (\text{d. h.} \quad \gamma = 180^\circ - \beta).$$

Folglich sind sowohl β als auch γ konstant. □

Abbildung 5.14.: Konstante Winkel liegen auf einem Kreisbogen

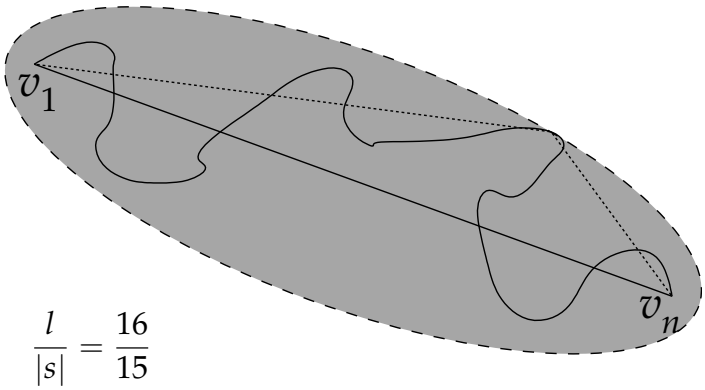


Abbildung 5.15.: Elliptischer Akzeptanzbereich

maximalen Längenverhältnisses liefert somit ein geeignetes Ähnlichkeitsmaß.

Länge und Breite der Ellipse (also die Achse R entlang s und die Achse r senkrecht zu s) hängen mit $|s|$ und l wie folgt zusammen:

$$R = \frac{l}{2}\,, \qquad r = \sqrt{\frac{l^2 - |s|^2}{4}}.$$

Durch Angabe dieser Achsen R und r statt eines Längenverhältnisses bekommt man mehr Flexibilität, da damit die Form der Ellipse in Bezug auf s freier gewählt werden kann.

Um den Aufwand für die Berechnung, ob ein bestimmter Punkt innerhalb einer Ellipse um s mit den Achsen R und r liegt, gering zu halten, führe ich sie auf die Berechnung eines Längenverhältnisses zurück: Sei m_s der Mittelpunkt von s, also $(m_s \in s) \wedge (|m_s v_1| = |m_s v_n|)$, so bilden eine Strecke $t = [ab]$ mit $t \in g = \overline{v_1 v_n}$ und $|am_s| = |bm_s|$ und eine Länge $l' = |av_k| + |v_k b|$ ebenfalls eine Ellipse (die Brennpunkte a und b der Ellipse werden also auf der Geraden $\overline{v_1 v_n}$ ausgehend von m_s nach außen verschoben). Mittels

$$R = \frac{l'}{2}\,, \quad r = \sqrt{\frac{l'^2 - |t|^2}{4}} \qquad \Longrightarrow \qquad l' = 2R, \quad |t| = \sqrt{l'^2 - 4r^2} = 2\sqrt{R^2 - r^2}.$$

lassen sich zu gegebenen Achsen R und r nun passende l' und t bestimmen, somit kann für jeden Punkt v_i durch Vergleich von $l'_i = |av_i| + |v_i b|$ mit $l' = 2R$ einfach getestet werden, ob er innerhalb der Ellipse liegt.

Der Radius R kann nicht als konstanter Parameter angegeben werden, da in der Generalisierung ständig unterschiedlich lange Strecken auftreten, und zumindest $2R > |s|$ gelten sollte. Vielmehr wird R als Vielfaches von $|s|$ angegeben (z. B. $R = 0.6|s|$). r kann konstant gewählt werden und gibt dann für die ganze Generalisierung einen konstanten maximalen Abstand vor, kann aber auch abhängig von $|s|$ angegeben werden, was zu lauter ähnlichen Ellipsen führt.

5.6.5. Eigenschaften dieser Ähnlichkeitsmaße

ε-Korridor, Linse und Ellipse sind nicht die einzig denkbaren Formen für Akzeptanzbereiche um s, theoretisch sind hier beliebige Flächen denkbar, jedoch die wenigsten sinnvoll. Die drei hier gezeigten haben dabei den großen Vorteil der einfachen Berechenbarkeit. Im Allgemeinen kann der Test, ob sich ein Punkt in einer beliebigen Fläche befindet, sehr aufwendig werden. Eine einfache Möglichkeit zur Erzeugung anderer Akzeptanzbereiche ist die (gewichtete) Kombination obiger Maße: $d_i^{\varepsilon\gamma l} = \lambda_\varepsilon\,\varepsilon_i + \lambda_\gamma\,\gamma_i + \lambda_l\,l_i$. Hierbei steuern λ_ε, λ_γ und λ_l, wie stark sich die einzelnen Maße auswirken (wie stark diese den Akzeptanzbereich „formen“).

Die meisten Generalisierungen sollten sich mit den hier gezeigten Maßen (und Modifikationen dieser Maße, z. B. rechteckige statt abgerundete ε-Korridore) abdecken lassen, für weitere Maße sollte die Grundidee zur Konstruktion klar sein.

Alle Generalisierungen, die mit konvexen Akzeptanzbereichen arbeiten, garantieren hierbei bestimmte Eigenschaften der Originalspur p: die Generalisierung $q = w_1 w_2 \ldots w_m$ weicht bei Kenntnis des Parameters (ε, γ oder andere) zwischen allen w_i, w_{i+1} um maximal einen festen Wert von q ab. Ist der Akzeptanzbereich nicht konvex, so gilt dies nur für die Eckpunkte $v_1, v_2, \ldots, v_n$ von p, nicht aber zwingend für die Punkte der Strecke $[v_i v_{i+1}]$ zwischen zwei Eckpunkten.

Das ε-Maß unterscheidet sich hier in einem wesentlichen Punkt grundlegend von den anderen Maßen. Für jeden Punkt der Originalspur p gilt, daß er eine Entfernung kleinergleich ε zur Generalisierung q (also zu einem Punkt $x \in q$) besitzt, d. h. der ε-Korridor wird von q und *nicht* nur von den Eckpunkten von q bestimmt, jeder beliebige Punkt $x \in q$ ist hier also gewissermaßen „gleichberechtigt“. Dies liegt daran, daß die Breite des Korridors nur von ε abhängt, wohingegen die Form der Akzeptanzbereiche der anderen vorgestellten Maße auch davon bestimmt wird, wie *weit* die Eckpunkte w_i und w_{i+1} auseinander liegen.

5.6.6. Skalierung

Bei der Betrachtung der unterschiedlichen Ähnlichkeitsmaße wurde die mögliche Lage eines Punktes v_k bezüglich vorgegebener Endpunkte v_1 und v_n betrachtet. Wie oben erwähnt, spielt der Abstand der Punkte v_1 und v_n zueinander je nach Maß eine mehr oder minder wichtige Rolle. Abbildung 5.16 zeigt drei einander ähnliche Segmente einer Bewegungsspur: Die Segmente $p = v_a \ldots v_e$, $p' = v'_a \ldots v'_e$ und $p'' = v''_a \ldots v''_e$, wobei p' ein in Längsrichtung gestrecktes und p'' ein vergrößertes Abbild von p darstellt. Die vorgestellten Ähnlichkeitsmaße reagieren unterschiedlich auf diese Skalierungen.

Bei der Generalisierung mittels ε-Korridor verhalten sich p und p'' gleich, da der Abstand $h = h' < \varepsilon$, wogegen $h'' > \varepsilon$. Das Winkelmaß hingegen verhält sich hier völlig anders: es gilt $\angle v_a v_k v_e = \angle v''_a v''_k v''_e$, wogegen $\angle v'_a v'_k v'_e$ wesentlich stumpfer ist. Während es auf den ersten Blick so scheint, als unterschieden sich Winkel- (Linse),

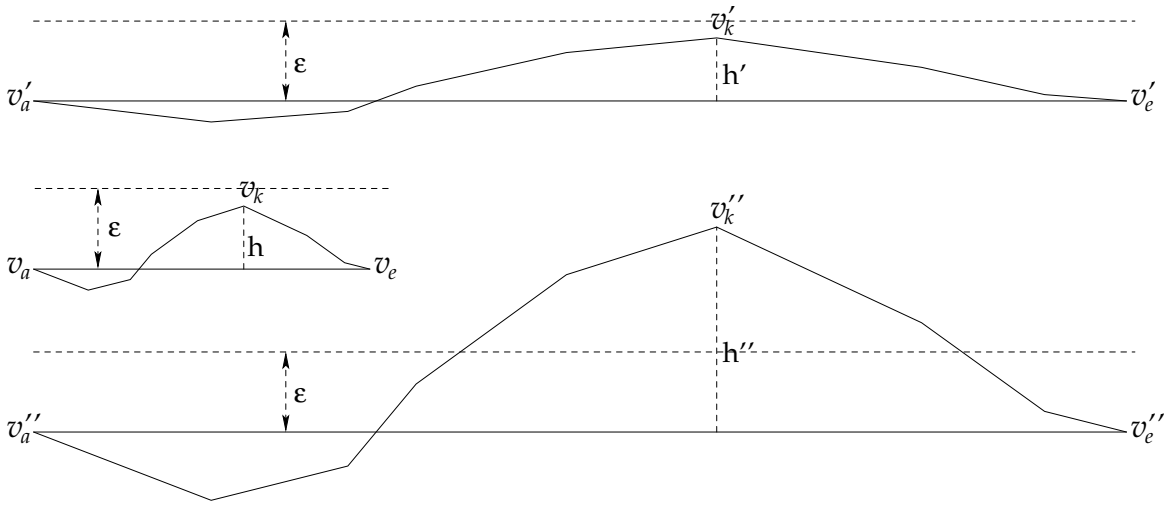


Abbildung 5.16.: „Ähnliche“ Segmente?

Ellipsen- und ε-Korridor-Maß in erster Linie in der Form des Akzeptanzbereichs, so sieht man hier deutlich, daß sie auf eine Größenänderung völlig unterschiedlich reagieren: Während für den ε-Korridor die Entfernung von Start- zu Endpunkt $|v_a v_e|$ völlig irrelevant ist, ist sie für das elliptische und das Winkelmaß maßgeblich.

Dies zu ändern, ist nicht weiter schwer. Damit der elliptische Akzeptanzbereich aus Abbildung 5.17 A bei größerem Abstand zwischen v_a und v_e zu B und nicht zu C skaliert, muß der zu $\overline{v_a v_e}$ rechtwinklige Radius r der Ellipse – und damit ihre Breite – konstant gehalten werden. Wie schon in Abschnitt 5.6.4 erwähnt, ist dies problemlos möglich. Für das Winkelmaß ist es natürlich ebenfalls möglich, die maximale Breite der Linse konstant zu halten, allerdings ist die Berechnung des Winkels hier etwas aufwendiger.[21]

Unter dem Flächenmaß (Abschnitt 5.6.2) in der vorgeschlagenen Form ($\frac{A}{|v_a v_e|}$ mit A Fläche unter der Kurve) sind die Bewegungsspuren p und p' identisch. Wählt man hingegen $\frac{A}{|v_a v_e|^2}$, so haben p und p'' gleiches Maß.

Neben der Wahl zwischen der Streckung in Längsrichtung bei Beibehaltung der Breite und zentrischer Streckung sind natürlich für alle Maße auch jederzeit beliebige Zwischenstufen möglich, in dem man beispielsweise die Breite der Ellipse mit $\sqrt{|v_a v_e|}$ wachsen läßt.

5.6.7. „Unscharfe“ Ähnlichkeitsmaße

Alle bislang vorgestellten Maße arbeiten mit einer harten Grenze: Entweder liegt ein bestimmter Parameter für alle v_i unterhalb eines Schwellwertes, dann ist die

[21] Umgekehrt ließe sich natürlich auch die Breite des ε-Korridors abhängig von $|v_a v_e|$ wählen, was aber wenig sinnvoll erscheint.

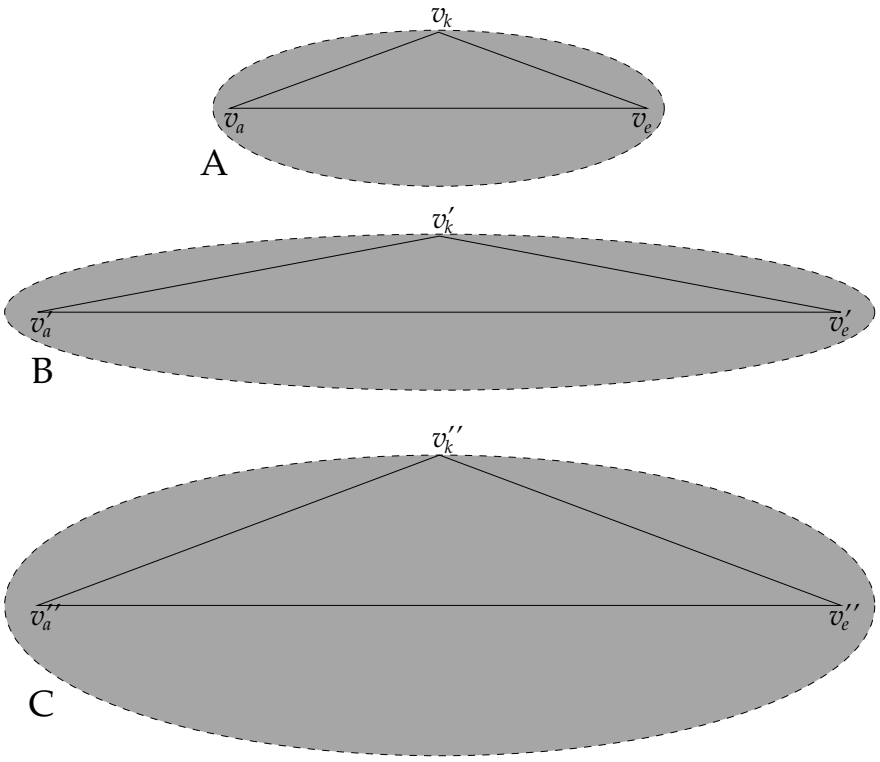


Abbildung 5.17.: Skalierung des elliptischen Akzeptanzbereichs

Generalisierung des Segmentes fertig, oder er liegt für mindestens ein v_k oberhalb, dann muß weiter unterteilt werden. Nicht berücksichtigt wird dabei, *wie weit* dieser Schwellwert überschritten wird, ebensowenig (bei Ähnlichkeitsmaßen, für die jeder Punkt einzeln getestet wird), *wie viele* v_i diese Schwelle überschreiten.

Ich betrachte diejenigen Maße, die einen Akzeptanzbereich um s bilden, aus der obigen Aufzählung sind dies ε, γ, l und ihre Kombinationen. Hier gibt für jeden Punkt v_i das zugehörige Maß $\varepsilon_i = |\text{dist}(s, v_i)|$, $\gamma_i = \angle v_1 v_i v_n$ und $l_i = (|v_1 v_i| + |v_i v_n|)$ den Grad der Abweichung von $s = [v_1 v_n]$ an. Sei im Folgenden δ_i dieses Maß

Statt nun zu prüfen, ob diese Abweichung für mindestens einen Punkt größer als ein bestimmter Schwellwert ist, wird der Durchschnitt aller Abweichungen bestimmt und mit einem Schwellwert $\tilde{\delta}$ verglichen.

Die Wahl des einfachen Durchschnitts

$$\overline{\delta} = \frac{1}{n} \sum_{i=1}^{n} \delta_i$$

kann dazu führen, daß einzelne v_k sehr weit von s entfernt liegen, solange der Abstand der meisten Punkte zu s sehr gering ist. Durch quadratische Abstandsmaße

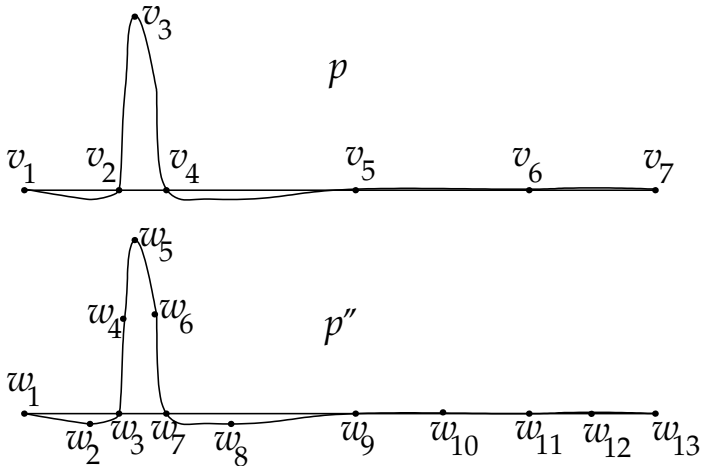


Abbildung 5.18.: Messung einer Bewegungsspur durch den Polygonzug p (mit nur einem von s abweichenden Punkt v_3) und den (doppelt so genauen) Polygonzug p''.

(oder auch höhere Potenzen) wird dieser Effekt gemildert, weil größere Entfernungen dadurch stärker gewichtet werden:

$$\overline{\delta^a} = \frac{1}{n}\sqrt[a]{\sum_{i=1}^{n} \delta_i^a}.$$

Sei $p = v_1 v_2 \dots v_n$ ein Polygonzug mit $\forall_{i \neq k} : \delta_i = 0$. Dann gilt $\overline{\delta^a} = \frac{\delta_k}{n}$. Wie weit v_k abweichen kann (wie groß d_i werden darf), wird also von der Anzahl der Punkte in p bestimmt. Ist also ein Polygonzug p' doppelt so lang wie p oder ein Polygonzug p'' doppelt so genau[22] wie p, so darf ein einzelner Punkt plötzlich doppelt so weit von s abweichen, ohne daß $\tilde{\delta}$ überschritten wird, und dies unabhängig von der Potenz a. Im ersten Fall skaliert die mögliche Abweichung mit der Länge der Bewegungsspur, was in bestimmten Fällen erwünscht sein kann.

Wird eine Bewegung hingegen mit einem feineren Zeitraster gemessen, so soll dies normalerweise nicht dazu führen, daß plötzlich anders generalisiert wird. Allerdings steigt durch genauere Messungen auch die Wahrscheinlichkeit, daß neben v''_k auch die zu v''_k benachbarten Punkte (v''_{k-1}, v''_{k+1}, …) weit von s entfernt liegen, was diesen Effekt etwas dämpft (Abb. 5.18).

Selbstverständlich ist eine „normale“ Bewegungsspur gewöhnlich nicht so extrem verteilt, so daß der oben beschriebene Effekt nie in dieser Stärke auftreten wird, dennoch wäre es es wünschenswert, „Ausreißer“ bei großen n zu dämpfen. Hierbei sind mehrere Ansätze praktikabel.

[22] Mit Genauigkeit ist hier das Zeitraster gemeint, in dem der Polygonzug gemessen wurde, ein doppelt so genauer Polygonzug enthält also doppelt so viele Punkte.

Eine einfache Möglichkeit besteht darin, nicht den Durchschnitt über ganz p zu nehmen, sondern jeweils $l = (2m + 1)$ Punkte zu mitteln:[23]

$$\overline{\delta_k^a} = \frac{1}{2m+1} \sqrt[a]{\sum_{i=k-m}^{k+m} \delta_i^a}.$$

Damit ist $\overline{\delta_k^a}$ unabhängig von der Länge des Polygonzuges p, nicht jedoch unabhängig von der Meßgenauigkeit. Wird beispielsweise über $l = 3$ benachbarte Werte gemittelt, so gilt im Beispiel von Abbildung 5.18 für den Punkt

$$v_3 \in p: \qquad \overline{\delta_3^a} = \frac{1}{3} \sqrt[a]{\delta_2^a + \delta_3^a + \delta_4^a} = \frac{\delta_3}{3}.$$

Betrachtet man nun die doppelt so genaue Messung p'', so ergibt sich

$$w_5 \in p'': \qquad \overline{\delta_5''^a} = \frac{1}{3} \sqrt[a]{\delta_4''^a + \delta_5''^a + \delta_6''^a}.$$

Da nun die Punkte w_4 und w_6 zum Tragen kommen, gilt im Punkt $w_5 = v_3$, daß sein Abstand in p kleiner ist als in $p'': \ \overline{\delta_5''^a} > \overline{\delta_3^a}$.

Der Parameter l (also die Größe des Bereichs, der gemittelt wird) muß demzufolge an die Genauigkeit der Messung angepaßt werden. Wird genauer gemessen, so muß l im gleichen Verhältnis erhöht werden.

Eine andere Möglichkeit, solche Ausreißer in den Griff zu bekommen, besteht darin, mit einem festen Akzeptanzbereich und einer Pufferzone zu arbeiten. Ein Schwellwert δ bestimmt wie gehabt einen Bereich, in dem alle Punkte des Polygonzuges liegen müssen. Sei nun

$$a \ominus b = \begin{cases} a - b & \text{für} \quad a \geq b \\ 0 & \text{für} \quad a < b \end{cases} \qquad \text{und} \quad \breve{\delta}_i = \delta_i \ominus \delta.$$

Dann ist

$$\overline{\breve{\delta}^a} = \frac{1}{n} \sqrt[a]{\sum_{i=1}^{n} \breve{\delta}_i^{\,a}} \quad = \frac{1}{n} \sqrt[a]{\sum_{i=1}^{n} (\delta_i \ominus \delta)^a}$$

ein Maß dafür, wie weit p im Schnitt über den Akzeptanzbereich hinausragt. Ein weiterer Schwellwert $\breve{\delta}$ legt fest, wie viel „Überstand“ erlaubt ist. Durch die Angabe zweier getrennter Werte für die Größe des Akzeptanzbereiches (δ) und den im Schnitt erlaubten Überstand ($\breve{\delta}$) läßt sich der Grad der Unschärfe genau steuern. Für $\delta = 0$ bekommt man den zuerst vorgestellten rein unscharfen Akzeptanzbereich, für $\breve{\delta} = 0$ den klassischen Akzeptanzbereich mit harten Grenzen.

[23] Wobei l an den Rändern kleiner (der Bereich also abgeschnitten) wird.

Beide Ansätze lassen sich natürlich kombinieren, indem auch hier wieder nicht über den gesamten Polygonzug p, sondern für jeden Punkt v_i über die $l = 2m + 1$ benachbarten Punkte $v_{i-m} \dots v_{i+m}$ gemittelt wird.

Die Bestimmung der durchschnittlichen Abweichung aller Punkte $v_i \in p$ und das Mitteln über l Punkte unterscheiden sich grundsätzlich. Im ersten Fall wird *ein* Durchschnittswert $\overline{\delta^a}$ über *alle* Punkte bestimmt, im zweiten Fall für *jeden* Punkt v_k der Durchschnitt $\overline{\delta^a_k}$ über seine l Nachbarn. Hier wird also jeder Punkt mehrfach betrachtet.

5.6.8. Drei und mehr Dimensionen

Ich habe die unterschiedlichen Ähnlichkeitsmaße der Einfachheit halber oben nur für den zweidimensionalen Fall betrachtet. Abgesehen vom Flächenmaß ist ihre Anwendung im mehrdimensionalen Raum ohne Einschränkung möglich. Die Berechnung der einzelnen Abstände wird dadurch zwar etwas aufwendiger (3 oder 4 Koordinaten pro Punkt statt 2), die Komplexität des Algorithmus steigt dadurch jedoch nicht.[24]

Bezüglich des Flächenmaßes kann man sich entscheiden, im dreidimensionalen zur Volumenberechnung überzugehen, was zu Problemen führt: Was für ein Volumen soll zwischen einer Strecke im Raum und einem Polygonzug berechnet werden? Meines Erachtens sinnvoller ist, auch im Mehrdimensionalen die Fläche zwischen Referenzstrecke und Polygonzug zu bestimmen. Hierzu wird zu jedem Punkt v_i des Polygonzuges das Lot l_i auf $\overline{v_1 v_n}$ bestimmt. Die Fläche bestimmt sich dann zu

$$A = \sum_{i=1}^{n-1} |\square l_i v_i v_{i+1} l_{i+1}| \,,$$

wobei $\square l_i v_i v_{i+1} l_{i+1}$ die minimale spannende Fläche zwischen den vier Punkten beschreibt.

5.7. ε mit anderen Ecken

Unter allen angegebenen Ähnlichkeitsmaßen ist der ε-Korridor in vielen Fällen Mittel der Wahl, er garantiert, daß sich die Generalisierung von der Originalspur nirgends weiter als ein bekanntes Maß entfernt, was für Routenplanung etc. nicht unerheblich ist. Diese Eigenschaft wird dadurch erreicht, daß so lange weitergeneralisiert wird, bis jedes Teilstück die ε-Bedingung erfüllt. Hierzu ist unerheblich, an welchem Punkt ein Segment geteilt wird, das die ε-Bedingung nicht erfüllt. Der Teilungspunkt entscheidet über die Qualität der Generalisierung, nicht aber über die ε-Eigenschaft.

[24] genauer: sie steigt linear in der Anzahl der Dimensionen, jedoch nicht in der Anzahl der Punkte

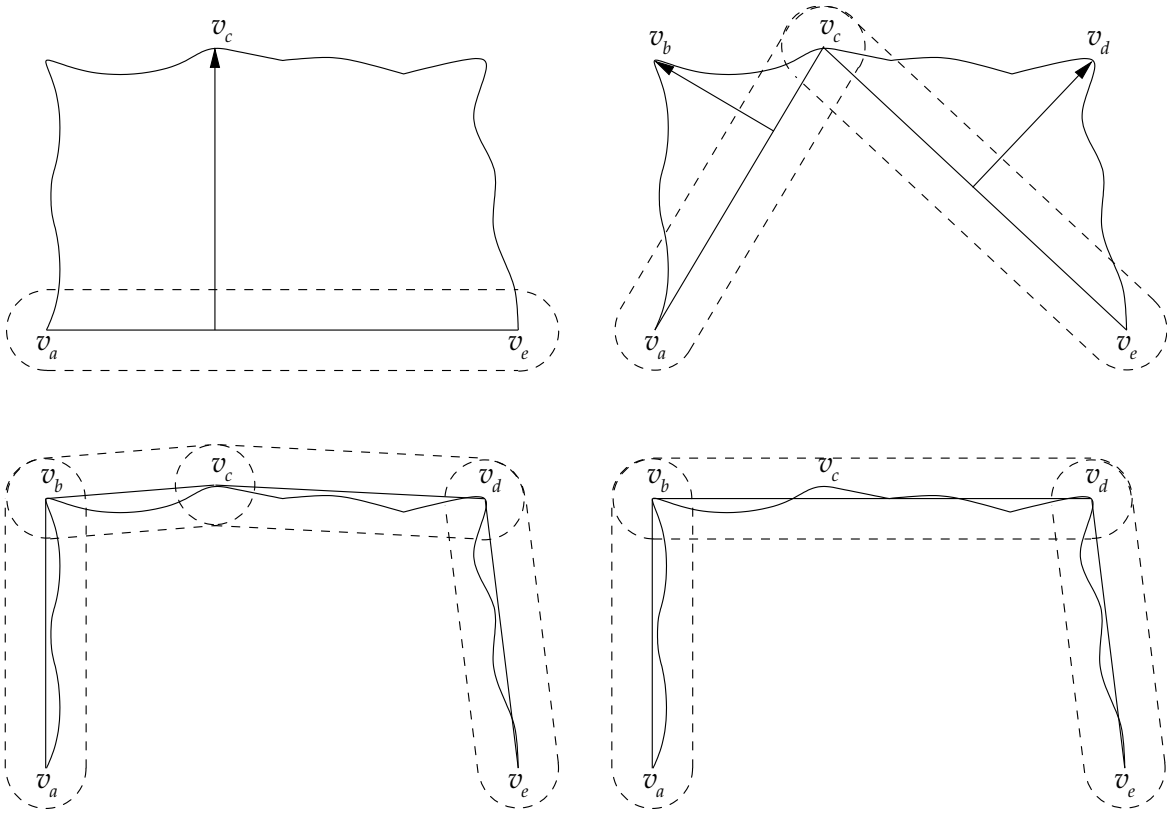


Abbildung 5.19.: Ecken finden bei Douglas/Peucker

Sowohl Douglas und Peucker als auch Jungert und Hernández wählen als Teilungspunkt den zur Referenzgeraden am weitesten entfernt liegenden Punkt. Dies bietet sich an, da diese Entfernung sowieso ermittelt werden muß und somit zur Verfügung steht. Außerdem besteht eine hohe Wahrscheinlichkeit, daß damit eine markante Ecke auch erfaßt wird. Wie Abbildung 5.19 zeigt, können dabei allerdings auch unbedeutende Punkte zu Eckpunkten der Generalisierung werden, wenn sie nur zufällig bezüglich der Referenzgeraden am weitesten entfernt liegen. Ein ähnlicher Effekt zeigt sich im Vergleich von Abb. 5.1 und Abb. 5.2. Der Douglas/Peucker Algorithmus „erwischt“ in Abb. 5.1 einen wenig ausgeprägten Eckpunkt, was dann auch zu einer gegenüber Persson/Jungert/Hernández um einen Punkt längeren Generalisierung führt. Der Polygonzug in Abb. 5.19 wird, wie man sieht, zu $v_a v_b v_c v_d v_e$ generalisiert, obgleich $v_a v_b v_d v_e$ ebenfalls die ε-Bedingung erfüllt, den Polygonzug besser beschreibt und zudem einen Punkt kürzer ist.

Ein Ansatz besteht darin, die Generalisierung nachträglich zu glätten, d. h. für einen Polygonzug $p = v_1 v_2 \dots v_n$ und seine Generalisierung $q = v_1 v_{q_2} v_{q_3} \dots v_n$ wird q wie folgt nachbearbeitet: Für alle Eckpunkte der Generalisierung wird geprüft, ob $[v_{q_k} v_{q_{k+2}}]$ eine gültige Generalisierung des Teilstücks $v_{q_k} v_{q_k+1} v_{q_k+2} \dots v_{q_{k+2}}$

darstellt. Um den Test zu beschleunigen, wird zunächst getestet, ob $v_{q_{k+1}}$ weiter als ε von $[v_{q_k} v_{q_{k+2}}]$ entfernt liegt. Ist dies der Fall, kann nicht geglättet werden. Liegt $v_{q_{k+1}}$ dagegen im ε-Korridor, so müssen noch die übrigen Punkte überprüft werden. Der „Vortest“ mit $v_{q_{k+1}}$ filtert hierbei die Mehrzahl der Fälle von vornherein aus, wodurch der zusätzliche Aufwand dieser nachträglichen Glättung reduziert wird. Bedenkt man weiterhin, daß alle in der letzten Rekursionsstufe gefundenen Eckpunkte nicht für eine Glättung in Frage kommen (sie wurden ja gerade eingefügt, *weil* sie nicht im ε-Korridor ihrer Nachbarn lagen), reduziert dies den Aufwand weiter.

Die nachträgliche Glättung entfernt „schwache“ Ecken, kann aber nicht dafür sorgen, daß markante Eckpunkte gewählt werden. Dies kann dadurch erreicht werden, daß eben nicht der am entferntesten liegende Punkt als Teilungspunkt gewählt wird, sondern die „beste“ Ecke.[25]

Die Auswahl dieser „besten Ecke“ sollte dabei in linearer Zeit erfolgen, um die Komplexität des Algorithmus nicht zu erhöhen. Die Antwort der Kartographen auf die Frage, was denn nun die „markanteste Ecke“ sei, lautet: die, die die meisten menschlichen Kartenzeichner als Ecke wählen. Leider liefert dies noch keinen Algorithmus. Betrachten wir nochmals Abb. 5.19. v_c liegt am weitesten von $[v_a v_e]$ entfernt, und wird somit gewählt. Nutzt man hingegen das Winkelabstandsmaß ($\min_i\{\angle v_a v_i v_e\}$), so wird im ersten Schritt v_b gewählt, was zur gewünschten Generalisierung $v_a v_b v_d v_e$ führt. Das Winkelmaß hat, wie schon oben erwähnt, das Problem, daß sehr schnell Punkte, die nahe v_a oder v_e liegen, als Eckpunkte gewählt werden (und dies sogar dann noch, wenn sie innerhalb des ε-Korridors liegen). Daher empfiehlt sich auch hier wieder der Einsatz des elliptischen Abstands (entweder in der Form $\max_i\{|v_a v_i| + |v_i v_e|\}$ oder wie im obigen Kapitel beschrieben mit verschobenen Brennpunkten). Dies verringert zwar die Wahrscheinlichkeit dafür, daß ein innerhalb des ε-Korridors liegender Punkt gewählt wird, schließt es aber nicht aus.

Schöner ist hier ein Maß, das für alle Punkte innerhalb des Korridors den Abstand 0 liefert und außerhalb kreisbogen- oder ellipsenförmige Isoplethen besitzt. Damit fallen Rekursionsbedingung und Trennpunktbestimmung auch wieder zusammen: Ist der maximale Abstand 0, so ist das Teilstück fertig generalisiert, ansonsten wird am Punkt mit maximalen Abstand geteilt. Als Maße bieten sich hier beispielsweise die in Abbildung 5.20 gezeigten Gummibandansätze an: Um Start- und Endpunkt v_a und v_e wird ein Kreis mit Radius ε gelegt. Nun wird um diese beiden Kreise sowie um den zu betrachtenden Punkt v_k ein Gummiband gespannt. Für den ε-Korridor allein beträgt die Länge des Gummibandes $\ell_0 = 2\pi\varepsilon + 2|v_a v_e|$, liegt v_i innerhalb des Korridors, gilt $\ell_i = \ell_0$, oder $\ell_i - \ell_0 = 0$. Für ein v_k außerhalb

[25] Teilt man jedes Segment in der Mitte, wird übrigens garantiert $O(n \log n)$ erreicht, was aber im Mittel eher zu einer verlängerten Laufzeit führt, und schließlich keine gute Generalisierung liefert.

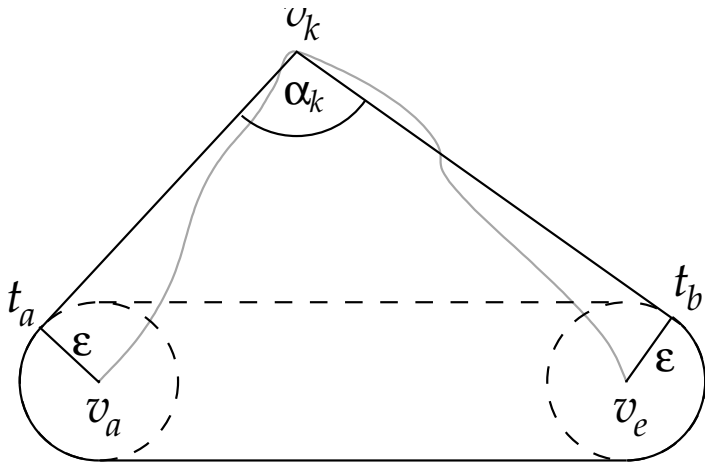


Abbildung 5.20.: Das Gummiband-Maß

des Korridors ist $\ell_k - \ell_0 > 0$. Bezüglich dieses Maßes äquidistante Punkte liegen auf einer ellipsenähnlichen Kurve um den ε-Korridor. Alternativ dazu läßt sich natürlich auch α_k bestimmen, wodurch die Isoplethen eher kreisförmig werden. Das Winkelmaß führt dazu, daß „vor" bzw. „hinter" s liegende Punkte[26] stärker berücksichtigt werden, weil die Winkel in diesen Bereichen schnell sehr spitz werden.

Die beschriebenen Maße setzen den betrachteten Punkt v_k in Beziehung zum Anfangs- und Endpunkt v_a und v_e des Segments. Zur Bestimmung der „markantesten Ecke" könnte ein lokales Maß geeigneter sein. Der naive Ansatz, den spitzesten Winkel $\angle v_{k-1}v_kv_{k+1}$ dreier benachbarter Punkte zu bestimmen, verspricht hier wenig Erfolg, zu schnell erwischt dies eine kleine lokale Spitze. Die Wahl etwas entfernter liegender Beipunkte stellt sicher, daß dies nicht auftritt: Sei $s_{s,t} = |v_sv_{s+1}| + |v_{s+1}v_{s+2}| + \ldots + |v_{t-1}v_t|$ Für jeden Punkt v_k bestimmt man für ein festgesetztes s einen Beipunkt v_{k-l} und einen Beipunkt v_{k+r} so, daß $s_{k-l+1,k} < s, \quad s_{k-l,k} \geq s$ und $s_{k,k+r-1} < s, \quad s_{k,k+r} \geq s$. Der Winkel $\angle v_{k-l}v_kv_{k+r}$ liefert nun ein lokales Eckenmaß, durch das der spitzeste Winkel gewählt wird. Punkte die näher als s am Start- oder Endpunkt liegen, kommen hierbei nicht in Betracht (die Wahl von $s = \varepsilon$ vermeidet hierbei Sonderfälle, wie man sich leicht überlegt). Der Aufwand zur Bestimmung dieses lokalen Eckenmaßes hält sich dabei in Grenzen. Die Punkte werden der Reihe nach durchlaufen. Sind für einen Punkt v_k die zugehörigen Beipunkte v_{k-l} mit $s_{k-l,k}$ und v_{k+r} mit $s_{k,k+r}$ bekannt, so müssen die für den nächsten Punkt v_{k+1} die zugehörigen Beipunkte lediglich aufsteigend gesucht werden: Linker Beipunkt von v_{k+1} ist einer der Punkte $v_{k-l}, v_{k-l+1}, \ldots$ mit $s_{k-l,k+1} = s_{k-l,k} + |v_kv_{k+1}|, \quad s_{k-l+1,k+1} = s_{k-l,k+1} - |v_{k-l}v_{k-l+1}|, \ldots$. Damit muß nicht für jeden Punkt v_k die Summe der Entfernungen $|v_{k-l}v_{k-l+1}| \ldots |v_{k+r-1}v_{k+r}|$

[26] Ein Punkt v_k liegt „vor" bzw. „hinter" $s = [v_av_e]$, wenn der Fußpunkt l_k des Lotes von v_k auf $g = \overline{v_av_e}$ nicht in s liegt ($l_k \notin [v_av_e]$). Siehe auch Abschnitt 5.3, Abb. 5.8

bestimmt (was eine Erhöhung der Laufzeit für jeden linearen Durchlauf um einen Faktor $\bar{l} + \bar{r}$ bedeuten würde[27]), sondern insgesamt für jedes v_k lediglich die beiden Beipunkte nachgeführt werden (was im Mittel einen Faktor 3 ausmacht).

Eine genauere Betrachtung lokaler Verfahren erfolgt in Kapitel 10.

5.8. Generalisierung „on the fly“?

Eine wesentliche Einschränkung des Algorithmus für die Verarbeitung von Bewegungsspuren besteht in seinem globalen Ansatz, d. h. die Bewegungsspur p muß vollständig vorliegen, um sie generalisieren zu können. Häufig soll schon während der Bewegung die bislang gefahrene Spur weiterverarbeitet werden. Mit den heutigen Rechengeschwindigkeiten wäre es zwar kein großes Problem, für jeden neuen Meßpunkt die bislang gemessene Spur komplett neu zu generalisieren (in Schritt k eine Komplexität von $O(k \log k)$ (average case), insgesamt also $O(n^2 \log n)$), dies ist jedoch nicht hinreichend: Da die Generalisierung im Schritt k für die Spur $p_k = v_1 v_2 \dots v_k$ mit der Strecke $s_{1,k} = [v_1 v_k]$ beginnt und dann rekursiv absteigt und im Schritt $k+1$ von $s_{1,k+1} = [v_1 v_{k+1}]$ ausgeht, können sich die Generalisierungen von p_k und p_{k+1} durchgehend unterscheiden. Entscheidend hierbei ist, daß eine Änderung des Endpunktes Auswirkungen auf das erste (und zweite, dritte, ...) generalisierte Segment hat. Damit ändert sich aber die komplette Generalisierung dauernd, d. h. das bislang gefahrene (und generalisierte) Teilstück kann nicht einfach weiterverarbeitet werden.

5.9. Partielle Generalisierung

Hernández und Jungert geben in [JH96] eine „partial generalization“ an, die den grundsätzlichen Ansatz ihres globalen Algorithmus aufgreift und die Bewegungsspur sequentiell durchläuft. Hierzu werden zunächst die k ersten Punkte $v_1, v_2, \dots, v_k$ der Bewegungsspur $v_1, v_2, \dots, v_n$ betrachtet. Erster Punkt der Generalisierung $w_1 = v_1$. Nun wird der zu $g = \overline{v_1 v_k}$ am weitesten entfernt liegende Punkt v_m ($m \in \{1, 2, \dots, k\}$) bestimmt. Liegt v_m weiter als ε von g entfernt ($\text{dist}(g, v_m) > \varepsilon$), so ist $w_2 = v_m$ der zweite Punkt der Generalisierung und der Algorithmus wird mit $v_m, v_{m+1}, \dots, v_{m+k-1}$ fortgesetzt. Liegen dagegen alle k Punkte näher als ε an g ($\forall v_i \in \{v_1, v_2, \dots, v_k\} : \text{dist}(g, v_i) \leq \varepsilon$), so ist $w_2 = v_k$ der zweite Punkt der Generalisierung und der Algorithmus wird mit $v_k, v_{k+1}, \dots, v_{2k-1}$ fortgesetzt.

Abbildung 5.21 zeigt die Generalisierung eines Polygonzuges mit $k = 5$. Im ersten Schritt wird der zur Geraden $\overline{v_1 v_5}$ am weitesten entfernt liegende Punkt v_4 bestimmt. Da $\text{dist}(\overline{v_1 v_5}, v_4) > \varepsilon$, ist v_4 neuer Startpunkt und v_8 ($8 = 4 + 5 - 1$) neuer

[27] da l und r von v_k abhängig sind, muß der Mittelwert genommen werden.

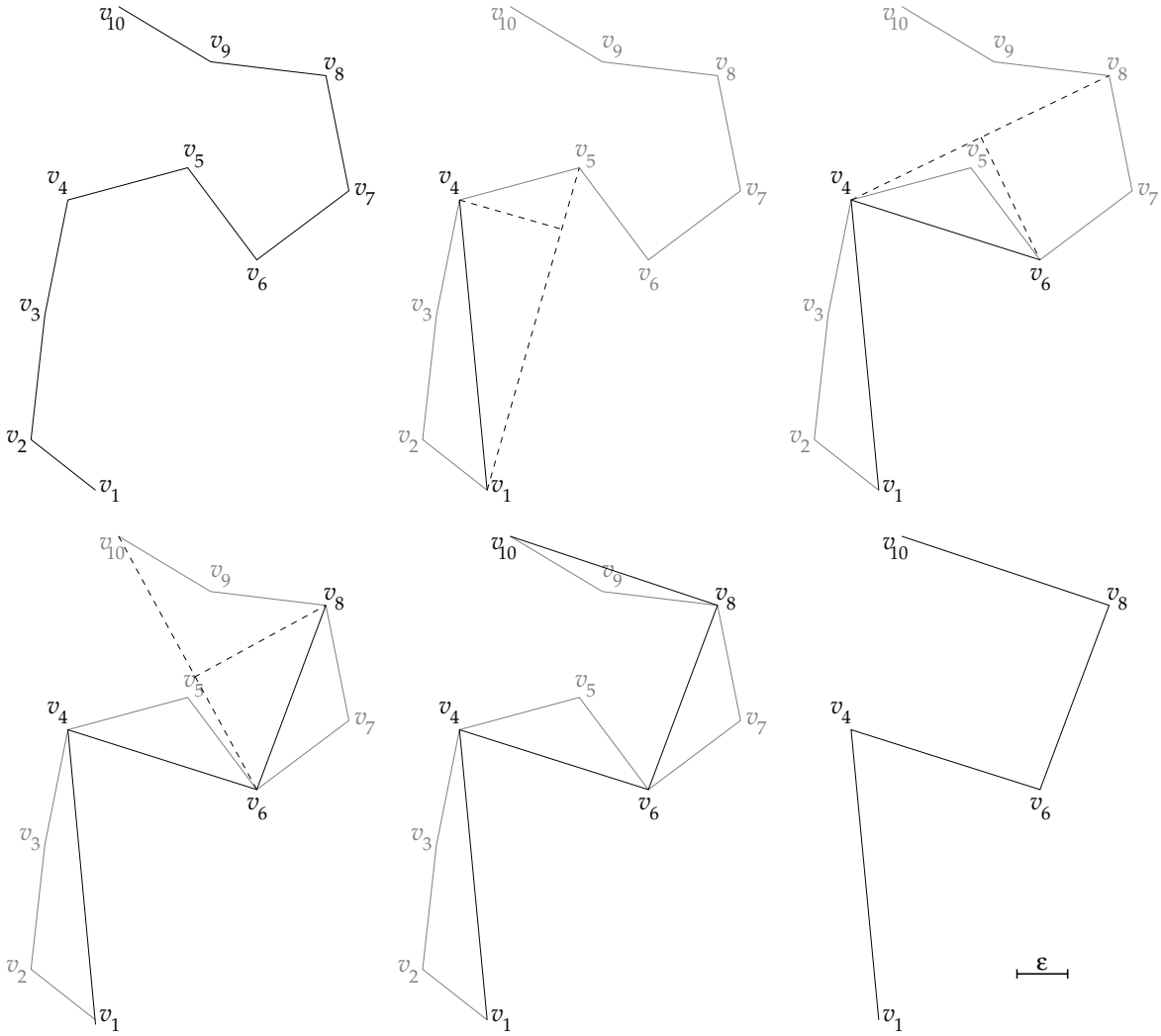


Abbildung 5.21.: „partial generalization“ mit $k = 5$

Endpunkt. Bezüglich $\overline{v_4v_8}$ liegt v_6 am weitesten entfernt, bezüglich $\overline{v_6v_{10}}$ ist dies v_8 und am Ende liegt v_9 weniger als ε von $\overline{v_8v_{10}}$ entfernt. Wir bekommen somit die Generalisierung $v_1v_4v_6v_8v_{10}$.

Mich überzeugt dieser Algorithmus in dieser Form vor allem aus zwei Gründen nicht:

- Der Algorithmus sucht den bezüglich zweier Punkte v_a und v_{a+k-1} zu $\overline{v_av_{a+k-1}}$ am weitesten entfernt liegenden Punkt v_m mit $m \in \{a, a+1, \ldots, a+k-1\}$ und arbeitet dann mit v_m als neuen Startpunkt weiter. Damit ist aber nicht sichergestellt, daß die *zwischen* v_a und v_m liegenden Punkte alle im ε-Korridor zu $\overline{v_av_m}$ liegen. So liegen im Beispiel aus Abbildung 5.21 der Punkt v_2 weiter als ε von $\overline{v_1v_4}$, v_5 weiter als ε von $\overline{v_4v_6}$

und v_7 weiter als ε von $\overline{v_6 v_8}$ entfernt.

Der Algorithmus behauptet zwar nicht, die ε-Bedingung zu erfüllen, dies ändert aber nichts daran, daß die auftretenden Abweichungen sehr groß werden können, was sicher ein unerwünschter Effekt ist. Die Stärke dieser Abweichungen ist dabei unter anderem von der Wahl der Länge k abhängig, je größer k, desto stärkere Abweichungen können auftreten.

- Selbst, wenn die Bewegung völlig geradlinig verläuft, wird im jeweils k-ten Punkt abgeschnitten, die Generalisierung besteht damit aus Segmenten der maximalen Länge k.

Darüberhinaus sollte natürlich auch hier wie schon oben der Abstand zur Strecke $s = [v_a v_{a+k-1}]$ statt zur Geraden $g = \overline{v_a v_{a+k-1}}$ gemessen werden.

Während das erste Argument ein möglichst kleines k nahelegt, fordert das zweite Argument ganz im Gegensatz dazu eine möglichst große Segmentlänge. Zudem sollte die zwischen v_a und v_{a+k-1} zurückgelegte Entfernung im Mittel wesentlich größer sein als ε, da sonst kaum Punkte auftreten können, die von $[v_a v_{a+k-1}]$ hinreichend weit entfernt sind. Nichtsdestotrotz vereinfacht der Algorithmus den Polygonzug, die Frage bleibt, ob die Generalisierung hinreichend ähnlich ist.

Eine einfache Verbesserung besteht darin, auch hier mit der maximalen links- und rechtsseitigen Distanz zu arbeiten. Falls die beiden so gefundenen maximal entfernten Punkte v_l und v_r beide weiter als ε von der Referenzstrecke entfernt liegen, so wird der erste der beiden Punkte (der Punkt mit kleinerem Index) gewählt.

Um eine Generalisierung zu erhalten, die die ε-Bedingung erfüllt, muß mehr geleistet werden. Im Grunde reicht es, hier Douglas/Peucker (oder auch Persson/Jungert/Hernández) rückwärts anzuwenden (wie oben gilt $v_a = v_1$, $w_1 = v_1$):

- Bestimme den zu $s \leftarrow [v_a v_{a+k-1}]$ maximal entfernten Punkt v_m.
- Ist $\mathrm{dist}(s, v_m) \leq \varepsilon$, so ist v_{a+k-1} neue Ecke der Generalisierung, $a \leftarrow a + k - 1$.
- Ist dagegen $\mathrm{dist}(s, v_m) > \varepsilon$, so bestimme den zu $s \leftarrow [v_a v_m]$ maximal entfernten Punkt $v_{m'}$ (mit $m' \in \{a, \ldots, m\}$) und teste wieder.

Das Vorgehen ist also ganz einfach: Der Punkt v_m wird wie oben gefunden, nur wird jetzt noch getestet, ob die Punkte zwischen v_a und v_m die ε-Bedingung erfüllen. Ist dies nicht der Fall, wird wieder der Punkt $v_{m'}$ mit maximalem Abstand gewählt und erneut getestet usw. Mit diesem Verfahren wird also schließlich ein Punkt $v_{m'}$ gefunden, so daß alle Punkte zwischen v_a und $v_{m'}$ weniger als ε von $[v_a v_{m'}]$ entfernt liegen und mit $v_{m'}$ als neuem Startpunkt weitergemacht.

Abbildung 5.22 zeigt die ersten Schritte des modifizierten Algorithmus am Beispiel aus Abbildung 5.21. Zunächst wird v_4 als der bezüglich $[v_1 v_5]$ am weitesten entfernt liegende Punkt gefunden. Da v_4 die ε-Bedingung nicht erfüllt, wird nun

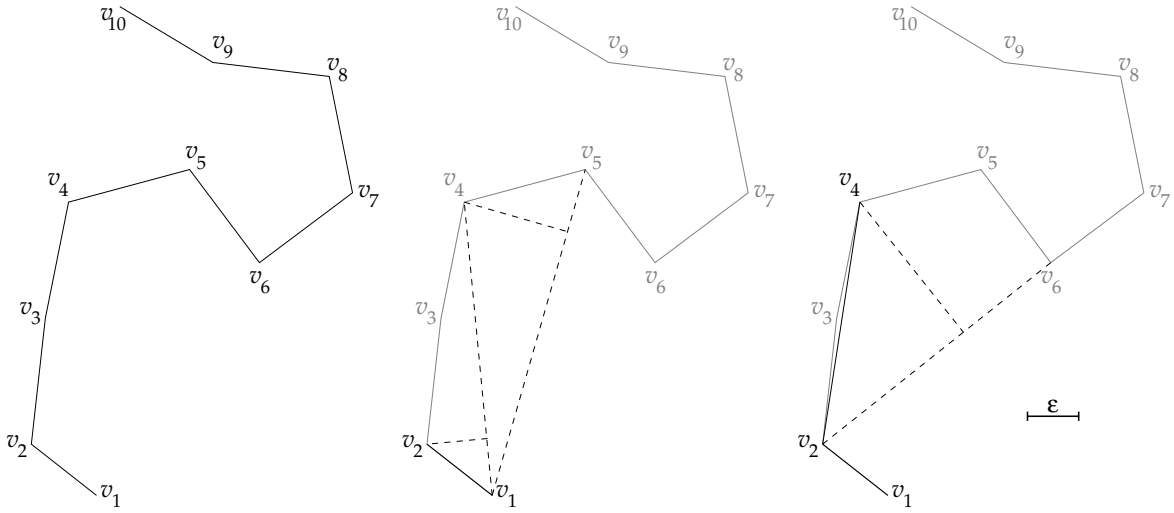


Abbildung 5.22.: Modifizierte „partial generalization“ mit $k = 5$

$[v_1v_4]$ zur Referenzstrecke. Bezüglich dieser liegt v_2 am weitesten (und weiter als ε) entfernt, und somit ist $[v_1v_2]$ neue Referenzstrecke. Da zwischen v_1 und v_2 keine weiteren Punkte liegen, wird hier abgebrochen, v_2 ist eine Ecke der Generalisierung und zugleich neuer Startpunkt. Neue Referenzstrecke ist demnach $[v_2v_6]$, zu dieser liegt v_4 am weitesten entfernt, also wird nun $[v_2v_4]$ getestet. v_3 liegt näher als ε, also ist v_4 die nächste Ecke der Generalisierung und neuer Startpunkt, es geht mit $[v_4v_8]$ weiter …

Damit ist das erste geschilderte Problem gelöst, der Algorithmus erfüllt die ε-Bedingung. Um nun geradlinige Bewegungen möglichst zu einer einzigen langen Strecke zu generalisieren, muß die Beschränkung auf k Punkte aufgehoben werden. Hierbei gilt: Liegen bezüglich der Referenzstrecke $[v_av_{a+k-1}]$ alle dazwischenliegenden Punkte näher als ε, so ist das betrachtete Segment zu kurz und wird um k Punkte verlängert, neue Referenzstrecke wird $[v_av_{a+2k-1}]$. Liegen auch hier wieder alle Punkte im ε-Korridor, so wird auf $[v_av_{a+3k-1}]$ verlängert usw.

Der so gewonnene Algorithmus erfüllt beide oben genannten Forderungen und arbeitet die Bewegungsspur sequentiell ab, ist also im Grundsatz für eine Bewegungsverarbeitung „on the fly“ nutzbar. Ich stelle im nächsten Kapitel einen Algorithmus vor, der die Idee des ε-Korridors aufgreift und konsequenter inkrementell umsetzt. Vor allem kommt er ohne Vorgabe einer Sequenzlänge k aus.

6. WεG

In diesem Kapitel werden (eckentreue) inkrementelle ε-Generalisierungen vorgestellt, die damit die Weiterverarbeitung einer Bewegungsspur „on the fly“ ermöglichen, da hier im Gegensatz zu den Ansätzen im letzten Kapitel die Bewegungsspur nicht vollständig vorliegen muß.

6.1. Ganz einfach ...

Eine einfache inkrementelle Umsetzung des ε-Ansatzes zeigt Algorithmus 4. Angefangen vom Startpunkt v_1 wird für jeden Punkt v_k getestet, ob alle Punkte zwischen v_1 und v_k in der ε-Umgebung von $g = \overline{v_1 v_k}$ liegen. Abbildung 6.1 zeigt die Generalisierung des Polygonzuges $p = v_1 v_2 v_3 v_4 v_5 v_6 \ldots$ (Bild a). Im ersten Schritt (Bild b) wird geprüft, ob v_2 in der ε-Umgebung zu $\overline{v_1 v_3}$ liegt, im zweiten Schritt (Bild c), ob v_2 und v_3 in der ε-Umgebung zu $\overline{v_1 v_4}$ liegen und im dritten Schritt (Bild d) liegt v_4 nicht mehr in der ε-Umgebung zu $\overline{v_1 v_5}$. Daher wird hier abgebrochen und mit v_4 als neuem Startpunkt weitergemacht (da $\overline{v_1 v_4}$ eine gültige Generalisierung für $v_1 v_2 v_3 v_4$ und $\overline{v_1 v_5}$ keine gültige Generalisierung für $v_1 v_2 v_3 v_4 v_5$ darstellt).

Auch hier läßt sich wieder mit $d' = |\text{dist}(s, v_k)|$ rechnen, um Schleifen zu vermeiden. Im Gegensatz zu Douglas/Peucker müssen hier keine weiteren Fälle betrachtet werden, da die Struktur des Algorithmus sicherstellt, daß damit alle Schleifen erkannt werden.

simpleinc$_\varepsilon$ (Algorithmus 4) ist eine historiefreie inkrementelle Generalisierung. Sie besitzt damit wie oben gezeigt über die Anzahl der Segmente s lineare Komplexität ($O(s)$). In der Anzahl der Punkte pro Segment r steigt die Laufzeit quadratisch ($O(r^2)$), da für jeden Punkt im Segment alle im gleichen Segment vor ihm liegenden Punkte erneut durchlaufen werden müssen. Dies führt insgesamt zu $O(r^2 s) = O(rn)$, da $n = rs$.

Douglas und Peucker schreiben in [DP73], daß schon in den sechziger Jahren die Firma AEG einen Plotter (GEAGRAPH 4000) zum Zeichnen von Karten verkaufte, in dem ein inkrementeller Algorithmus implementiert war. Den Algorithmus selbst geben sie leider nicht an, er scheint aber (nach der Beschreibung im Fließtext) simpleinc$_\varepsilon$ zu entsprechen. Dies würde auch mit der quadratischen Kom-

Algorithmus 4: Einfache inkrementelle ε-Generalisierung

```
1:  function in? = (Gerade g, Polygonzug p = v_1 ... v_k, Distanz ε) → Boolean :
2:  begin
3:    for all v_i ∈ {v_2, ... v_{k-1}} do      ▷ Wir testen alle Punkte
4:      d ← |dist(g, v_k)|;                    ▷ Distanz zur Geraden
5:      if (d > ε) then                        ▷ Punkt mit größerem Abstand gefunden
6:        return false;
7:      end if
8:    end for
9:    return true;                             ▷ Alles im Korridor
10: end

11: function rest = (Polygonzug p = v_1 ... v_n, Distanz ε) → Polygonzug :
12: begin
13:   k ← 3;
14:   while (k ≤ n) ∧ in?(v_1v_k, v_1 ... v_k, ε) do   ▷ Alle Punkte im Korridor
15:     k++;
16:   end while
17:   return v_{k-1}v_k ... v_n                ▷ v_{k-1} ist der letzte Punkt im Segment
18: end

19: function simpleinc_ε = (Polygonzug p = v_1 ... v_n, Distanz ε) → Polygonzug :
20: begin
21:   if n ≤ 2 then                            ▷ Erst ab 3 Punkten wird generalisiert
22:     return p
23:   else
24:     return v_1 ∘ simpleinc_ε(rest(p, ε))   ▷ rekursiv die Segmente ausgeben
25:   end if
26: end
```

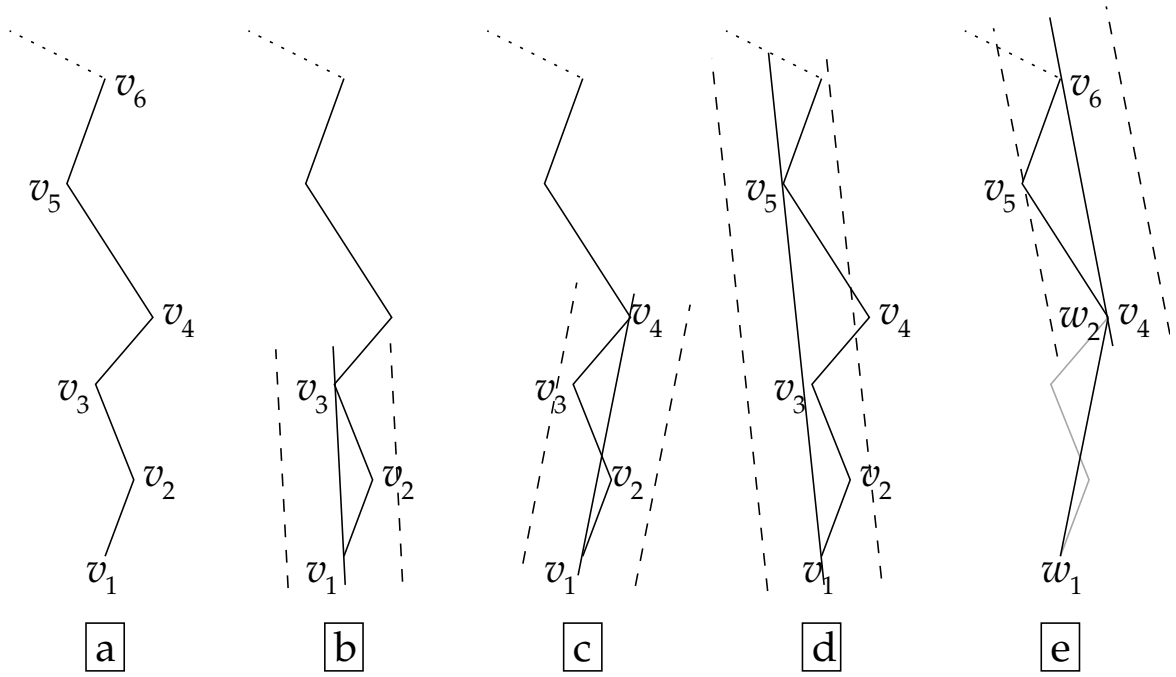


Abbildung 6.1.: Generalisierung mittels simpleinc$_\varepsilon$.

plexität von simpleinc$_\varepsilon$ zusammenpassen: „...it required far too much computer time for the on-line processing system being operated at the time.“ [1]

Grundsätzlich läßt sich simpleinc natürlich auch mit anderen Ähnlichkeitsmaßen benutzen, in dem wie bei Douglas/Peucker $d' = |\text{dist}(g, v_k)|$ durch ein anderes Maß ersetzt wird. Für Ähnlichkeitsmaße, die einen Akzeptanzbereich aufspannen, liefert der Algorithmus einen generalisierten Polygonzug q, für den gilt, daß die Originalspur p in ihrem Akzeptanzbereich liegt.

Die gefundene Generalisierung q ist hierbei häufig nicht optimal: im Beispiel aus Abbildung 6.1 wird in Schritt d abgebrochen, da ein Punkt (v_4) nicht mehr im Korridor zu $[v_1v_5]$ liegt. Diese Entscheidung ist voreilig, da alle Punkte v_2 bis v_5 im ε-Korridor zu $[v_1v_6]$ liegen, womit die Strecke $[v_1v_6]$ eine gültige (und bessere) Generalisierung darstellt. Dieser Effekt tritt auch bei allen anderen Maßen auf.

Eine einfache Möglichkeit damit umzugehen besteht darin, nicht sofort abzubrechen, sobald ein Punkt $v_2 \dots v_{k-1}$ das Ähnlichkeitsmaß bezüglich $[v_1v_k]$ nicht erfüllt sondern noch bis zu m Schritte weiterzusuchen, ob für eine der Strecken $[v_1v_{k+1}]$ bis $[v_1v_{k+m}]$ alle zugehörigen Punkte $v_2 \dots v_k$ bis $v_2 \dots v_{k+m-1}$ das Ähnlichkeitsmaß erfüllen. Alg. 5 zeigt die veränderte Funktion rest'.

[1] Im gleichen Paper geben Douglas und Peucker auch eine Modifikation des Algorithmus an, die mittels einiger Abschätzungen die Laufzeit im Mittel reduziert. Sie erreichen damit allerdings nicht die lineare Komplexität des unten angegebenen WεG, weshalb ich an dieser Stelle nicht weiter darauf eingehe.

Algorithmus 5: Vorausschau um m Schritte

```
1:  function rest' = (Polygonzug p = v_1 ... v_n, Distanz ε, Index m) → Polygonzug :
2:  begin
3:      k ← 2;
4:      l ← 2;                                  ▷ Segmentende +1
5:      while (k ≤ n) ∧ (k − l ≤ m) do          ▷ Nicht zu weit...
6:          if in?(v_1v_k, v_1 ... v_k, ε) then
7:              l++;
8:          end if
9:          k++;
10:     end while
11:     return v_{l−1}v_l ... v_n               ▷ v_{l−1} ist der letzte Punkt im Segment
12: end
```

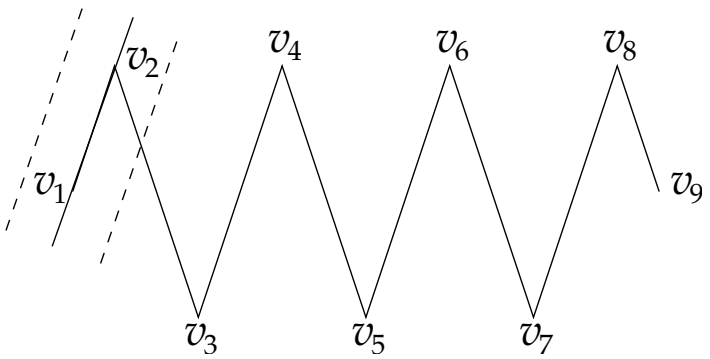


Abbildung 6.2.: simpleinc worst case mit ε-Maß

Ohne ein beschränkendes m (also $m > n$) würde der Algorithmus in jedem Schritt den kompletten verbleibenden Polygonzug bis zum Endpunkt v_n durchsuchen. Dies führt im worst case (Abb. 6.2) zu einer Komplexität von $O(n^3)$. Beschränkt m dagegen die Vorausschau, so wächst die Laufzeit im worst case nur linear um den Faktor m. Darüberhinaus wäre der Algorithmus ohne beschränkendes m nicht mehr zur Verarbeitung „on the fly“ geeignet, da er dann schon zur Bestimmung des ersten Segmentes den gesamten Polygonzug durchlaufen muß.

Abbildung 6.2 zeigt aber noch etwas anderes. Schon bei der Betrachtung der Punkte v_1, v_2 und v_3 ist klar, daß es keinen ε-Korridor um eine beliebige Gerade $\overline{v_1 v_i}$ geben kann, so daß v_2 und v_3 beide innerhalb des Korridors liegen. Sobald der Algorithmus dies bemerkt, kann er also abbrechen. Daß dies geht, liegt an der schon in Kapitel 5.6 angesprochenen Eigenschaft des ε-Korridors: Seine Breite (2ε) bestimmt sich über die ganze Länge hinweg ausschließlich aus dem Parameter ε.

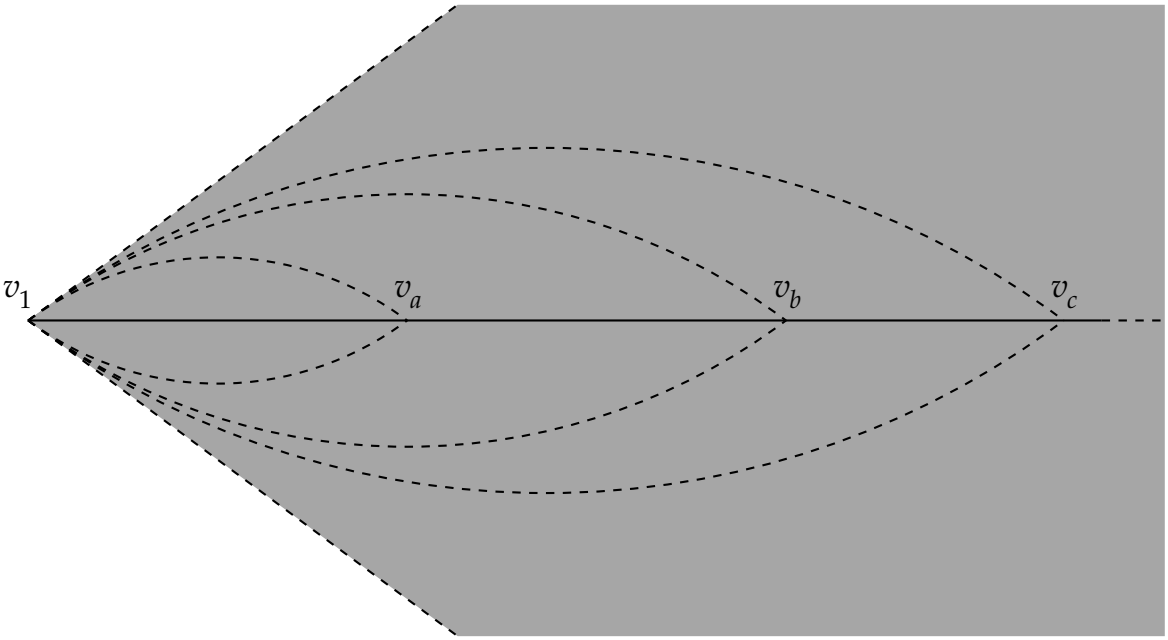


Abbildung 6.3.: Winkel-Akzeptanzbereich mit zunehmendem $|s|$ und konstantem γ

Die anderen beschriebenen Ähnlichkeitsmaße haben diese Eigenschaft nicht. So kann das Längenverhältnis $\frac{|p'|}{|s'|}$ der ersten 5 Punkte $p' = v_1v_2v_3v_4v_5$ zu $s' = [v_1v_5]$ eines Polygonzuges $p = v_1v_2 \ldots v_n$ sehr schlecht sein, ohne daß damit ausgeschlossen ist, daß $\frac{|p|}{|s|}$ trotzdem nahe 1 liegt (mit $s = [v_1v_n]$). Gleiches gilt für die Betrachtung der Fläche. Ebenso hängt beim elliptischen Akzeptanzbereich seine Breite in einer bestimmten Entfernung zu v_1 von der Länge $|s''|$ von $s'' = [v_1v_k]$ (also von der Entfernung von v_k zu v_1) ab. Aus der Kenntnis der Punkte $v_1, \ldots, v_i$ ist nicht ersichtlich, ob es noch ein v_k (mit $k > i$) in p gibt, so daß v_1 bis v_i in der zugehörigen Ellipse liegen.

Der durch das Winkel-Ähnlichkeitsmaß bestimmte Akzeptanzbereich bestimmt sich im hohen Maße durch $|s|$, wird aber auch durch den Winkel γ beschränkt, wie Abbildung 6.3 zeigt: mit zunehmend längerem s vergrößert sich der Akzeptanzbereich entsprechend, für $|s| \to \infty$ entspricht er der Fläche zwischen den Schenkeln des Winkels $\bar{\gamma} = 360° - 2\gamma$ (für $\gamma > 90°$, siehe auch Abb. 5.14) mit Scheitelpunkt v_1. Hieraus läßt sich zwar eine Abbruchbedingung konstruieren, die aber für die allermeisten Fälle höchst unzureichend ist, da der erlaubte Bereich zu groß wird.

Eine andere Möglichkeit, die Größe des Akzeptanzbereiches nach oben hin abzuschätzen, besteht darin, die maximale Länge von s abzuschätzen. Dies ist häufig problemlos möglich. Möchte man beispielsweise die Bewegung eines Serviceroboters in einem Gebäudekomplex betrachten, so kann bekannt sein, daß das maxima-

le gerade Teilstück weniger als 500 m beträgt (weil z. B. alle vorhandenen Gänge kürzer sind). Dies liefert eine Schranke für s und somit eine Abschätzung des Akzeptanzbereiches.

6.2. Der WεG Algorithmus

Wie schon oben angedeutet, läßt sich für den ε-Korridor die Tatsache nutzen, daß seine Breite nicht von $|s|$ beeinflußt wird. Damit kann ich einen Algorithmus angeben[2], der die oben angegebenen Unzulänglichkeiten nicht besitzt.[3]

Abbildung 6.4 zeigt die schrittweise Generalisierung eines Polygonzuges $p = v_1 v_2 \dots v_n$. Gesucht ist eine Strecke $s = [v_1 v_i]$, so daß alle Punkte $v_2, v_3, \dots v_{i-1}$ im durch s bestimmten ε-Korridor liegen. v_2 liegt weniger als ε von v_1 entfernt und liegt somit, unabhängig davon, in welche Richtung s zeigt, im ε-Korridor um s. v_3 liegt weiter als ε von v_1 entfernt und schränkt dadurch die mögliche Lage von s ein. Solange s im Keil K_3 liegt, liegt v_3 im ε-Korridor von s. v_3 legt also den Winkel fest, um den s gedreht werden kann.

Im nächsten Schritt kommt v_4 hinzu. Hier sind zwei Sachen wichtig: $s_{1,4} = [v_1 v_4]$ liegt im Keil K_3, d. h. $s_{1,4}$ ist eine mögliche Generalisierung des Teilstücks $v_1 v_2 v_3 v_4$. Weiterhin schränkt v_4 die mögliche Lage von s weiter ein, der Keil wird schmaler. Für den nächsten Punkt ist die Situation anders: v_5 liegt nicht in K_4, somit ist $s_{1,5} = [v_1 v_5]$ keine gültige Generalisierung des Teilstücks $v_1 v_2 v_3 v_4 v_5$.

Dies ist eine mögliche Abbruchbedingung: $v_1 v_2 v_3 v_4$ wird zu $[v_1 v_4]$ generalisiert und der Algorithmus arbeitet mit v_4 als Startpunkt des nächsten Segmentes weiter. Hiermit (Algorithmus 6) erhält man die gleiche Generalisierung, die auch $\text{simpleinc}_\varepsilon$ liefert[4], der Algorithmus benötigt hierfür jedoch nur lineare (und nicht quadratische) Laufzeit.

Abbildung 6.4 zeigt weiterhin, daß zwar $v_5 \notin K_4$, daß es aber trotzdem noch eine Strecke $s_{1,i} \in K_5$ mit $i > 5$ geben kann, so daß $v_1, v_2, \dots v_5$, da v_5 in der ε-Umgebung von K_4 liegt (dies ist gleichbedeutend mit $K_5 \neq \emptyset$). Schon im nächsten Schritt findet sich $v_6 \in K_5$, also ist $[v_1 v_6]$ eine gültige Generalisierung. K_6 (nicht mehr dargestellt) unterscheidet sich nicht von K_5, da v_6 aufgrund seiner Lage die Möglichkeiten für s nicht weiter einschränkt. v_7 liegt wieder außerhalb von $K_6 = K_5$, allerdings diesmal zu weit von K_6 entfernt (außerhalb der ε-Umgebung), als daß es noch einen ε-Korridor geben könnte, der seinen Startpunkt in v_1 hat und $v_1, v_2, \dots v_7$ umfaßt.

Liegt also v_i außerhalb der ε-Umgebung von K_{i-1} (gleichbedeutend damit: ist $K_i = \emptyset$), so kann es keinen zugehörigen ε-Korridor mehr geben. Die Generalisierung des Segmentes ist damit beendet, der letzte gültige Endpunkt v_k

[2] siehe auch [Ste98]
[3] und zudem im average case lineare Komplexität aufweist
[4] siehe aber Kapitel 6.3

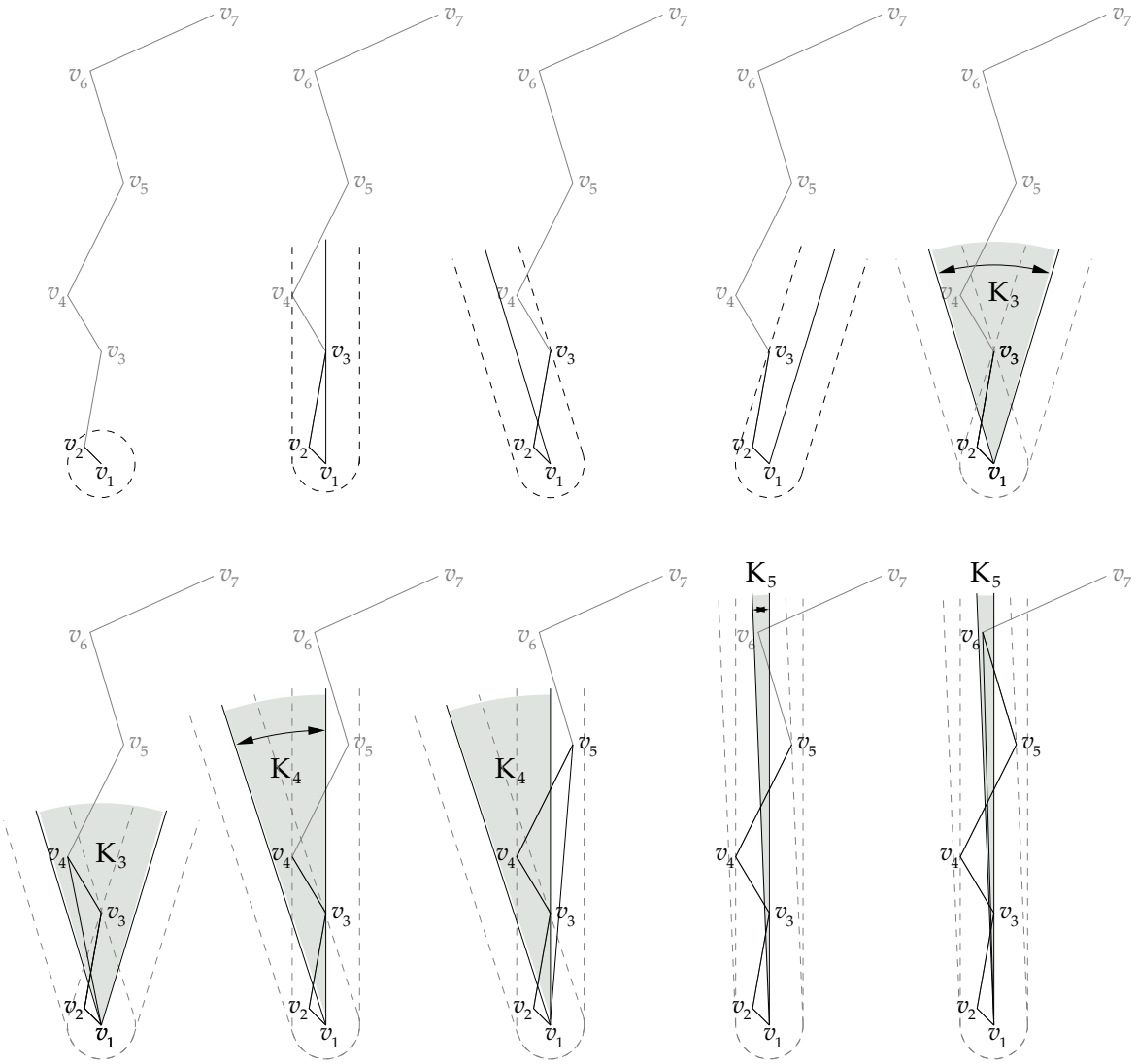


Abbildung 6.4.: WεG Algorithmus

Algorithmus 6: Linearer WεG

```
1: function rest = (Polygonzug p = v_1 ... v_n, Distanz ε) → Polygonzug :
2: begin
3:    k ← 2;
4:    K ← keil(v_1, v_2);                ▷ Keil bestimmen
5:    while (k < n) ∧ in?(v_k, K) do     ▷ Alle Punkte im Korridor
6:       k++;
7:       K ← K ∩ keil(v_1, v_k);         ▷ Keil weiter verkleinern
8:    end while
9:    if in?(v_k, K) then                ▷ Abbruch wegen k = n?
10:      return v_k v_{k+1} ... v_n       ▷ v_k ist der letzte Punkt im Segment
11:   else
12:      return v_{k-1} v_k ... v_n       ▷ v_{k-1} ist der letzte Punkt im Segment
13:   end if
14: end

15: function linWεG = (Polygonzug p = v_1 ... v_n, Distanz ε) → Polygonzug :
16: begin
17:   if n ≤ 2 then                      ▷ Erst ab 3 Punkten wird generalisiert
18:      return p
19:   else
20:      return v_1 ∘ linWεG(rest(p, ε))  ▷ rekursiv die Segmente ausgeben
21:   end if
22: end
```

($k < i, v_k \in K_{k-1}$) liefert die Generalisierung $[v_1 v_k]$ des ersten Segmentes und ist neuer Startpunkt. Dies führt zu Algorithmus 7: WεG (Wedging ε Generalization[5]). WεG liefert meist längere Segmente als linWεG und somit eine Generalisierung mit weniger Punkten.[6]

Die Berechnung des Keils ist einfach. In Abbildung 6.5 ist die Situation nochmal dargestellt: $v_2 = v_1 + \varepsilon r^{\perp} + \alpha r$, wobei r ein auf Länge 1 normierter Richtungsvektor und $r^{\perp}$ der dazu orthogonale Vektor ist. Im zweidimensionalen Fall hat diese

[5] *wedge:* Keil, Keilform; *wedging ε generalization:* einzwängende ε-Generalisierung

[6] Bei gleichem Startpunkt liefert WεG ein gleichlanges oder längeres Segment als linWεG. Unterscheiden sich beide Algorithmen im ersten Segment, so haben sie für das zweite Segment unterschiedliche Startpunkte. Dadurch kann es vorkommen, daß das zweite Segment von linWεG länger wird als das zweite Segment bei WεG. Im Mittel liefert aber WεG die knappere Generalisierung.

Algorithmus 7: WεG

function rest = (Polygonzug $p = v_1 \dots v_n$, Distanz ε) $\rightarrow$ Polygonzug :
begin
  $k \leftarrow 2$;
  $l \leftarrow k$;
  $\mathrm{K} \leftarrow \mathrm{keil}(v_1, v_2)$; ▷ *Keil bestimmen*
  while $(k < n) \wedge (\mathrm{K} \neq \emptyset)$ **do** ▷ *Ein möglicher Korridor existiert*
    k++;
    if $v_k \in \mathrm{K}$ **then** ▷ $[v_1 v_k]$ *ist Generalisierung*
      $l \leftarrow k$;
    end if
    $\mathrm{K} \leftarrow \mathrm{K} \cap \mathrm{keil}(v_1, v_k)$; ▷ *Keil weiter verkleinern*
  end while
  if $v_k \in \mathrm{K}$ **then** ▷ *Abbruch wegen k = n?*
    return $v_k v_{k+1} \dots v_n$ ▷ v_k *ist der letzte Punkt im Segment*
  else
    return $v_l v_{l+1} \dots v_n$ ▷ v_l *ist der letzte Punkt im Segment*
  end if
end

function WεG = (Polygonzug $p = v_1 \dots v_n$, Distanz ε) $\rightarrow$ Polygonzug :
begin
  if $|p| \leq 2$ **then** ▷ *Erst ab 3 Punkten wird generalisiert*
    return p
  else
    return $v_1 \circ \mathrm{W\varepsilon G}(\mathrm{rest}(p, \varepsilon))$ ▷ *rekursiv die Segmente ausgeben*
  end if
end

Gleichung genau zwei Lösungen (für $|v_1 v_2| > \varepsilon$):

$$\begin{pmatrix} x_2 \\ y_2 \end{pmatrix} = \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} + \varepsilon \begin{pmatrix} y_r \\ -x_r \end{pmatrix} + \alpha \begin{pmatrix} x_r \\ y_r \end{pmatrix}; \quad x_r^2 + y_r^2 = 1$$

Zusätzlich kann man noch nutzen, daß $\varepsilon^2 + \alpha^2 = |v_1 v_2|^2$. Hieraus berechnen sich zwei Lösungen r_L und r_R für r, die sofort $s_\mathrm{L} = v_1 + \beta r_\mathrm{L}$ und $s_\mathrm{R} = v_1 + \beta r_\mathrm{R}$ (mit $\beta \geq 0$) liefern. Gilt für einen Punkt v_i: $|v_1 v_i| \leq \varepsilon$, so besteht der Keil aus der ganzen Ebene.

Auch das Schneiden zweier Keile ist trivial. Der Winkel zwischen s_L und s_R ist immer kleiner 180°. Um zwei Keile $\mathrm{K}_1 = (s_{1\mathrm{L}}; s_{1\mathrm{R}})$ und $\mathrm{K}_2 = (s_{2\mathrm{L}}; s_{2\mathrm{R}})$ zu schneiden ($\mathrm{K} = \mathrm{K}_1 \cap \mathrm{K}_2$), müssen lediglich die Halbgeraden $s_{1\mathrm{L}}, s_{1\mathrm{R}}, s_{2\mathrm{L}}, s_{2\mathrm{R}}$ sortiert werden.

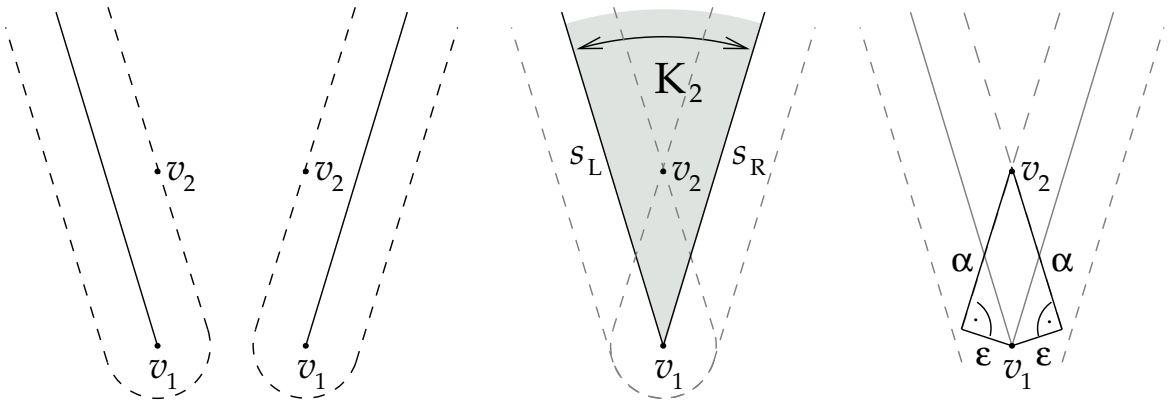


Abbildung 6.5.: Keilberechnung

Es gibt eine überschaubare Anzahl von Fällen:

- Liegt s_{2L} rechts (in der rechten Halbebene) von s_{1L}, dann $s_L = s_{2L}$,
  sonst $s_L = s_{1L}$.
- Liegt s_{2R} links (in der linken Halbebene) von s_{1R}, dann $s_R = s_{2R}$,
  sonst $s_R = s_{1R}$.
- Liegt s_R rechts (in der rechten Halbebene) von s_L, dann $K = (s_L; s_R)$,
  sonst $K = \emptyset$.

Der Keilansatz liefert nicht nur schnellere Algorithmen als simpleinc$_\varepsilon$, sondern darüberhinaus Zusatzinformationen: Wird während der Bewegung eines Objektes dessen Spur generalisiert, so hängt die Generalisierung der Bewegung immer ein Stück hinterher (die Generalisierung muß gewissermaßen „warten“, bis ein neuer Eckpunkt erreicht ist). linWεG und WεG sind für dieses letzte Stück jedoch nicht völlig blind, vielmehr liefert der berechnete Keil Informationen darüber, welche Richtung das aktuelle Segment haben wird. Diese Schätzung wird schnell ziemlich genau, wie in Abbildung 6.4 sichtbar wird: Schon K_3 schränkt die mögliche Richtung auf einen Bereich von 37° ein, K_4 auf 18° und K_5 schließlich auf nur noch 3°, die Richtungsschätzung gewinnt also schnell an Genauigkeit. Allgemein gilt für einen Punkt v_k, daß K_k maximal einen Winkel $\gamma = 2 \arcsin \frac{\varepsilon}{|v_1 v_k|}$ einschließt. Da K_k den Schnitt der Keile zu v_2 bis v_k darstellt, ist γ im Mittel noch wesentlich spitzer. Konkret beschränkt schon ein Punkt v_k, der nur 4ε von v_1 entfernt liegt ($|v_1 v_k| = 4\varepsilon$), γ auf weniger als 30°.

linWεG liefert also schon während der Generalisierung des aktuellen Segmentes eine gute Schätzung für dessen Richtung, was den Algorithmus für die Verarbeitung „on the fly“ gut geeignet macht. Für WεG ist diese Richtungsschätzung grundsätzlich natürlich ebenso möglich, allerdings nicht ganz so einfach. In linWεG

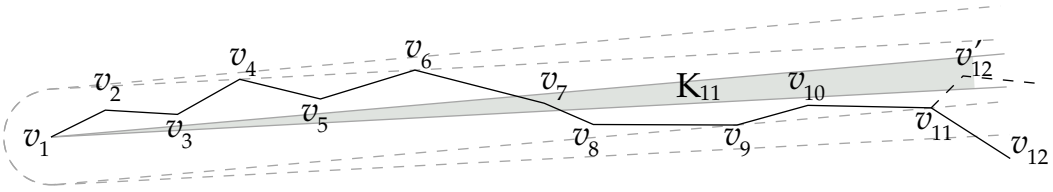


Abbildung 6.6.: Probleme mit WεG

ist jeder betrachtete Punkt möglicher Endpunkt, beim ersten außerhalb des Keils liegenden Punkt wird abgebrochen. WεG hingegen läuft auch weiter, solange ein Punkt v_i zwar außerhalb des Keils K_i aber innerhalb dessen ε-Umgebung liegt. In Abbildung 6.6 ist v_7 der letzte gültige Endpunkt, linWεG würde hier abbrechen, WεG noch nicht. v_8, v_9, v_{10} und v_{11} liegen zwar nicht mehr in den jeweiligen Keilen K_7, K_8, K_9 bzw. K_{10}, jedoch in der ε-Umgebung des entsprechenden Keils, oder anders: nach v_{11} könnte ja noch ein Punkt $v'_{12} \in K_{11}$ kommen, womit $[v_1 v'_{12}]$ eine gültige Generalisierung darstellt. Da v_{12} aber außerhalb der ε-Umgebung um K_{11} liegt, bricht hier nun auch WεG ab, und da v_7 der letzte gültige Endpunkt war, wird das erste Segment zu $v_1 v_7$ generalisiert und die Generalisierung des zweiten Segmentes mit v_7 als neuem Startpunkt neu begonnen.

Neben erhöhtem Rechenaufwand bedeutet dies, daß die Daten der Richtungsschätzung ab v_7 nicht mehr korrekt sind. Sie sind nicht *falsch*, der Keil K_{11} liefert eine korrekte Schätzung der durch die Punkte v_1 bis v_{11} beschriebenen Richtung, die aber ab v_7 nicht mehr mit der endgültigen Generalisierung übereinstimmen muß. Ist dies nicht hinreichend, so bietet sich entweder an, auf die längeren Segmente von WεG zu verzichten und linWεG zu benutzen, oder eine doppelte Vorausschau zu betreiben: Da v_7 der letzte gültige Endpunkt ist, werden zusätzlich zur gezeigten Keilberechnung vorsorglich mit v_7 als Startpunkt auch die Keile $K'_8, K'_9, \ldots$ bestimmt. K_i liefert also nun die Schätzung für den Fall, daß noch ein weiterer gültiger Eckpunkt im Segment gefunden wird, K'_i für den Fall, daß v_7 sich als letzter gültiger Eckpunkt des Segmentes erweist.[7]

Findet sich noch ein weiterer gültiger Endpunkt v'_{12}, so sind die K'_i veraltet und die Parallelberechnung startet mit v'_{12} als potentiellem Startpunkt erneut, ist dagegen v_7 letzter gültiger Endpunkt im Segment, so liegen $K'_8, K'_9, \ldots$ schon vor und müssen nicht erneut berechnet werden.

Ob und wie ein weiterverarbeitender Algorithmus mit diesen Schätzungen etwas anfangen kann, hängt von der konkreten Anwendung ab, in einigen Fällen wird die pragmatische Lösung, sich auf linWεG zu beschränken, das Mittel der Wahl sein, in anderen ist schon die einfache Schätzung hinreichend.[8]

[7] Auch bei der Berechnung der K'_i können wieder Punkte v_j vorkommen, die nicht in K'_{j-1}, aber in der ε-Umgebung zu K'_{j-1} liegen. In diesem seltenen Fall müßte theoretisch eine weitere Vorausschau hinzukommen, in der Praxis sollte dies aber nicht nötig sein.

[8] Die hier beschriebene doppelte Vorausschau ist bei Einsatz der in Kapitel 6.4 beschriebenen

6.3. Schleifenerkennung

simpleinc$_\varepsilon$ erkennt wie oben beschrieben Schleifen und Rückläufigkeiten ganz einfach dadurch, daß in jedem Schritt der Abstand zur Strecke s statt zur Geraden g bestimmt wird. Der Keilansatz von WεG liefert diese Schleifenerkennung nicht so einfach mit, hier ist zusätzliche Arbeit nötig.

Ein Wendepunkt v_w ist einfach zu bestimmen: Liegt v_w weiter von v_1 entfernt als v_{w+1} ($|v_1v_w| > |v_1v_{w+1}|$), so ist v_w ein Wendepunkt im Segment. Damit ist auch eine sehr einfache Schleifenbestimmung möglich: Sei $\delta_l = \max_{i\leq l}\{|v_1v_i|\}$. Für $|v_1v_l| < \delta_l$ befindet sich die Bewegung im Punkt v_l in einer Schleife oder Rückläufigkeit. Wird diese zu groß ($\delta_l - |v_1v_l| > \varepsilon'$, ε' gibt also wieder an, wie groß die Schleifen maximal werden dürfen, um nicht erfaßt zu werden. Es bietet sich auch hier an, $\varepsilon' = \varepsilon$ zu wählen), so bricht die Generalisierung an dieser Stelle ab. Der letzte gültige Endpunkt v_k liefert wie gehabt mit $[v_1v_k]$ die Generalisierung des Segmentes und dient als neuer Startpunkt.

Dieses Vorgehen liefert allerdings (in den Schleifen) andere Generalisierungen als simpleinc$_\varepsilon$, da hier der Unterschied im Abstand zweier Punkte zu v_1 bestimmt wird, wogegen in simpleinc$_\varepsilon$ der Abstand beider Punkte zueinander relevant ist. Dieses Verhalten ließe sich allerdings nur wesentlich aufwendiger nachbilden.

Beide Algorithmen (simpleinc$_\varepsilon$ und WεG) erkennen nun also Schleifen und brechen die Generalisierung an ihnen ab. Die Bestimmung des Endpunktes v_k liefert jedoch in bestimmten Fällen sehr schlechte Ergebnisse, wie Abbildung 6.7 zeigt:[9] Der Polygonzug $p = v_1v_2\ldots v_9$ enthält zwischen v_4 und v_7 eine Schleife, die auch richtig erkannt wird. $[v_1v_5]$ ist eine gültige Generalisierung, da v_2, v_3 und v_4 in der zugehörigen ε-Umgebung liegen. Im nächsten Schritt liegt v_5 nicht in der ε-Umgebung von $[v_1v_6]$, also bricht der Algorithmus hier ab, $[v_1v_5]$ ist Generalisierung des ersten Segmentes, v_5 Startpunkt für die weitere Generalisierung. Dies liefert am Ende die Generalisierung $q = v_1v_5v_9$. Die Schleife wird also erkannt, ohne Schleifenerkennung würde zu v_1v_9 generalisiert, ist aber in q nicht als Schleife sichtbar.

Dies liegt daran, daß $|v_4v_7| \leq 2\varepsilon$ und $|v_4v_5| \leq \varepsilon, |v_7v_5| \leq \varepsilon$. Dies bedeutet, daß sowohl $p_a = v_1v_2v_3v_4v_5$ bezüglich $q_a = v_1v_5$ als auch $p_b = v_5v_6v_7v_8v_9$ bezüglich $q_b = v_5v_9$ die ε-Bedingung erfüllt, die Generalisierung ist also „formal korrekt“, die markanten Ecken der Bewegungsspur werden allerdings nicht gefunden.

Eckensuche nicht mehr (oder nur unter wesentlich erhöhtem Aufwand) möglich.

[9] Die Abbildung zeigt den Effekt am Beispiel von simpleinc$_\varepsilon$, er tritt aber, wie man sich leicht überlegt, in ähnlicher Form auch bei WεG auf.

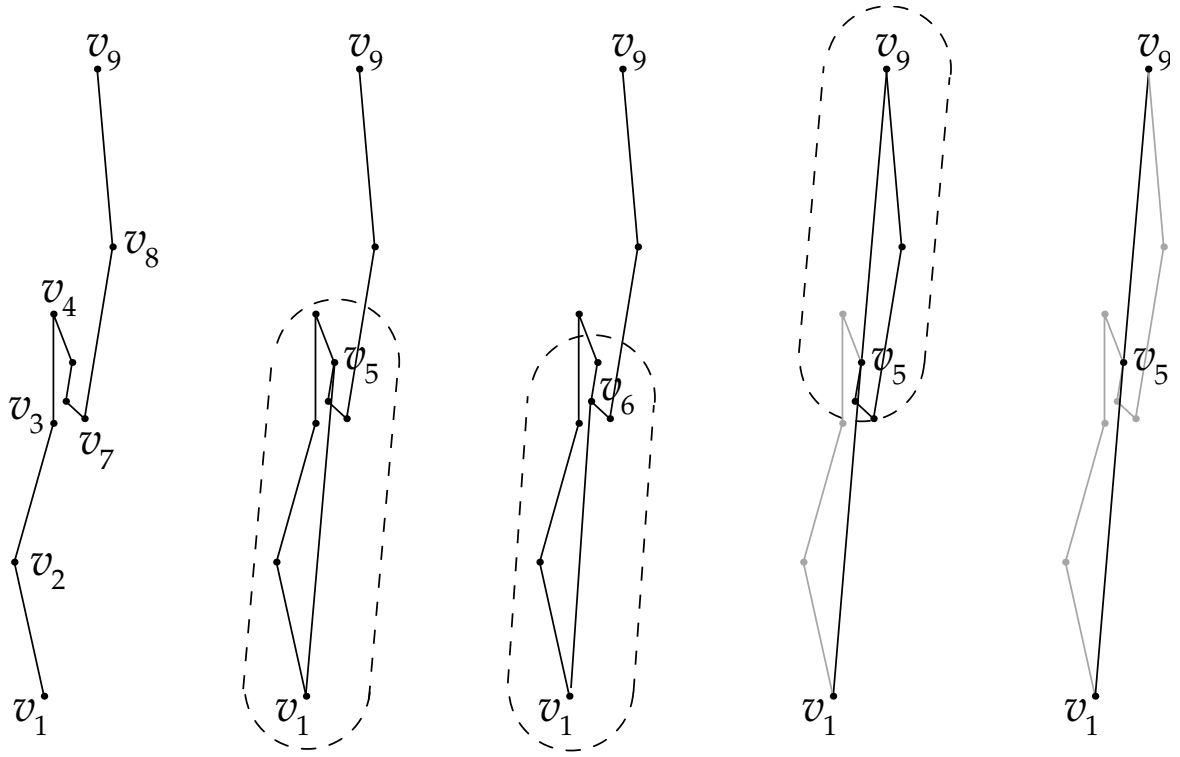


Abbildung 6.7.: Probleme bei der Schleifenerkennung mit simpleinc$_\varepsilon$

6.4. Eckensuche

Im gezeigten Fall würde die Generalisierung $v_1v_4v_7v_9$ die Bewegungsspur (vor allem die Schleife) weit besser beschreiben. Ein möglicher Ansatz besteht darin, bei der Schleifenerkennung nicht den letzten gültigen Endpunkt v_5, sondern den Wendepunkt v_4 zu wählen. Für simpleinc$_\varepsilon$ muß hierzu ein Wendepunkt gesondert bestimmt werden. WεG bestimmt in seiner Schleifenerkennung zwar sowieso alle Wendepunkte, hier kann es allerdings passieren, daß für einen Wendepunkt v_w die Strecke $[v_1v_w]$ keine gültige Generalisierung darstellt. In diesem Fall muß dann zusätzlich von v_w ausgehend eine gültige Generalisierung gesucht werden.

Doch nicht nur bei Schleifen ist die Erkennung markanter Ecken durch die beiden Algorithmen nicht sonderlich gut. Abbildung 6.8 zeigt die inkrementelle Generalisierung des schon aus Abbildung 5.1 bekannten Polygonzuges (dies erlaubt einen Vergleich mit Douglas/Peucker), in Abbildung 6.9 wird der gleiche Polygonzug von der anderen Seite aus inkrementell durchlaufen. Man sieht deutlich, daß die Generalisierung nicht auf den markanten Ecken der Originalspur beruht.

Die Douglas/Peucker-Generalisierung wählt den zu einer Referenzstrecke am weitesten entfernt liegenden Punkt als Ecke, die Wahrscheinlichkeit, daß dieser Punkt ein markanter Eckpunkt der Originalspur ist, ist daher hoch (wenn auch

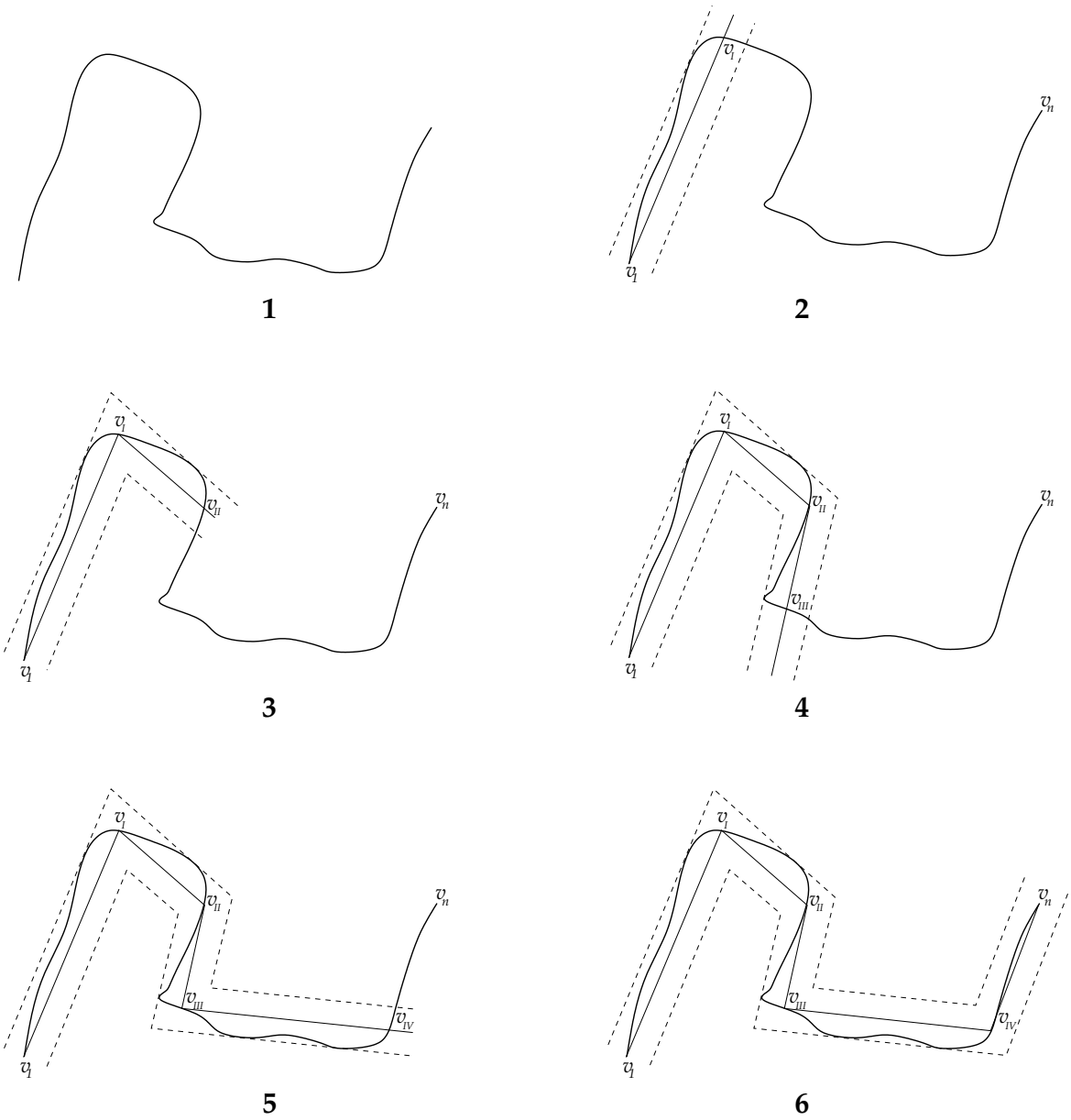


Abbildung 6.8.: Inkrementelle Generalisierung (von links nach rechts)

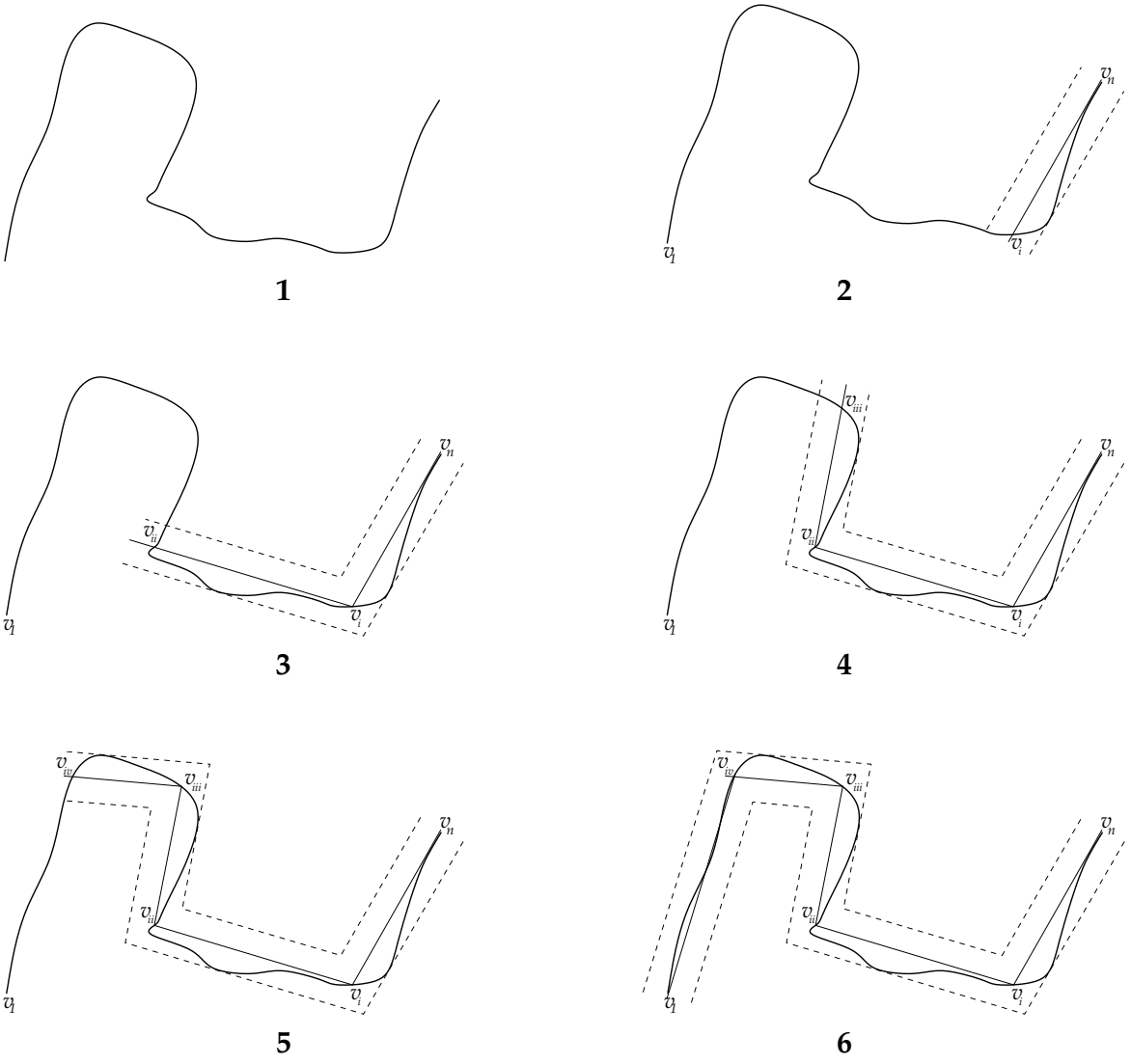


Abbildung 6.9.: inkrementelle Generalisierung (von rechts nach links)

nicht garantiert, siehe auch Kapitel 5.7). Die inkrementelle Generalisierung dagegen bricht ein Segment an der Stelle ab, wo der Polygonzug den betrachteten Korridor seitlich verläßt, die Wahrscheinlichkeit, daß dies ein markanter Endpunkt ist, ist also eher gering, wie man auch in den Abbildungen sieht: Die Generalisierung eines Segmentes „schneidet“ hier gewissermaßen die Kurve.

Der durch die inkrementelle Generalisierung gewählte Endpunkt des Segmentes ist allein dadurch ausgezeichnet, daß er der letzte Punkt ist, für den der Polygonzug die ε-Bedingung erfüllt. Damit repräsentiert er allerdings in den meisten Fällen keine besonders markante Ecke der Bewegungsspur.

Da kein zwingender Grund besteht, gerade den letzten gefundenen Punkt zum Endpunkt des betrachteten Segmentes zu machen, kann durch die Wahl eines „markanteren“ Endpunktes eine „bessere“ Generalisierung erreicht werden.

Der Algorithmus ändert sich dadurch wie folgt: Ausgehend vom Startpunkt v_1 wird das erste Segment wie gehabt abgearbeitet, bis die Abbruchbedingung erfüllt ist. Danach wird nicht der letzte Punkt v_n im Segment als Endpunkt des Segmentes und neuer Startpunkt gewählt, sondern unter allen Punkten des Segmentes die markanteste Ecke $v_\angle$ gesucht. $v_\angle$ ist dann Endpunkt des Segmentes und neuer Startpunkt des nächsten Segmentes.

Bei der Auswahl von $v_\angle$ unterscheiden sich linWεG und simpleinc$_\varepsilon$ zum einen[10] und WεG zum anderen in einem wesentlichen Punkt. Während bei ersterem alle Punkte $v_1, v_2, \ldots v_n$ als mögliche Eckpunkte in Frage kommen, kann es bei WεG Punkte $v_k, v_l, \ldots$ geben, die zwar im ε-Korridor zu $[v_1 v_n]$ liegen, die selbst aber keine gültigen Generalisierungen $[v_1 v_k], [v_1 v_l], \ldots$ darstellen.

Der WεG Algorithmus muß hier lediglich so modifiziert werden, daß er sich zu jedem Punkt v_k merkt, ob er Endpunkt einer gültigen Generalisierung $[v_1 v_k]$ ist. Praktischerweise erzeugt WεG hierzu beim Durchlaufen des Segmentes eine Liste möglicher gültiger Endpunkte.

Nun benötigt man noch eine Bewertungsfunktion, um aus diesen möglichen Eckpunkten den „markantesten“ auszuwählen. Die Situation entspricht hierbei nicht ganz der Suche nach dem besten Teilungspunkt für Douglas/Peucker in Kapitel 5.7, da hier noch keine Informationen über den weiteren Verlauf der Bewegungsspur vorliegen, eine einfache Teilung also nicht unbedingt gewünscht ist. Trotzdem ist der Teilungsansatz geeignet, „bessere“ Eckpunkte zu bestimmen.

In ihrer Betrachtung des AEG-Algorithmus (simpleinc$_\varepsilon$) schlagen Douglas und Peucker vor, nicht den letzten gültigen Punkt v_k, sondern den zu $[v_1 v_k]$ am weitesten entfernt liegenden Punkt $v_i \in \{v_2, v_3, \ldots v_{k-1}\}$ als Ecke zu wählen. Damit imitieren sie in gewisser Weise die Eckenwahl ihres eigenen Algorithmus im Kleinen. Es erscheint mir jedoch nicht einsichtig, hier den Abstand zu $[v_1 v_k]$ als Maß zu nutzen. Wie z. B. in Abbildung 6.10 zu sehen ist, verspricht der Abstand zu

[10] linWεG und simpleinc$_\varepsilon$ liefern ja (abgesehen von der Schleifenerkennung) identische Generalisierungen

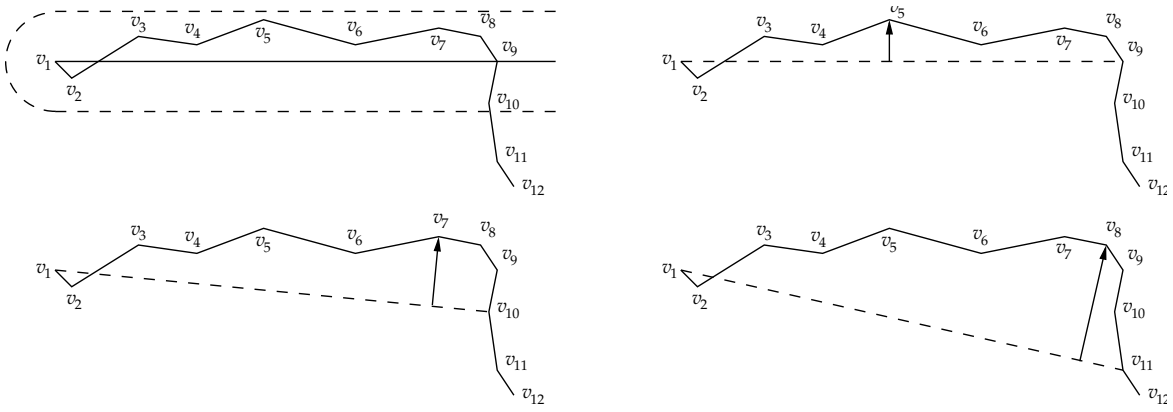


Abbildung 6.10.: Eckensuche mittels maximaler Distanz für WεG und simpleinc$_\varepsilon$

$[v_1 v_m]$ mit $v_m = v_{k+1}$ bessere Ergebnisse. Im Fall von WεG sollte man sogar den ersten völlig außerhalb des Korridors liegenden Punkt als Endpunkt der Referenzstrecke wählen. Die Abbildung zeigt links oben den ε-Korridor der letzten gültigen Generalisierung $[v_1 v_9]$. Der dazu maximal entfernte Punkt ist v_5, was nicht unbedingt einen guten Eckpunkt darstellt. v_{10} ist der Abbruchpunkt für simpleinc$_\varepsilon$ und die Referenzstrecke $[v_1 v_{10}]$ liefert v_7 als entferntesten Punkt, was schon besser ist. v_{11} schließlich ist der erste Punkt, der außerhalb jedes möglichen ε-Korridors liegt, somit die Abbruchbedingung für WεG, und $[v_1 v_{11}]$ liefert mit dem am weitesten entfernt liegenden Punkt v_8 auch die beste Ecke. Dies muß natürlich nicht so sein, allgemein gilt jedoch: Je stärker die Referenzstrecke gegenüber dem Korridor geneigt ist, desto weiter liegt die gewählte Ecke $v_\angle$ von v_1 entfernt, was zu längeren Segmenten führt. Der Wahl des Eckpunktes durch maximalen Abstand zu $[v_1 v_k]$ führt im Mittel zu einer Verdoppelung der Punktzahl der Generalisierung, die Referenzstrecke $[v_1 v_{k+1}]$ verbessert dies, und im Fall von WεG kann $[v_1 v_m]$ (wobei v_m der erste Punkt mit $K_m = \emptyset$ ist) gewählt werden, was eine weitere Verbesserung darstellt.

Auch hier ließe sich, wie schon bei der Eckensuche für Douglas/Peucker, statt dem maximalen Abstand grundsätzlich wieder ein elliptisches Maß wählen. Hierdurch steigt die Wahrscheinlichkeit, einen Punkt nahe v_k zu erwischen.[11]

Abbildung 6.11 macht die Grundsituation nochmals deutlich: Bei der Generalisierung des Polygonzuges $v_1 v_2 \ldots v_{17}$ ist $[v_1 v_8]$ die letzte gültige Generalisierung des ersten Segmentes. Für simpleinc$_\varepsilon$ (bzw. linWεG) ist demnach v_9 der letzte betrachtete Punkt. v_{10} ist der letzte Punkt, für den es einen ε-Korridor geben kann ($K_{10} \neq \emptyset, K_{11} = \emptyset$), somit ist für WεG v_{11} der letzte betrachtete Punkt.

In der Abbildung ist oben der Polygonzug mit ε-Korridor abgebildet, in der

[11] Das elliptische Maß gewichtet Punkte in Nähe des Start- und Endpunktes stärker als solche in der Mitte. Punkte nahe v_1 liegen im ε-Korridor zu $v_1 v_k$, also sehr nahe an v_1, Punkte nahe v_k hingegen zwar in der Gegend von v_m, jedoch häufig auch weiter als ε entfernt.

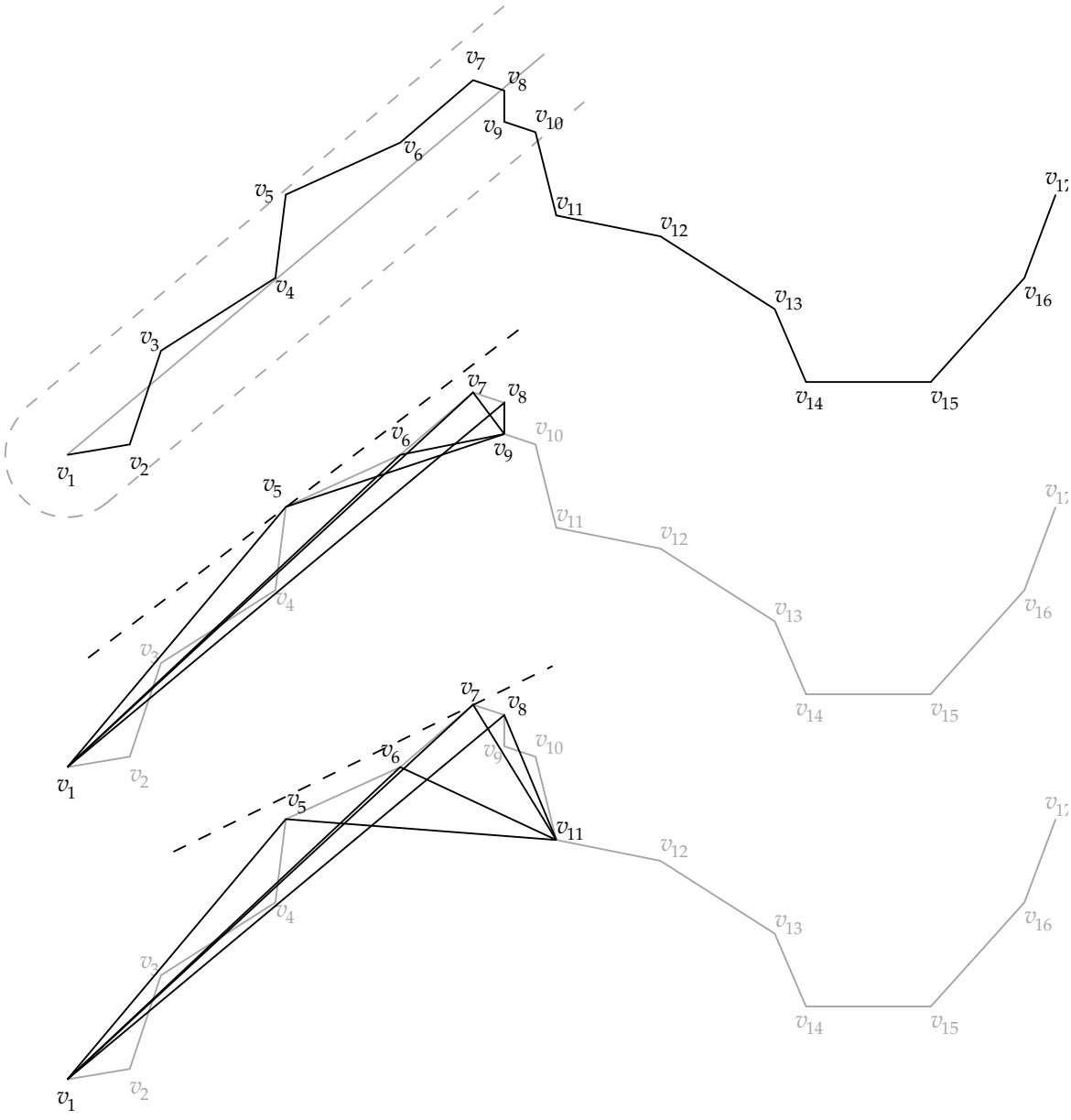


Abbildung 6.11.: Ecken finden

Mitte die Eckensuche für simpleinc$_\varepsilon$ und unten die Eckensuche für WεG. Im ersten Fall wird ein Punkt v_m bezüglich v_1 und v_9 getestet, im zweiten bezüglich v_1 und v_{11}. Im Vergleich zeigen sich hier die Unterschiede deutlich. Bezüglich $[v_1v_9]$ ist v_5 der am weitesten entfernt liegende Punkt, $|v_1v_7| + |v_7v_9|$ die längste Strecke (elliptisches Maß), und $\measuredangle v_1v_8v_9$ der spitzeste Winkel. Die drei unterschiedlichen Maße liefern also auch drei unterschiedliche Ecken.

Bezüglich $[v_1v_{11}]$ liegt v_7 am weitesten entfernt, ebenso ist $|v_1v_7| + |v_7v_{11}|$ die längste Strecke, und schließlich ist $\measuredangle v_1v_8v_{11}$ der spitzeste Winkel. Die durch die unterschiedlichen Maße gewählten Eckpunkte rutschen hier also zusammen, was daran liegt, daß v_{11} schon ein gutes Stück weit entfernt vom gezeigten ε-Korridor liegt.

Ein Referenzpunkt in ausreichender Entfernung verbessert also die Eckensuche. Nun ist im Beispiel der Punkt v_{11} hinreichend weit entfernt: Sei l_{11} der Fußpunkt des Lotes von v_{11} auf die Referenzgerade $g = \overline{v_1v_8}$, so gilt $|v_1l_{11}| \approx 3.5|v_{11}l_{11}|$. Dies gilt aber nicht, wenn die Länge des betrachteten Segmentes im Verhältnis zu ε sehr groß ist, was zu Relationen von $|v_1l_t| > 10|v_tl_t|$ oder mehr führen kann.[12] Optimal für die Eckensuche wäre ein Referenzpunkt, der ähnlich weit von der potentiellen Ecke entfernt liegt, wie der Startpunkt v_1. Im betrachteten Beispiel wäre demnach v_{15} eine gute Wahl.

Da zum Zeitpunkt der Eckensuche nur die Punkte bis v_{11} betrachtet wurden, ist eine Vorschau nötig: v_8 als letzter gültiger Punkt wird als vorläufiger Eckpunkt akzeptiert und von v_8 ausgehend ein neues Segment generalisiert. Der letzte gültige Punkt dieses neuen Segmentes nun (im Beispiel v_{14}) bildet schließlich den Referenzpunkt, bezüglich dem die Ecke bestimmt wird (Abb. 6.12 oben und Mitte). Bezüglich $[v_1v_{14}]$ liegt v_7 am weitesten entfernt, $|v_1v_7| + |v_7v_{14}|$ ist die größte Entfernung und $\measuredangle v_1v_7v_{14}$ der spitzeste Winkel. Daß die Maxima der drei Maße zusammenfallen, ist natürlich nicht zwingend. Es ist jedoch anschaulich klar, daß ein vom ε-Korridor des betrachteten Segmentes weiter entfernt liegender Referenzpunkt für die Eckensuche geeigneter ist.

Der Nachteil dieser Vorausschau besteht zum einen darin, daß sie selbstverständlich die Laufzeit des Algorithmus erhöht, da für jede Ecke zusätzlich ein Segment vorausberechnet werden muß,[13] im Mittel also eine Verdoppelung der Laufzeit. Dieser erhöhte Rechenaufwand ist aber (vor allem bei der Geschwindigkeit heutiger Computer) nicht das große Problem (die Komplexität steigt hierdurch nicht). Gravierender ist, daß durch so eine Vorausschau die Verzögerung für die Bewegungsverarbeitung „on the fly“ zu groß wird. Wird die Spur eines Objekts während seiner Bewegung generalisiert, um mit diesen Daten seine weiteren Aktionen (oder die Aktionen anderer Roboter) zu steuern, so müssen selbstverständlich

[12] mit v_t Referenzpunkt

[13] lediglich wenn zufällig der vorläufige Eckpunkt und der gefundene Eckpunkt übereinstimmen, ist eine Einsparung möglich

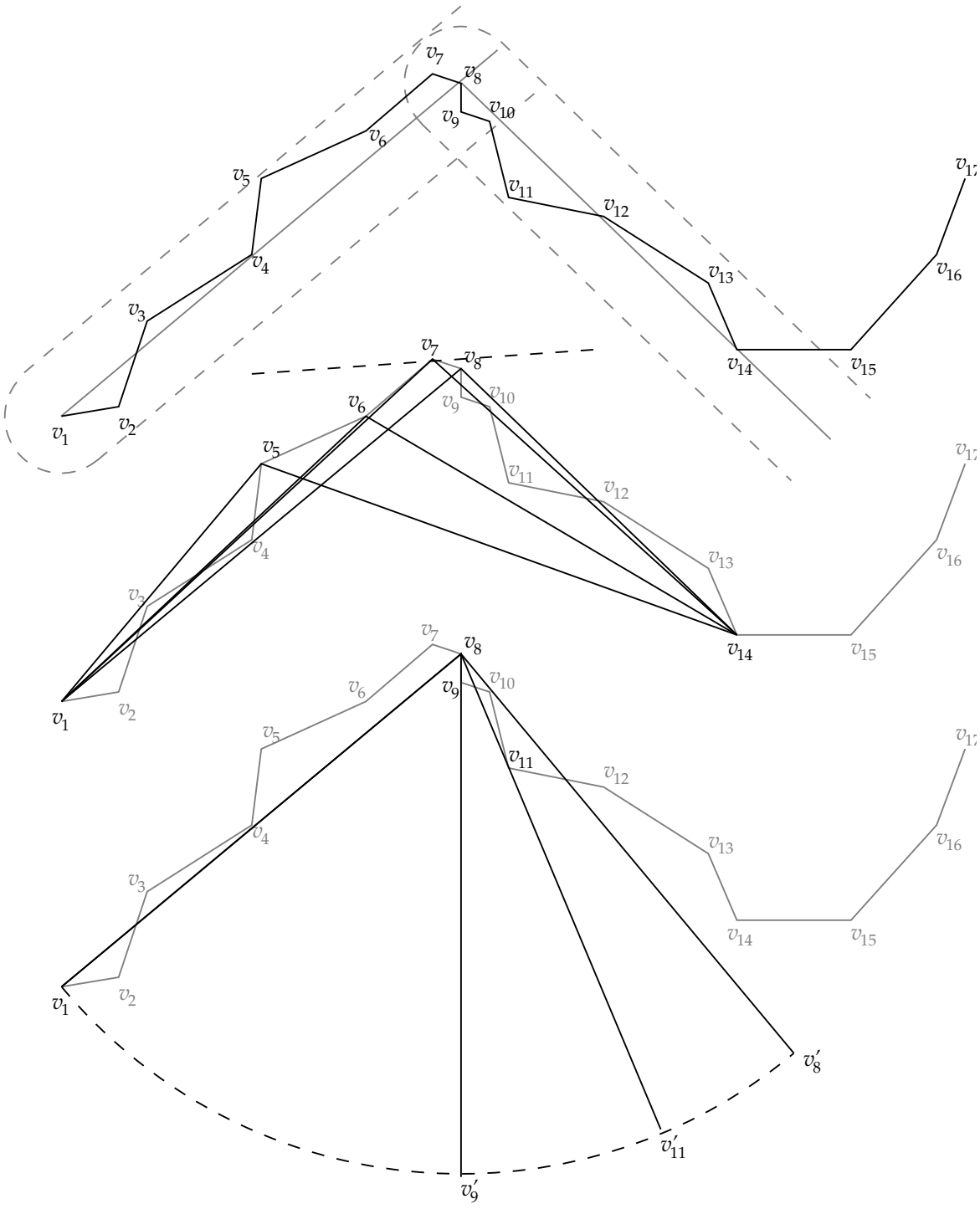


Abbildung 6.12.: Vorschau um ein Segment

Ecken (Kurven in der Bewegung, das Abbiegen des Gefährts) möglichst frühzeitig erkannt werden, Verzögerungen in den gezeigten Größenordnungen sind dann häufig nicht hinnehmbar (die Generalisierung würde durchgehend mehr als ein Segment hinter der aktuellen Position herhinken).

Wenn ich also nicht die Möglichkeit habe zu warten, bis sich beispielsweise mein Roboter ein ganzes Segment vorausbewegt hat, um die nächste Ecke zu bestimmen, aber andererseits ein möglichst entfernt liegender Referenzpunkt für die Eckensuche wünschenswert ist, bleibt, einen möglichen Referenzpunkt zu schätzen.

Hierzu kann ich zum einen von der letzten bekannten Bewegungsrichtung ausgehen und annehmen, daß sich die Bewegung in diese Richtung fortsetzt.[14] Sei v_k der letzte gültige Endpunkt eines Segmentes und v_m wie oben der letzte betrachtete Punkt (im Fall von linWεG ist $v_m = v_{k+1}$, im Fall von WεG der erste Punkt mit $K_m = \emptyset$). Die Gerade $\overline{v_k v_m}$ gibt also an, in welche Richtung sich die Bewegung von v_k aus fortsetzt. Nun verschiebt man v_m entlang dieser Geraden (von v_k weg) und erhält einen Punkt $v'_m \in \overline{v_k v_m}$ mit $|v_k v'_m| = |v_1 v_k|$. Abbildung 6.12 unten zeigt die Bestimmung von v'_9 (als Referenzpunkt für linWεG) bzw. v'_{11} (als Referenzpunkt für WεG).

Wie man sieht, liegen die beiden Referenzpunkte ziemlich weit auseinander, was daran liegt, daß $\overline{v_8 v_9}$ bzw. $\overline{v_8 v_{11}}$ (denkbar wäre außerdem $\overline{v_{10} v_{11}}$) eine sehr lokale und damit fast zufällige Richtungsinformation liefert: Ein kleiner Schlenker an dieser Stelle kann den Referenzpunkt schnell um 60° drehen ($\overline{v_8 v_{11}}$ ist hier noch der robusteste Wert, da er die Richtung über mehrere Punkte mittelt).

Da diese lokale Richtungsschätzung also sehr ungenaue Ergebnisse liefert und das Wesentliche am gesuchten Referenzpunkt ist, daß er ein gutes Stück vom ε-Korridor entfernt liegt, bietet es sich an, einfach von v_k aus rechtwinklig zu g einen Punkt v'_k im Abstand von $|v_1 v_k|$ zu v_k als Referenzpunkt zu bestimmen. Verläßt die Bewegungsspur den Korridor auf der rechten Seite (v_m liegt rechts von g), so liegt auch v'_k auf der rechten Seite, verläßt sie den Korridor auf der linken Seite, dann liegt er links ($\triangle v_1 v_k v'_k$ ist ein gleichschenkliges rechtwinkliges Dreieck). Im Beispiel aus Abbildung 6.12 unten gilt für den Referenzpunkt v'_8: $\measuredangle v_1 v_8 v'_8 = 90°$ und $|v_1 v_8| = |v'_8 v_8|$.

Unabhängig davon, für welche Schätzung man sich entscheidet, dient der so gefundene Punkt v'_m (im Beispiel v'_9, v'_{11} bzw. v'_8) als Referenzpunkt für die Eckensuche mit einem der angegebenen Maße (maximale Entfernung zu $|v_1 v'_m|$ (Distanz), Maximum der Weglänge $|v_1 v_\angle| + |v_\angle v'_m|$ (Ellipse), oder spitzester Winkel $\measuredangle v_1 v_\angle v'_m$ (Kreisbogen)). $v_\angle$ bildet den Endpunkt des betrachteten Segmentes und den Startpunkt für die Generalisierung des nächsten Segmentes.

Einen Sonderfall schließlich bietet das letzte Segment. Wird die Generalisierung im Punkt v_n abgebrochen, weil hier der Polygonzug p zu Ende ist, so findet

[14] Vorschlag von Thomas Röfer [Röf99] für die in Kapitel 7 beschriebene Rolland-Generalisierung.

keine Eckensuche mehr statt, v_n ist Endpunkt.

6.5. Komplexität

simpleinc$_\varepsilon$ und linWεG sind historiefreie inkrementelle Generalisierungen und besitzen demzufolge über die Länge der Spur lineare Komplexität. Wie schon oben gezeigt, besitzt simpleinc$_\varepsilon$ in der Länge eines Segmentes eine Komplexität von $O(n^2)$, linWεG ist auch hier linear. Abgesehen von der etwas umständlicheren Schleifenerkennung ist daher linWεG der simpleinc$_\varepsilon$-Generalisierung auf alle Fälle vorzuziehen (und liefert zusätzlich die Richtungsschätzung durch die Angabe des letzten Keils).

WεG ist *fast* eine historiefreie inkrementelle Generalisierung, entspricht aber nicht ganz der vorne (Kapitel 4.2) angegebenen Definition. Dies liegt daran, daß WεG eine Vorausschau betreibt. Bei der Generalisierung eines Segmentes kann es vorkommen (und kommt es nicht selten vor), daß schon Punkte des darauffolgenden oder auch mehrerer darauf folgender Segmente betrachtet werden müssen. Diese Vorausschau betrifft meist nur das folgende Segment – in diesem (durchschnittlichen) Fall ist WεG eine historiefreie inkrementelle Generalisierung und damit wie linWεG $O(n)$.

Der worst case für WεG sieht so aus, daß bei der Generalisierung einer Bewegungsspur $p = v_1 v_2 \dots v_n$ der Punkt v_2 den letzten gültigen Endpunkt darstellt, aber alle weiteren Punkte v_i in der ε-Umgebung von K_{i-1} liegen, so daß zur Generalisierung des ersten Segment der gesamte Polygonzug durchlaufen wird, obwohl schließlich das erste Segment zu $v_1 v_2$ generalisiert wird. Soweit steigt die Komplexität noch nicht, statt einmal wird p zweimal durchlaufen, und $O(2n) = O(n)$. Nur wenn nun bei der Generalisierung des zweiten Segmentes, ausgehend von v_2, wieder v_3 als letzter gültiger Endpunkt erkannt wird *und* die weiteren Punkte *wieder* in den ε-Umgebungen der jeweiligen Keile bezüglich v_2 liegen, und dies auch für die folgenden Segmente fortgesetzt gilt, wird die Laufzeit quadratisch.[15]

Diese Struktur kann aber fast nicht auftreten, wie Abbildung 6.13 anschaulich macht. v_3 liegt in der ε-Umgebung K_2^ε zu K_2, das heißt, daß v_3 zwar kein gültiger Endpunkt ist, aber WεG weitersucht. Um die oben beschriebene „worst case“ Struktur zu erfüllen, muß nun v_4 im markierten Bereich M liegen, weil nur in diesem Bereich sowohl die Bedingung für das erste als auch für das zweite Segment erfüllt ist, in der ε-Umgebung des jeweiligen Keils zu liegen.

Um nun auf eine Laufzeit von $O(n^2)$ zu kommen, muß nun v_4 wieder letzter gültiger Endpunkt für das Segment mit Startpunkt v_3 sein, und v_5 muß im Schnitt

[15] Hierbei muß nicht immer direkt der nächste Punkt Endpunkt sein und auch nicht immer *ganz* bis zum Ende von p gesucht werden, die gezeigte Struktur muß aber erfüllt sein, d. h. in *fast* jedem Schritt liegt der letzte gültige Endpunkt *sehr nahe* am Startpunkt und gleichzeitig geht die Vorausschau dabei jedes Mal bis *fast* zum Ende von p.

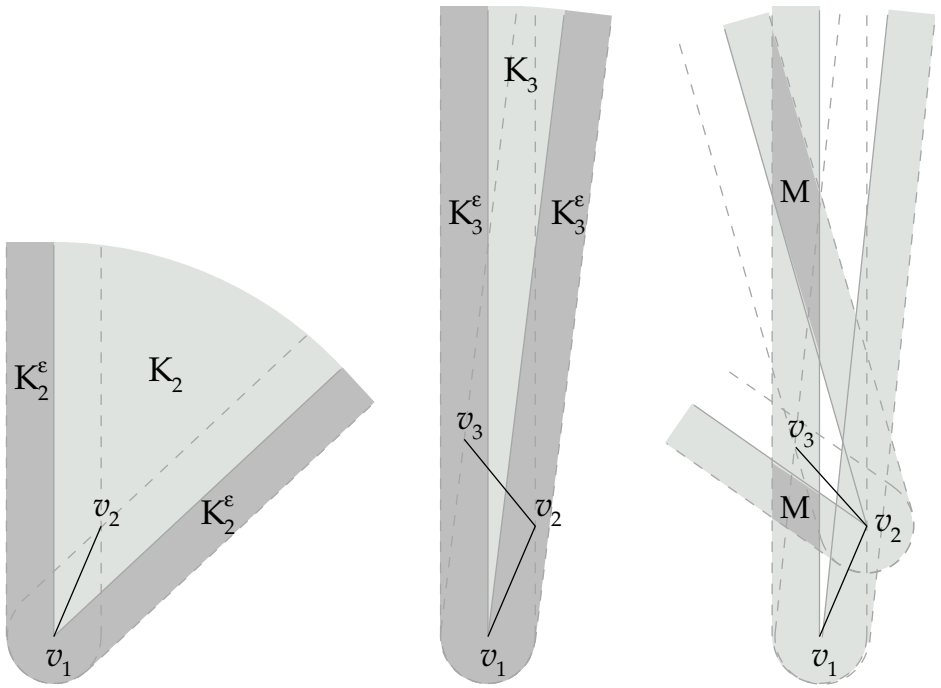


Abbildung 6.13.: Vorausschau bei WεG

der jeweiligen ε-Umgebungen $K^\varepsilon_{4,1}$, $K^\varepsilon_{4,2}$ und $K^\varepsilon_{4,3}$ der Keile[16] $K_{4,1}$, $K_{4,2}$ und $K_{4,3}$, gleiches wiederum auch für die folgenden Punkte. Offensichtlich können nur sehr konstruierte Beispiele dies schaffen.

Nun ist es zwar für $O(n^2)$ nicht nötig, daß wirklich *jeder* Punkt obiges erfüllt, es würde zum Beispiel auch reichen, wenn die Punkte $v_1, v_3, v_5, \ldots$ wie oben beschrieben liegen (auch $O(n + (n-2) + (n-4) + (n-6) + \ldots) = O(n^2)$). Dies erlaubt den dazwischenliegenden Punkten $v_2, v_4, \ldots$ aber keine beliebige Lage. Liegt beispielsweise v_4 außerhalb von $K^\varepsilon_{3,1}$, so unterbricht er die Vorausschau. Somit besitzt auch WεG in fast allen Fällen eine Komplexität von $O(n)$.

Die oben beschriebene Eckensuche verschlechtert natürlich die Laufzeit aller beschriebener Algorithmen. Da nicht der letzte gültige Endpunkt des betrachteten Segmentes gewählt wird, werden weitere Punkte doppelt betrachtet. Solange der Eckenauswahlalgorithmus nicht zu schlecht ist, führt dies allerdings nicht zu einer Erhöhung der Komplexität. Selbst wenn die Eckenwahl dazu führt, daß in jedem Segment jeweils ein Punkt aus der Mitte des Segmentes als Ecke ausgewählt wird (d. h. bei der Generalisierung des ersten Segmentes von v_1 aus wird im Punkt v_k abgebrochen und dann $v_{\lfloor \frac{k}{2} \rfloor}$ als Ecke gewählt), wird im Schnitt jeder Punkt dadurch nur doppelt so oft betrachtet, und $O(2n) = O(n)$. Damit bedeutet die Eckensuche zwar erhöhten Aufwand, aber keine höhere Komplexität.

[16] $K_{4,i}$ ist der Keil K_4 mit Spitze v_i

6.6. Mehr Dimensionen

Der simpleinc$_\varepsilon$-Algorithmus funktioniert natürlich nicht nur im zweidimensionalen Fall. Auch bei mehr Dimensionen kann die Entfernung eines Punktes zu einer Geraden einfach bestimmt werden. Für linWεG und WεG wird die Sache leider komplizierter. Im Dreidimensionalen wird der Keil zu einem Kreiskegel[17] (die oben angegebene Gleichung hat unendlich viele Lösungen, die den Mantel des Kegels beschreiben), was zunächst kein Problem darstellt. Während aber der Schnitt zweier Keile (mit identischer Spitze) wieder einen Keil liefert, ist das Ergebnis des Schnitts zweier Kreiskegel mit gleicher Spitze ein Kegel mit einer Linse als Grundfläche (der Schnitt zweier Kreise liefert eine Linse). Der Schnitt dieses Linsenkegels mit weiteren Kegeln erzeugt eine immer komplexere (konvexe) Grundfläche. Der Aufwand für den Test, ob sich ein Punkt innerhalb dieses Gebildes befindet, steigt mit jedem am Schnitt beteiligten Kegel an.

linWεG besitzt daher im Mehrdimensionalen keine Vorteile mehr gegenüber simpleinc$_\varepsilon$ (da normalerweise bei weitem nicht alle Kegel am Schnitt beteiligt sind, ist zwar linWεG vom Grundsatz her leicht im Vorteil, was aber durch den wesentlich höheren Berechnungsaufwand wieder ausgeglichen wird). WεG kann natürlich auch im Mehrdimensionalen längere Segmente liefern als simpleinc$_\varepsilon$, diesen Nachteil kann simpleinc$_\varepsilon$ jedoch mit der oben beschriebenen beschränkten Vorausschau um m Punkte kompensieren.

Im zweidimensionalen Fall ist linWεG die Generalisierung der Wahl, wenn keine Vorausschau gewünscht ist, im mehrdimensionalen simpleinc$_\varepsilon$. Ist eine Vorausschau gewünscht, ist im Zweidimensionalen WεG im Vorteil, im Mehrdimensionalen fast immer simpleinc$_\varepsilon$ mit Vorausschau um m Punkte. Die oben beschriebene Eckensuche ist dabei im Mehrdimensionalen ohne Einschränkung genauso möglich.

6.7. Minimale WεG-Generalisierung

Der WεG-Algorithmus liefert auf $p = v_1v_2 \ldots v_n$ von v_1 ausgehend das Segment maximaler Länge, das die ε-Bedingung erfüllt. Dies bedeutet allerdings nicht automatisch, daß WεG damit auch die Generalisierung mit den wenigsten Segmenten liefert. In Abbildung 6.14 ist v_3 der letzte gültige Endpunkt, folglich wird das erste Segment zu v_1v_3 generalisiert. Ausgehend von v_3 ist v_5 der letzte gültige Endpunkt, was schließlich zur Generalisierung $v_1v_3v_5v_6$ führt. Wie die Abbildung zeigt, ist auch der *kürzere* Polygonzug $v_1v_2v_6$ eine gültige Generalisierung, die die ε-Bedingung erfüllt. Dadurch, daß das erste Segment kürzer gewählt wird, wird

[17] Ein Kegel mit einem Kreis als Grundfläche, wiewohl man bei den hier betrachteten Gebilden nur schlecht von einer Grundfläche sprechen kann.

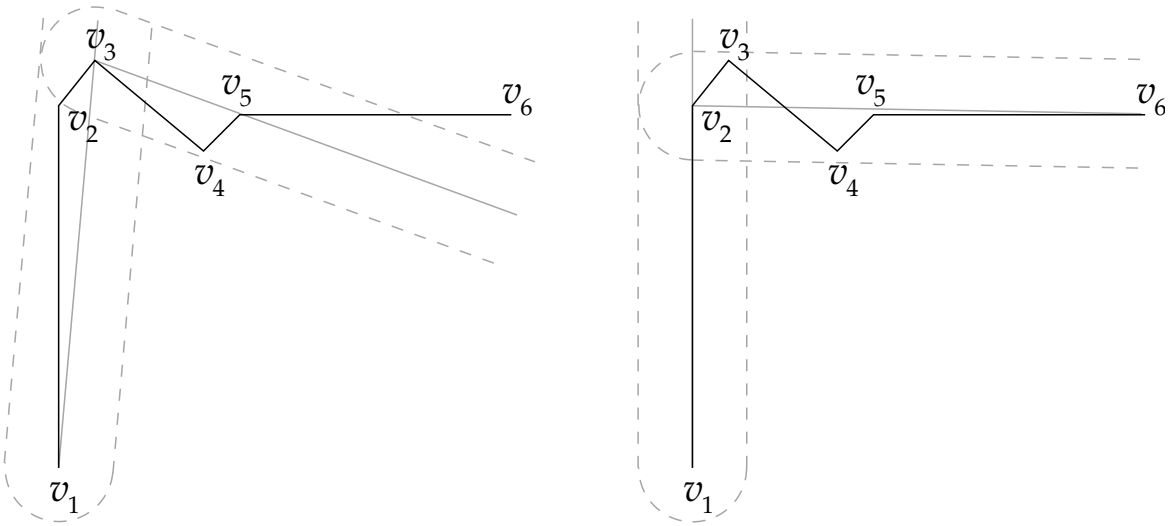


Abbildung 6.14.: WεG ist nicht immer optimal

hier die Generalisierung besser.[18]

Nun kann man grundsätzlich schon alle möglichen Generalisierungen von p durchtesten und die Kürzeste bestimmen, indem man systematisch Punkte aus p streicht und testet, ob die so erzeugten Polygonzüge die ε-Bedingung erfüllen. Wie man sich leicht überlegt, führt dies zu exponentieller Laufzeit und ist somit in der Praxis nicht durchführbar. Man kann jedoch auch geschickter vorgehen.

Die Keilberechnung des WεG-Algorithmus liefert nicht nur den letzten gültigen Endpunkt, sondern alle gültigen Endpunkte bezüglich des Startpunktes v_1. Jeder dieser Punkte ist also ein möglicher Startpunkt für das zweite Segment. Auch im zweiten Segment liefert die Keilberechnung wieder die Liste aller gültigen Endpunkte, die wiederum Startpunkte für das nächste Segment bilden. Die Verbesserung, die so erreicht wird, besteht darin, daß nur die Polygonzüge getestet werden, die von vornherein die ε-Bedingung erfüllen.

Aber es sind noch weitere Optimierungen möglich. Beim gezeigten Vorgehen hat vom Startpunkt v_1 ausgehend das erste Segment k mögliche Endpunkte, diese bilden somit wieder Startpunkte, die wiederum $k', k'', \ldots$, mögliche Endpunkte haben usw., wodurch der Suchraum weiterhin sehr groß bleibt. Nun haben häufig zwei Startpunkte v_a und v_b mehrere mögliche Endpunkte gemeinsam. Ein möglicher Endpunkt v_h ist dann in beiden Varianten wieder Startpunkt. Selbstverständlich müssen nun aber die zu v_h gehörigen Endpunkte nicht mehrmals berechnet werden, sondern können nach der ersten Berechnung gespeichert werden und stehen dann zur Verfügung.

Sei l_i die Anzahl möglicher und l'_i die Anzahl gültiger Endpunkte für Start-

[18] sofern man eine kürzere Generalisierung auch als besser empfindet.

punkt v_i (l_i ist also Anzahl der Punkte, die durchsucht werden müssen, das sind alle Punkte v_k, die noch in $K^{\varepsilon}_{k-1,i}$ liegen, l'_i die Anzahl der Punkte, die in $K_{k-1,i}$ liegen) und $\bar{l} = \frac{1}{n}\sum_{i=1}^{n} l_i$, $\bar{l'} = \frac{1}{n}\sum_{i=1}^{n} l'_i$, dann können in $O(\bar{l}n)$ die möglichen Endpunkte zu allen Punkten von p berechnet werden: Die Berechnung der Endpunkte von v_i ist in der Segmentlänge linear, kostet also l_i Schritte. Hierbei ist $\bar{l}$ wesentlich kleiner als die mittlere Segmentlänge einer Generalisierung q durch WεG, da WεG von vornherein möglichst lange Segmente selektiert, wogegen in $\bar{l}$ alle in p möglichen Segmente eingehen.

Die Berechnung liefert einen gerichteten Graphen, in dem alle Pfade von v_1 nach v_n gültige ε-Generalisierungen von p darstellen und der keine weiteren Kanten enthält. Insbesondere enthält der Graph weder Schleifen noch Sackgassen (d. h. alle Pfade enden in v_n). v_1 besitzt Kanten zu l'_1 Nachfolgern, v_2 zu l'_2 Nachfolgern usw., insgesamt besitzt der Graph also n Knoten und $\bar{l'}n$ Kanten. Eine Breitensuche (Algorithmus 8) findet den kürzesten Pfad zwischen zwei Punkten[19] (v_1 und v_n) in $O(n + \bar{l'}n) = O(\bar{l'}n)$: Im ersten Schritt werden alle möglichen Endpunkte zu v_1 bestimmt. Diese Punkte besitzen v_1 als Vorgänger und damit einen Abstand von 1 zu v_1. Zu den so erhaltenen Punkten $M_1 = \{v_{1_1}, v_{1_2}, \ldots\}$ werden nun wiederum die jeweiligen Nachfolger bestimmt, die demzufolge den Abstand 2 zu v_1 besitzen. Punkte, die schon im ersten Schritt gefunden wurden (Abstand 1 zu v_1), werden nicht weiter betrachtet, es bleibt die Menge $M_2 = \{v_{2_1}, v_{2_2}, \ldots\}$ der Punkte mit Abstand 2 zu v_1. Ausgehend von M_2 wird nun M_3 bestimmt usw., bis $v_n \in M_m$. Damit ist m der minimale Abstand von v_n zu v_1 und somit auch die minimale Anzahl der Segmente einer ε-Generalisierung von p.

Bei diesem Durchlauf werden zu jedem betrachteten Punkt v_k alle direkten Vorgänger gespeichert, von denen aus v_k mit minimalem Abstand zu v_1 erreichbar ist (die Teil eines minimalen Pfades von v_1 nach v_k sind). Wählt man nun ausgehend vom Punkt v_n einen beliebigen Vorgänger $v_{n'_1}$, zu diesem wieder einen beliebigen Vorgänger $v_{n'_2}$ usw., so erreicht man in einer minimalen Anzahl von Schritten den Punkt v_1 und enthält einen minimalen Pfad.

Algorithmus 8 berechnet einen Graphen über p, der genau alle minimalen Pfade von v_1 nach v_n enthält, die eine ε-Generalisierung von p darstellen. Hierbei bestimmt die Funktion step in Breitensuche die möglichen Kanten, bis v_n erreicht ist, und die Funktion tighten durchläuft die so entstandene Struktur von v_n aus rückwärts, wodurch alle Kanten entfernt werden, die nicht Teil eines minimalen Pfades sind.

Damit ist die Berechnung einer minimalen Generalisierung in $O(\bar{l}n)$ möglich (da $\bar{l} \geq \bar{l'}$), also linear in der Segmentlänge und linear in der Polygonlänge. Betrachtet man einen doppelt so langen Polygonzug, so verdoppelt sich n, $\bar{l}$ bleibt unverändert.[20] Betrachtet man hingegen einen Polygonzug mit doppelter Segmentlänge

[19] [Ste01][S. 66, Satz 2.27]

[20] Man kann sich überlegen, daß $\bar{l}$ ein klein wenig größer wird, da die Punkte am Ende von p selbst-

Algorithmus 8: Shortest Path Generalization

```
1:  type Punkt = { N_0 Abstand;  Punkte Vorgänger;  Punkte Nachfolger }

2:  function step = (Punkte M, N_0 m, Polygonzug p, Distanz ε) → Polygonzug :
3:  begin
4:    if (v_n ∈ M) return p;                    ▷ Wir sind fertig
5:    M' ← ∅;
6:    for all v_i ∈ M do                        ▷ Alle Punkte durchgehen
7:      for all v_k ∈ Endpunkte(v_i, p, ε) do   ▷ Menge aller gültigen Endpunkte zu v_i
8:        if (v_k.Abstand = ∞) then             ▷ Endpunkt v_k wurde noch nicht betrachtet
9:          v_k.Abstand ← m;                    ▷ besitzt also Abstand m zu v_1
10:         M' ← M' ∪ {v_k};                    ▷ Im nächsten Schritt bearbeiten
11:       end if
12:       if (v_k.Abstand = m) then             ▷ v_k wurde in diesem Schritt gefunden
13:         v_k.Vorgänger ← v_k.Vorgänger ∪ {v_i};   ▷ v_i zu den Vorgängern hinzufügen
14:       end if
15:     end for
16:   end for
17:   return step(M', m + 1, p);                ▷ Nächster Schritt
18: end

19: function tighten = (Polygonzug p, Punkte M, Punkte N) → Punkte :
20: begin
21:   if (M = ∅) return N;                      ▷ Wir sind fertig
22:   for all v_i ∈ M do                        ▷ Alle Punkte durchgehen
23:     for all v_k ∈ v_i.Vorgänger do          ▷ Kanten erzeugen
24:       v_k.Nachfolger ← v_k.Nachfolger ∪ {v_i};
25:     end for
26:   end for
27:   return tighten(p, v_i.Vorgänger, M ∪ N);
28: end

29: function generateGraph = (Polygonzug p = v_1 ... v_n, Distanz ε) → Punkte :
30: begin
31:   for all v_i ∈ {v_1, ... v_n} do           ▷ Initialisierung
32:     v_i.Abstand ← ∞;  v_i.Vorgänger ← [ ];  v_i.Nachfolger ← [ ];
33:   end for
34:   return tighten(step({v_1}, 1, p), {v_n}, ∅);
35: end
```

(z. B. eine weniger stark gebogene Bewegungsspur), so verdoppelt sich $\bar{l}$, n bleibt konstant, was ebenfalls einen linearen Anstieg darstellt.

Lediglich eine höhere Genauigkeit läßt die Kosten quadratisch steigen. Wird eine Bewegungsspur mit doppelt so vielen Punkten beschrieben, so verdoppelt das sowohl die Anzahl der Punkte in p als auch die Anzahl der Punkte pro Segment, was zu quadratischer Komplexität führt.

Der so entstandene Graph liefert nicht nur eine, sondern ein ganzes Bündel minimaler Generalisierungen. Die Knoten des Graphen bilden hierbei die Eckpunkte der minimalen Generalisierungen. Alle diese Pfade erfüllen die ε-Bedingung und sind minimal, das bedeutet aber nicht, daß sie die Form der Bewegung gleich gut beschreiben. Die Generalisierungen in Abbildung 5.2, Abbildung 6.8 und Abbildung 6.9 arbeiten alle auf dem gleichen Polygonzug und besitzen alle 6 Eckpunkte, trotzdem beschreiben sie die Form der Originalspur nicht gleich gut.[21]

verständlich weniger Nachfolger haben und diese Punkte bei einem längeren Polygonzug weniger ins Gewicht fallen. Diese Verzerrung ist allerdings vernachlässigbar klein.

[21] Das hier beschriebene Verfahren ist weder das erste noch das einzige, das eine minimale ε-Generalisierung liefert, so wird z. B. in [dBvKS95] ein Ansatz von Chan und Chin beschrieben, der auf sehr ähnliche Weise eine minimale Segmentierung generiert (im Gegensatz zu WεG legen sie um jeden Punkt einen Kreis mit Radius ε, der dann mit allen möglichen Halbgeraden geschnitten wird, wodurch ebenfalls Keile mit möglichen Lagen der betrachteten Halbgeraden bestimmt werden können. Die Berechnung dieser Keile ist letztlich zur Berechnung bei WεG äquivalent).

7. Rolland-Generalisierung

Als inkrementelle Korridorgeneralisierung bot sich WεG auch an, um die Bewegungsspur eines autonom fahrenden Rollstuhls (Rolland) bei seiner Fahrt durch die Korridore eines Bürokomplexes zu generalisieren. Für Korridore der Breite b wählt man $\varepsilon = b/2$. Fährt Rolland nun einen geraden Korridor entlang, so wird seine Bewegung zu einem einzigen Segment generalisiert, biegt er ab, erkennt die Generalisierung dies schnell, ein neues Segment beginnt.

Im in Abbildung 7.1 gezeigten Beispiel funktioniert die Sache auch wunderbar. Beginnt die beobachtete Bewegung jedoch nicht in der Mitte des Ganges, sondern an dessen Rand,[1] so wird nicht mehr der ganze Korridor zu einem Segment generalisiert, wie Abbildung 7.2 zeigt. Der Algorithmus gibt die Form der Bewegung dabei gut wieder, die Kurven werden erkannt, nur eben für die Aufgabenstellung zu gut.

Wählt man $\varepsilon = b$ (Gangbreite), so erfaßt die Generalisierung zwar den gesamten Gang in einem Segment, allerdings wird nun wesentlich später erkannt, wenn Rolland in einen anderen Gang abbiegt. Möchte man die Generalisierung der Bewegung on-the-fly zur Steuerung des Rolland nutzen, so ist dies sehr unbefriedigend, hier ist eine möglichst frühzeitige Erkennung wünschenswert.

Dies liegt daran, daß bei WεG die Position des Startpunktes wesentlich ist. Es existiert ja offensichtlich ein ε-Korridor, der den Teil der Bewegung umfaßt, die im Korridor stattfindet, nur muß dafür der Startpunkt der Bewegung in der Gangmitte liegen. Gesucht ist also eine Generalisierung, die von der Lage des Startpunktes unabhängig ist.

7.1. Der Rolland-Algorithmus

Betrachtet man die Bewegung eines Fahrzeugs ausgehend von einem Startpunkt v_1 entlang eines $b = 2\varepsilon$ breiten Korridors,[2] so gilt für eine zu den Korridorwänden parallele Gerade g durch v_1 zum einen, daß die Bewegungsspur p in jedem

[1] Dies tritt häufig auf, da zum einen selten in der Mitte des Ganges geparkt wird und zum anderen Kurven häufig geschnitten werden, was den gleichen Effekt hat.

[2] Der selbstfahrende Rollstuhl Rolland ist mit Ultraschallsensoren ausgestattet, mit denen unter anderem ständig die Breite des Korridors gemessen wird, durch den Rolland gerade fährt. Die Korridorbreite wird in diesem Fall nicht für die gesamte Bewegung konstant gehalten, sondern

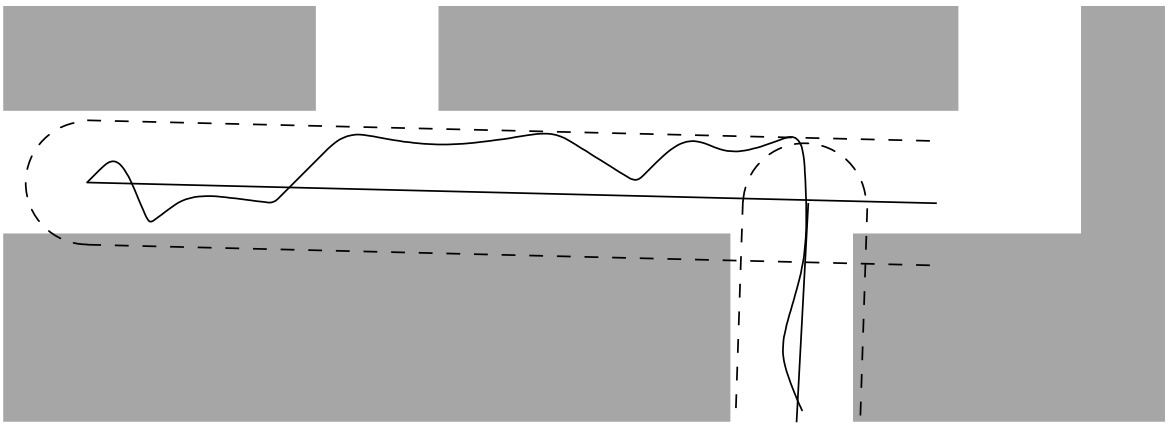


Abbildung 7.1.: Gangerkennung mittels WεG

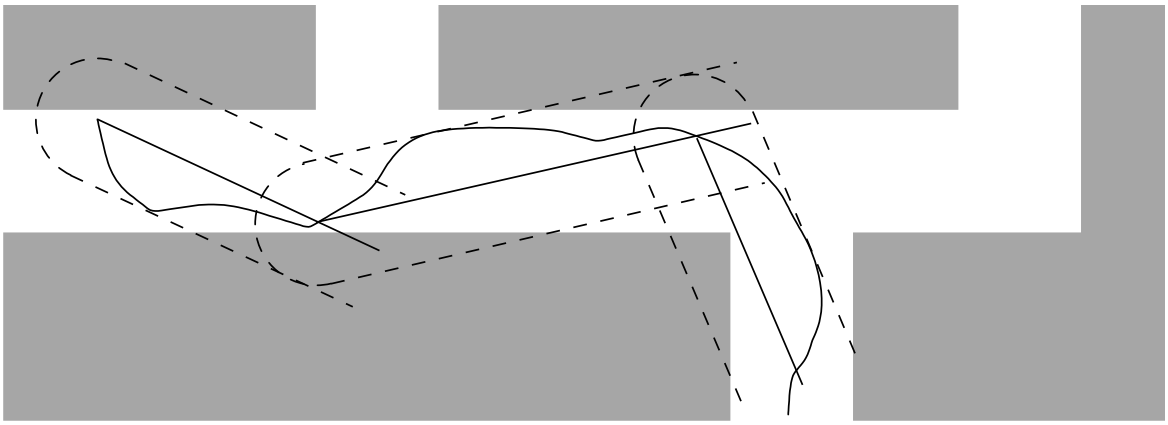


Abbildung 7.2.: Probleme der Gangerkennung mittels WεG

Punkt einen Abstand weniger 2ε zu g besitzt. Darüberhinaus gilt für den maximalen linksseitigen (d_l) und rechtsseitigen (d_r) Abstand von p zu g: $d_l + d_r \leq 2\varepsilon$, d. h. wenn Punkte aus p auf der einen Seite weiter als ε von g entfernt liegen, so dürfen sie auf der anderen Seite entsprechend weniger von g abweichen.

Algorithmus 9 (rolland) zeigt eine modifizierte Version von simpleinc$_\varepsilon$, die auf dieser Bedingung basiert. Ausgehend vom Startpunkt v_1 wird für jeden Punkt v_k getestet, ob alle dazwischenliegenden Punkte in einem Korridor der Breite b entlang der Geraden $g = \overline{v_1 v_k}$ liegen. Im Gegensatz zu WεG oder Douglas/Peucker muß g hier jedoch nicht in der Mitte des Korridors liegen. Die Unterschiede zu simpleinc$_\varepsilon$ liegen also lediglich in der Funktion in?, beide Algorithmen sind strukturell gleich, simplekorr besitzt daher ebenfalls eine Komplexität von $O(n^2)$ in der Segmentlänge.

ständig angepaßt [Röf99].

Algorithmus 9: Generalisierung mittels Korridorbreite b

```
1:  function in? = (Gerade g, Polygonzug p = v_1 ... v_k, Breite b) → Boolean :
2:  begin
3:    d_l ← 0;   d_r ← 0;
4:    for all v_i ∈ v_2, ... v_{k-1} do      ▷ Wir testen alle Punkte
5:      d ← dist(g, v_k);                   ▷ vorzeichenbehaftete Distanz zur Geraden
6:      if (d < d_l) then                   ▷ weiter links
7:        d_l ← d;
8:      else if (d > d_r) then              ▷ weiter rechts
9:        d_r ← d;
10:     end if
11:     if (d_r − d_l > b) then             ▷ Korridor paßt nicht
12:       return false;
13:     end if
14:   end for
15:   return true;                          ▷ Alles im Korridor
16: end

17: function rest = (Polygonzug p = v_1 ... v_n, Breite b) → Polygonzug :
18: begin
19:   k ← 3;
20:   while (k ≤ n) ∧ in?(v_1 v_k ,  v_1 ... v_k ,  b) do   ▷ Alle Punkte im Korridor
21:     k++;
22:   end while
23:   return v_{k-1} v_k ... v_n            ▷ v_{k-1} ist der letzte Punkt im Segment
24: end

25: function simplekorr = (Polygonzug p = v_1 ... v_n, Breite b) → Polygonzug :
26: begin
27:   if n ≤ 2 then                         ▷ Erst ab 3 Punkten wird generalisiert
28:     return p
29:   else
30:     return v_1 ∘ simplekorr(rest(p, ε))   ▷ rekursiv die Segmente ausgeben
31:   end if
32: end
```

Algorithmus 10: Rolland-Generalisierung

```
1:  function rolland = (Polygonzug p = v_1 ... v_n, Breite b) → Polygonzug :
2:  begin
3:    if n ≤ 2 then                          ▷ Erst ab 3 Punkten wird generalisiert
4:      return p;
5:    end if
6:    a ← 1;   i ← 2;                        ▷ Initialisierung
7:    g = overline(v_a v_i);
8:    d_l ← 0;   d_r ← 0;
9:    repeat
10:     i++;
11:     d ← dist(g, v_i);                    ▷ vorzeichenbehaftete Distanz zur Geraden
12:     if (d < d_l) then                    ▷ weiter links
13:       d_l ← d;
14:     else if (d > d_r) then               ▷ weiter rechts
15:       d_r ← d;
16:     end if
17:     if (d_r − d_l > b) then              ▷ Korridor paßt nicht
18:       g ← overline(v_a v_i);             ▷ nachführen
19:       d_l ← mindist(g, v_a ... v_i);     ▷ linker Abstand
20:       d_r ← maxdist(g, v_a ... v_i);     ▷ rechter Abstand
21:     end if
22:   until (i = n) ∨ (d_r − d_l > b);
23:   if (d_r − d_l > b) then                ▷ kein Gang mehr möglich
24:     return v_1 ∘ rolland(v_{i−1} ... v_n, b);   ▷ neues Segment
25:   else
26:     return v_1 v_n;                      ▷ fertig
27:   end if
28: end
```

Thomas Röfer[3] schlägt einen etwas modifizierten Algorithmus vor[4] (Algorithmus 10), bei dem g nicht in jedem Schritt neu bestimmt wird und somit nicht in jedem Schritt der Abstand aller Punkte zu g erneut berechnet werden muß: $g = \overline{v_1 v_2}$ ist die erste Referenzgerade, die die Richtung eines Korridors der Breite b bestimmt. Daher wird nun für alle folgenden Punkte getestet, wie weit sie auf der rechten bzw. linken Seite abweichen. Ist die Summe der maximalen rechtsseitigen und linksseitigen Abweichung größer b, so erfüllt ein durch g bestimmter Korridor die Anforderung nicht mehr, alle Punkte der Bewegungsspur zu enthalten. Daher wird mit $g' = \overline{v_1 v_k}$ (v_k der letzte betrachtete Punkt) die Referenzgerade „nachgeführt“, wobei nun der maximale linke und rechte Abstand aller Punkte zu g' neu bestimmt werden muß. Ist die Summe dieser Abstände immer noch größer b, so existiert keine gültige Referenzgerade, und es wird abgebrochen. Nun wird getestet, ob $v_1 v_{k-1}$ eine gültige Generalisierung ist, wenn nicht, wird $v_1 v_{k-2}$ getestet usw.

7.2. Schleifenerkennung

Wie schon in Kapitel 4.4 und Kapitel 5.3 beschrieben, können in Korridor-Generalisierungen interne und externe Schleifen auftreten (siehe auch Abbildung 4.3 und 5.3).

Die Rolland-Generalisierung erkennt in der vorgestellten Form weder interne noch externe Schleifen. Eine interne Schleife entsteht beispielsweise, wenn Rolland[5] 10 m einen Korridor entlang, im gleichen Korridor 5 m zurück und danach wieder 8 m vorwärts fährt. Wird diese Schleife nicht erkannt, so wird zwar die *Form* der Bewegungsspur nicht richtig erkannt, die Erkennung des umgebenden Korridors hingegen funktioniert problemlos. Anders ist die Situation bei externen Schleifen, wenn also die Bewegungsspur über das generalisierte Segment herausragt: Rolland fährt 10 m den Gang entlang, 5 m zurück und biegt dann nach links in einen anderen Gang ab (4 m lang). Dies wird ohne Schleifenerkennung zu 5 m geradeaus gefolgt von 4 m rechts generalisiert.

Da der Rolland-Algorithmus speziell zur Verarbeitung von Bewegungen in Korridorumgebungen entwickelt wurde, stört kaum, daß interne Schleifen nicht erkannt werden, es interessiert ja weniger, wie Rolland genau gefahren ist, sondern vielmehr, welche Korridore er entlang gefahren ist. Werden externe Schleifen nicht erkannt, so geht hingegen Information verloren. Im Beispiel oben steckt in der generalisierten Bewegungsspur nur noch die Information, daß vom Startpunkt der Bewegung aus der Korridor mindestens 5 m lang ist und daß in dieser Entfernung

[3] Mitarbeiter im Bremer Rolland-Projekt

[4] Der Algorithmus wird in [Röf99] erwähnt, allerdings nicht ausführlich beschrieben. Die hier wiedergegebene Form habe ich aus direkter Rücksprache mit Thomas Röfer.

[5] oder welches Gefährt auch immer

ein weiterer Gang abgeht, der mindestens 4 m lang ist. Daß der erste Gang noch mindestens 5 weitere Meter geradeaus geht, ist hingegen verloren gegangen.

Für viele Anwendungen kann man auf diese Schleifenerkennung also sicher völlig verzichten, andernfalls gibt es mehrere Möglichkeiten des Umgangs mit Schleifen.

Eine einfache Änderung des Algorithmus besteht darin, nicht mit der Geraden $g = \overline{v_1 v_k}$ sondern mit der Halbgeraden $h = [v_1 v_k$ zu arbeiten. Genau wie bei Jungert/Hernández tritt auch hier das Problem auf, daß ein „hinter" v_1 liegender Punkt v_i irgendwie in das rechts/links Schema eingeordnet werden muß. Die einfache Lösung besteht darin, zum einen den Abstand zu g zu bestimmen, der dann entsprechend rechts bzw. links eingeht, und zum anderen zusätzlich den Abstand des Fußpunktes l_i vom Punkt v_i auf g zum Punkt v_1 mit einem weiteren Abstand b' zu vergleichen ($b' \geq |v_1 l_i|$).

Dies liefert ein rechteckiges Korridorende im Gegensatz zu den Halbkreisen bei Douglas/Peucker oder WεG. Es wäre hier falsch, den euklidischen Abstand von $|v_1 v_i|$ zu bestimmen, der bei den anderen Algorithmen diese Halbkreise liefert, da v_1 hier nicht unbedingt in der Mitte des Korridors liegt. Zum anderen haben zumindest in der westlichen Welt (selbst in der modernen Architektur) die allermeisten Korridore in Bürokomplexen die Angewohnheit, rechteckige und nicht abgerundete Formen zu besitzen.

Je nachdem, wie große Schleifen erkannt werden sollen, wählt man $b' = \frac{b}{2}$ bis $b' = b$. Es gibt aber auch gute Gründe, b' unabhängig von b zu wählen. Sollen beispielsweise Rangierbewegungen weggeneralisiert und nicht als Schleifen erkannt werden, so wird man b' entsprechend groß wählen. Die Wahl von b' hängt dann vom Fahrzeug und nicht von der Gangbreite ab.

Die Erkennung von Rückläufigkeiten am Startpunkt v_1 ist natürlich nicht hinreichend, eine Erweiterung einfach möglich. Wird zu jedem Punkt v_i der Fußpunkt l_i seines Lotes auf g bestimmt, so lassen sich Rückläufigkeiten leicht feststellen, indem der Startpunkt der Halbgeraden ständig verschoben wird: Liegt $l_i \in h$, so wird h um das Stück $[v_1 l_i]$ verkürzt. In jedem Schritt wird also von h vorne ein Stück abgeschnitten. Damit wird erreicht, daß eine Bewegung entlang des Korridors nur in Richtung von v_1 weg verlaufen kann, jedes Umdrehen wird sofort erkannt.

Damit werden sowohl innere als auch äußere Schleifen erfaßt. Möchte man auf die Erkennung innerer Schleifen bewußt verzichten, ist dies ebenso einfach möglich. Der Algorithmus generalisiert (wie in Alg. 10 beschrieben) ein Segment und liefert als letzte gültige Gerade $g = \overline{v_1 v_k}$. Nun wird zu jedem Punkt v_i der Fußpunkt l_i auf g bestimmt. Die kürzeste Strecke $s \subset g$, die alle l_i enthält, ist dann Generalisierung des Segments. s wird bestimmt, indem ausgehend von $t_0 = [v_1 v_k]$ für jeden Fußpunkt l_i die neue Strecke $t_i = t_{i-1} \cup [v_1 l_i]$ bestimmt wird. (Für $l_i \in t_{i-1}$ gilt somit $t_i = t_{i-1}$, andernfalls wird l_i neuer Start- oder Endpunkt von t_i, je nachdem, ob l_i auf g vor oder hinter t_{i-1} liegt). Mit $s = t_k$ gilt: $\forall l_i : l_i \in s$, und zudem: $[v_1 v_k] \subset s$. s kann also auf beiden Seiten über $[v_1 v_k]$ „hinausragen", womit $s \setminus [v_1 v_k]$ genau die

externen Schleifen repräsentiert.

7.3. Eckensuche

Wie schon für WεG beschrieben, gilt auch für die Rolland-Generalisierung, daß der letzte Eckpunkt in einem Korridor nicht unbedingt die markanteste Ecke darstellt. Die in Kapitel 6.4 beschriebenen Verfahren zur Eckensuche lassen sich auch hier anwenden. Hierbei muß für jeden so ausgewählten Punkt v_k überprüft werden, ob $[v_1 v_k]$ einen gültigen Korridor für die Punkte $v_2, v_3, \ldots v_{k-1}$ beschreibt.

Ist v_k kein gültiger Eckpunkt, wird die zweitbeste Ecke gewählt, usw. Für jede Überprüfung müssen $k-2$ Punkte getestet werden, was im worst case eine Komplexität von $O(k^2)$ bedeutet. Allerdings ist die Wahrscheinlichkeit, im ersten oder zweiten Versuch eine gültige Ecke zu erwischen, relativ groß.

7.4. Komplexität

Der Rolland-Algorithmus ist eine historiefreie inkrementelle Generalisierung und damit über die Länge der Spur linear. Innerhalb eines Segments kann die Komplexität quadratisch werden, wenn der Korridor immer neu berechnet werden muß, im Mittel bleibt sie aber linear.

7.5. Korridor versus ε-Kriterium

Mit $q = \text{rolland}(p)$ gilt zunächst, daß alle Punkte von p zu q einen Abstand kleinergleich b besitzen. q liefert aber noch mehr Information. Liegt beispielsweise der Punkt $v_i \in p$ zum zugehörigen Teilstück $t_k = [w_k w_{k+1}] \subset q$ auf der linken Seite $\frac{2}{3}b$ entfernt, so kann ein Punkt v_j auf der rechten Seite maximal $\frac{1}{3}b$ entfernt liegen. Diese Information ist nutzbar. Zum einen kann man zu jedem Teilstück von g zusätzlich den maximalen rechten und linken Abstand mit abspeichern, wodurch man einen attributierten Polygonzug bekommt. Bei Messungen von Bewegungen in Gängen (z. B. in Bürogebäuden) liefern die Sensoren häufig noch Informationen über die Breite der durchfahrenen Gänge, die zusätzlich mit eingehen. Im Bremer Rolland-Projekt hat sich gezeigt, daß die mittels Ultraschall gemessene ungefähre Gangbreite sehr gut genutzt werden kann, um den Parameter b zu justieren. b ist dann nicht über die ganze Generalisierung konstant, sondern wird für jedes Teilstück neu bestimmt. Jedes Teilstück $t_k \subset q$ wird also mit der aktuellen Gangbreite b_k sowie der maximalen linken und rechten Distanz $d_{k,l}$ und $d_{k,r}$ attributiert. Gilt $d_{k,l} + d_{k,r} = b_k$, so ist die Lage des Korridors für das Teilstück t_k eindeutig bestimmt, meist gilt jedoch $d_{k,l} + d_{k,r} < b_k$, was dem Korridor etwas Spiel läßt.

Verzichtet man auf eine eckentreue Generalisierung,[6] so läßt sich durch eine kleine Modifikation ein Algorithmus $\text{rolland}_\varepsilon$ angeben, der eine ε-Generalisierung darstellt. Hierzu wird lediglich die Referenzstrecke für jeden Teilkorridor zentriert. Sei $t_k = [w_k w_{k+1}]$ ein Teilstück von q mit $d_{k,l}$ maximalem linken und $d_{k,r}$ maximalem rechten Abstand,[7] so existiert dazu genau eine parallele Gerade g'_k mit $|d'_{k,l}| = |d'_{k,r}|$. Hierzu muß $g_k = \overline{w_k w_{k+1}}$ lediglich um $\frac{d_{k,l}+d_{k,r}}{2}$ verschoben werden. Bezüglich dieses Teilstücks liegen also alle Punkte im ε-Korridor um g'_k mit $\varepsilon = b/2$. Um den Start- und Endpunkt von t'_k zu bestimmen, wird g'_k mit g'_{k-1} und g'_{k+1} geschnitten, die Schnittpunkte bilden die Ecken. Hierbei kann es allerdings zu „Übersteuern“ kommen, worauf ich in Kapitel 8.3.2 genauer eingehe. Als Startpunkt der Generalisierung t'_1 des ersten Segments und der Endpunkt der Generalisierung t'_m des letzten Segments dient dabei der Fußpunkt des Lotes von w_1 bzw. w_m auf g'_1 bzw. g'_m.

[6] d. h. die Eckpunkte von q sind auch Eckpunkte von p

[7] beide Abstände vorzeichenbehaftet, d. h. $d_{k,l} \leq 0$ und $d_{k,r} \geq 0$

8. Convex Hull Generalisierung

Die große Schwäche des WεG-Algorithmus liegt in seiner Abhängigkeit von Start- und Endpunkt jedes Segmentes. Der im letzten Kapitel vorgestellte rolland-Algorithmus ist – basierend auf der Grundidee von simpleinc$_\varepsilon$– in direkter Reaktion auf diese Schwäche entstanden und optimiert die Generalisierung für Gangszenarien, z. B. in Bürogebäuden.

Der in diesem Kapitel vorgestellte Algorithmus ist wie WεG und rolland eine inkrementelle Korridor-Generalisierung.

Existiert um einen Polygonzug $\hat{p}$ einen Korridor K der Breite 2ε, der $\hat{p}$ vollständig enthält, so gibt es (mindestens) eine Strecke s, die eine ε-Generalisierung von $\hat{p}$ darstellt. Der hier vorgestellte Algorithmus basiert nun auf der einfachen Grundidee, einen Polygonzug p sequentiell in Segmente $\hat{p}_1$, $\hat{p}_2$, ... maximaler Länge zu zerlegen, daß für jedes Teilstück $\hat{p}_i$ ein Korridor K_i der Breite 2ε existiert, der $\hat{p}_i$ vollständig enthält, und diese Segmente anschließend zusammenzusetzen:

Der Polygonzug $p = v_1 v_2 \dots v_n$ wird beginnend bei $v_s = v_1$ der Reihe nach abgearbeitet.

1. Solange es für den Eckpunkt v_i des Polygonzuges einen Korridor K der Breite 2ε gibt, der das Teilstück $\hat{p} = v_s \dots v_i$ vollständig enthält, wähle den nächsten Eckpunkt (also $i \leftarrow i + 1$).

2. Gilt dies nicht, so bildet der Polygonzug $\hat{p}_k = v_s \dots v_{i-1}$ ein Segment der Generalisierung. Zu diesem existiert eine Strecke s_k, die Generalisierung von $\hat{p}$ ist. Nun setze $v_s \leftarrow v_{i-1}$ und fahre wieder bei 1 fort.[1]

3. Die einzelnen Strecken s_k werden nun zu einem Gesamtstreckenzug q verbunden, der die Generalisierung von p darstellt. Hierbei kann es nötig sein, Zwischenstücke zwischen die einzelnen s_k einzufügen.

Dieses Kapitel liefert eine effiziente Implementierung dieser Idee, wobei an mehreren Stellen zwischen unterschiedlichen Ausprägungen gewählt werden kann, sowie eine modifizierte Version, bei der der Zusammenbau der einzelnen Segmente schon bei der Segmentbestimmung berücksichtigt wird.

[1] Alternativ wäre es auch möglich, $v_s \leftarrow v_i$ zu wählen.

ab	Gerade durch die Punkte a und b
$[ab]$	Strecke mit den Endpunkten a und b
$ab]$	Halbgerade mit Endpunkt b
$\triangle abc$	Dreieck mit den Eckpunkten a, b, c
$\square abcd$	Rechteck mit den Eckpunkten a, b, c, d
$[a_1 a_2 \ldots a_m a_1]$	Polygon mit Eckpunkten a_i
$\angle([ab], [ac])$	Winkel zwischen den Strecken $[ab]$ und $[ac]$
∂H	Rand der Menge H
$\overset{\circ}{H} = (H \setminus \partial H)$	Inneres von H
$\overline{H} = (H \cup \partial H)$	Abschluß von H.
$\lfloor x \rfloor$	Gaußklammer: ganzzahliger Anteil von x

Tabelle 8.1.: Bezeichner

8.1. Grundlagen

In diesem Kapitel wird das mathematische Rüstzeug gesammelt, das zur effizienten Implementierung des Algorithmus nötig ist. Alle folgenden Definitionen und Sätze beziehen sich auf den $\mathcal{R}^2$ (auch wenn ein Großteil problemlos auf den $\mathcal{R}^n$ erweiterbar ist). Die aufgestellten Beziehungen sollten größtenteils bekannt oder intuitiv klar sein, einen Überblick über Polygone und konvexe Mengen unter dem Blickwinkel der computerunterstützten Geometrie bietet z. B. [AS93].

Definition 10 *Die Breite b_R eines Rechtecks $R = \square abcd$ ist die Länge der kürzeren Seite, also:*

$$b_R \leftarrow \min\{\big|[ab]\big|, \big|[bc]\big|\}.$$

Definition 11 *Die Breite b_M einer Menge M ist die Breite des schmalsten Rechtecks, das M enthält, also:*

$$b_M \leftarrow \min\{b_{R_k} \mid M \subset R_k\}.$$

Definition 12 *Eine Menge H heißt konvex, wenn sie die Verbindungsstrecken aller ihrer Punkte enthält, also:*

$$\forall a, b \in H : [ab] \in H.$$

Satz 2 *Der Durchschnitt beliebig vieler konvexer Mengen ist konvex.*

Beweis. Seien H_i konvexe Mengen und $H = \bigcap H_i$. Dann gilt:

$$(a, b \in H) \;\Rightarrow\; (\forall H_i : a, b \in H_i) \;\Rightarrow\; (\forall H_i : [ab] \in H_i) \;\Rightarrow\; ([ab] \in H).$$

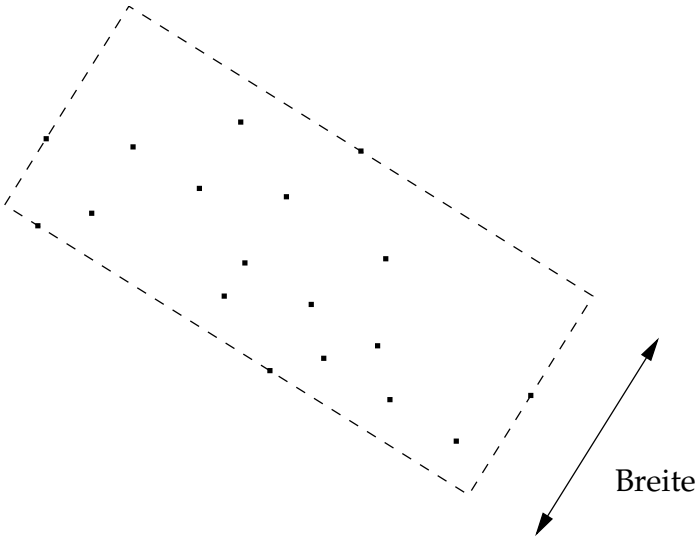


Abbildung 8.1.: Breite einer Menge M.

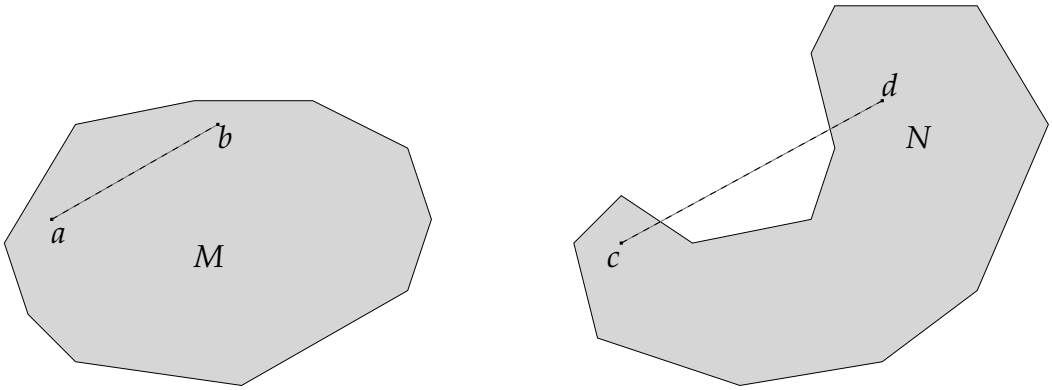


Abbildung 8.2.: Konvexe (M) und nicht konvexe (N) Mengen.

Somit ist nach Definition 12 H konvex. □

Mit Satz 2 kann man nun definieren:

Definition 13 *Die konvexe Hülle H einer Menge M ist der Durchschnitt aller konvexen Mengen H_i, die M enthalten, also:*

$$H = \bigcap H_i : \ H_i \text{ konvex } \wedge \ M \subset H_i.$$

Satz 3 *Jeder Polygonzug $p = v_1 \dots v_n$ ist in der konvexen Hülle H seiner Eckpunkte enthalten:*

$$(\forall v_i \in H) \Rightarrow (p \subset H).$$

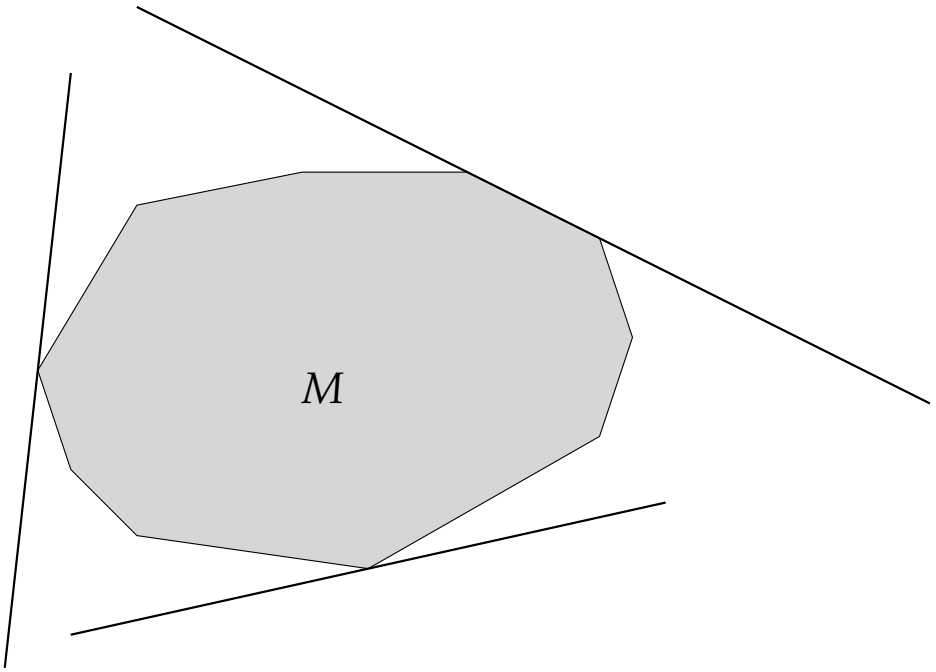


Abbildung 8.3.: Einige Stützgeraden der Menge M

Beweis. Folgt direkt aus Definition 12. □

Satz 4 *Die Breite b_M einer Menge M ist identisch mit der Breite b_{H_M} ihrer konvexen Hülle H_M:*

$$b_{H_M} = b_M.$$

Beweis. Wir zeigen zunächst $b_M \leq b_{H_M}$: Nach Definition 11 existiert ein Rechteck R mit $b_R = b_{H_M}$ und $H \subset R$. Weiterhin gilt $M \subset H$, somit auch $M \subset R$. Somit ist wiederum nach Definition 11: $b_M \leq b_R = b_{H_M}$.

Nun $b_M \geq b_{H_M}$: Nach Definition 11 existiert ein Rechteck S mit $b_S = b_M$ und $M \subset S$. Ein Rechteck ist ein konvexes Polygon, H ist nach Definition 13 Schnitt aller konvexer Polygone, die P enthalten, somit $H \subset S$. Weiterhin gilt nach Definition 11, daß $b_S \geq b_R$, und somit $b_M = b_S \geq b_R = b_{H_M}$. □

Definition 14 *Eine Gerade g heißt Stützgerade oder Tangente einer Menge M, wenn g M „seitlich berührt“, also:*

1. $g \cap \partial M \neq \emptyset$.
2. *Alle Punkte von M liegen in einer durch g bestimmten Halbebene.*

Satz 5 *Die konvexe Hülle H einer Punktmenge $P = \{p_1, p_2, p_3\}$ ist das Dreieck mit den Eckpunkten p_1, p_2, p_3:*

$$H = \triangle p_1 p_2 p_3.$$

Beweis. $[p_1 p_2] \in H \Rightarrow \forall q \in [p_1 p_2] : [p_3 q] \in H$. Dies gilt auch für den „entarteten" Fall, daß p_1, p_2, p_3 auf einer Linie liegen. □

Satz 6 *Die konvexe Hülle H einer endlichen Punktmenge $P = \{p_1 \dots p_n\}$ ist die Vereinigung aller Dreiecke mit Eckpunkten aus P:*

1. $|P| = 1$*:* $H = P$.
2. $|P| = 2$*:* $H = [p_1 p_2]$.
3. $|P| \geq 3$*:* $H = \bigcup_{i,j,k \in \{1 \dots n\}} \triangle p_i p_j p_k$.

Beweis. Fall 1 und 2 sind trivial. Fall 3: Sei H die konvexe Hülle von P. Damit gilt

$$\begin{aligned} p_i, p_j \in P \quad &\Rightarrow \quad [p_i p_j] \in H, \\ \forall p_i p_j p_k \in P : p_l \in \triangle p_i p_j p_k \quad &\Rightarrow \quad \exists p_m \in [p_i p_j] : p_l \in [p_k p_m] \\ &\Rightarrow \quad \triangle p_i p_j p_k \in H \\ \Longrightarrow \quad \forall p_i, p_j, p_k \in P : \triangle p_i p_j p_k \in H \quad &\Rightarrow \quad H \supset \bigcup_{i,j,k \in \{1 \dots n\}} \triangle p_i p_j p_k. \end{aligned}$$

Weiterhin gilt für alle Punkte

$$\begin{aligned} p_a, p_b \in \bigcup_{i,j,k \in \{1 \dots n\}} \triangle p_i p_j p_k \quad &\Rightarrow \quad \exists p_c, p_d, p_e \in P : p_a \in \triangle p_c p_d p_e \quad \wedge \\ &\qquad \exists p_f, p_g, p_h \in P : p_b \in \triangle p_f p_g p_h. \end{aligned}$$

$$\begin{aligned} p_c, p_d, p_e, p_f, p_g, p_h \in P \quad &\Rightarrow \bigcup_{i,j,k \in \{c,d,e,f,g,h\}} \triangle p_i p_j p_k \subset \bigcup_{i,j,k \in \{1 \dots n\}} \triangle p_i p_j p_k \\ &\Rightarrow \quad [p_a p_b] \in \bigcup_{i,j,k \in \{1 \dots n\}} \triangle p_i p_j p_k \\ &\Rightarrow \quad H \subset \bigcup_{i,j,k \in \{1 \dots n\}} \triangle p_i p_j p_k, \end{aligned}$$

und somit $H = \bigcup \triangle p_i p_j p_k$. □

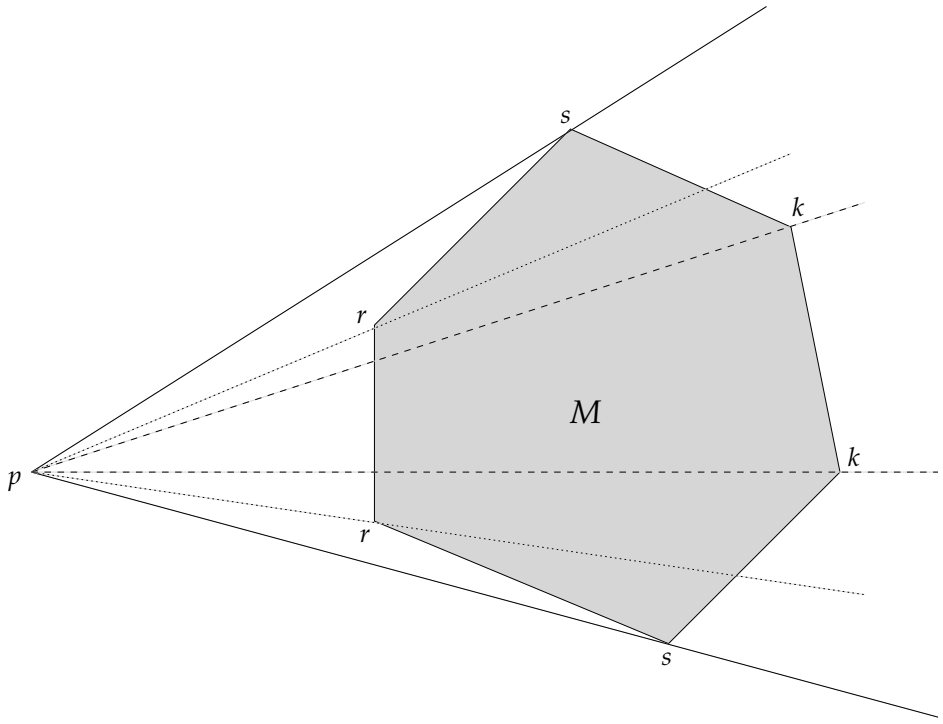


Abbildung 8.4.: Konkave, stützende und reflexe Punkte

Satz 7 *Die konvexe Hülle H einer endlichen Punktmenge $P = \{p_1 \dots p_n\}$ ist ein Polygon $[v_1 v_2 \dots v_m v_1]$ mit m „echten" Eckpunkten:*

$$H = [v_1 v_2 \dots v_m v_1], \qquad \forall v_i v_j v_k : v_i \notin v_j v_k, \quad v_i \in \{p_1 \dots p_n\}.$$

Beweis. Folgt direkt aus Satz 6. □

In den folgenden Sätzen und Beweisen werden auf einer gegebenen konvexen Hülle $H = [v_1 v_2 \dots v_m v_1]$ häufiger die zu einem Punkt v_i benachbarten Punkte v_{i-1} und v_{i+1} betrachtet. Für $i = 1$ bezeichnet v_{i-1} dann den Punkt v_m, für $i = m$ bezeichnet v_{i+1} analog dazu den Punkt v_1. Alle Rechnungen auf den Indices der Eckpunkte der konvexen Hülle werden also auf der endlichen Gruppe $\langle \{1, 2, \dots, m\}, + \rangle$ durchgeführt.

Definition 15 *Für einen beliebigen Punkt p heißt ein Punkt $q \in \partial H$*

1. *stützend, wenn $p = q$ oder pq Stützgerade von H.*
2. *konkav, wenn die Strecke $[pq]$ das Innere von H trifft: $[pq] \cap \overset{\circ}{H} \neq \emptyset$.*
3. *reflex sonst.*

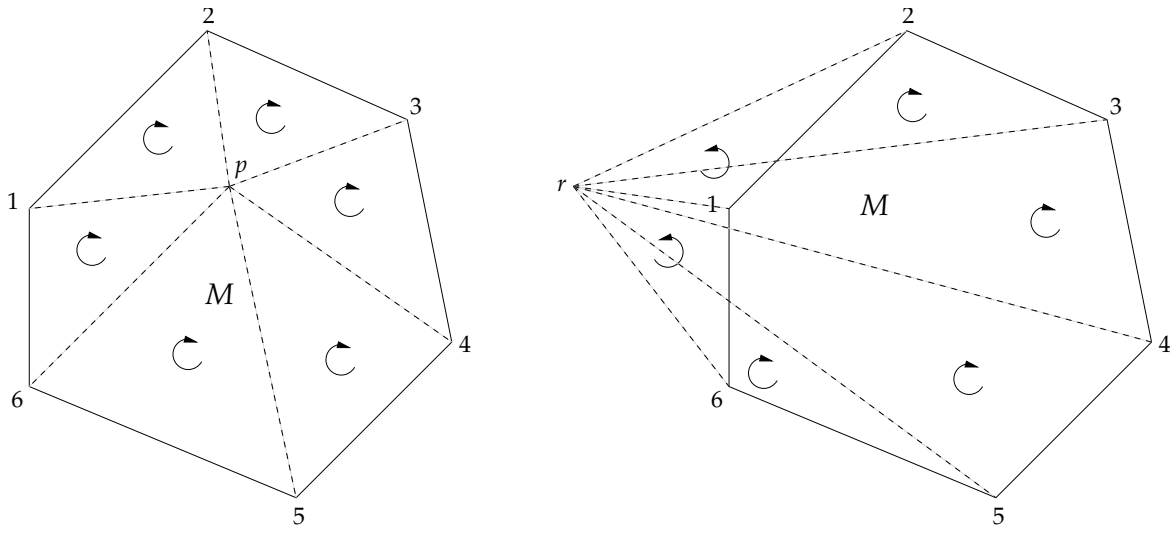


Abbildung 8.5.: Innere und äußere Punkte

Satz 8 *Für ein konvexes Polygon $H = [v_1 v_2 \dots v_m v_1]$ wie in Satz 7 mit mindestens drei Eckpunkten ($m \geq 3$) gilt: Ein Punkt p liegt genau dann in H, wenn alle Randpunkte von H bezüglich p konkav oder stützend sind und außerhalb von H, wenn es reflexe Randpunkte gibt.*

Beweis. Sei $[v_1 v_2 \dots v_m v_1]$ im Uhrzeigersinn orientiert:[2] $\angle([v_i v_{i-1}], [v_i v_{i+1}]) \in (0, \pi)$. Dann gilt für einen Punkt p:

1. $p \in H \Rightarrow \forall v_i, v_{i+1} : \angle([v_i p], [v_i v_{i+1}]) \in [0, \pi)$ (das Dreieck $\triangle p v_i v_{i+1}$ ist ebenfalls im Uhrzeigersinn orientiert), da $v_i v_{i+1}$ eine Stützgerade von H ist und somit alle $q \in H$ in der gleichen Halbebene von $v_i v_{i+1}$ liegen müssen. v_{i+2} liegt rechts von $v_i v_{i+1}$, und somit auch p.
   Somit gilt: $\forall v_i : \angle([v_i v_{i+1}], [v_i p]) \in [0, \angle([v_i v_{i+1}], [v_i v_{i-1}])]$, woraus folgt, daß v_i bezüglich p konkav oder stützend ist.
2. $p \notin H \Rightarrow \exists v_i, v_{i+1} : \angle([v_i p], [v_i v_{i+1}]) \in (-\pi, 0)$ (das Dreieck $\triangle p v_i v_{i+1}$ ist im Gegenuhrzeigersinn orientiert). Sei nun $q := v_i + (v_{i+1} - v_i)/2$. Dann ist q bezüglich p reflex: pq ist keine Stützgerade und $[pq] \notin H$.

Aufgrund der Orientierung der Dreiecke $\triangle v_{i-1} v_i p$ und $\triangle v_i v_{i+1} p$ ist also ersichtlich, ob v_i bezüglich p konkav, stützend oder reflex ist (Siehe auch Abbildung 8.5).

□

Satz 9 *Ein konvexes Polygon[3] wie in Satz 7 $H = [v_1 v_2 \dots v_m v_1]$ besitzt bezüglich eines Punktes*

[2] Im Folgenden: $v_{m+1} = v_1$ und $v_0 = v_m$
[3] Sei H im Folgenden im Uhrzeigersinn orientiert

1. $p \in \overset{\circ}{H}$ *keine stützenden und reflexen Eckpunkte,*
2. $p = v_i$ *die drei stützenden Eckpunkte* v_{i-1}, v_i, v_{i+1} *und keine reflexen Eckpunkte,*
3. $p \in [v_i v_{i+1}]$ *zwei stützende Eckpunkte* v_i, v_{i+1} *und keine reflexen Eckpunkte,*
4. $p \notin H$ *mindestens 2 stützende Eckpunkte* v_j, v_k *(mit* $j < k$*) und genau dann noch einen oder zwei weitere stützende Eckpunkte* v_{j+1}, v_{k-1}*, wenn* $p \in v_j v_{j+1}$ *bzw.* $p \in v_{k-1} v_k$*. Liegt nun* p *links von* $v_j v_k$ *(also* $\triangle p v_j v_k$ *gegen den Uhrzeigersinn orientiert), so sind alle* v_i *mit* $(i < j) \vee (i > k)$ *konkav und alle zwischen* v_j *(bzw.* v_{j+1}*) und* v_k *(bzw.* v_{k-1}*) reflex.*

Für die Eckpunkte v_i *gilt also bezüglich eines Punktes* p*: Alle konkaven Punkte liegen direkt nebeneinander, alle reflexen Punkte ebenso, beide Gruppen sind auf jeder Seite durch jeweils einen bis zwei stützende Punkte voneinander getrennt.*

Beweis. 1, 2 und 3 folgen aus Satz 6 und Satz 8. Siehe hierzu vor allem den Beweis von Satz 8: Sind die Dreiecke $\triangle v_{i-1} v_i p$ und $\triangle v_i v_{i+1} p$ beide im Uhrzeigersinn orientiert, so ist v_i konkav, sind beide gegen den Uhrzeigersinn orientiert, dann reflex. Sind die Orientierungen unterschiedlich (oder liegen v_{i-1}, v_i, p bzw v_i, v_{i+1}, p auf einer Geraden), so ist v_i stützend.

Offensichtlich muß es auch für $p \notin H$ nicht zwingend reflexe Eckpunkte geben, in diesem Fall ist nur ein einziges Dreieck $\triangle v_l v_{l+1} p$ gegen den Uhrzeigersinn orientiert. □

Satz 10 *Ist das konvexe Polygon* $H = [v_1 v_2 \dots v_m v_1]$ *(wie in Satz 7) die konvexe Hülle einer Menge* M*, so gilt für die konvexe Hülle* H' *von* $M \cup \{p\}$*:*

1. $H' = H$*, wenn alle* v_i *bezüglich* p *konkav sind.*
2. $H' = H \cup \triangle v_s v_t p = [v_1 v_2 \dots v_{s-1} v_s p v_t \dots v_m v_1]$*, wenn es stützende Punkte* v_s, v_t *bezüglich* p *gibt.*
   Hat H *bezüglich* p *mehr als zwei stützende Punkte* v_i, v_j, v_k *(und evtl.* v_l*), so liegen (je) zwei davon auf einer Geraden mit* p*, sei oBdA:* $v_j \in [v_i p]$*, so wähle* $s := i$*. Für* t *analog.*

Beweis. Fall 1 ist klar. Fall 2: Sei oBdA $\triangle p v_s v_t$ im Gegenuhrzeigersinn orientiert. Zunächst gilt $[v_s v_t] \in H$ (nach Definition 12) und

$$\forall v_i \in \{v_{s+1}, \dots v_{t-1}\} : \triangle v_i v_s v_t \subset \triangle p v_s v_t \quad \Rightarrow \quad \bigcup_{i=s+1}^{t-1} \triangle v_i v_s v_t \subset \triangle p v_s v_t.$$

Nach Satz 6 gilt (mit $v_{m+1} := v_1$ und $v_{1-1} := v_m$) weiterhin:

$$H' = H \cup \bigcup_{i,j=1}^{m} \triangle p v_i v_j = H \cup \triangle p v_s v_t$$

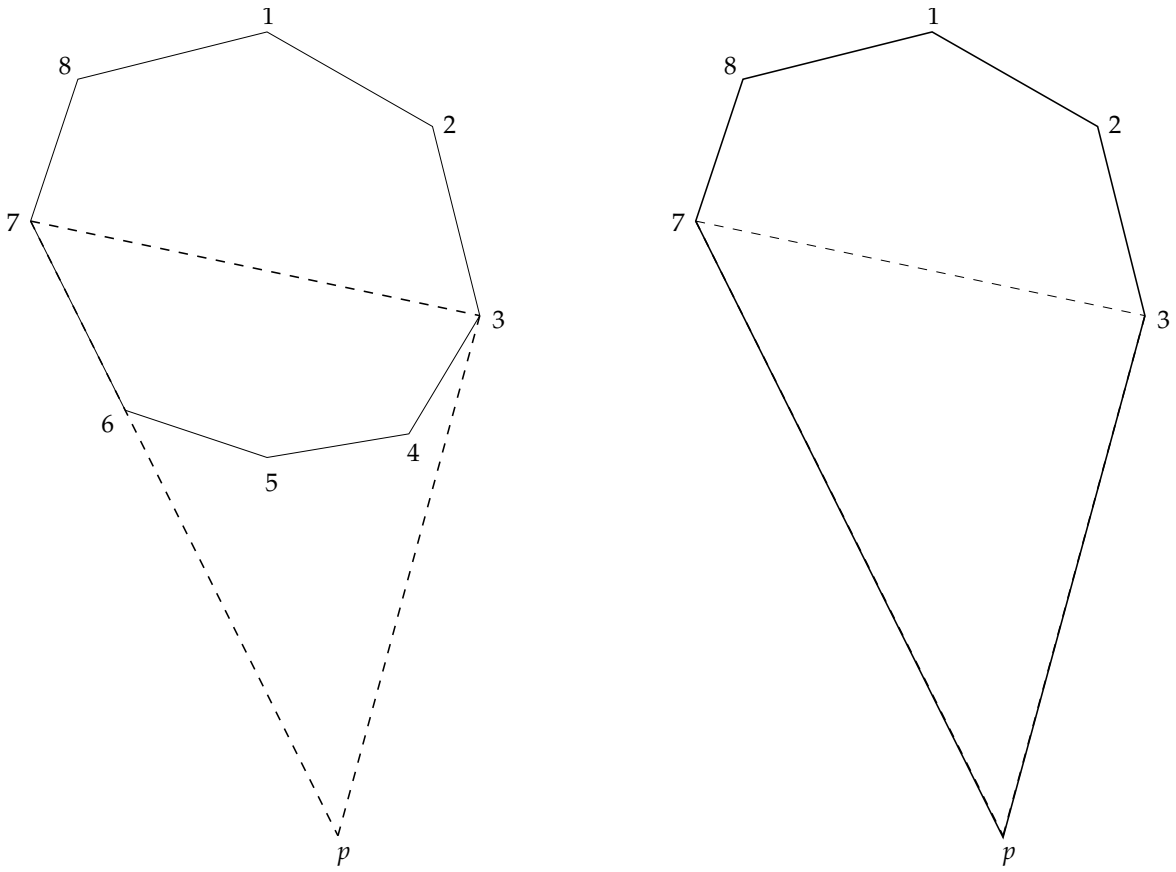


Abbildung 8.6.: Erweiterung der konvexen Hülle um einen Punkt

und mit

$$H = \bigcup_{i=1}^{m} \triangle v_i v_s v_t = \left(\left(\bigcup_{i=s+1}^{t-1} \triangle v_i v_s v_t \right) \cup \left(\bigcup_{i=t+1}^{s-1} \triangle v_i v_s v_t \right) \right),$$

somit also schließlich

$$H' = \triangle p v_s v_t \cup \left(\bigcup_{i=t+1}^{s-1} \triangle v_i v_s v_t \right).$$

□

Definition 16 *Zwei Punkte $p, q \in \partial M$ heißen antipodisch, wenn es zwei Stützgeraden $g \parallel h$ von M mit $p \in g, q \in h, p \neq q$ gibt.*

Ein Punkt $p \in \partial M$ heißt antipodisch zu einer Stützgeraden g von M, wenn $p \notin g$ und eine Stützgerade h von M mit $p \in h$ und $g \parallel h$ existiert.

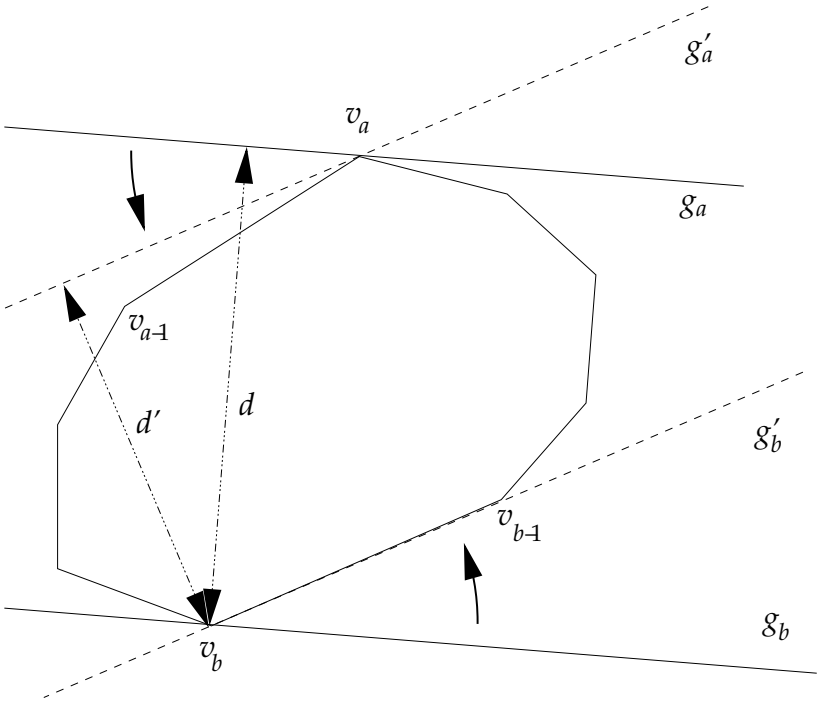


Abbildung 8.7.: Bestimmung der Breite eines Polygons

Satz 11 *Die Breite b eines konvexen Polygons $H = [v_1 v_2 \dots v_m v_1]$ wie in Satz 7 mit mindestens drei Punkten ($m \geq 3$) ist der minimale Abstand eines Eckpunktes v_i zu seiner antipodischen Stützgeraden $v_k v_{k+1}$:*

$$b = \min_{i,k} \{ d(v_i, v_k v_{k+1}) \mid v_i \text{ antipodisch zu } v_k v_{k+1} \}.$$

Die Breite eines „entarteten" ($m < 3$) Polygons $H = [v_1 v_2 v_1]$ (also der Strecke $[v_1 v_2]$) oder $H = [v_1]$ (also des Punktes v_1) ist 0.

Beweis. Die Spezialfälle mit $m < 3$ sind trivial. Sei im Folgenden also $m \geq 3$, und somit $\overset{\circ}{H} \neq \emptyset$. Zwei parallele Stützgeraden einer Menge H bilden einen Korridor, der H enthält:

$$g \parallel h, \quad g \neq h, \quad g \cap \partial H \neq \emptyset, \quad h \cap \partial H \neq \emptyset$$

Somit existieren $p \in (g \cap \partial H)$, $p \notin h$ und $q \in (h \cap \partial H)$, $q \notin g$. Alle Punkte $r \in H$ liegen nun nach Definition 14 auf der selben Seite von g wie q und auf der selben Seite von h wie p, somit zwischen g und h.

Hieraus folgt direkt, daß es ein Rechteck der Breite b gibt, das H enthält. Es bleibt also noch zu zeigen, daß es kein Rechteck R der Breite $b_R < b$ mit $H \subset R$ gibt. Offensichtlich ist jedes Rechteck R', $H \subset R'$, dessen Längsseiten H nicht berühren, breiter als H. Sei also R ein Rechteck, das H in den zueinander antipodischen Punkten v_a und v_b berührt, g_a Stützgerade durch v_a und $g^{\perp}_{a,b}$ Senkrechte zu

g_a durch v_b:

$$v_a \in g_a, \quad g_a \text{ Stützgerade}, \quad v_b \in g_{a,b}^{\perp}, \quad s := g_a \cap g_{a,b}^{\perp} \quad \Longrightarrow \quad b_R := |[v_b s]|.$$

Liege nun oBdA s links von v_a: $v_{a-1}v_a] \cap [v_b s] \neq \emptyset$. g_a ist Stützgerade durch v_a, g_b Stützgerade durch v_b, mit $g_a \parallel g_b$. Dreht man nun g_a und g_b gemeinsam (d. h. so, daß sie weiterhin parallel bleiben) gegen den Uhrzeigersinn um v_a bzw. v_b, bis g_a an v_{a-1} oder g_b an v_{b-1} anstößt, so gilt für die so erhaltenen Stützgeraden g_a' und g_b': $d(g_a', v_b) \leq d(g_a, v_b)$. Nun geht also g_a' durch v_{a-1} und v_a oder g_b' durch v_{b-1} und v_b, wie oben behauptet.

Liegt s rechts von v_a, so gilt Analoges für eine Drehung im Uhrzeigersinn bis zum Anschlag an v_{a+1} bzw. v_{b+1}.

Somit reicht es also, das Minimum der Abstände aller durch zwei Eckpunkte v_i, v_{i+1} gehenden Stützgeraden zum zugehörigen antipodischen Eckpunkt v_k zu betrachten (entweder v_{k-1} oder v_{k+1} könnten dabei ebenfalls antipodischer Eckpunkt zu $v_i v_{i+1}$ sein). □

8.2. Segmentbestimmung

Der erste Schritt des Algorithmus besteht darin, die ersten k Punkte $p_1 \ldots p_k$ des Polygonzuges zu bestimmen, die noch in einen Korridor der Breite $b \leq 2\varepsilon$ passen. Wie aus Satz 4 ersichtlich, reicht es hierzu, inkrementell für jeden neu hinzugekommenen Punkt p_{k+1} die Breite der konvexen Hülle von $\{p_1 \ldots p_{k+1}\}$ zu bestimmen.

8.2.1. Berechnung der konvexen Hülle

Die ersten beiden Punkte passen immer ($b = 0$). Ist H die konvexe Hülle der ersten k Punkte, so bestimmt sich die konvexe Hülle H' der ersten $k+1$ Punkte nach Satz 10:

1. Liegt $q := p_{k+1}$ in H, so ist $H' = H$ und es ist nichts weiter zu tun.
2. Liegt q außerhalb von H, so gibt es die beiden stützenden Punkte v_s und v_t (oBdA $s < t$), und H' ist das Polygon, das entsteht, wenn man die Sequenz $v_{s+1} \ldots v_{t-1}$ durch q ersetzt ($H' = [v_1 \ldots v_s q v_t \ldots v_m v_1]$, wenn das Dreieck $\triangle v_s q v_t$ im Uhrzeigersinn orientiert ist, und $H' = [v_s \ldots v_t q v_s]$, wenn $\triangle v_s q v_t$ gegen den Uhrzeigersinn orientiert ist (siehe hierzu auch Abbildung 8.6).

Beide Schritte lassen sich auf einmal erledigen: Finden sich keine stützenden Eckpunkte v_s und v_t, so liegt q in H.

Da weiterhin nach Satz 9 die stützenden Eckpunkte *zwischen* den konkaven und reflexen liegen, ist eine binäre Suche möglich:

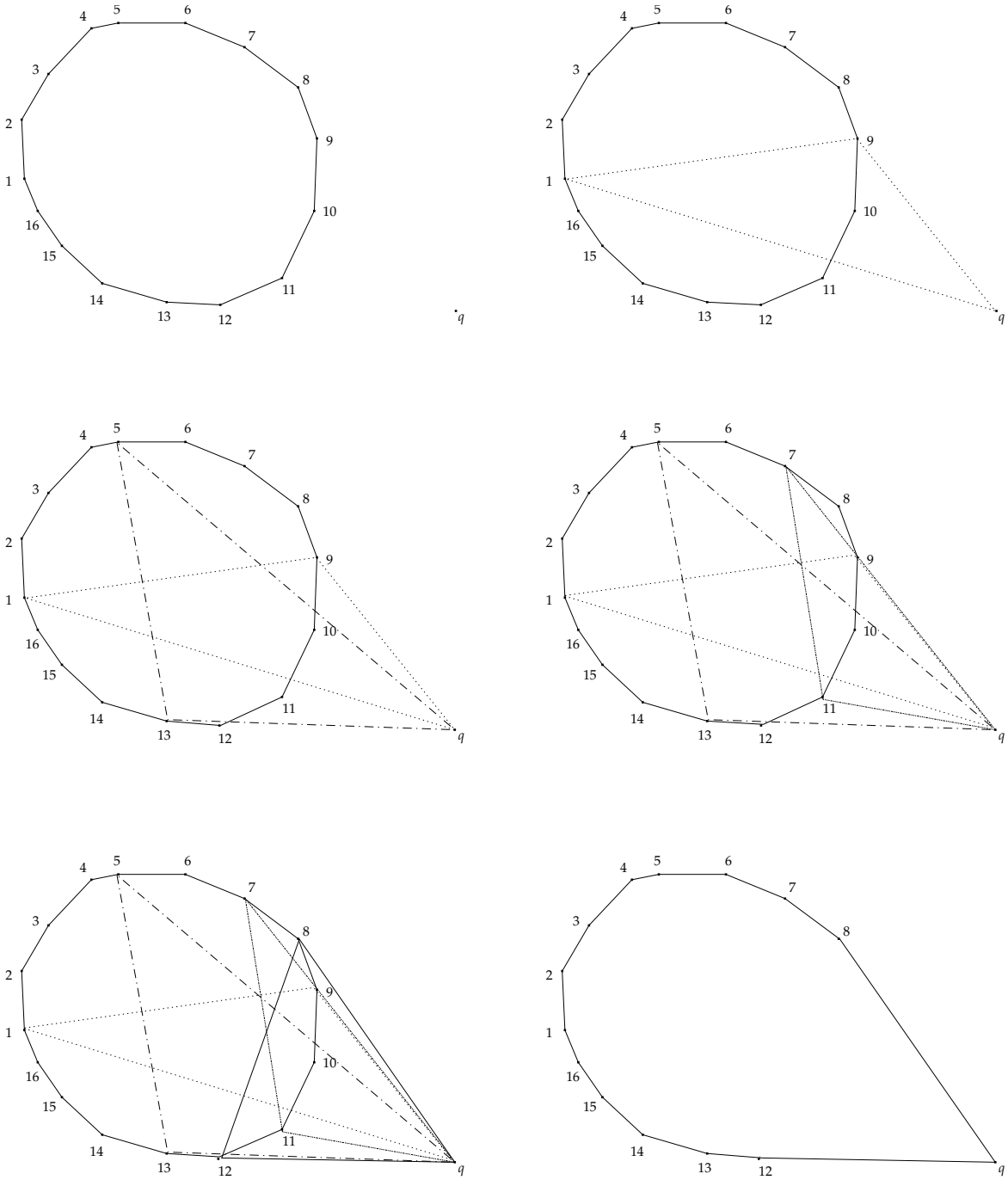


Abbildung 8.8.: Die Bestimmung der Stützpunkte und der erweiterten konvexen Hülle mit einem konkaven und einem reflexen Startpunkt

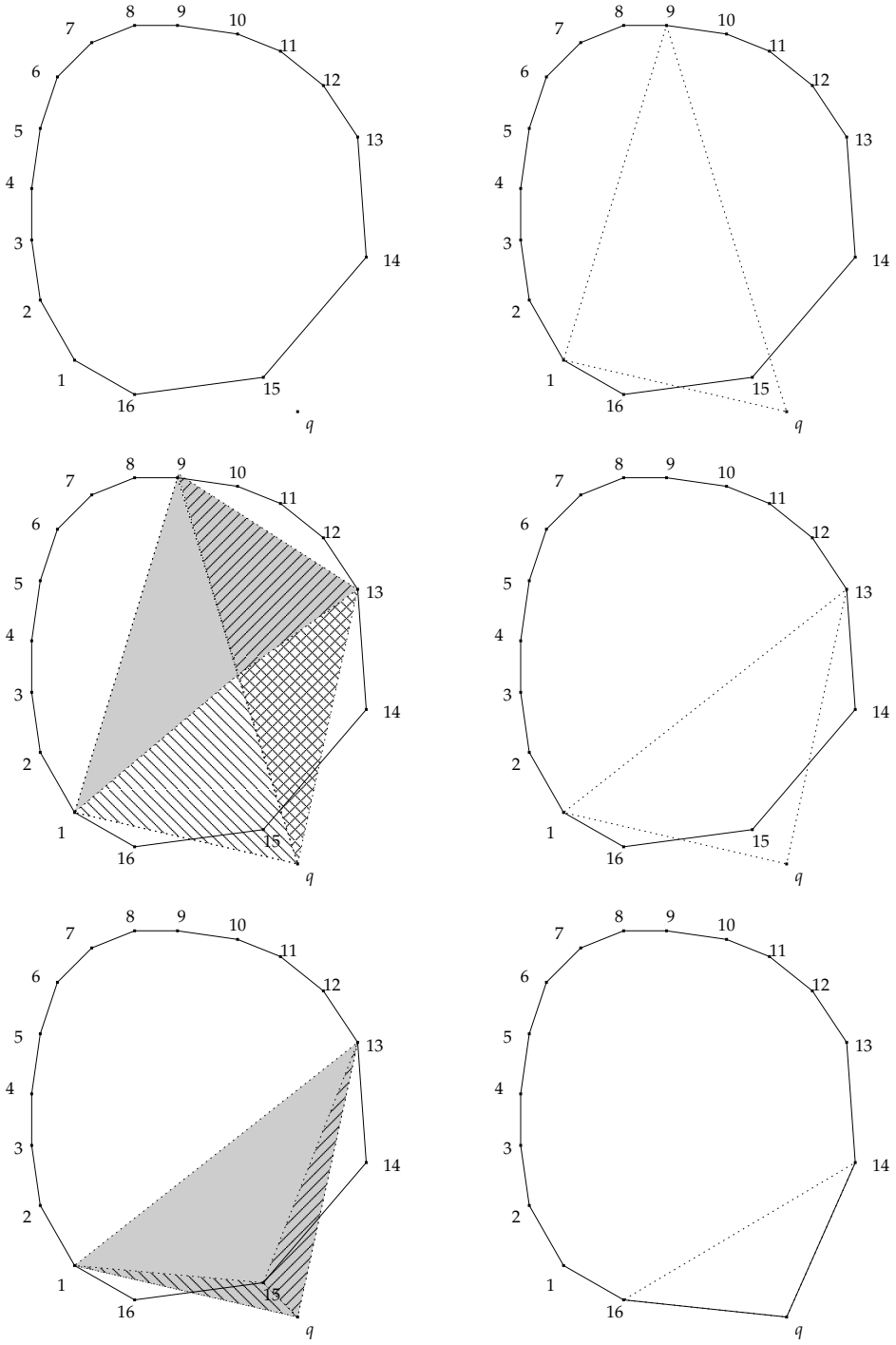


Abbildung 8.9.: Die Bestimmung der Stützpunkte und der erweiterten konvexen Hülle mit zwei konkaven Startpunkten

1. Wähle 2 beliebige[4] Eckpunkte von H, z. B. $v_l := v_1$ und $v_r := v_{\lfloor \frac{m}{2} \rfloor + 1}$.
   Liegen sie mit q auf einer Linie ($q \in v_l v_r$) und
   a) $q \in [v_l v_r]$, so ist $q \in H$: fertig.
   b) $q \notin [v_l v_r]$, wähle zwei neue Eckpunkte $v_l := v_{\lfloor \frac{m}{4} \rfloor}$ und $v_r := v_{\lfloor \frac{3m}{4} \rfloor}$.
2. Betrachte nun das Dreieck $\triangle q v_l v_r$ (oBdA $\triangle q v_l v_r$ im Uhrzeigersinn orientiert): $[v_l v_r] \in H$. Liegt $[v_l q$ zwischen $[v_l v_{l-1}$ und $[v_l v_r$, so ist v_l konkav bezüglich q, schneidet $v_l q$ das Dreieck $\triangle v_l v_{l+1} v_r$, so ist v_l reflex zu q, ansonsten ist $v_l q$ Stützgerade und somit v_l stützend zu q. In diesem Fall gilt:
   a) $v_l v_{l+1} \neq v_l q$ (also $v_{l+1} \notin v_l q$): der „linke" Stützpunkt $v_s := v_l$ zu q ist gefunden.
   b) $v_l v_{l+1} = v_l q$ (also $v_{l+1} \in v_l q$): v_{l+1} ist ebenfalls stützend zu q und $v_l \in [q v_{l+1}]$, somit also $v_s := v_{l+1}$.

   Analog das Vorgehen für v_r, nur daß hier im Fall, daß v_r stützend ist, v_{r-1} als potentieller weiterer Stützpunkt getestet werden muß.
3. Ist mindestens einer der beiden Punkte nicht stützend, so muß nun rekursiv nach den Stützstellen gesucht werden, wobei folgende Möglichkeiten in Betracht kommen:
   a) Ein reflexer und ein konkaver Punkt (oBdA: v_l konkav, v_r reflex): Da nach Satz 9 die stützenden immer zwischen der Menge der reflexen und der Menge der konkaven Punkten liegen müssen, ist nun binäre Suche nach den beiden stützenden Punkten möglich: v_a und v_b sind die zwischen v_l und v_r liegenden Punkte $v_a := v_{\lfloor \frac{l+r}{2} \rfloor}, v_b := v_{r+\lfloor \frac{r-l}{2} \rfloor}$.
      i. v_a konkav: Ein stützender Punkt muß also zwischen v_a und v_r liegen, hier rekursiv weitersuchen.
      ii. v_a reflex: Zwischen v_l und v_a rekursiv weitersuchen.
      iii. v_a stützend: Prüfen, ob der von q entfernter liegende Punkt v_{a-1} ebenfalls stützend ist ($v_a \in [v_{a-1} q]$):
         A. $v_a \in [v_{a-1} q] : v_s := v_{a-1}$
         B. $v_a \notin [v_{a-1} q] : v_s := v_a$

         v_s ist gefunden.

      v_b analog.
   b) Ein stützender Punkt (oBdA v_l stützend, das Dreieck $\triangle v_l v_r q$ ist im Uhrzeigersinn orientiert):
      i. v_r konkav: Die Punkte zwischen v_l und v_r müssen somit ebenfalls konkav sein (sie liegen auf der anderen Seite von $v_l v_r$ wie q), somit findet sich der zweite stützende Punkt durch binäre Suche in $\{v_{r+1}, v_{r+2}, \ldots v_{l-1}\}$

[4] Für eine Reihe von Generalisierungsaufgaben empfiehlt es sich, mit den in der vorhergehenden Iteration gefundenen Stützstellen anzufangen.

ii. v_r reflex: Die Punkte zwischen v_r und v_l: $\{v_{r+1}, \dots v_{l-1}\}$ müssen also ebenfalls reflex sein[5], somit findet sich der zweite stützende Punkt durch binäre Suche in $\{v_{i+1}, \dots v_{r-1}\}$.

c) Beide Punkte konkav ($\triangle v_l v_r q$ im Uhrzeigersinn orientiert): Wenn es stützende und/oder reflexe Punkte gibt, so müssen diese auf der gleichen Seite von $v_l v_r$ wie q liegen, also in $\{v_{r+1}, \dots v_{l-1}\}$.
Wähle $v_c := v_{r+\lfloor \frac{r-l}{2} \rfloor}$:

i. v_c reflex: Die beiden stützenden Punkte müssen also zwischen v_c und v_l sowie v_r und v_c liegen und können durch binäre Suche gefunden werden.

ii. v_c stützend (ist v_{c+1} oder v_{c-1} ebenfalls eine Stützstelle mit $v_c \in [q v_{c+1}]$ bzw. $v_c \in [q v_{c-1}]$, so betrachte diese): Ist das Dreieck $\triangle v_c v_r q$ im Uhrzeigersinn orientiert, also q auf der anderen Seite von $v_c v_r$ wie v_l, so sind $\{v_{c+1}, \dots v_l\}$ konkav, die zweite Stützstelle kann also durch binäre Suche zwischen v_r und v_c gefunden werden. Ist $\triangle v_c v_r q$ gegen den Uhrzeigersinn orientiert (dies ist gleichbedeutend mit $\triangle v_l v_c q$ im Uhrzeigersinn orientiert, da v_c Stützstelle ist) entsprechend binäre Suche zwischen v_c und v_l.

iii. v_c konkav:

A. $(q \in \triangle v_c v_l v_r) \Leftrightarrow (\triangle v_c v_l q$ und $\triangle v_r v_c q$ sind beide im Uhrzeigersinn orientiert): $q \in H$, fertig.

B. $\triangle v_c v_l q$ gegen den Uhrzeigersinn orientiert: rekursiv weiter prüfen mit $v'_r := v_c$.

C. $\triangle v_r v_c q$ gegen den Uhrzeigersinn orientiert: rekursiv weiter prüfen mit $v'_l := v_c$.

d) Beide Punkte reflex: Die konkaven und somit auch die stützenden Punkte müssen also zwischen v_l und v_r liegen, also auf der *anderen* Seite von $v_l v_r$ wie q. Nun wähle[6] $v_a := v_{l+\lfloor \frac{r-l}{4} \rfloor}, v_b := v_{r-\lfloor \frac{r-l}{4} \rfloor}$. Gilt nun

i. v_a oder v_b ist stützend oder konkav, oder beide sind weiterhin reflex und $\triangle q v_a v_b$ ist wiederum im Uhrzeigersinn orientiert ($\triangle q v_a v_b$ und $\triangle q v_l v_r$ sind gleich orientiert), wähle $v'_l := v_a$ und $v'_r := v_b$ und teste erneut.

ii. v_a und v_b sind beide reflex *und* $\triangle q v_a v_b$ ist *gegen* den Uhrzeigersinn orientiert:

[5] allenfalls v_{l-1} könnte noch der zu v_l gehörige stützende Punkt sein, wenn $v_{l-1} v_l$ eine Stützgerade bezüglich q ist.

[6] Der Bereich zwischen v_l und v_r wird hier geviertelt und nicht wie vielleicht zu erwarten gedrittelt, da im Mittel die Anzahl der konkaven Ecken die der reflexen übersteigt, dies um so mehr, je näher q an H liegt. Gerade im Kontext der Generalisierung von Polygonzügen ist also 4 als Divisor angemessen, je nach Art der Daten könnte auch 5 oder 6 sinnvoll sein. Die Komplexität wird durch den Wert des Divisors nicht verändert, sie bleibt logarithmisch.

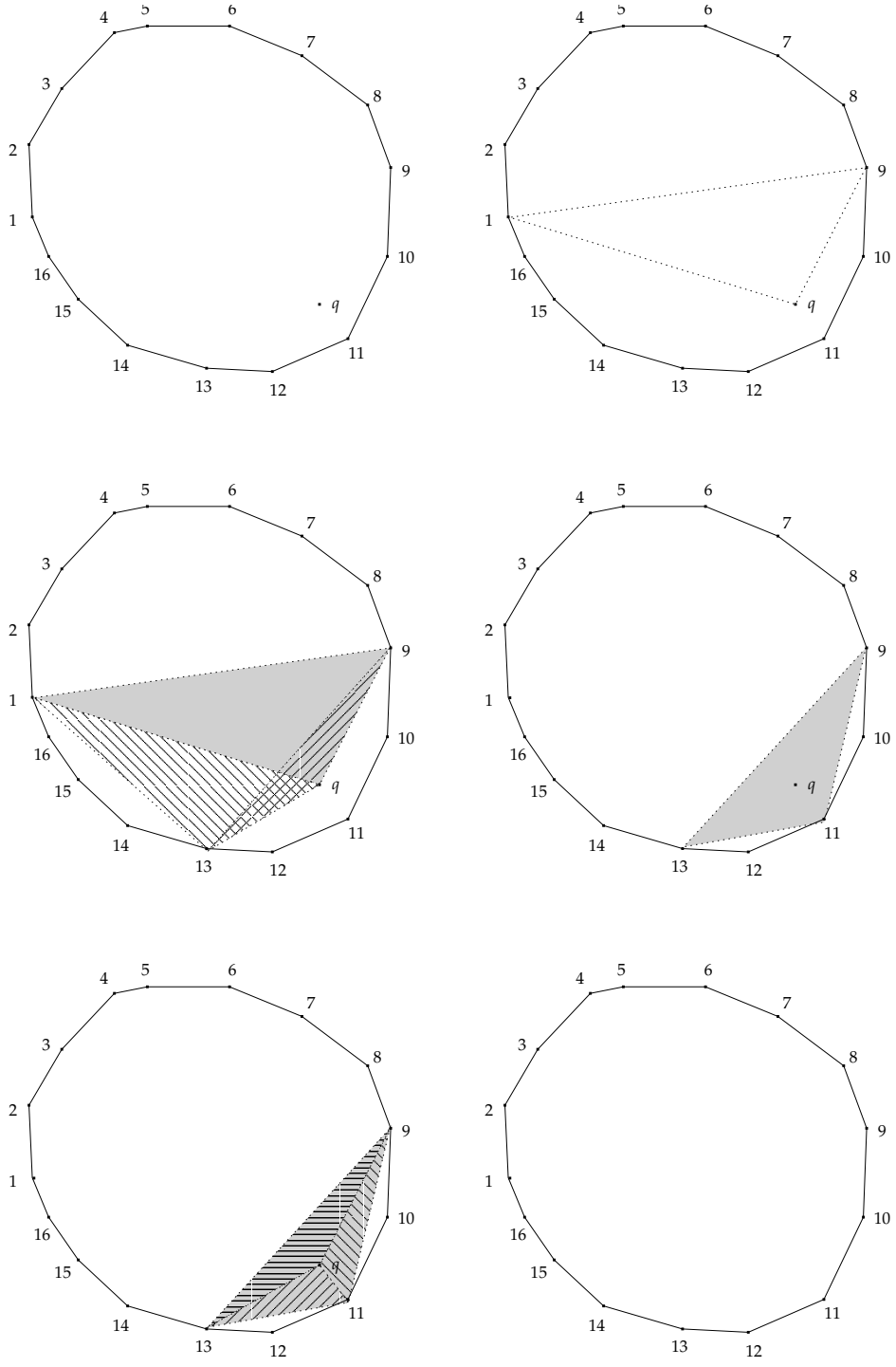


Abbildung 8.10.: Der Algorithmus zur Bestimmung der neuen konvexen Hülle mit $q \in H$.

A. $\triangle q v_l v_a$ im Uhrzeigersinn orientiert (das ist äquivalent zu: $\triangle q v_r v_b$ gegen den Uhrzeigersinn orientiert): $v'_l := v_l, v'_r := v_a$.
B. $\triangle q v_l v_a$ gegen den Uhrzeigersinn orientiert (äquivalent: $\triangle q v_r v_b$ im Uhrzeigersinn orientiert): $v'_l := v_b, v'_r := v_r$.

In der weit überwiegenden Anzahl der Fälle tritt hierbei Fall 3(d)i auf. Fall 3(d)ii kann nur vorkommen, wenn der konkave Bereich von H weniger als ein Viertel der Eckpunkte umfaßt (normalerweise mehr als die Hälfte) und gleichzeitig v_l und v_r geeignet liegen.

Wurden durch obiges Verfahren zwei Stützstellen v_s und v_t gefunden, so muß abschließend noch geprüft werden, ob $q \in [v_s v_t]$, da in diesem Fall $q \in H$.

Dieses Verfahren stellt also fest, ob zu q Stützstellen existieren und liefert diese, beides in logarithmischer Zeit. Somit ist dies ein $O(n \log n)$ Verfahren zur Berechnung der konvexen Hülle von n Punkten, in dieser Hinsicht also optimal.

Da für viele praktische Anwendungen die vorkommenden konvexen Hüllen eine überschaubare Anzahl Eckpunkte haben, kann häufig die einfacher zu implementierende lineare Suche nach den Stützstellen hinreichend sein. Die Gesamtkomplexität der Hüllenberechnung steigt dadurch zwar von $O(n \log n)$ auf $O(n^2)$, allerdings gehen hier ja nicht alle Punkte des zu generalisierenden Polygonzuges ein sondern nur die Eckpunkte der konvexen Hülle eines einzelnen Polygonzugsegments.

8.2.2. Bestimmung der Breite

Um die Breite einer konvexen Hülle zu bestimmen, reicht es nach Satz 11, zu jeder Geraden $v_i v_{i+1}$ den zugehörigen antipodischen Punkt v_k zu bestimmen und dessen Abstand zu $v_i v_{i+1}$ messen. Der minimale Abstand ist dann die Breite des Polygons.

Die Bestimmung des antipodischen Punktes v_k zu $v_i v_{i+1}$ geht mit binärer Suche in logarithmischer Zeit, was für die Bestimmung der antipodischen Punkte zu allen Seiten des Polygons schon eine Komplexität von $O(n \log n)$ bedeuten würde. Berücksichtigt man allerdings, daß aus der Tatsache, daß v_k antipodischer Punkt zu $v_i v_{i+1}$ ist, folgt, daß der zu $v_{i+1} v_{i+2}$ antipodische Punkt $v_l \in \{v_k, v_{k+1} \dots v_i\}$, also „rechts“ von v_k liegt, kann man leicht einen Algorithmus angeben, der die Entfernung aller Kanten des Polygons zu ihren antipodischen Punkten und somit auch die Breite des gesamten Polygons in linearer Zeit bestimmt:

1. Der zu $v_1 v_2$ antipodische Punkt $w_{12} := v_w$ kann durch binäre Suche in $O(\log n)$ gefunden werden.
2. Um den antipodischen Punkt w_{23} zu $v_2 v_3$ zu finden, prüfe der Reihe nach, ob $v_w, v_{w+1}, \dots$ antipodisch zu $v_2 v_3$ ist.
3. Allgemein also: Bestimme den zu $v_i v_{i+1}$ antipodischen Punkt $w_{i,i+1}$ durch lineare Suche in $v_w, v_{w+1}, v_{w+2}, \dots$.

Da dabei *nur* der zuletzt gefundene Punkt v_w nochmals betrachtet wird, hat dieses Verfahren $O(n)$. Es stört hierbei nicht, daß ein einzelner Punkt w antipodischer Punkt zu mehreren nebeneinanderliegenden Kanten sein kann, da hierdurch andere Eckpunkte zu keiner Kante antipodisch sind. Die Komplexität steigt dadurch also nicht (es ist egal, ob ein Punkt fünf mal und dafür weitere drei Punkte nur einmal betrachtet werden, oder ob jeder von vier Punkten zweimal betrachtet wird).

Für die Segmentierung muß die Breite der konvexen Hülle H in jedem Schritt neu bestimmt werden. Braucht dies wie beschrieben lineare Zeit, so steigt damit die Komplexität des Gesamtalgorithmus auf $O(n^2)$. Daher nutzt das folgende Verfahren den inkrementellen Aufbau der konvexen Hülle durch geschicktes Zwischenspeichern von Werten, wodurch die Komplexität bei $O(n \log n)$ bleibt:

1. Im Dreieck $H := [v_1 v_2 v_3 v_1]$ ist v_1 antipodisch zu $v_2 v_3$, v_2 zu $v_1 v_3$ und v_3 zu $v_1 v_2$.
2. Speichere nun zu jeder Kante $v_i v_{i+1}$ den zugehörigen antipodischen Punkt w und dessen Distanz $d_{v_i v_{i+1}}$ zu $v_i v_{i+1}$, und
3. zu jedem Punkt die zugehörigen antipodischen Kanten (hierzu reicht es, die Randpunkte des Bereichs zu speichern: Ist w antipodisch zu $v_i v_{i+1}, v_{i+1} v_{i+2}, \ldots v_{i+k-1} v_{i+k}$, so reicht es, zu w die Punkte v_i und v_{i+k} zu speichern, da alle dazwischenliegenden Kanten w als antipodischen Punkt haben müssen).
4. Die Kanten werden weiterhin in einer nach den Distanzen $d_{v_i v_{i+1}}$ sortierten Liste D gehalten.
5. Wird nun ein neuer Punkt q wie in Abschnitt 8.2.1 beschrieben hinzugefügt ($H' := H \cup \{q\}$), so muß die Breite von H' neu bestimmt werden, falls $q \notin H$, also $H' \neq H$. Hierbei gilt:

   a) Für alle Kanten $v_i v_{i+1}$, deren antipodischer Punkt $w_{i,i+1}$ bezüglich q konkav ist, ändert sich nichts.
   b) Ist $w_{i,i+1}$ bezüglich q reflex, so ist q neuer antipodischer Punkt zu $v_i v_{i+1}$, und $d_{v_i v_{i+1}}$ muß neu berechnet und in D einsortiert werden.
   c) Ist $w_{i,i+1}$ bezüglich q stützend, so muß getestet werden, ob $w_{i,i+1}$ oder q neuer antipodischer Punkt zu $v_i v_{i+1}$ ist (Test, ob $d(v_i v_{i+1}, w_{i,i+1}) < d(v_i v_{i+1}, q)$), $d_{v_i v_{i+1}}$ muß gegebenenfalls neu berechnet und in D einsortiert werden.

   Die Stützstellen v_l und v_r von H bezüglich q wurden bereits für die Berechnung von H' (Abschnitt 8.2.1) bestimmt, sind also bekannt. Wie in Satz 9 gezeigt, liegen alle reflexen Eckpunkte auf der q näher liegenden Seite zwischen v_r und v_l (es sind genau die Punkte, die keine Eckpunkte von H' sind).
6. Für die Punkte[7] $\{v_{r+1}, v_{r+2}, \ldots v_{l-1}\}$ gilt also: Die zugehörige antipodische

[7] Diese Punkte sind alle reflex, lediglich v_{r+1} und v_{l-1} könnten zu v_r bzw. v_l gehörige weitere Stützstellen sein (also $v_{r+1} \in [v_r q]$ bzw. $v_{l-1} \in [v_l q]$), die dann gar nicht betrachtet werden

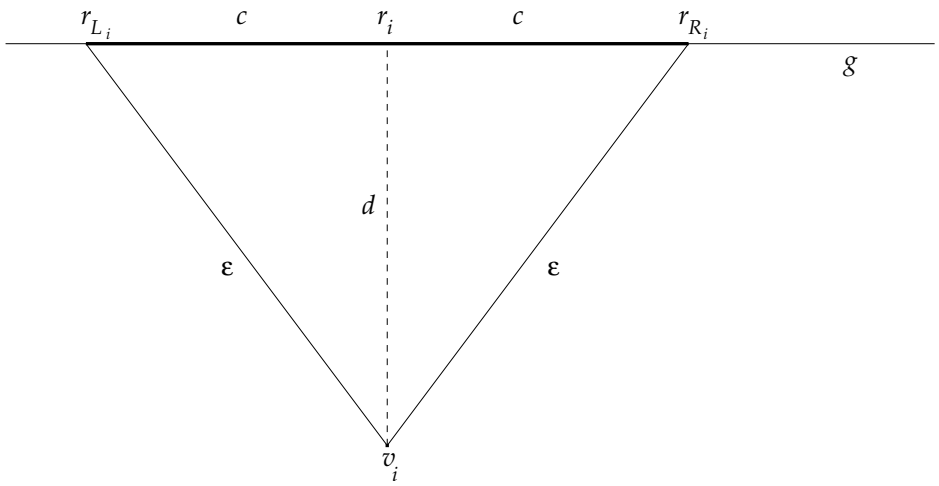


Abbildung 8.11.: Bestimmung der Punkte $r \in g$ mit $d(v_i, r) \leq \varepsilon$

Kante $v_j v_{j+1}$ hat q als neuen antipodischen Punkt und $d'_{v_j v_{j+1}} := d(v_j v_{j+1}, q)$ als neue Distanz. Nun muß $d_{v_j v_{j+1}}$ aus D gestrichen und $d'_{v_j v_{j+1}}$ neu einsortiert werden. Dies[8] geht bekanntlich in logarithmischer Zeit (wobei man hier zusätzlich nutzen kann, daß $d'_{v_j v_{j+1}} > d_{v_j v_{j+1}}$), dies für $k := l - r - 1$ Werte (und eventuell zusätzlich für die beiden Stützstellen v_r und v_l). Somit benötigt die Berechnung der neuen Breite $O(k \log n)$.
Glücklicherweise führt dies nicht dazu, daß der Gesamtalgorithmus wie auf den ersten Blick zu vermuten damit eine Komplexität von $O(n^2 \log n)$ besitzt. Dies liegt daran, daß die k oben betrachteten Punkte nicht mehr Teil des Polygons H' sind und somit im Folgenden überhaupt nicht mehr betrachtet werden.
Die Gesamtkomplexität des Algorithmus beträgt also $O(n \log n)$.

7. Die Breite $b_{H'}$ von H' ist nun die kleinste Distanz aus D. Ist $b_{H'} \leq 2\varepsilon$, so gehört q noch zum aktuellen Segment und der nächste Punkt des Polygonzuges wird geprüft. Ist $b_{H'} > 2\varepsilon$, so gehört q nicht mehr zum aktuellen Segment, das Segment ist abgeschlossen.

8.2.3. Bestimmung des Streckenstücks

Für ein durch das beschriebene Verfahren gefundenes Polygonzugsegment $\hat{p}$ gibt es nun mindestens eine Strecke s, so daß alle Punkte $p_i \in \hat{p}$ maximal ε von s entfernt liegen: $\forall p_i : d(s, p_i) \leq \varepsilon$.

Ist $d_{v_i v_{i+1}}$ die kleinste Distanz aus D, $w_{i,i+1}$ der zugehörige antipodische Punkt

brauchen, da sie als neue antipodische Punkte nicht in Frage kommen.
[8] das Einsortieren eines Wertes in eine sortierte Liste

und g die zu v_iv_{i+1} parallele Gerade, die zwischen v_iv_{i+1} und $w_{i,i+1}$ liegt:

$$d(v_iv_{i+1}, g) = d(g, w_{i,i+1}) = \frac{1}{2}d_{v_iv_{i+1}},$$

so liegen alle $p_i \in \hat{p}$ maximal $\frac{1}{2}d_{v_iv_{i+1}}$ von g entfernt, eine Strecke s, die Teil von g ist ($s \subset g$) liegt also diesbezüglich optimal.

Ist h_i die zu g rechtwinklige Gerade durch v_i (also $h_i \perp g,\ v_i \in h_i$) und $r_i := (g \cap h_i)$, so berechnet sich der minimale Abstand eines Punktes v_i zur Geraden g aus der Länge der Strecke $[v_ir_i]$. Ist $|[v_ir_i]| < \varepsilon$, so existieren neben r_i weitere Punkte auf g, zu denen v_i einen Abstand kleinergleich ε besitzt (siehe Abb. 8.11):

$$\forall r \in [r_{L_i}r_{R_i}] \subset g : d(r, v_i) \leq \varepsilon.$$

Ist nun $c := d(r_{L_i}, r_i) = d(r_i, r_{R_i})$ und $d := d(r_i, v_i)$, so gilt: $d^2 + c^2 = \varepsilon^2$ und somit $c = \sqrt{\varepsilon^2 - d^2}$. g und v_i sind bekannt, also lassen sich r_i und c und somit r_{L_i} und r_{R_i} (in konstanter Zeit) bestimmen.

Für alle Punkte v_i seien nun r_{L_i} und r_{R_i} bezüglich g gleich sortiert, d. h. für alle v_j auf der einen Seite von g ist $\triangle v_jr_{L_i}r_{R_i}$ im Uhrzeigersinn orientiert und für alle v_k auf der anderen Seite von g ist $\triangle v_kr_{L_i}r_{R_i}$ gegen den Uhrzeigersinn orientiert. Mit anderen Worten: r_{L_i} liegt jeweils „links“ von r_{R_i}.

Ist nun s_L das Teilstück von g, das alle r_{L_i} enthält, und s_R das Teilstück, das alle r_{R_i} enthält:

$$s_L := \bigcup_{i,j\in\{1,\ldots m\}} [r_{L_i}r_{L_j}], \qquad s_R := \bigcup_{i,j\in\{1,\ldots m\}} [r_{R_i}r_{R_j}],$$

so gilt:

$$\forall v_i : \quad d(v_i, s_L) \leq \varepsilon \quad \wedge \quad d(v_i, s_R) \leq \varepsilon,$$

und schließlich:

$$s := s_L \cap s_R \quad \Rightarrow \quad \forall v_i :\ d(v_i, s) \leq \varepsilon.$$

Beweis. $s_L \subset g$ und $s_R \subset g$ enthalten für jeden Punkt v_i einen Punkt r_{L_i} bzw. r_{R_i}, der von v_i den Abstand ε besitzt, somit ist die ε-Bedingung erfüllt. Nun ist noch zu zeigen, daß dies auch für $s = s_L \cap s_R$ gilt.

Der Schnitt von s_L und s_R ist nicht leer ($s \neq \emptyset$), da sonst weitere Punkte in den Korridor gepaßt hätten: Angenommen $s = \emptyset$, so existiert ein Punkt $t \in g$ mit $\forall v_i : t \in [r_{L_i}r_{R_i}]$. t liegt also „zwischen “ s_L und s_R und besitzt zu allen v_i einen Abstand kleiner ε. Somit liegen alle v_i in einem Kreis um t mit Radius ε. Fügt man nun einen weiteren Punkt p_{n+1} zu $\hat{p}$ hinzu, so gilt, daß alle $m+1$ Punkte zur Strecke $[qp_{n+1}]$ einen Abstand kleinergleich ε besitzen, die um p_{m+1} erweiterte konvexe Hülle H' hätte also eine Breite $b' \leq \varepsilon$, somit kann die Segmentbestimmung nicht schon bei p_n beendet gewesen sein: Das Segment wäre also nicht vollständig.[9]

[9] Dies kann höchstens bei der Bearbeitung des allerletzten Segmentes eines Polygonzuges auftreten, stellt dann aber kein großes Problem dar, da man in diesem Fall t als Endpunkt der Generalisierung wählen kann.

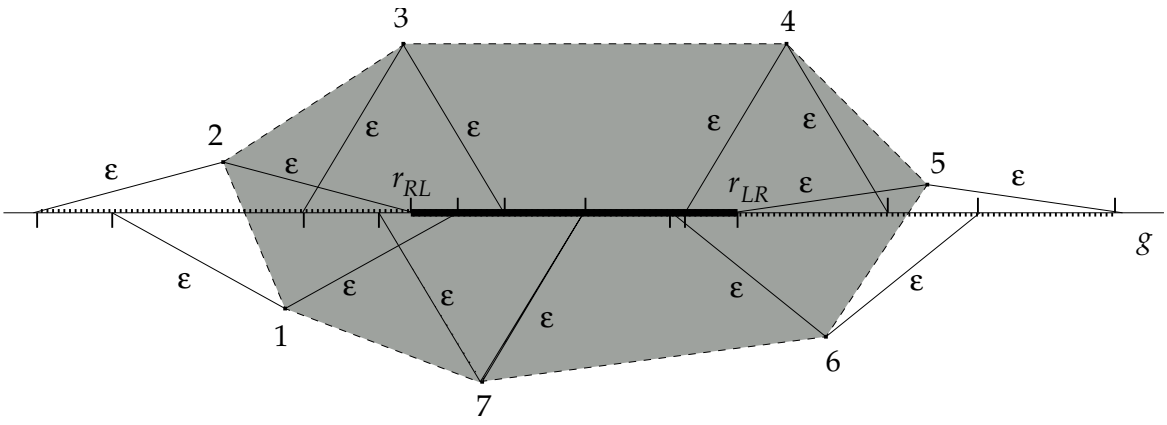


Abbildung 8.12.: Bestimmung von r_{LR} und r_{RL}.

Weiterhin gilt:

$$s = s_L \cap s_R = \left(\bigcup_{a,b=1}^{m} [r_{L_a} r_{L_b}] \right) \cap \left(\bigcup_{c,d=1}^{m} [r_{R_c} r_{R_d}] \right) = \bigcup_{a,b,c,d=1}^{m} \left([r_{L_a} r_{L_b}] \cap [r_{R_c} r_{R_d}] \right)$$

Obiges ist nur eine umständliche Schreibweise für $s_L = [r_{LL} r_{LR}]$ und $s_R = [r_{RL} r_{RR}]$ mit r_{LL} der am weitesten „links“ und r_{LR} der am weitesten „rechts“ liegende Punkt aller r_{L_i}, analog für r_{R_i}:

$$\forall r_{L_i} : r_{L_i} \in [r_{LL} r_{LR}] \quad \wedge \quad \forall r_{R_i} : r_{R_i} \in [r_{RL} r_{RR}] \quad \Rightarrow \quad \forall i : [r_{L_i} r_{R_i}] \cap [r_{RL} r_{LR}] \neq \emptyset$$

Somit gilt also für $s = [r_{RL} r_{LR}]$: $\forall v_i : d(v_i, s) \leq \varepsilon$. □

Zur Bestimmung von r_{LR} und r_{RL} müssen nicht alle r_{L_i} und r_{R_i} bestimmt werden. Abbildung 8.12 zeigt dies:

$d(v_7, [v_3 v_4])$ ist minimal (v_7 ist antipodischer Punkt zu $[v_3 v_4]$). Somit haben v_3, v_4 und v_7 den maximalen Abstand aller v_i zu g. Der am weitesten „links“ liegende Punkt r_{RL} kann also nur von Punkten im Bereich v_7 und v_3 bestimmt sein: $r_{RL} \in \{r_{R_7}, r_{R_1}, r_{R_2}, r_{R_3}\}$, analog $r_{LR} \in \{r_{L_4}, r_{L_5}, r_{L_6}, r_{L_7}\}$.

Allgemein: Ist $d(w_{i,i+1}, v_i v_{i+1})$ die kleinste Distanz aus D und somit g die zugehörige zu $v_i v_{i+1}$ parallele Gerade, so bestimmen die Punkte $v_{i+1}, v_{i+2}, \ldots, w_{i,i+1}$ die Position von r_{LR} und die Punkte $w_{i,i+1}, \ldots, v_{i-1}, v_i$ die Position von r_{RL}.

8.2.4. Bestimmung aller Streckenstücke

Das oben ermittelte Streckenstück stellt die Basis des kürzesten Korridorsegments minimaler Breite dar, das die ermittelte konvexe Hülle H enthält. Allerdings kann es noch weitere 2ε-breite Korridore geben, die H enthalten.

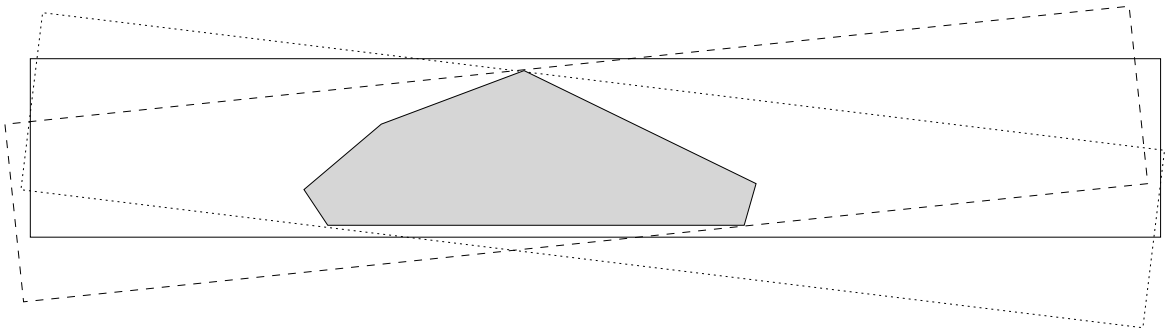


Abbildung 8.13.: Drehung des 2ε-Korridors

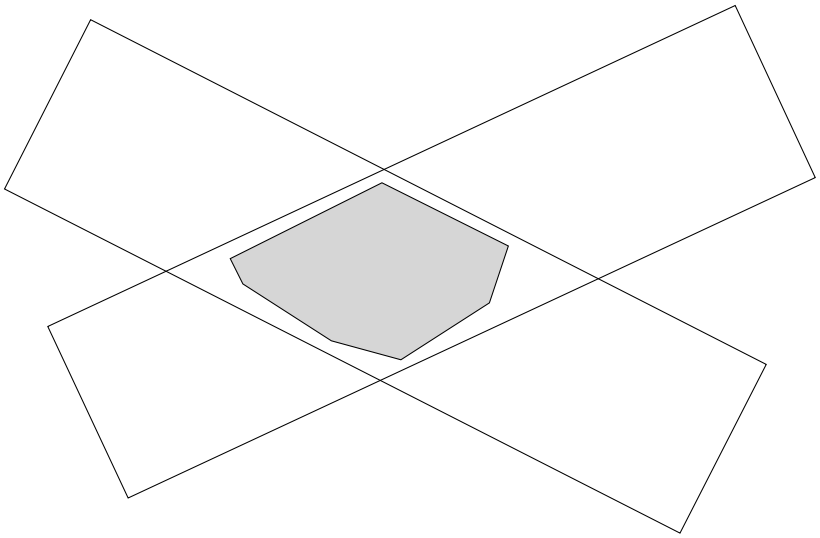


Abbildung 8.14.: Alternative Lagen des 2ε-Korridors

- Ist die minimale Breite von H kleiner als 2ε, so kann der 2ε breite Korridor natürlich ein kleines Stück in der Breite verschoben werden (H hat „Spiel“ im Korridor).
- Weiterhin existieren dann weitere „gedrehte“ Korridore, die H enthalten (Abbildung 8.13).
- Gibt es neben der minimalen Breite $d(w_{i,i+1}, v_i v_{i+1})$ weitere Eckpunkte mit $d(w_{k,k+1}, v_k v_{k+1}) \leq 2\varepsilon$, so existiert parallel zu $v_k v_{k+1}$ selbstverständlich ebenfalls ein Korridor, der H enthält (Abbildung 8.14).

Hierbei kann man zwei alternative Lagen eines Korridors nicht immer durch eine Drehung so ineinander überführen, daß H dabei die ganze Zeit innerhalb des Korridors liegt. Dies gilt selbst dann, wenn (wie in Abbildung 8.13) die beiden Lagen zu benachbarten Kanten $v_{i-1}v_i, v_i v_{i+1}$ von H gehören.

Ist nun v_l der zu $v_{i-1}v_i$ und v_k der zu v_iv_{i+1} gehörige antipodische Punkt (also $d(v_l, v_{i-1}v_i) \leq 2\varepsilon$ und $d(v_k, v_iv_{i+1}) \leq 2\varepsilon$), so gilt: Zu den zwischen v_k und v_l liegenden Kanten $v_kv_{k+1}, v_{k+1}v_{k+2}, \ldots, v_{l-1}v_l$ ist v_i antipodischer Punkt. Gilt nun weiter:

$$\forall v_m \in \{v_k, v_{k+1}, \ldots v_{l-1}\} : d(v_i, v_mv_{m+1}) \leq 2\varepsilon,$$

so ist die Drehung von $v_{i-1}v_i$ nach v_iv_{i+1} möglich.

Die möglichen Lagen des 2ε-Korridors sind also effizient berechenbar:

- Die Breite von H bezüglich der unterschiedlichen v_iv_{i+1} liegt als sortierte Liste vor.
- Somit sind Bereiche v_j bis v_k bestimmbar, in denen der Korridor drehbar und verschiebbar ist.
- Die möglichen Positionen des Korridors sind ebenfalls effizient bestimmbar. Sie sind innerhalb jedes der oben ermittelten Bereiche sowohl in Translation als auch in Rotation zusammenhängend.
- Folglich ist die Menge der möglichen Lagen des zu H gehörenden Streckenstücks effizient bestimmbar.

Da hier allerdings einige Freiheitsgrade bestehen, ist die Auswahl des für eine Generalisierung *optimalen* Streckenstücks keineswegs trivial und meist nicht effizient möglich. Hier bietet es sich an, die in 8.2.3 beschriebene Auswahl zu verwenden.

8.3. Zusammenbau

Das obige Verfahren bestimmt eine Reihe von Segmenten $S_1, S_2, \ldots S_n$. Zu jedem Segment S_i existiert (mindestens) eine Strecke s_i, für die die ε-Bedingung erfüllt ist. Allerdings ist eine geordnete Menge von Strecken noch nicht automatisch ein Polygonzug.

Für viele Aufgaben ist diese Beschreibung hinreichend und sinnvoll. Soll z. B. überprüft werden, ob zwei Trajektorien p, q einander ähnlich sind, ist es völlig ausreichend, eine Segmentierung von p zu bestimmen und iterativ zu überprüfen, ob q in der ε-Umgebung der Segmente von p liegt.

Um eine Generalisierung nach der vorne gemachten Definition zu bekommen, muß man die Streckenstücke noch zu einem Polygonzug verbinden. Hierzu betrachtet man die zu den Strecken s_i zugehörigen Geraden g_i ($s_i \in g_i$). Die Schnitte dieser Geraden liefern die Eckpunkte des Polygonzuges: $p_2 = g_1 \cap g_2, p_3 = g_2 \cap g_3, \ldots, p_n = g_{n-1} \cap g_n$, wobei p_1 der Startpunkt von s_1 und p_{n+1} der Endpunkt von s_n ist. Abbildung 8.15 zeigt die Konstruktion des Polygonzuges $q = \mathbf{abcd}$.

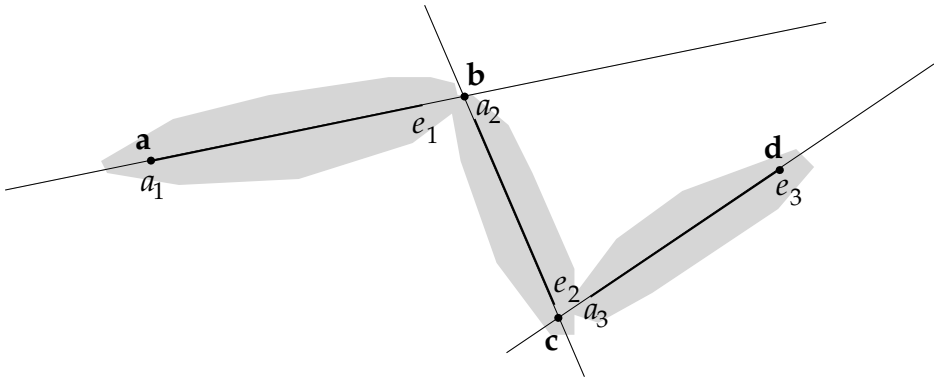


Abbildung 8.15.: Zusammenstecken des Polygonzuges

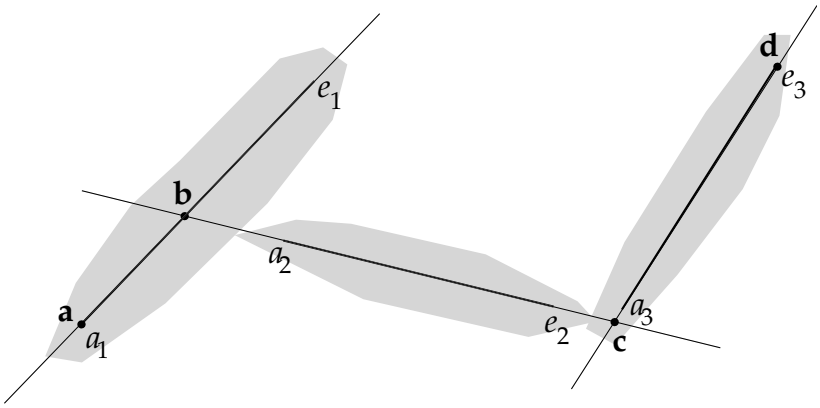


Abbildung 8.16.: Probleme mit Rückläufigkeiten

Wie schon vorne erwähnt, hat man bei der Bestimmung der Segmente die Wahl, ob man den letzten Punkt v_k im Segment S_{i-i} als ersten Punkt im Segment S_i wählt oder ob man das Segment S_i mit dem Punkt v_{k+1} beginnt. Im ersten Fall überlappen die Segmente und damit auch die zugehörigen konvexen Hüllen, im zweiten Fall nicht. Der Zusammenbau funktioniert für beide Segmentierungen gleich. Die Graphiken in diesem Abschnitt basieren auf nicht überlappenden Segmentierungen, da dann die zugehörigen konvexen Hüllen in den meisten Fällen nicht überlappen, wodurch die Darstellungen klarer werden.

8.3.1. Probleme

Leider klappt dies nicht immer. Wie bei der inkrementellen ε-Generalisierung (und auch der Douglas/Peucker-Generalisierung) können auch hier Rückläufigkeiten und andere Artefakte auftreten. Zum einen kann es passieren, daß der Schnittpunkt $r = g_i \cap g_{i+1}$ auf g_i vor dem Endpunkt e_i von s_i oder auf g_{i+1} nach dem Startpunkt

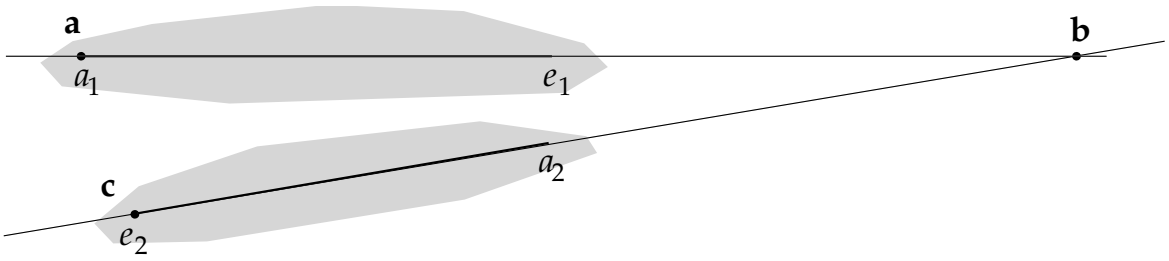


Abbildung 8.17.: Probleme mit Überständen

a_i von s_{i+1} liegt. Abbildung 8.16 zeigt das Problem.

Zum anderen kann der Schnittpunkt zweier Geraden weit außerhalb der gemessenen Punkte liegen (Abbildung 8.17). Das so entstehende Polygon suggeriert also einen Trajektorienverlauf, der in dieser Form nicht stattgefunden hat: das Polygon „übersteuert“.

Dieser Fehler kann etwas dadurch gemildert werden, daß bei der Segmentierung der letzte in H_i liegende Punkt als erster zur Berechnung von H_{i+1} benutzter Punkt dient. Dadurch wird die Neigung zum Übersteuern zwar gedämpft aber nicht ausgeschlossen.

Die triviale Lösung, einfach durchgängig End- und Anfangspunkte der einzelnen Strecken zu verbinden (also $q = a_1e_1a_2e_2 \ldots a_ne_n$) funktioniert zwar, verdoppelt aber die Anzahl der Punkte des Polygons, führt also zu einer ziemlich schlechten Generalisierung. Daher bietet sich ein Kompromiß an:

Liegt der Schnittpunkt $r = g_i \cap g_{i+1}$ innerhalb der Halbgeraden $a_ie_i]$ oder $[a_{i+1}e_{i+1}$, oder liegt r zu weit außerhalb ($\min\{d(r,e_i), d(r,a_{i+1})\} > \delta$, wobei δ eventuell abhängig von $\angle g_ig_{i+1}$ gewählt wird), so wird ein zusätzliches Segment eingefügt, sonst nicht.

Der Algorithmus liefert aufgrund seiner Struktur keine Informationen über Schleifen und Rückläufigkeiten, die innerhalb eines Segmentes auftreten, die Schleifenerkennung beschränkt sich im Gegensatz zu den bislang beschriebenen Algorithmen daher auf beim Zusammenbau der Segmente auftretende Schleifen.

8.3.2. Übersteuern

Das in Abbildung 8.17 gezeigte Problem tritt vor allem bei sehr spitzen Winkeln zwischen g_i und g_{i+1} auf. Um es in den Griff zu bekommen, benötigt man zunächst ein Maß U für die Stärke des Übersteuerns und weiterhin einen Schwellwert δ, den U nicht übersteigen darf. Um $\angle g_ig_{i+1}$ in U einfließen zu lassen, bietet sich an, $U := d(r, [e_ia_{i+1}])$ zu wählen. Hierdurch wird auch Trajektorien mit geringer Abtastrate Rechnung getragen. Ein winkelunabhängiges Maß ist hingegen $U' := \min\{d(r,e_i), d(r,a_{i+1})\}$.

Der Schwellwert δ ist abhängig von der konkreten Aufgabenstellung und

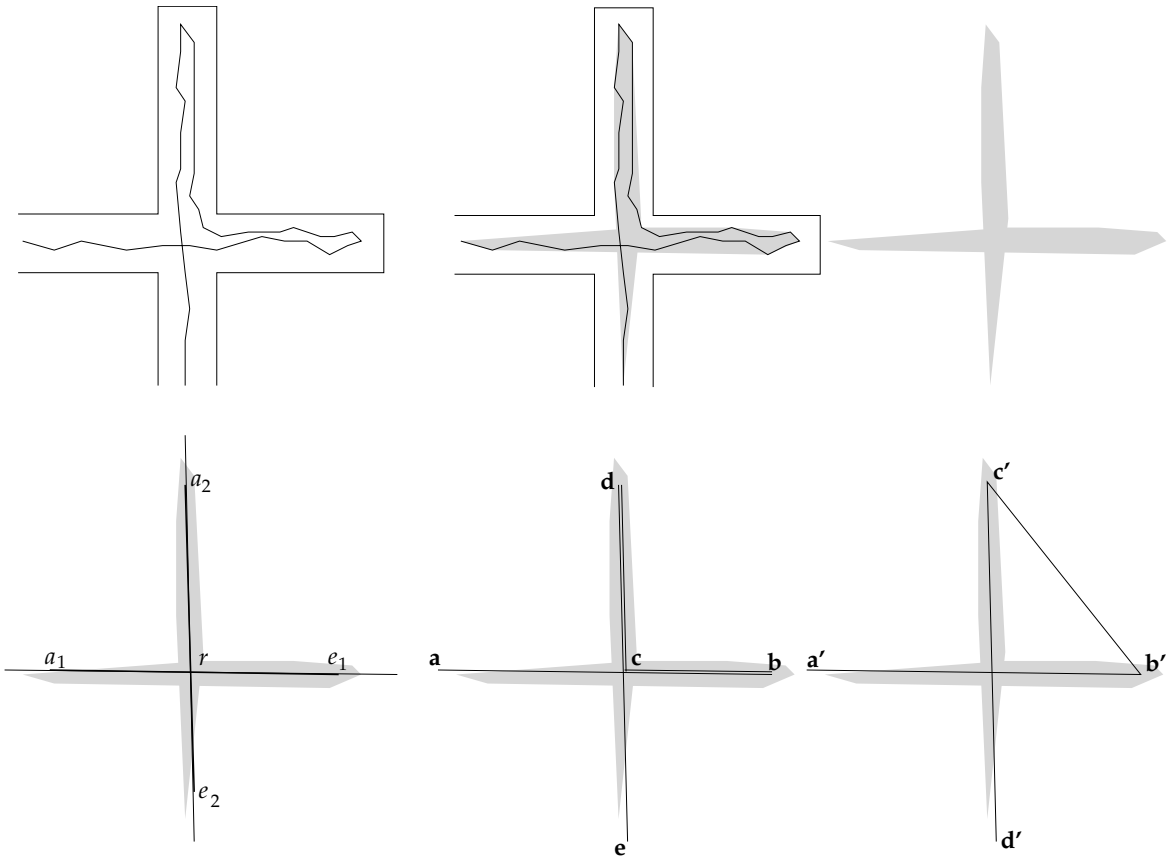


Abbildung 8.18.: Alternative Lösungen für Rückläufigkeiten

eventuell ebenfalls von der Abtastrate. Als Basis empfiehlt sich eine Wahl von $\delta := \varepsilon$ oder $\delta := 2\varepsilon$.

Findet nun an einer Stelle ein Übersteuern statt, wird dies durch Einsetzen der zusätzlichen Strecke $[e_i, a_{i+1}]$ behoben.

8.3.3. Rückläufigkeiten

Nun zum Problem der rückläufigen Bewegung. Seien wie bisher

$$g_i = a_i e_i, \quad g_{i+1} = a_{i+1} e_{i+1}, \quad r = g_i \cap g_{i+1}, \quad s_i = [a_i e_i], \quad s_{i+1} = [a_{i+1} e_{i+1}].$$

Liegt nun der Schnittpunkt $r \in a_i e_i]$ (oder alternativ $r \in [a_{i+1} e_{i+1})$, so bieten sich zwei mögliche Verbindungsstrecken für den Zusammenbau des Polygons an:

$[e_i r]$**:** Der Endpunkt von s_i wird mit dem Schnittpunkt der Geraden verbunden.

$[e_i a_{i+1}]$**:** Der Endpunkt von s_i wird mit dem Startpunkt von s_{i+1} verbunden.

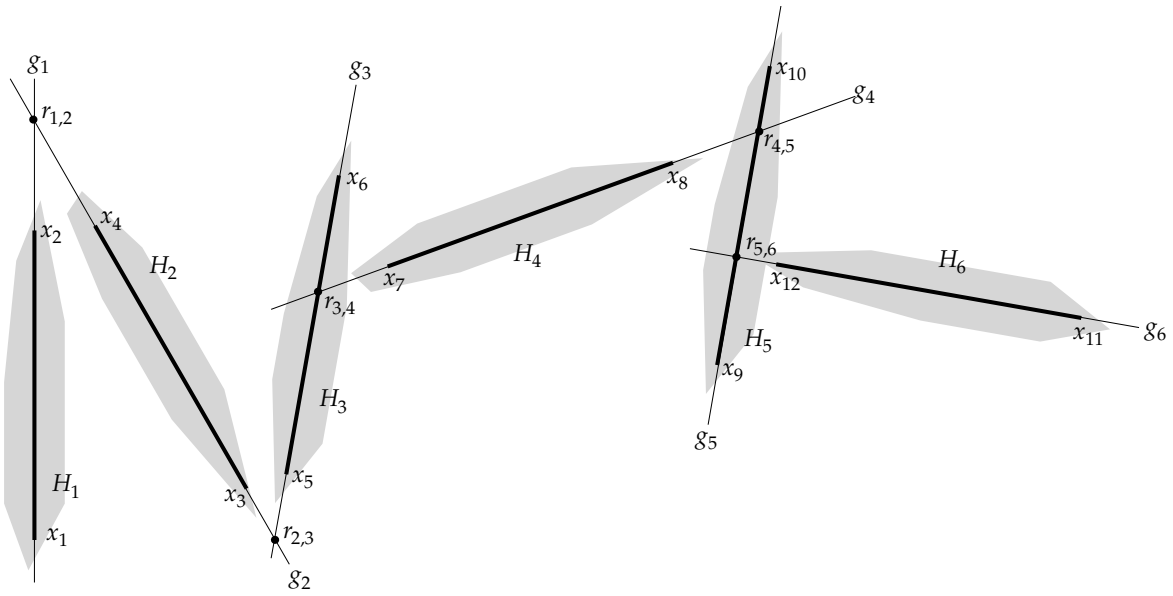


Abbildung 8.19.: Orientierung der einzelnen konvexen Hüllen beim Zusammenbau

Beide Verfahren unterscheiden sich vor allem dann, wenn sich beide Strecken überschneiden, also $r \in a_i e_i] \wedge r \in [a_{i+1} e_{i+1}$.

In Abbildung 8.18 zeigt sich der Unterschied zwischen beiden Ansätzen. Dargestellt ist die Trajektorie einer Bewegung zunächst in einen Korridor hinein, als sich dieser als Sackgasse erweist, aus diesem hinaus, in den nächsten Korridor hinein, und wie dort ebenfalls kein Durchkommen ist, weiter in die letzte verbleibende Alternative. Die Bewegung hat wie auch der Korridor selbst die Form eines Kreuzes. Dies spiegelt sich sowohl in der Form der konvexen Hüllen H_1 und H_2 als auch in den zugehörigen Strecken s_1 und s_2 wieder. Wird nun der Schnittpunkt r zur Bildung der Generalisierung benutzt ($q = \mathbf{abcde}$), so werden 2 zusätzliche Strecken ($[e_1 r]$ und $[r a_2]$) eingefügt, wodurch die Form der Bewegung erhalten bleibt. Werden dagegen s_1 und s_2 direkt verbunden ($q = \mathbf{a'b'c'd'}$), benötigt man zwar nur eine zusätzliche Strecke, allerdings wird die Form der Bewegung zerstört, die Bewegung führt mitten durch die Korridorwand.

8.3.4. Wo ist vorne?

In den obigen Beispielen zum Zusammenbau der Segmente wurden die zu den konvexen Hüllen H_i gehörigen minimalen Strecken $[a_i e_i]$ benutzt. Die Strecken sind wie in 8.2.3 beschrieben zu bestimmen. Was dabei aber nicht klar ist: Welcher Endpunkt ist a_i und welcher e_i? Konvexe Hüllen haben zwar eine Breite, aber keine Orientierung. Abbildung 8.19 zeigt 6 Segmente eines Polygonzuges (konvexe Hüllen H_1 mit H_6 sowie die Schnittpunkte der zugehörigen Geraden. g_1 und g_2 schneiden

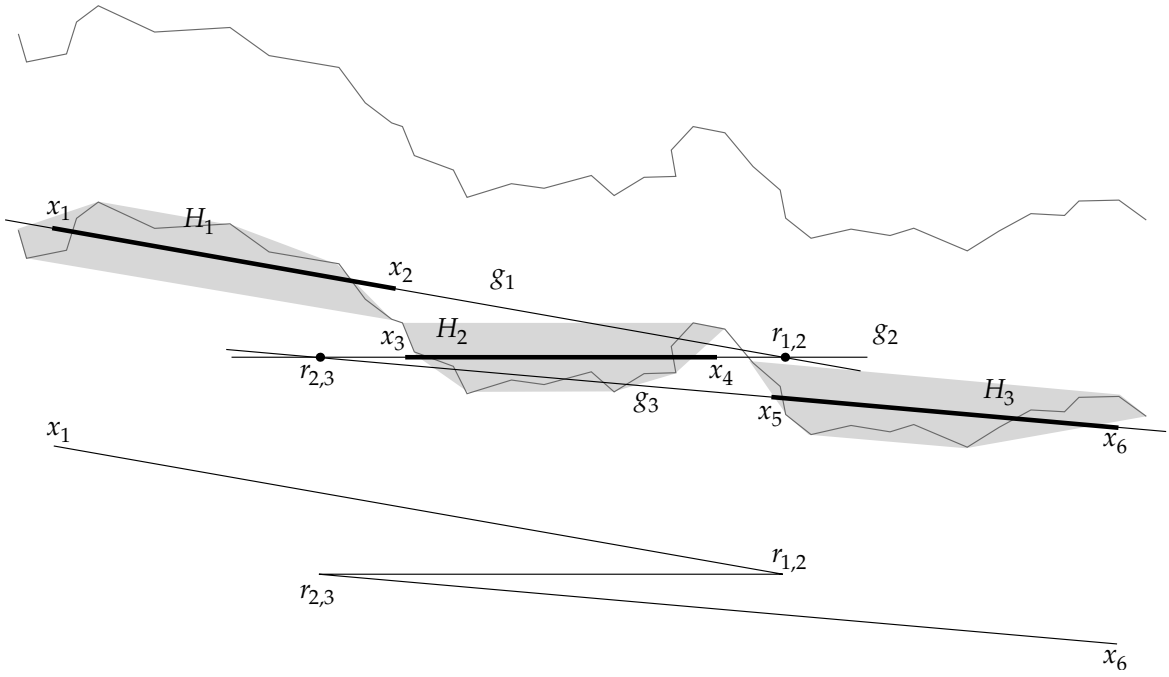


Abbildung 8.20.: Fehlerhafter Zusammenbau

sich in $r_{1,2}$, g_2 und g_3 in $r_{2,3}$. Da auf den Segmenten (und somit auf den Geraden) eine Ordnung existiert, ist klar, daß $r_{1,2}$ im generalisierten Polygonzug *vor* $r_{2,3}$ liegt. Daraus läßt sich folgern, daß $a_2 = x_4$ und $e_2 = x_3$ (auch wenn diese Werte hier nicht mehr benötigt werden). Ebenso läßt sich daraus für das Startsegment schließen, daß $a_1 = x_1$ und $e_1 = x_2$ (da $r_{1,2}$ „hinter" e_1 liegen soll), womit x_1 Startpunkt des generalisierten Polygonzuges ist.

Im dritten Segment liegt der Schnittpunkt $r_{3,4}$ innerhalb der minimalen Strecke zu H_3, gibt also keine Auskunft darüber, wie x_5 und x_6 zu sortieren sind. Hier reicht jedoch die Information, daß $r_{2,3}$ auf g_3 „vor" x_5 liegt, um $a_3 = x_5$ und $e_3 = x_6$ zu bestimmen.

Unklar sind nun die Fälle, in denen beide Schnittpunkte $r_{i-1,i}$ und $r_{i,i+1}$ innerhalb der zugehörigen minimalen Strecke $[x_k x_{k+1}]$ ($i = \lceil \frac{k}{2} \rceil$) oder beide auf der gleichen Seite dieser Strecke liegen, im Beispiel: Segment 5. In diesem Fall scheint die Wahl $a_5 = x_{10}$ und $e_5 = x_9$ darin begründet, daß dann $r_{4,5}$ und $r_{5,6}$ auf g_5 in gleicher Weise angeordnet sind wie a_5 und e_5.

Diese Regel gilt allgemein und unabhängig davon, ob die Schnittpunkte $r_{i-1,i}$ und $r_{i,i+1}$ nun innerhalb oder außerhalb der minimalen Strecke liegen:[10] wähle a_i und e_i so, daß sie auf g_i in gleicher Weise angeordnet sind wie $r_{i-1.i}$ und $r_{i,i+1}$. Eine Sonderbehandlung benötigen lediglich das erste und letzte Segment, da hier nur

[10] Im seltenen Sonderfall $r_{i-1.i} = r_{i,i+1}$ können a_i und e_i beliebig gewählt werden, hier ist die Anordnung unwichtig.

ein Schnittpunkt zur Verfügung steht: Der näher am Schnittpunkt liegende Endpunkt der Strecke wird e_1 bzw a_m, der weiter entfernt liegende a_1 bzw. e_m.

Leider ist diese Regel in bestimmten Fällen fehlerhaft, wie Abbildung 8.20 zeigt: der Originalpolygonzug wird hier in drei Teile segmentiert, die Schnittpunkte der zugehörigen Geraden g_1, g_2 und g_3 liegen wie gewünscht außerhalb der Strecken $[x_1x_2]$, $[x_3x_4]$ und $[x_5x_6]$. Leider sieht die gewonnene Generalisierung $x_1r_{1,2}r_{2,3}x_6$ dem Originalpolygonzug nicht sehr ähnlich. Sie ist schlicht falsch.

Dies rührt daher, daß bei sehr stumpfen Winkeln (wenn also beide Geraden fast parallel liegen) eine sehr kleine Veränderung in der Neigung eine sehr große Verschiebung des Schnittpunktes verursacht. Die Lage von $r_{1,2}$ und $r_{2,3}$ ist stark abhängig von der der konvexen Hüllen H_1 und H_3 bzw. von der Neigung der Gerade g_1 und g_3 bezüglich g_2. Daher ist ein von $r_{1,2}$ und $r_{2,3}$ unabhängiges Maß besser geeignet.

Die Orientierung der Strecke $[a_ie_i]$ soll den mittleren Verlauf der Bewegung innerhalb von H_i wiedergeben. Damit ist die Lage des ersten und letzten Punktes des durch H_i überdeckten Segmentes des Originalpolygonzuges ein guter Anhaltspunkt: Sei H_i die konvexe Hülle der Punkte v_j mit v_k. Dann wähle a_i und e_i so, daß sie auf $g_i = \overline{a_ie_i}$ in gleicher Weise angeordnet sind, wie die Fußpunkte des Lotes von v_j und v_k auf g (oder gleichbedeutend damit: $|\angle((e_i - a_i), (v_k - v_j))| \leq 180°$). Somit bestimmt sich die Orientierung von H_i unabhängig von der Lage des vorhergehenden oder nachfolgenden Segmentes.

8.3.5. Stumpfe Rückläufigkeiten

Neben den gezeigten großen Rückläufigkeiten treten bei eher runden Bewegungsverläufen häufig Artefakte auf, die nach dem oben eingeführten Kriterium wie Rückläufigkeiten wirken, aber häufig keine sind. Abbildung 8.21 zeigt die Segmentierung eines Polygonzuges in ein erstes Segment v_1 mit v_{13} und ein zweites Segment v_{14} mit v_{22}. Zu beiden Segmenten werden nun die zugehörigen Geraden bestimmt. Wie man sieht schneidet die Gerade g_s die Gerade g_1 innerhalb des Teilstücks $[a_1e_1]$, was eigentlich auf eine Rückläufigkeit hinweist.[11] Wie das Einzeichnen des ε-Korridors zeigt, liefert der Polygonzug a_1re_2 aber durchaus eine gültige Generalisierung. Daß diese nicht erkannt wird, liegt daran, daß die zum ersten Segment gehörigen Punkte v_{12} und v_{13} nicht mehr in der ε-Umgebung zu $[a_1r]$ liegen, sondern durch die ε-Umgebung von $[re_2]$ abgedeckt werden.

Die Erkennung von Rückläufigkeiten muß also verfeinert werden. Interessant ist hierbei der Fall, daß der Schnittpunkt r beider Geraden g_1 und g_2 innerhalb der minimalen Strecke $[a_ie_i]$ einer konvexen Hülle (aber nicht beider) liegt. Sei oBdA $r \in [a_1e_1]$. Die zu klärende Frage ist nun: Kann die Strecke $[a_1e_1]$ zu $[a_1r]$ verkürzt

[11] Im gezeigten Beispiel ist der Endpunkt des ersten Segmentes *nicht* Startpunkt des zweiten Segmentes. Das gezeigte Problem kann aber ebenso auftreten, wenn dies der Fall ist.

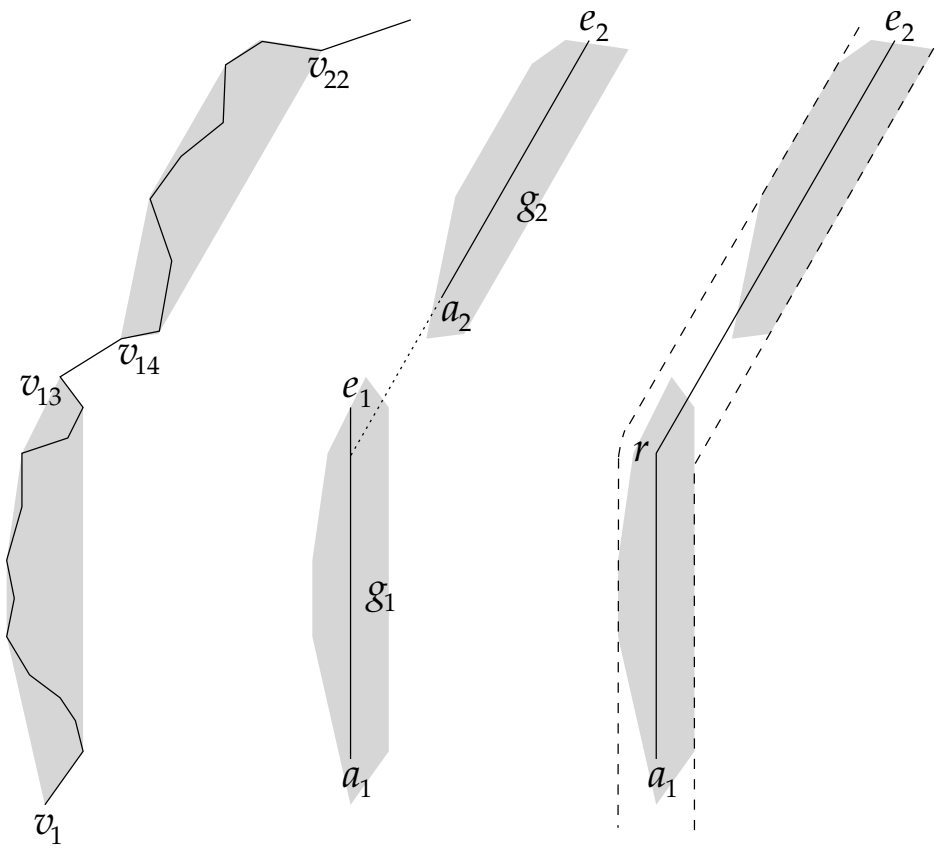


Abbildung 8.21.: Fälschlicherweise erkannte Probleme beim Zusammenbau

werden, ohne daß dadurch Punkte aus der ε-Umgebung herausragen?

Die von der Verkürzung der Strecke betroffenen Punkte sind leicht zu ermitteln (Punkte im Inneren der Hülle können dabei vernachlässigt werden, liegen die Randpunkte innerhalb der ε-Umgebung, dann auch die komplette Hülle): Abbildung 8.22 zeigt den Bereich um den Schnittpunkt r der beiden Geraden g_1 und g_2. Um zu testen, ob die Verkürzung von $[a_1e_1]$ auf $[a_1r]$ möglich ist, muß lediglich überprüft werden, ob irgendwelche Punkte aus der konvexen Hülle H_1 in den Bereich A ragen. Ist $H_1 \cap A = \emptyset$, so kann verkürzt werden.

Zunächst werden alle Punkte v_i auf dem Rand von H_1 ausgewählt, für die der Fußpunkt ihres Lotes auf g_1 innerhalb der Strecke $[re_1]$ liegt (der erste Punkt kann in logarithmischer Zeit gefunden werden, alle weiteren Punkte liegen benachbart). Gilt nun für alle diese Punkte: $\text{dist}([re_2], v_i) \leq \varepsilon$, so liegt keiner von ihnen in A und die Verkürzung ist möglich. Durch diesen einfachen Test kann ein Großteil der auftretenden Rückläufigkeiten vermieden werden, was die Qualität der Generalisierung stark verbessert.

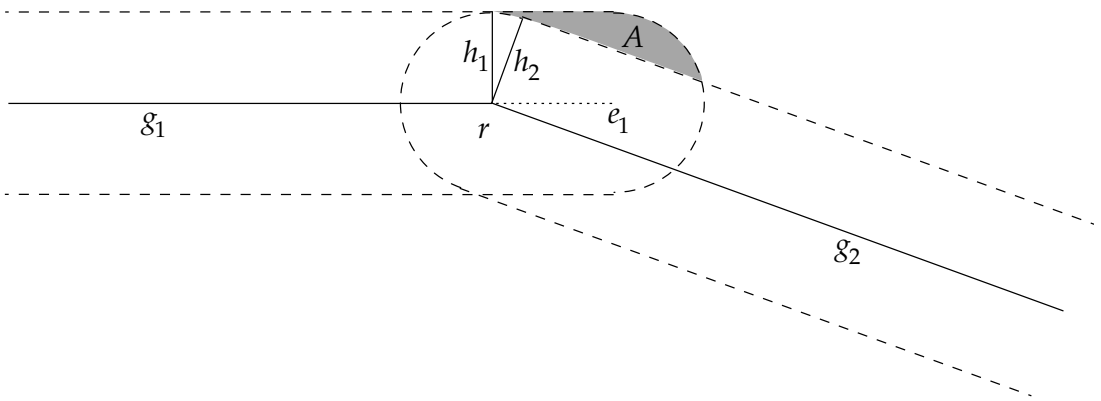


Abbildung 8.22.: Überprüfung des ε-Korridors

8.4. Minimalität

8.4.1. Eine minimale Segmentierung

Der beschriebene Algorithmus liefert eine Segmentierung eines gegebenen Polygonzuges p in Teilstücke $\{p_1, p_2, \ldots p_m\}$, so daß zu jeder p_i eine Strecke s_i besteht, zu der p_i einen Abstand kleinergleich ε hat ($\forall a \in p_i : d(a, s_i) \leq \varepsilon$). Diese Segmentierung ist minimal.

Beweis. Eine Segmentierung ist minimal, wenn keine andere Unterteilung des Polygonzuges p in Teilstücke $\{p'_1, p'_2, \ldots p'_{m'}\}$ mit $m' < m$ existiert. Angenommen es gäbe eine Segmentierung, die p in $m' < m$ Segmente zerlegt und die ε-Bedingung erfüllt. Dann muß es ein Teilstück p'_i geben, das nicht mit dem Teilstück p_i der oben gefundenen Segmentierung $\{p_1, p_2, \ldots p_m\}$ übereinstimmt ($p'_i \neq p_i$). Sei p'_k das erste solche Teilstück:

$$\forall i \in \{1, \ldots k-1\} : p_i = p'_i \qquad \wedge \qquad p_k \neq p'_k.$$

Die konvexe Hülle H'_k kann nun keine Punkte enthalten, die nicht auch in H_k liegen ($H'_k \subsetneq H_k$). Folglich gibt es in H_k mindestens einen Punkt, der nicht mehr in H'_k liegt. Sei a_j der erste Punkt aus H_k, der nicht mehr in H'_k liegt. Damit liegt a_j also in H'_{k+1}. Somit besitzt H'_{k+1} also mindestens einen Punkt, der nicht in H_{k+1} liegt.

Sei nun a_l der letzte Punkt in H_{k+1}, also $a_{l+1} \notin H_{k+1}$. Dann kann a_{l+1} auch nicht in H'_{k+1} liegen: a_{l+1} liegt nach der beschriebenen Konstruktion *nicht* in H_{k+1}, da H_{k+1} um a_{l+1} erweitert breiter als 2ε ist. Läge a_{l+1} in H'_{k+1} so enthielte H'_{k+1} den Punkt a_j, alle Punkte aus H_{k+1} und zusätzlich a_{l+1}. Durch Erweiterung von H_{k+1} um a_j kann H_{k+1} aber nicht schmaler (höchstens breiter) werden. Somit muß H'_{k+1} erweitert um a_{l+1} mindestens so breit sein, wie H_{k+1} um a_{l+1} erweitert, also breiter als 2ε.

Ist in der Segmentierung $\{p_1, p_2, \ldots p_m\}$ der Punkt b_i der Endpunkt zum Teilstück p_i, so kann also in keiner Segmentierung $\{p'_1, p'_2, \ldots p'_{m'}\}$ das Teilstück p'_i den

Punkt b_{i+1} oder nachfolgende Punkte enthalten. Somit muß jede vollständige Segmentierung von p mindestens aus m Teilstücken bestehen. □

Für das so erzeugte generalisierte Polygon q muß diese Minimalität nicht gelten. Sind zur Korrektur von Rückläufigkeiten oder Übersteuern viele zusätzliche Segmente nötig, so kann es andere Segmentierungen von p geben, deren generalisiertes Polygon q' weniger Ecken enthält. Ebenso garantieren die beschriebenen Korrekturmechanismen nicht, daß der entstehende Polygonzug minimal ist. Somit gilt: Der beschriebene Algorithmus liefert eine minimale Sequenz von Gangstücken, die die Bewegungsspur überdecken, *nicht* aber zwingend einen minimalen Polygonzug.

8.4.2. Alle minimalen Segmentierungen

Neben der oben gefundenen Segmentierung können noch weitere Segmentierungen mit m Teilstücken existieren. Die Menge aller möglichen Segmentierungen läßt sich hierbei mit einigem Aufwand bestimmen.

Der oben beschriebene Algorithmus liefert die Segmentierung eines Polygonzuges $p = v_1 v_2 \dots v_n$ in m Teilstücke:[12]

$$(v_{l_1} \dots v_{r_1})\ (v_{l_2} \dots v_{r_2})\ \dots\ (v_{l_m} \dots v_{r_m}) \qquad \text{mit} \quad v_{l_1} = v_1, \quad v_{r_m} = v_n, \quad v_{r_i+1} = v_{l_{i+1}}.$$

Durchläuft man mit dem gleichen Algorithmus den Polygonzug rückwärts, enthält man eine Segmentierung

$$(v_{l'_1} \dots v_{r'_1})\ (v_{l'_2} \dots v_{r'_2})\ \dots\ (v_{l'_m} \dots v_{r'_m}) \qquad \text{mit} \quad v_{l'_1} = v_1, \quad v_{r'_m} = v_n, \quad v_{r'_i+1} = v_{l'_{i+1}},$$

wobei gilt: $l'_i \le l_i < l'_{i+1}$ (und damit auch $r_i \ge r'_i > r_{i-1}$).

Das bedeutet für einen Punkt $v_{L_2} \in \{v_{l'_2}, \dots, v_{l_2}\}$: Das Segment $(v_1 v_{L_2-1})$ ist gültig und der Polygonzug $p_{L_2,n} = v_{L_2} \dots v_n$ läßt sich in $m-1$ Teilstücke segmentieren. Ebenso gilt für einen Punkt $v_{L_3} \in \{v_{l'_3}, \dots, v_{l_3}\}$, daß der Polygonzug p_{1,L_3-1} sich in 2 Teilstücke und der Polygonzug $p_{L_3,n}$ sich in $m-2$ Teilstücke segmentieren läßt usw., allgemein: für einen Punkt $v_{L_k} \in \{v_{l'_k}, \dots, v_{l_k}\}$ existiert für den ersten Teilpolygonzug p_{1,L_k-1} eine Segmentierung in $k-1$ Teilstücke und für den zweiten Teilpolygonzug $p_{L_k,n}$ eine Segmentierung in $m-k+1$ Teilstücke.

Leider sind die einzelnen v_{L_k} nicht unabhängig voneinander wählbar. Zu einem Punkt v_{L_k} existiert ein maximaler Punkt v_{R_k}, so daß für $v_{L_k}, v_{L_k+1}, \dots, v_{R_k}$ eine konvexe Hülle der Breite kleinergleich 2ε existiert (und für v_{R_k+1} nicht mehr). Damit kann der Startpunkt des nächsten Segmentes $v_{L_{k+1}}$ für eine minimale Segmentierung nun aber nur noch aus der Menge $\{v_{l'_{k+1}}, \dots, v_{R_k+1}\}$ gewählt werden.

[12] Die einzelnen Segmente überlappen hier nicht. Die folgenden Überlegungen sind auch für eine überlappende Segmentierung gültig, bei der der letzte Punkt eines Segmentes gleichzeitig Startpunkt des nächsten ist, hier gilt dann $v_{r_i} = v_{l_{i+1}}$.

Um also nun die Menge aller minimalen Segmentierungen zu bekommen, muß zu jedem $v_{L_k} \in \{v_{l'_k}, \ldots, v_{l_k}\}$ das zugehörige v_{R_k} bestimmt werden. Dies ist zwar einfach möglich, indem von v_{L_k} ausgehend die konvexe Hülle aufgebaut wird, um v_{R_k} zu bestimmen, aber auch aufwendig, wenn man bedenkt, daß dies für jedes mögliche v_{L_k} gemacht werden muß. Da diese konvexen Hüllen sich aber stark überlappen, lassen sie sich effizienter berechnen.

Die konvexe Hülle H_{l_k,r'_k} der Punkte $v_{l_k}, \ldots, v_{r'_k}$ ist sicher Teil aller dieser konvexen Hüllen. Beim Rückwärtsdurchlaufen des Polygonzuges werden hier darauf aufbauend die konvexen Hüllen H_{l_k-j,r'_k} der Punkte $v_{l_k-1}, \ldots, v_{r'_k}$, $v_{l_k-2}, \ldots, v_{r'_k}$ bis $v_{l'_k}, \ldots, v_{r'_k}$ wie oben beschrieben bestimmt und als Stack gespeichert (dies kann sehr effizient geschehen, da sie sich nur in wenigen Punkten unterscheiden). Nun wird das oberste Element $H_{l'_k,r'_k}$ vom Stack geholt, um den Punkt $v_{r'_k+1}$ erweitert, und getestet, ob $H_{l'_k,r'_k+1}$ ein erlaubtes Segment (Breite kleinergleich 2ε) darstellt, wenn ja, wird um $v_{r'_k+2}$ erweitert und getestet usw. Können zur so erzeugten konvexen Hülle $H_{l'_k,r'_k+j}$ keine weiteren Punkte mehr angefügt werden, sind damit die Punkte $r'_k, \ldots, r'_k + j$ als mögliche Endpunkte des Segmentes zu l'_k gefunden.

Nun wird das nächste Element $H_{l'_k+1,r'_k}$ vom Stack geholt, um den Punkt $v_{r'_k+1}$ erweitert, getestet usw., um die Endpunkte $r'_k, \ldots, r'_k + j'$ ($j' \geq j$) zu $v_{l'_k+1}$ zu erhalten, bis am Ende zu jedem Punkt v_{L_k} die Menge seiner möglichen Endpunkte (die noch eine minimale Segmentierung erlauben) bekannt sind.[13]

Damit läßt sich (analog zu Abschnitt 6.7) ein (gerichteter) Graph aufbauen, der alle minimalen Segmentierungen als Pfade enthält. Aus diesem Graph läßt sich nun beispielsweise ein Pfad auswählen, der keine (oder zumindest die wenigsten) Probleme beim Zusammenbau der einzelnen Segmente zu einem Polygonzug hat, womit man im besten Fall eine minimale Generalisierung bekommt. Allerdings ist der Aufwand zur Erzeugung dieses Graphen wie gezeigt relativ groß, so daß er in den meisten Fällen nicht lohnt.

8.5. Convex Hull im Dreidimensionalen

Der Grundidee nach funktioniert der beschriebene Algorithmus auch für Bewegungen (Polygonzüge) im dreidimensionalen Raum. Leider treten hier aber Probleme

[13] Auch hier wird noch einige Arbeit doppelt erledigt. Da die konvexen Hüllen über den Eckpunkten eines Polygonzuges (eines Bewegungsverlaufes) aufgespannt werden, ist die Wahrscheinlichkeit hoch, daß die Erweiterung von $H_{l'_k,r'_k}$ um den Punkt $v_{r'_k+1}$ mit der gleichen Modifikation erfolgt wie die Erweiterung von $H_{l'_k+1,r'_k}$ um den selben Punkt (beide konvexen Hüllen haben bezüglich $v_{r'_k+1}$ die gleichen Stützstellen). Speichert man sich also zusätzlich zu jedem Punkt die gefundenen Stützstellen, beschleunigt dies das Verfahren zusätzlich. Es sind noch weitere Modifikationen denkbar, dabei ist jedoch fraglich, ob der Laufzeitgewinn den zusätzlichen Aufwand wert ist.

auf, die ihn in der Praxis (zumindest in dieser Form) als ungeeignet erscheinen lassen:

- Der Aufwand, einen neuen Punkt zu einer dreidimensionalen konvexen Hülle hinzuzufügen, steigt, das vorne beschriebene Verfahren funktioniert nicht mehr.[14]
- Statt der Korridorbreite muß ein Zylinderdurchmesser bestimmt werden. Gesucht ist jetzt der Zylinder mit dem geringsten Durchmesser, der die konvexe Hülle vollständig enthält. Auch hier sind neue Ansätze nötig.
- Im Zusammenbau der einzelnen Strecken gibt es (fast) keine Schnittpunkte mehr, da fast alle Geraden im dreidimensionalen zueinander windschief sind.

Dies führt dazu, daß die Komplexität des Algorithmus pro Segment ansteigt (die Komplexität bezogen auf die Gesamtlänge bleibt linear) und die Qualität des Zusammenbaus sinkt, da viele Verbindungsstücke eingefügt werden müssen.

8.6. Direkter Zusammenbau

Während ConvexHull einen Polygonzug einfach und elegant segmentiert, ist der Zusammenbau dieser Segmente eher unbefriedigendes Stückwerk, was daher rührt, daß die einzelnen Segmente unabhängig voneinander erzeugt werden. Berücksichtigt man dagegen schon während der Segmentierung den späteren Zusammenbau, so treten die ganzen Probleme von vornherein nicht auf. Hierzu muß bei der Konstruktion einer konvexen Hülle darauf geachtet werden, daß sie nur so lange erweitert wird, wie ein Zusammenbau ohne Einfügen von Zwischenstücken möglich ist.

8.6.1. Vorüberlegungen

Definition 17 *Die durch zwei parallele Geraden $t' \parallel t''$ beschränkte Fläche $K_{t',t''}$ heißt Korridor mit Wänden t' und t'': $K_{t',t''} = A_1 \cap A_2$, wobei A_1 die durch t' beschränkte Halbebene ist, die t'' enthält und A_2 die durch t'' beschränkte Halbebene, die t' enthält.*

Definition 18 *Ein Korridor $K_{t',t''}$ schneidet eine Fläche H einseitig, wenn*

$$(H \subset A_1) \vee (H \subset A_2).$$

Dies ist gleichwertig damit, daß höchstens eine der beiden Korridorwände Sekante von H ist (die andere ist höchstens Tangente von H oder berührt H gar nicht).

[14] Zur effizienten Berechnung einer komplexen Hülle im mehrdimensionalen Raum siehe z. B. [BDH96]

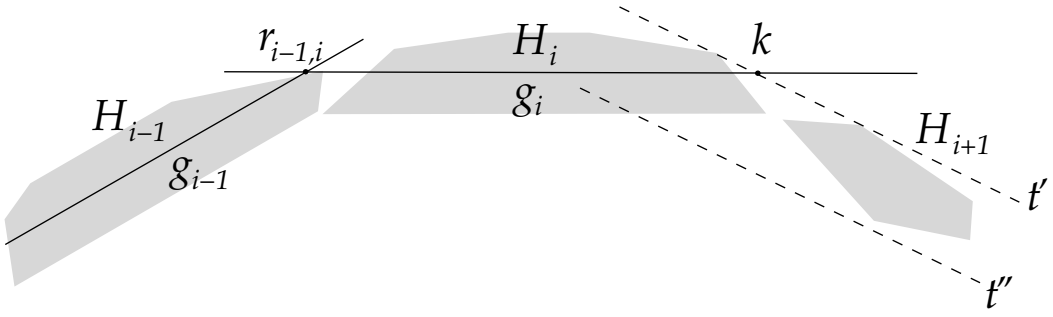


Abbildung 8.23.: Inkrementeller Zusammenbau

Definition 19 *Ein Korridor $K_{t',t''}$ schneidet eine Fläche H beidseitig, wenn*

$$(H \cap A_1 \neq H) \wedge (H \cap A_2 \neq H).$$

Beide Korridorwände sind Sekanten von H.

Definition 20 *Die Breite b_K eines Korridors $K_{t',t''}$ ist der Abstand der Geraden t' und t''.*

Die Breite eines Korridors $K_{t',t''}$ ist damit gleich der Breite des breitesten Rechtecks $R \subset K_{t',t''}$.

Definition 21 *Ein Korridor $K_{t',t''}$ der Breite b_K heißt durch eine Gerade g definiert, wenn $t' \parallel g \parallel t''$ und t' und t'' jeweils den Abstand $\frac{b_K}{2}$ zu g besitzen.*

g liegt in der Mitte des Korridors.

Sei H_i die konvexe Hülle eines schon fertig generalisierten Segmentes und g_i die zugehörige Gerade (Abbildung 8.23). Die konvexe Hülle H_{i+1} muß nun so aufgebaut werden, daß die zugehörige Gerade g_{i+1} die zu H_i gehörige Gerade g_i so schneidet, daß keine Rückläufigkeiten entstehen. Dies ist genau dann der Fall, wenn

- ein 2ε breiter Korridor existiert, der H_{i+1} enthält und die konvexe Hülle H_i einseitig schneidet (in der Abbildung liegt H_i vollständig auf einer Seite der Korridorwand t'), und dies auf der „richtigen“ Seite.[15]
- der durch g_i definierte 2ε breite Korridor die konvexe Hülle H_{i+1} einseitig schneidet.

Beide Forderungen folgen direkt aus den Beobachtungen aus Abschnitt 8.3.5: der durch g_i definierte 2ε-Korridor enthält H_i, der durch g_{i+1} definierte 2ε-Korridor H_{i+1}. Der Zusammenbau der einzelnen Segmente liefert die Schnittpunkte $r_{i-1,i} = g_{i-1} \cap g_i$ und $r_{i,i+1} = g_i \cap g_{i+1}$, das Segment S_i wird also durch die Strecke $s_i = [r_{i-1,i}, r_{i,i+1}]$ repräsentiert.

[15] siehe unten

Liegen nicht alle Punkte aus H_i in der ε-Umgebung von s_i, so ist s_i zu kurz. Dies stört genau dann nicht, wenn diese „überstehenden" Punkte dafür in der ε-Umgebung von s_{i-1} oder s_{i+1} liegen. Schneidet der durch g_{i+1} definierte Korridor nun H_i beidseitig, so liegt $r_{i,i+1}$ zu weit innerhalb von H_i: es gibt Punkte in H_i, die weder in der ε-Umgebung von s_i noch im durch g_{i+1} definierten 2ε-Korridor liegen. Schneidet umgekehrt der durch g_i definierte 2ε-Korridor H_{i+1} beidseitig, so gibt es Punkte in H_{i+1}, die weder in der ε-Umgebung von s_{i+1} noch in der ε-Umgebung von s_i liegen.

Schneidet der durch g_{i+1} definierte Korridor H_i hingegen einseitig, so muß dies auf der „richtigen" Seite passieren. Welches ist nun die „richtige" Seite?

$r_{i-1,i}$ ist Startpunkt der das Segment S_i repräsentierenden Strecke s_i. Solange ihr Endpunkt $r_{i,i+1}$ noch nicht bekannt ist, haben wir zunächst eine Halbgerade $h_i \subset g_i$ mit Startpunkt $r_{i-1,i}$. Unklar ist nun noch, in welcher Hälfte von g_i die Halbgerade h_i liegt.

Der durch g_{i-1} definierte Korridor $K_{t'_{i-1},t''_{i-1}}$ schneidet die konvexe Hülle H_i einseitig und H_i liegt nicht völlig in diesem Korridor (sonst wären die Punkte aus H_i noch zu H_{i-1} hinzugefügt worden). Damit ist genau eine der beiden Korridorwände de Sekante von H_i (gesucht ist also die Seite, auf der H_i aus dem Korridor ragt). Sei oBdA t''_{i-1} diese Korridorwand und $a = t''_{i-1} \cap g_i$ der Schnittpunkt der zu H_i gehörigen Geraden g_i mit der Korridorwand, dann ist $h_i = [r_{i-1,i}a$ die gesuchte Halbgerade.

Gilt nun für g_{i+1}: $g_i \cap g_{i+1} = r_{i,i+1} \in h_i$ und H_i liegt bezüglich des durch g_{i+1} definierten 2ε-Korridors in der selben Halbebene wie $r_{i,i+1}$, so liegt H_i auf der „richtigen" Seite, d. h. alle Punkte in H_i liegen entweder in der ε-Umgebung von s_i oder in den durch g_{i-1} und g_{i+1} definierten 2ε-Korridoren.

8.6.2. Aufbau der Segmente

Der Aufbau eines Segmentes funktioniert grundsätzlich genauso wie vorne, es werden so lange Punkte zu einer konvexen Hülle H_i hinzugefügt, wie diese im zugehörigen 2ε-Korridor liegen, sobald es für einen Punkt v_e keinen 2ε-Korridor mehr gibt, der H_i und v_e enthält, wird abgebrochen und ein neues Segment begonnen. Der Unterschied zur vorne beschriebenen Segmentierung besteht lediglich darin, daß es nun Einschränkungen für die Lage des Korridors gibt.

Wir gehen zunächst davon aus, daß das Segment S_i abgeschlossen ist, damit sind $r_{i-1,i}$, H_i, g_i und h_i bekannt. Sei v_{e-1} der letzte noch in H_i liegende Punkt des Polygonzuges. v_e liegt also nicht mehr in H_i, das bedeutet, er liegt außerhalb des durch g_i definierten 2ε-Korridors. Nun soll das neue Segment S_{i+1} konstruiert werden, indem die konvexe Hülle H_{i+1} Punkt um Punkt aufgebaut wird.

Wie schon vorne kann man auch hier entweder den letzten noch in H_i liegenden Punkt v_{e-1} oder den ersten nicht mehr in H_i liegenden Punkt v_e als ersten Punkt

von H_{i+1} wählen. In beiden Fällen[16] ist $v_e \in H_{i+1}$ und damit ist festgelegt, auf welcher Seite H_{i+1} aus dem durch g_i definierten Korridor ragt (sei dies im Folgenden oBdA die Korridorwand t_i''). Schon hier steht also die Richtung der Halbgeraden h_{i+1} fest (ihre genaue Lage hingegen noch nicht).

Es gibt zwei Tangenten von H_i, die durch den Punkt v_e führen. Wir wählen diejenige, deren Schnittpunkt k mit h_i von $r_{i-1,i}$ weiter entfernt liegt (besitzt nur eine der beiden Tangenten einen Schnittpunkt mit h_i, wird diese gewählt). Nun wird H_{i+1} um den Punkt v_{e+1} erweitert. H_i und H_{i+1} besitzen nun bis zu vier gemeinsame Tangenten. Hiervon interessieren nur die beiden, bei denen beide konvexen Hüllen auf der gleichen Seite liegen, und von diesen beiden nun wie oben nur diejenige, deren Schnittpunkt k mit h_i weiter von $r_{i-1,i}$ entfernt liegt.

Hierzu müssen die vier Tangenten gar nicht bestimmt werden. Im ersten Schritt werden mit logarithmischer Suche auf H_i die beiden Tangenten zu v_e bestimmt und die richtige Tangente t_e ausgewählt (sei w_e ein Eckpunkt von H_i, an dem die Tangente t_e die konvexe Hülle H_i berührt, d. h. w_e ist bezüglich v_e ein stützender Punkt von H_i). Wird nun der Punkt v_{e+1} hinzugefügt, so wird getestet, ob er auf der gleichen Seite von t_e liegt wie H_i. Ist dies der Fall, ist nichts weiter zu tun ($t_{e+1} = t_e$). Liegt er auf der anderen Seite von t_e, so ist er neuer tangentialer Punkt von H_{i+1}. Nun muß lediglich noch getestet werden, ob w_e bezüglich v_{e+1} stützend ist. Ist dies nicht der Fall, so ist w_e bezüglich v_{e+1} reflex und es muß ein neuer stützender Punkt w_{e+1} gesucht werden. Hierzu werden einfach die auf der Hülle H_i zu w_{e+1} benachbart liegenden Punkte getestet.[17]

Wird ein neuer Punkt v_{e+c} zur konvexen Hülle H_{i+1} hinzugefügt, so ist die neue Tangente t_{e+c} wie gezeigt aus t_{e+c-1} einfach bestimmbar.

Nun werden für jeden neu hinzukommenden Punkt v_{e+c} zwei Sachen getestet:

- Der Korridor $K_{t_i', t_i''}$ darf H_{i+1} nur einseitig schneiden. Ich habe oben angenommen, daß t_i'' Sekante von H_{i+1} ist. Damit müssen alle Punkte aus H_{i+1} auf der selben Seite von t_i' liegen. Der Test ist einfach: Am Anfang wird der vorzeichenbehaftete Abstand von v_e zu t_i' bestimmt und dann für jeden neuen Punkt v_{e+c} getestet, ob sein Abstand zu t_i' das gleiche Vorzeichen besitzt.

- Um H_{i+1} muß ein 2ε-Korridor existieren, der H_i nur einseitig schneidet (und zwar auf der „richtigen“) Seite. Auch dies kann einfach getestet werden: Ein beliebiger Korridor $K_{t',t''}$, der H_{i+1} enthält, schneidet H_i genau dann auf der „richtigen“ Seite (oder auch gar nicht), wenn für mindestens eine der beiden Korridorwände $t \in \{t', t''\}$ gilt: $t \cap g_i = l \in h_i$ und $|r_{i-1,i}l| \geq |r_{i-1,i}k|$.

[16] Theoretisch könnte es vorkommen, daß v_{e-1} und v_e so ungünstig liegen, daß kein gültiger Korridor um beide Punkte existiert, hierzu müssen beide Punkte aber sowohl einen Abstand von weit mehr als 2ε zueinander also auch eine sehr ungünstige Lage bezüglich h_i besitzen. Möchte man auf der sicheren Seite sein, wählt man v_e als Startpunkt des neuen Segmentes.

[17] In welche Richtung dabei getestet werden muß, ist klar. Es interessieren die Punkte, die auf der anderen Seite von $\overline{w_e v_{e+1}}$ liegen wie v_e.

Alle 2ε-Korridore, die H_{i+1} enthalten, können wie in 8.2.4 beschrieben bestimmt werden: wir erhalten einen oder mehrere Bereiche, in denen ein 2ε-Korridor um H_{i+1} frei drehbar ist. Hierbei interessieren nur die Grenzlagen: Liefert eine der Grenzlagen einen gültigen Korridor (also einen, dessen Wand obige Bedingung erfüllt), so ist der Bereich relevant.

Die Bestimmung dieser Bereiche kann inkrementell geschehen (jeder neu zu H_{i+1} hinzukommende Punkt schränkt die mögliche Lage des Korridors weiter ein).

Ist eine dieser beiden Bedingungen für einen Punkt v_{e+c} nicht erfüllt, so wird an diesem Punkt abgebrochen und H_{i+1} enthält v_{e+c-1} als letzten Punkt. Unter den oben bestimmten möglichen Lagen des Korridors wird nun optimalerweise die Richtung gewählt, bezüglich derer H_{i+1} am schmalsten ist. Allerdings ist dies etwas aufwendiger, da nun für die entsprechenden Bereiche alle möglichen Lagen des Korridors bestimmt werden müssen. Nimmt man einfach die oben ermittelte Grenzlage, spart man sich zusätzlichen Aufwand.[18]

Damit steht auch die Lage von g_{i+1} fest, und das liefert $r_{i,i+1} = g_i \cap g_{i+1}$ und $h_{i+1} = [r_{i,i+1}u$ mit $u = g_{i+1} \cap t_i''$. Auf diese Weise können alle Segmente bestimmt werden. Lediglich das erste Segment bedarf einer Sonderbehandlung, da $r_{0,1}$ nicht existiert. Die Gerade g_1 wird wie gehabt bestimmt (Abschnitt 8.2.3). Auf ihr werden nun einfach (wie in Abschnitt 8.3.4) a_1 und e_1 bestimmt und $h_1 = [a_1e_1$ ist die gesuchte Halbgerade.

Somit lassen sich alle Schnittpunkte $r_{i,i+1}$ bestimmen. Startpunkt und Endpunkt des generalisierten Polygonzuges werden gesondert bestimmt. Die Halbgerade $[r_{1,2}a_1$ besitzt einen oder zwei Schnittpunkte mit ∂H_1. Diese sind mit logarithmischer Suche zu finden. Sei m der von $r_{1,2}$ weiter entfernt liegende Schnittpunkt (oder der einzige Schnittpunkt, falls es nur einen gibt). Liegt nun m weiter von $r_{1,2}$ entfernt als a_1 so ist $a = m$ Startpunkt der Generalisierung, andernfalls $a = a_1$. Genauso wird auch der Endpunkt e der Generalisierung bestimmt, wir bekommen: $ar_{1,2}r_{2,3} \ldots e$.

Das Verfahren liefert also eine Segmentierung des Originalpolygonzuges, die ohne Einsatz von Zwischenstücken einen Polygonzug liefert, der das ε-Kriterium erfüllt. Die Komplexität wird dabei gegenüber der zuerst angegebenen Segmentierung nicht erhöht, allerdings ist der Algorithmus etwas komplizierter.

8.6.3. Übersteuern

Der modifizierte Algorithmus verhindert zwar Rückläufigkeiten, Übersteuern kann aber weiterhin auftreten. Um dies zu verhindern, ist nur eine leichte Modi-

[18] In den meisten Fällen ist die mögliche Lage des Korridors sowieso schon stark eingeschränkt, da ja im nächsten Schritt abgebrochen wurde, weil es *keine* gültigen Korridore mehr gibt.

fikation nötig. Oben werden aus allen möglichen Korridorlagen diejenigen ausgewählt, die H_i auf der „richtigen“ Seite schneiden. Werden nun noch die Korridorlagen entfernt, die zu stark übersteuern, bleiben nur gewünschte Korridore übrig.

Die Halbgerade h_i besitzt mit ∂H_i einen oder zwei Schnittpunkte. Sei m der weiter von $r_{i-1,i}$ entfernt liegende Schnittpunkt (oder der einzige Schnittpunkt, falls es nur einen gibt). Nun gibt $U = |r_{i-1,i}r_{i,i+1}| - |r_{i-1,i}m|$ an, wie weit $r_{i,i+1}$ außerhalb von H_i liegt, d. h. wie stark hier übersteuert wird (für $U < 0$ liegt $r_{i,i+1}$ innerhalb von H_i).

Wie schon oben werden auch hier nur die Grenzlagen der Korridore in den gefundenen Bereichen getestet.[19] Liegt in einem Bereich für eine der Grenzlagen U unterhalb eines Schwellwertes δ, so enthält der Bereich gültige Korridore.

Wie oben wird abgebrochen, wenn für einen Punkt v_{e+c} keine gültigen Korridore mehr existieren. Nun kann es vorkommen, daß in einem gültigen Bereich beide Grenzlagen keine gültigen Korridore enthalten (der eine scheidet wegen Rückläufigkeit aus, der andere wegen Übersteuern). In diesem Fall muß in der Mitte des Bereiches ein gültiger Korridor gesucht werden, was durch logarithmische Suche geschehen kann.

In allen Fällen sollte zunächst getestet werden, ob der zur Tangente t parallel liegende Korridor gültig ist, da diese den Bewegungsverlauf gut wiedergibt. Dieser Test erspart vielfach die Suche.

8.6.4. Untersteuern

Der direkte Zusammenbau arbeitet mit Schnitten von Korridoren. Damit kann das schon in Abschnitt 4.4 gezeigte Phänomen (Abb. 4.2) auftreten: zwar liegen alle Punkte des Originalpolygonzuges p in einem durch die Generalisierung q definierten 2ε breiten Korridor, aber trotzdem erfüllt q die ε-Bedingung nicht.

Diesen Fehler korrekt zu beheben ist möglich, aber sehr aufwendig. Einfacher ist es, große Ausreißer zu verhindern: Zur Geraden g_i wird wie im Originalalgorithmus der Endpunkt e_i bestimmt. Verlangt man nun für alle Korridore, die mit g_i einen spitzeren Winkel als 90° einschließen, daß der Schnittpunkt $r_{i,i+1}$ der zugehörigen Geraden g_{i+1} mit g_i außerhalb liegt: $[r_{i-1,i}e_i] \subset [r_{i-1,i}r_{i,i+1}]$, so kann ein Punkt aus p maximal $\sqrt{2}\varepsilon$ von q entfernt liegen.

Dieser Test ist nicht sonderlich elegant, vor allem, da bei 90° ein Sprung auftritt, eine exakte Lösung verlangt jedoch, daß für alle möglichen Korridorlagen das Kreissegment (mit Radius ε) um $r_{i,i+1}$ bestimmt wird, das die beiden zu H_i und H_{i+1} gehörenden Korridore abschließt (dies ist das Segment zwischen den Strecken

[19] Man kann Fälle konstruieren, in denen dies nicht optimal ist, aber zum einen treten diese in der Praxis nicht auf, zum anderen ist der entstehende Fehler klein. Ein genauerer Test würde unverhältnismäßig mehr Aufwand bedeuten.

h_1 und h_2 aus Abbildung 8.22), um dann zu testen, ob Punkte aus H_i oder H_{i+1} außerhalb liegen.

In vielen Fällen wird man auch völlig auf diesen Test verzichten, zum einen tritt das Phänomen selten auf, zum anderen liefert die Generalisierung auch ohne diese Korrektur einen gültigen 2ε-Korridor um p, der eben schlimmstenfalls ein paar spitze Ecken besitzt.

8.6.5. Ablauf

Damit der eigentliche Algorithmus nicht in den vielen Detailüberlegungen untergeht, hier noch einmal ein Überblick über den Ablauf. Die Darstellung beschränkt sich auf die großen Schritte, Einzelentscheidungen (wann wie welcher Punkt/Korridor/... gewählt wird) müssen oben nachgelesen werden.

- Zunächst wird wie im Originalalgorithmus das erste Segment bestimmt. Der Startpunkt a der Generalisierung wird berechnet, ebenso die Halbgerade h_1.
- Für jedes weitere Segment S_{i+1} wird die konvexe Hülle H_{i+1} aufgebaut. Ein neuer Punkt v darf zu H_{i+1} hinzugefügt werden, solange um die um v erweiterte Hülle H_{i+1} ein 2ε-Korridor existiert, der
  - die konvexe Hülle nur einseitig (und auf der „richtigen“ Seite) schneidet,
  - nicht übersteuert,
  - nicht untersteuert.

  Die erste Forderung ist dabei zwingend, ob die zweite und dritte Forderung eingehalten werden müssen, hängt von der Anwendung ab.

  Die erste Forderung läßt sich dabei effizient prüfen, da die Menge aller Bereiche, in denen ein 2ε-Korridor um H_{i+1} liegen kann, inkrementell aufgebaut werden kann (jeder zu H_{i+1} hinzukommende Punkt schränkt die Bereiche weiter ein).
- Schließlich wird noch der Endpunkt e des letzten Segmentes S_w bestimmt, wir erhalten die Generalisierung $q = ar_{1,2}r_{2,3}\ldots r_{w-1,w}e$.

8.7. Komplexität

Die Convex Hull Generalisierung ist in beiden gezeigten Varianten historiefrei inkrementell und besitzt damit bezüglich der Segmentzahl lineare Komplexität. Der Aufbau eines einzelnen Segmentes erfolgt wie gezeigt in $O(n \log n)$.

9. Kumulative Generalisierung

9.1. k-Step-Generalisierung

Alle gezeigten eckentreuen Generalisierungen (dpgen, jhgen, simpleinc$_\varepsilon$, WεG, rolland) vereinfachen einen Polygonzug, indem sie Punkte aus p weglassen bzw. Punkte aus p auswählen, je nach Sichtweise. Die einfachste Möglichkeit ist hierbei sicher, jeden k-ten Punkt aus p zu wählen, d. h. für $p = v_1 v_2 \dots v_n$ ist $q = v_1 v_{k+1} v_{2k+1} \dots v_n$ (Algorithmus 11): simplestep ist eine historiefreie inkrementelle Generalisierung.

Das Vorgehen ist trivial, zu entscheiden ist lediglich, ob der letzte Punkt v_n von p auch Endpunkt von q wird, wenn $n \neq ik + 1 \quad (i \in \mathcal{N}_0)$. Liegt p vollständig vor, verzichtet man also auf inkrementelles Vorgehen, kann man natürlich k entsprechend wählen, bzw. für ein gewünschtes k die am Ende überzähligen Punkte über ganz p zu verteilen: Sei $n = ik + j + 1 \quad (i \in \mathcal{N}_0, j < k - 1$, so besitzt p entweder j Punkte zu viel oder $k - j$ Punkte zu wenig.

Für große i ($j \leq i$) können diese Punkte auf die einzelnen Segmente verteilt werden: für $j \leq k/2$ werden die j Punkte zu viel nun auf die i Segmente verteilt, j Segmente mit $k + 1$ Punkten, wobei jedes $\lfloor i/j \rfloor$-te Segment $k + 1$ Punkte bekommt, für $j > k/2$ dementsprechend $k - j$ der $i + 1$ Segmente mit nur $k - 1$ Punkten.

Allerdings wird dadurch die gleichmäßige Struktur der Generalisierung zerstört: Wenn eine Bewegungsspur p, wie in Kapitel 2 beschrieben, mit einer konstanten Abtastrate gemessen wurde, d. h. v_2 wurde die Zeitspanne Δ_t nach v_1 gemessen, v_3 die Zeitspanne Δ_t nach v_2 usw., dann gilt für $q = \text{simplestep}(p, k) = v_1 v_{k+1} v_{2k+1} \dots v_n$ selbstverständlich: v_{k+1} ist der Punkt, der $k\Delta_t$ nach v_1 erreicht wurde, usw., d. h. die Zeitinformation und somit die Geschwindigkeitsinformation bleibt erhalten, was durch die im letzten Absatz beschriebene Verteilung der „überzähligen“ Punkte zerstört würde.

Leider ist der Erhalt der Geschwindigkeitsinformation neben der Einfachheit des Algorithmus der einzige Vorteil von simplestep. Aus einem mit $\Delta_t = 30\text{ms}$ gemessenen Polygonzug jeden zweiten Punkt zu streichen entspricht einer Messung der Bewegung mit $\Delta_t' = 60\text{ms}$, also einer schlichten Vergröberung, oder anders gesagt: Die Hälfte der gemessenen Information wird einfach weggeworfen. Dieser Ansatz führt also schnell dazu, daß q einer mit zu geringer Abtastrate gemesse-

Algorithmus 11: Generalisierung mit Schrittweite k

```
1: function stepk = (Polygonzug p = v_1 ... v_n, Schrittweite k, Step i) → Polygonzug :
2: begin
3:    if n = 1 then                          ▷ wir sind fertig
4:       return p;
5:    else if i ≡ 1 (mod k) then             ▷ ein k-ter Punkt
6:       return v_1 ∘ stepk(v_2 v_3 ... v_n, k, i++);
7:    else
8:       return stepk(v_2 v_3 ... v_n, k, i++);
9:    end if
10: end

11: function simplestep = (Polygonzug p = v_1 ... v_n, Schrittweite k) → Polygonzug :
12: begin
13:    return stepk(p, k, 1);
14: end
```

nen Darstellung der Bewegung entspricht, wodurch q die Form der Bewegung nur unzureichend wiedergibt.

Bei diesem Algorithmus ist also der Grad der Vereinfachung[1] gleich dem Grad des Informationsverlusts: Alle Information, die in den gelöschten Punkten steckte, ist verloren. Die anderen bislang vorgestellten Algorithmen sind diesbezüglich besser. Ist q beispielsweise das Ergebnis der Douglas-Peucker-Generalisierung einer Bewegungsspur p, und seien v und w zwei aufeinanderfolgende Eckpunkte in q, so ist in q implizit die Information kodiert, daß alle Punkte zwischen v und w in p nicht weiter als den bekannten Parameter ε von $[vw]$ entfernt liegen. In q ist somit zusätzliche Information kodiert. Der einzige Vorteil von simpleinc – der Erhalt der Geschwindigkeitsinformation – ist für die meisten anderen Algorithmen leicht nachrüstbar. Hierzu muß lediglich zu jedem Punkt v_i in q gespeichert werden, wie viele Zeiteinheiten zwischen v_{i-1} und v_i liegen (also wie viele Punkte aus p zwischen v_{i-1} und v_i gelöscht wurden).

[1] Verhältnis der Anzahl Punkte des generalisierten Polygonzuges zur Anzahl Punkte im Originalpolygonzug

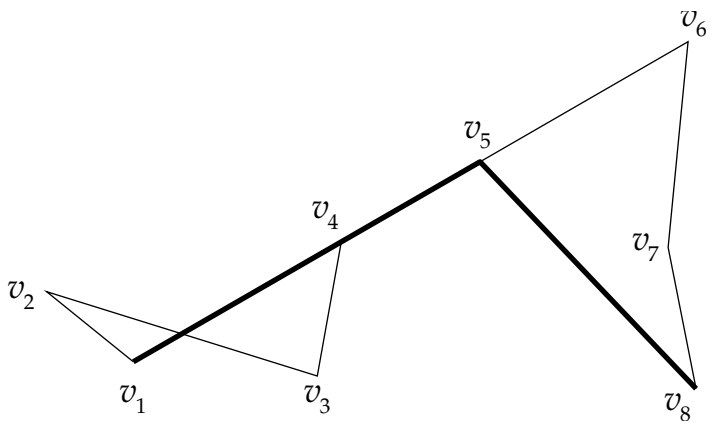


Abbildung 9.1.: Σ-Generalisierung mit unerwünschten Effekten

9.2. Σ-Generalisierung

simplestep faßt jeweils k aufeinander folgende Punkte und somit $k\Delta_t$ Zeitabschnitte zusammen, sowohl in p als auch in q repräsentiert jeder Punkt implizit eine Zeitinformation. Dies ist, wie vorne erwähnt, nützlich, sofern man mit Geschwindigkeiten hantieren möchte. Statt in einem konstanten zeitlichen Abstand die jeweilige Position zu messen, kann das Raster ebensogut durch die zurückgelegte Entfernung bestimmt werden, d. h. die Position eines Objekts wird im Startpunkt, nach dem es 30 cm, 60 cm, 90 cm, 120 cm usw. gefahren ist, gemessen und daraus eine Bewegungsspur erzeugt.

Die von Alexandra Musto (zur Generalisierung von QMV-Sequenzen) vorgeschlagene Σ-Generalisierung[2] benutzt diese Idee und wandelt eine in der Zeit gerasterte Bewegungsspur p in eine „entfernungsgerasterte“ Spur q um. Die Idee ist trivial: Ausgehend vom Startpunkt v_1 wird die Länge $|v_i v_{i+1}|$ der einzelnen Teilstücke aufsummiert, bis ein Schwellwert σ erreicht ist:

$$\sum_{i=2}^{k} |v_{i-1}v_i| = \sigma_{1,k} \quad \leq \quad \sigma \quad < \quad \sigma_{1,k+1} = \sum_{i=2}^{k+1} |v_{i-1}v_i|.$$

Für $(\sigma - \sigma_{1,k}) \leq (\sigma_{1,k+1} - \sigma)$ wird v_k als Endpunkt des Segments und neuer Startpunkt gewählt, ansonsten v_{k+1} (Algorithmus 12).

Der Schwellwert σ sollte natürlich hierbei wesentlich größer als die durchschnittliche Entfernung $\bar{d} = |v_i v_{i+1}|$ zwischen zwei Punkten gewählt werden. Mit $\sigma = k\bar{d}$ werden im Mittel je k Punkte in einem Segment zusammengefaßt.[3]

[2] [Mus00]

[3] Auch hier läßt sich wie oben beschrieben der generalisierte Polygonzug leicht mit einem Zeitraster attributieren.

Algorithmus 12: Σ-Generalisierung

```
1:  function sum = (Polygonzug p = v_1 ... v_n, Summe σ, Summe σ') → Polygonzug
    :
2:  begin
3:    if n = 1 then                          ▷ wir sind fertig
4:      return p;
5:    else
6:      σ'' = σ' + |v1v2|;                   ▷ neues Streckenstück aufaddieren
7:      if σ'' > σ then
8:        if ((σ − σ') ≤ (σ'' − σ)) ∧ (σ' > 0) then
9:          return v_1 ∘ sum(v_1v_2 ... v_n, σ, 0);   ▷ etwas zu kurz
10:       else
11:         return v_2 ∘ sum(v_2v_3 ... v_n, σ, 0);   ▷ etwas zu lang
12:       end if
13:     end if
14:   else
15:     return sum(v_2v_3 ... v_n, σ, σ'');   ▷ noch zu kurz
16:   end if
17: end

18: function Σ = (Polygonzug p = v_1 ... v_n, Summe σ) → Polygonzug :
19: begin
20:   return sum(p, σ, 0);
21: end
```

Die Σ-Generalisierung vereinfacht eine Bewegungsspur (mit ausreichend großem σ) sicherlich, gibt die Form aber nur höchst unzureichend wieder, wie z. B. Abbildung 9.1 zeigt. Ecken werden nicht erkannt und ein im Grunde völlig gerades Teilstück $v_4v_5v_6$ wird hier von der Generalisierung plötzlich „gebrochen“. Abhilfe schafft hier, aufeinanderfolgende gleichgerichtete Streckenstücke zusammenzufassen (was zum Polygonzug $v_1v_2v_3v_4v_6v_7v_8$ führt), wodurch solche „Brüche“ nicht mehr auftreten können. Bei einer gemessenen Bewegungsspur werden zwei aufeinanderfolgende Streckenstücke fast nie genau die gleiche Richtung besitzen, damit würde dieses Zusammenfassen fast nie auftreten. Bedenkt man jedoch, daß die Generalisierungsalgorithmen mittels der vorne beschriebenen QMV-Algebra nicht nur auf numerischen sondern auch auf QMV-Sequenzen arbeiten können, macht dieser Zusatz Sinn: Hier treten häufig gleichgerichtete Vektoren auf.[4]

[4]Die erste Version des Σ-Algorithmus arbeitete wie oben erwähnt auf QMV-Sequenzen

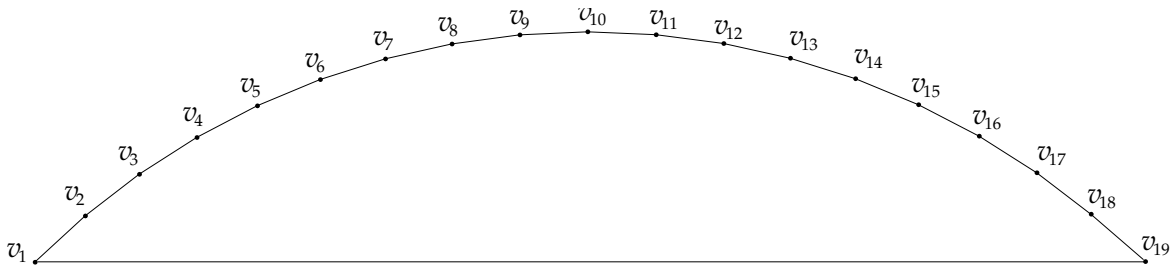


Abbildung 9.2.: 18 um jeweils 5° verdrehte Streckenstücke

9.3. Streckenkonkatenation

Da im Numerischen die Forderung „exakt gleiche Richtung“ hingegen zu streng ist, habe ich den Algorithmus dergestalt angepaßt, daß ein paar Grad Winkelabweichung toleriert wird. Haben zwei aufeinanderfolgende Streckenstücke $[v_i v_{i+1}]$ und $[v_{i+1} v_{i+2}]$ ungefähr die gleiche Richtung (eine Abweichung um max. α), so darf im Punkt v_{i+1} nicht getrennt werden, v_{i+1} ist kein gültiger Eckpunkt für die Generalisierung. Dies kann selbstverständlich dazu führen, daß ganze Kreisbögen als „gleich ausgerichtet“ erscheinen: 10 Streckenstücke, von denen jedes gegenüber seinem Vorgänger nur um 5° weiter nach rechts gedreht ist, beschreiben insgesamt eine Kurve von 50°, d. h. mit Schwellwert $\alpha = 5°$ wird die Bewegungsspur im ganzen Teilstück zu einem Segment generalisiert.

Wie Abbildung 9.2 zeigt, ist dieser Effekt geringer als auf den ersten Blick vermutet: Sie zeigt 18 gegeneinander um je 5° gegeneinander verdrehte aufeinanderfolgende Streckenstücke $[v_1 v_2]$, $[v_2, v_3]$, …, $[v_{18} v_{19}]$, die somit zu $[v_1 v_{19}]$ generalisiert werden. Die Richtung der Bewegung dreht sich hier um $18 \cdot 5° = 90°$. Die Generalisierung $[v_1 v_{19}]$ bildet die Sehne dieses Kreisbogens, der in seinem entferntesten Punkt v_{10} um $\left(1 - \frac{1}{\sqrt{2}}\right) \cdot |v_1 v_{19}| \approx 0.29 \cdot |v_1 v_{19}|$ von ihr abweicht.[5]

Wurde die betrachtete Bewegung allozentrisch gemessen (siehe Kapitel 2.3.3), so sind diese Überlegungen von geringer Relevanz für die Praxis, die Wahrscheinlichkeit, daß eine Bewegungsspur eine (relativ gleichmäßige) 90° Kurve (oder

[5] Die vordergründige Ähnlichkeit mit dem linsenförmigen Akzeptanzbereich aus Kapitel 5.6.3 sollte nicht darüber hinwegtäuschen, daß es sich hier um 2 grundsätzlich unterschiedliche Ansätze handelt: Im vorliegenden Ansatz wird nur die relative Winkelabweichung eines Streckenstückes zu seinem Vorgänger bestimmt, dies kann theoretisch bis zu einem Vollkreis oder noch weiter führen, genauso auch zu beliebigen Schlangenlinien etc., bricht aber ab, sobald auch nur ein Stück um mehr als diese erlaubte Toleranz verdreht ist. Das Winkelmaß aus Kapitel 5.6.3 hingegen beschreibt einen linsenförmigen Akzeptanz*bereich*, den die Bewegungsspur nicht verlassen darf, in dem sie sich aber annähernd beliebig verhalten darf. Außerdem arbeitet dieses Maß auf globalen Algorithmen und ist nur schwer (siehe Kapitel 6.1) für inkrementelle Nutzung anzupassen, wogegen die Konkatenation „gleich“gerichteter Streckenstücke von sich aus inkrementell ist.

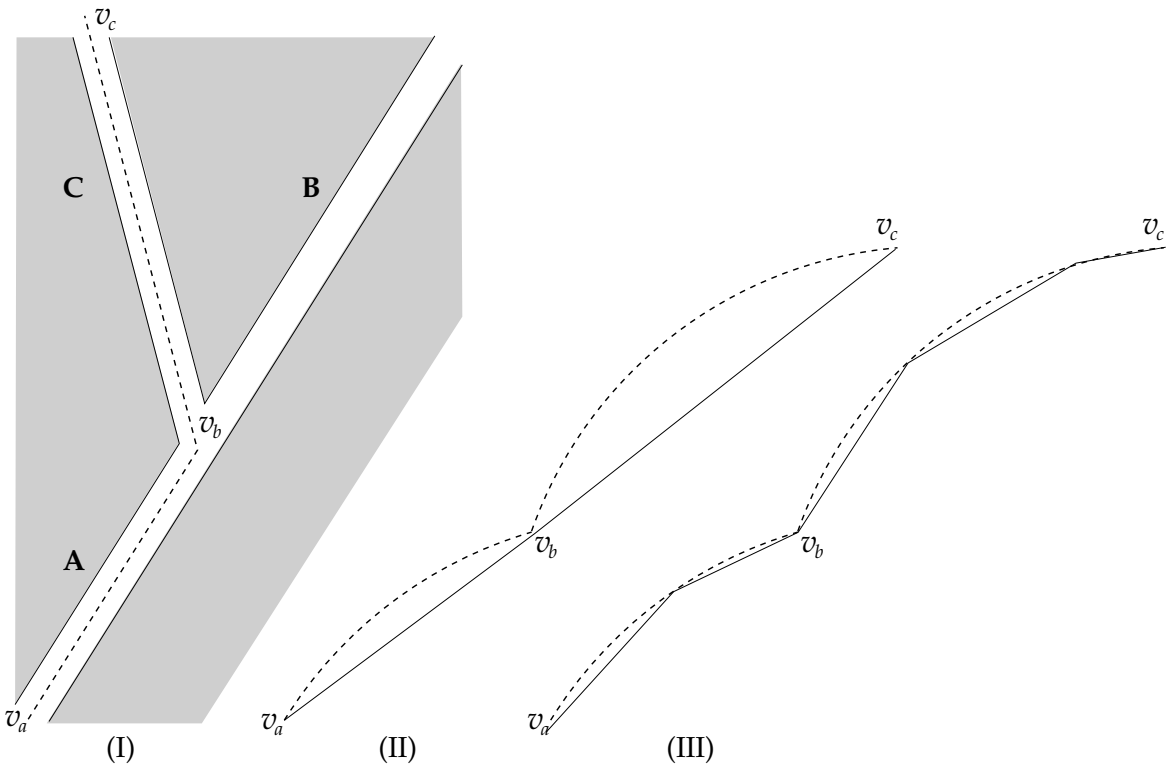


Abbildung 9.3.: Egozentrische Messung einer Bewegung mit „Rechtsdrall“

mehr) beschreibt, bei der jeder Schritt gerade unterhalb der Toleranzschwelle liegt, ist gering.

Leider ist dies nicht unabhängig von der Abtastrate: Wird die in Abbildung 9.2 gezeigte Kurve mit halber Abtastrate gemessen (wir betrachten also den Polygonzug $v_1v_3v_5v_7v_9v_{11}v_{13}v_{15}v_{17}v_{19}$), so beträgt der Winkel zwischen zwei Streckenstücken 10°. Je höher die Abtastrate (je genauer gemessen wird), desto kleiner muß also die Winkeltoleranz gewählt werden.

Wurde die Bewegungsspur egozentrisch gemessen, treten dagegen schnell Probleme auf. Wie in Kapitel 2.3.3 beschrieben, existiert bei rein egozentrischer Messung keine Möglichkeit, exakt die Richtung zu halten. Wird beispielsweise die Bewegung eines Serviceroboters gemessen, der einen langen geraden Gang entlangfährt, so passiert es leicht, daß die gemessene Bewegungsspur eine leichte Krümmung in die eine oder andere Richtung aufweist, so könnte der Polygonzug aus Abb. 9.2 auch das Ergebnis einer egozentrischen Messung der geraden Bewegung eines Serviceroboters sein, dessen linkes Rad etwas stärker abgefahren ist als das rechte, oder dessen Lenkung etwas Spiel hat.[6]

[6] Auch bei optimaler Einstellung der Sensoren sind diese Meßfehler prinzipiell nicht vermeidbar, da sie auch von externen Gegebenheiten wie Bodenbelag etc. abhängen.

Diese Fehler können bei jeder egozentrischen Messung auftreten und alle vorgestellten Generalisierungsalgorithmen müssen auf diesen „falschen“ Daten arbeiten. Allerdings ist hier die Konkatenation mittels Toleranzwinkel besonders anfällig. Zunächst scheint die Generalisierung eines solchen Bogens zu einer einzigen Strecke unter diesem Gesichtspunkt ein positiver Nebenaspekt zu sein. Abbildung 9.3 zeigt, wieso dies problematisch ist: Ein Serviceroboter kommt aus Gang A und biegt in Gang C ab (I). Da seine Bewegung egozentrisch gemessen wird und sich dabei ein leichter Rechtsdrall einschleicht, ist die gemessene Bewegungsspur gebogen (II). Die Konkatenation aller aufeinanderfolgender Streckenstücke mit hinreichend ähnlicher Richtung führt im Ergebnis zu zwei langen Strecken, die auch noch gleiche Richtung besitzen. Versucht man nun, den so gewonnenen Polygonzug auf die Gänge (I) zu matchen, so führt dies fälschlicherweise zur Annahme, der Serviceroboter wäre von Gang A nach Gang B gefahren.Die hier auftretenden Probleme entsprechen im Kleinen der Problematik der egozentrischen QMV-Sequenzen (Kapitel 3.3.2).

Andere Generalisierungsalgorithmen würden hier natürlich auch *falsche* Ergebnisse liefern – eine fehlerbehaftete Eingabe führt zu einer fehlerhaften Ausgabe –, jedoch sind diese Fehler weniger kritisch. Wird der bogenförmige Polygonzug aus (II) beispielsweise mittels einer inkrementellen ε-Generalisierung verarbeitet, so führt dies (je nach Wahl von ε)[7] zu einer Folge kürzerer Streckenstücke (III). Dies ist zwar auch kein zufriedenstellendes Ergebnis, führt aber nicht sofort zu falschen Wegzuschreibungen, der generalisierte Polygonzug paßt nicht richtig. In dieser generalisierten Bewegungsspur ist die leichte Krümmung weiterhin erkennbar, was Rückschlüsse auf Meßfehler zuläßt.

Bei allozentrisch (oder zumindest mit Kompaß) gemessenen Bewegungsspuren ist die Konkatenation „gleich“gerichteter Streckenstücke (bei geeignet, abhängig von der Abtastrate gewähltem Toleranzwinkel α) ein sinnvoller erster Schritt der Generalisierung, wenn man so will auch ein eigenständiger Generalisierungsalgorithmus. Bei egozentrisch gemessenen Bewegungsspuren hingegen treten die gezeigten Probleme auf. Ob diese Probleme in einer konkreten Anwendung zum Tragen kommen, hängt von einer ganzen Reihe von Randbedingungen ab.[8]

9.4. Modifizierte Streckenkonkatenation

Das Hauptproblem der oben gezeigten Konkatenation ist, daß sich die kleinen Winkelabweichungen über viele Streckenstücke aufsummieren können. Dies liegt dar-

[7] Wie vorne gezeigt, bietet sich an, ε in Größenordnung einer halben Gangbreite zu wählen.

[8] Um systematische Meßfehler zu erkennen, scheint die Betrachtung der Originalbewegungsspur erfolgversprechender, trotzdem hat sich im Bremer Rollandprojekt gezeigt, daß es einfacher ist, die generalisierte Bewegungsspur zu betrachten, womit – zumindest in Gangszenarien – gute Ergebnisse erreicht werden.

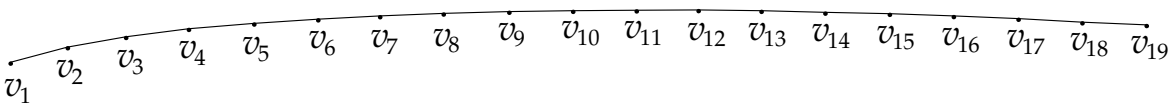


Abbildung 9.4.: Maximale Krümmung bei inkrementeller Konkatenation gleichlanger Streckenstücke mit Schwellwert $\alpha = 5^\circ$

an, daß jeweils nur die Teilstücke $[v_k v_{k+1}]$ und $[v_{k+1} v_{k+2}]$ miteinander verglichen werden. Dies ist nicht selbstverständlich, wenn man bedenkt, daß dies ein inkrementelles Verfahren ist.

Zunächst werden die Streckenstücke $[v_1 v_2]$ und $[v_2 v_3]$ miteinander verglichen. Falls ihre Richtung sich um weniger als α unterscheidet, darf in v_2 nicht getrennt werden. Technisch entspricht dies einer Zusammenfassung der beiden Teilstücke zur Strecke $[v_1 v_3]$. Im nächsten Schritt werden nun die Richtungen der Teilstücke $[v_2 v_3]$ und $[v_3 v_4]$ betrachtet. Sind diese wiederum hinreichend ähnlich, so darf in v_3 ebenfalls nicht getrennt werden, was technisch einer Zusammenfassung der ersten drei Teilstücke zur Strecke $[v_1 v_4]$ entspricht usw. Nun gibt es aber keinen Grund, im zweiten Schritt das Teilstück $[v_2 v_3]$ mit $[v_3 v_4]$ zu vergleichen. Die ersten zwei Teilstücke sind ja schon zusammengefaßt, es bietet sich also an, direkt die Richtung von $[v_1 v_3]$ und $[v_3 v_4]$ zu vergleichen.

Durch diese einfache Modifikation werden die im letzten Kapitel beschriebenen Probleme gelöst, wie ein erneuter Blick auf das Beispiel aus Abbildung 9.2 zeigt. Aufeinanderfolgende Teilstücke $[v_k v_{k+1}]$ und $[v_{k+1} v_{k+2}]$ sind wie schon erwähnt zueinander um 5° verdreht. Zwischen den Strecken $[v_k v_{k+2}]$ und $[v_{k+2} v_{k+3}]$ beträgt der Drehwinkel dann schon 7.5°, zwischen $[v_k v_{k+3}]$ und $[v_{k+3} v_{k+4}]$ sind es 10° usw. Damit addieren sich kleine Winkelabweichungen nicht mehr über viele Teilstücke auf. Abbildung 9.4 zeigt einen Polygonzug, bei dem alle Strecken $[v_1 v_k]$ und $[v_k v_{k+1}]$ jeweils um 5° zueinander verdreht sind. Man sieht leicht, daß ein ganz leichtes Abwandern weiterhin stattfinden kann, dieser Effekt ist aber in der Praxis nicht relevant.

Damit läßt sich diese modifizierte Konkatenation sowohl als Vorstufe einer Σ-Generalisierung als auch als eigenständige Generalisierung einsetzen (Algorithmus 13). Allerdings eignet sich concat als eigenständiger Algorithmus nicht, um die Feinstruktur einer Bewegung zu unterdrücken. Muß unser Serviceroboter beispielsweise vor einer Tür rangieren, so enthält die gemessene Bewegungsspur viele sehr kurze aufeinanderfolgende Streckenstücke, die zueinander um fast 180° verdreht sind. Diese Rangierbewegung bleibt bei der Generalisierung mittels concat unverändert erhalten.

Algorithmus 13: Inkrementelle Konkatenation

1: **function** con = (Strecke $s = [v_a v_1]$, Polygonzug $p = v_1 v_2 \dots v_n$, Winkel α) $\rightarrow$ Polygonzug :
2: **begin**
3: **if** $n = 1$ **then** ▷ *wir sind fertig*
4: **return** $v_a v_1$;
5: **else if** drehwinkel$(s, [v_1 v_2]) \leq \alpha$) **then** ▷ *„gleiche“ Richtung*
6: **return** con$([v_a v_2], v_2 v_3 \dots v_n, \alpha)$;
7: **else**
8: **return** $v_a v_1 \circ$ con$([v_1 v_2], v_2 v_3 \dots v_n, \alpha)$;
9: **end if**
10: **end**

11: **function** concat = (Polygonzug $p = v_1 \dots v_n$, Winkel α) $\rightarrow$ Polygonzug :
12: **begin**
13: **if** $n < 2$ **then** ▷ *wir sind fertig*
14: **return** p;
15: **else**
16: **return** con$([v_1 v_2], v_2 v_3 \dots v_n, \alpha)$;
17: **end if**
18: **end**

10. Generalisierung mit lokaler Gewichtung

Nach den globalen Divide-and-Conquer- und den inkrementellen Ansätzen zur Generalisierung stellt dieses Kapitel einen Ansatz mit lokalem Generalisierungskriterium vor, bei dem für jeden Punkt der Bewegungsspur bestimmt wird, wie „wichtig“ er für die Beschreibung der Bewegung ist. Während die globalen Ansätze ausgehend vom Bild der gesamten Spur so lange rekursiv die „extremen“ („herausragenden“) Punkte bestimmen, bis ein bestimmtes Ähnlichkeitsmaß erreicht ist (z. B. bis das Original von der Generalisierung nur noch um ε abweicht) – also aus globaler Sicht auf den Polygonzug bestimmt wird, welches die „wichtigen“ Punkte sind, wird dies im folgenden lokal entschieden.

10.1. Das Grundgerüst

Gegeben ist ein Polygonzug $p = v_1 v_2 \dots v_n$:

1. Berechne für jeden Punkt v_i ein Maß[1] κ_i. Hierbei bestimmt sich κ_i aus der Lage von v_i zu seinen direkten Nachbarn v_{i-1} und v_{i+1}.[2]

2. Lösche den Punkt v_m mit minimalem κ_m.

Damit erhält man einen Polygonzug $p' = v_1 v_2 \dots v_{m-1} v_{m+1} v_{m+2} \dots v_n$, für welchen man erneut die obigen beiden Schritte durchführt usw. Damit werden Schritt für Schritt so lange Punkte aus p gelöscht, bis für alle Punkte v_i ein Schwellwert κ erreicht ist, so daß $\kappa_i > \kappa$. Alternativ läßt sich auch angeben, wie viele Punkte der generalisierte Polygonzug im Verhältnis zur Originalspur noch enthalten soll.

Sofern sich κ_i für jeden Punkt in konstanter Zeit bestimmen läßt, arbeitet der Algorithmus in der vorliegenden Form in $O(n^2)$. Durch Umstrukturierung des obigen Ablaufs läßt sich die Komplexität auf $O(n \log n)$ drücken. Zunächst werden alle Punkte v_i nach ihren κ_i sortiert. Dies geht in $O(n \log n)$. Nun wird der Punkt v_m mit kleinstem κ_m gelöscht (also der letzte Punkt in dieser sortierten Liste). Dadurch

[1] eine Kostenfunktion

[2] Allgemein: zu seinen Nachbarn $v_{i-l}, v_{i-l+1}, \dots v_{i-1}, v_{i+1}, v_{i+2} \dots v_{i+l}$, mit konstantem l.

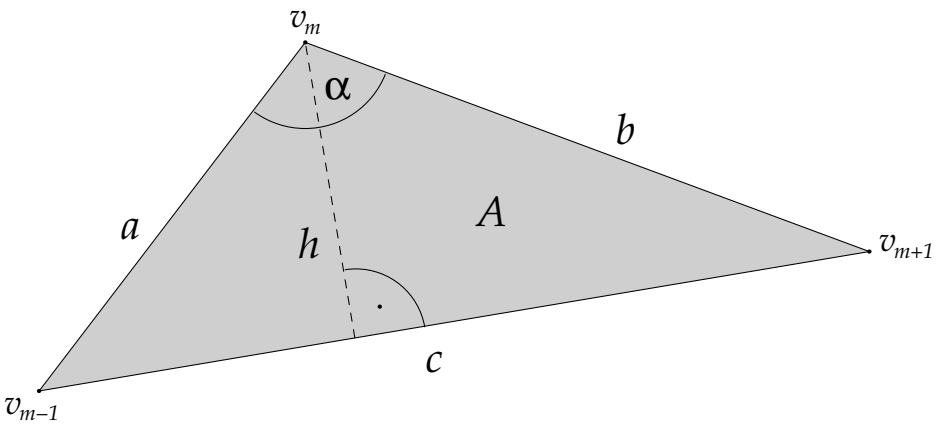


Abbildung 10.1.: Bestimmung des Gewichtes κ_m zum Punkt v_m

ändern sich die Maße κ_i der meisten Punkte nicht, lediglich κ_{m-1} und κ_{m+1} müssen neu bestimmt werden.[3] Folglich müssen nun noch v_{m-1} und v_{m+1} neu einsortiert werden, dies geht in $O(\log n)$ (Algorithmus 14).

Die konkrete Ausprägung des Algorithmus hängt nun nur noch von der Wahl der Gewichtungsfunktion weight ab.

10.2. Wichtige Punkte

Wann ist ein Punkt – bezüglich seiner Nachbarn – wichtig? Ein Punkt v_m ist *unwichtig*, wenn er nichts zum Bild der Bewegungsspur beiträgt. Für $v_m \in [v_{m-1}v_{m+1}]$ gilt daher: $\kappa_m = 0$. Wie *wichtig* ein Punkt, der nicht auf dieser Strecke liegt, bezüglich seiner Nachbarn ist, scheint anschaulich klar: je weiter von $[v_{m-1}v_{m+1}]$ entfernt, desto wichtiger. Es gibt nun eine ganze Reihe von Möglichkeiten, diesen Abstand messen, wie Abbildung 10.1 zeigt:

- $\kappa_m = \text{dist}(v_m, [v_{m-1}v_{m+1}])$: Der Abstand des Punktes v_m zur Strecke $[v_{m-1}v_{m+1}]$.
- $\kappa_m = \text{dist}(v_m, \overline{v_{m-1}v_{m+1}})$: Der Abstand des Punktes v_m zur Geraden $\overline{v_{m-1}v_{m+1}}$.
- $\kappa_m = A_{\triangle v_{m-1}v_m v_{m+1}}$: Die Dreiecksfläche.
- $\kappa_m = \dfrac{|a| + |b|}{|c|}$: Längenverhältnis.
- $\kappa_m = 180^\circ - \alpha$: Der Drehwinkel.

[3] Allgemein: bestimmt sich das Maß eines Punktes aus seiner Lage zu seinen l linken und l rechten (direkten) Nachbarn, so müssen $\kappa_{m-l}, \kappa_{m-l+1}, \ldots \kappa_{m-1}, \kappa_{m+1} \ldots \kappa_{m+l}$ neu bestimmt werden.

Algorithmus 14: Generalisierung mit lokalen Gewichten

```
1: function delpoints = (Punkte K, Gewicht κ) → Punkte :
2: begin
3:    v_m ← last(K);                    ▷ Punkt mit kleinstem Gewicht holen
4:    if κ_m > κ then                   ▷ Schwellwert erreicht
5:       return K;
6:    else
7:       K ← K ∖ {v_{m−1}, v_m, v_{m+1}};   ▷ Punkte löschen
8:       if m > 2 then                  ▷ v_{m−1} ist kein Randpunkt
9:          κ_{m−1} ← weight(v_{m−2}, v_{m−1}, v_{m+1});
10:      end if
11:      if m < n − 1 then              ▷ v_{m+1} ist kein Randpunkt
12:         κ_{m+1} ← weight(v_{m−1}, v_{m+1}, v_{m+2});
13:      end if
14:      insert(v_{m−1},K);             ▷ v_{m−1} wieder neu in K einsortieren
15:      insert(v_{m+1},K);             ▷ v_{m+1} wieder neu in K einsortieren
16:      return delpoints(K,κ);
17:   end if
18: end

19: function weightgen = (Polygonzug p = v_1 ... v_n, Gewicht κ) → Polygonzug :
20: begin
21:   if n ≤ 2 then                     ▷ wir sind fertig
22:      return p;
23:   else
24:      for all v_i ∈ {v_2 ... v_{n−1}} do
25:         κ_i ← weight(v_{i−1}, v_i, v_{i+1});
26:      end for
27:      κ_1 ← ∞;
28:      κ_n ← ∞;
29:      K ← sort_κ(v_1, v_2, ..., v_n);   ▷ Punkte nach ihren Gewichten κ_i sortieren
30:      K ← delpoints(K, κ);
31:      return polygonzug(K);          ▷ Polygonzug aus den restlichen Punkten
32:   end if
33: end
```

Im obigen Algorithmus ist das Gewicht jedes Punktes v_i nur von seinen direkten Nachbarn v_{i-1} und v_{i+1} abhängig, die kanonische Erweiterung auf l linke und l rechte Nachbarn $v_{i-l}, v_{i-l+1}, \ldots, v_{i-1}, v_{i+1}, \ldots, v_{l+m}$ sollte klar sein.

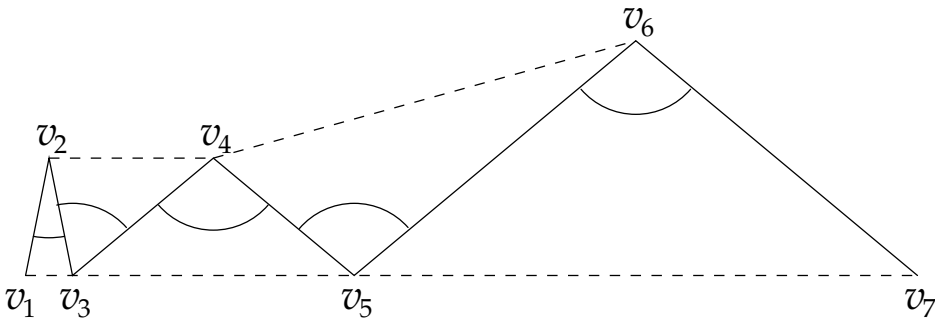


Abbildung 10.2.: Vergleich unterschiedlicher Gewichte

- $\kappa_m = \frac{|c|}{\alpha}$: Der eingeschlossene Winkel relativ zur Strecke.
- $\kappa_m = \ldots$

Im Grunde können hier beliebige Maße eingesetzt werden, das ganze in Abschnitt 5.6 vorgestellte Instrumentarium steht zur Verfügung. Und auch hier stellt sich wieder die Frage, wie unterschiedlich skalierte Dreiecke zu gewichten sind. Bestimmt man die Gewichte der Punkte $v_2, v_3, \ldots, v_6$ aus Abbildung 10.2 auf Basis des zugehörigen Winkels, so besitzt v_2 das größte Gewicht, gefolgt von v_3, und v_4, v_5 und v_6 besitzen das gleiche Gewicht und sind am „unwichtigsten“. Benutzt man stattdessen das Längenverhältnis (die Ellipse), so besitzen v_4 und v_6 weiterhin das gleiche Gewicht, nun ist aber v_5 am „unwichtigsten“.

Ganz anders sieht die Situation hingegen aus, wenn man ein reines Abstandsmaß benutzt, hier ist plötzlich v_6 am wichtigsten, gefolgt von v_5, und v_2, v_3 und v_4 sind gleich unwichtig. Weder das eine noch das andere Extrem scheint wünschenswert. Ein vernünftiges Maß sollte einen Kompromiß darstellen, so daß sowohl v_6 stärker gewichtet wird als v_4 als auch v_2, v_3 und v_4 unterschiedliche Gewichte besitzen.

Die Ergebnisse der Betrachtungen aus Abschnitt 5.6 gelten in beschränktem Maß auch für die Verwendung in weightgen, allerdings mit dem Unterschied, daß die Betrachtung vorne ausgehend von einer durch Start- und Endpunkt gegebenen Referenzstrecke $[v_a v_b]$ angibt, ob die dazwischenliegenden Punkte innerhalb einer bestimmten Toleranz liegen oder nicht, wogegen hier das Verhältnis dreier benachbarter Punkte zueinander bestimmt wird. Weiterhin gibt vorne ein Schwellwert an, ob ein bestimmtes Segment weiter bearbeitet werden muß oder nicht (liegen alle Punkte innerhalb der Toleranz, so ist das Segment fertig bearbeitet), wogegen das Maß hier eine Sortierung der Punkte bestimmt, wobei sich durch Entfernen eines Punktes das Maß seiner Nachbarn ändert.

Dies bedeutet, daß zwar die grundlegenden Überlegungen ähnlich sind (z. B. Form von Akzeptanzbereichen etc.), die Auswirkungen auf oder die Eignung zur Generalisierung durchaus unterschiedlich sein können.

10.2.1. Discrete Curve Evolution

Zur Bestimmung von Ähnlichkeiten zweidimensionaler Objekte vergleichen Longin Jan Latecki und Rolf Lakämper deren Umrißlinien. Hierzu müssen diese Polygone geglättet werden, um Rauschen zu filtern. Sie geben hierzu einen Generalisierungsalgorithmus „Discrete Curve Evolution“ (DCE) an,[4] der auf den oben erwähnten Schritten beruht. Die in [LL99a] in Pseudocode angegebene „Curve Evolution Procedure“ entspricht dem oben beschriebenen Grundgerüst und besitzt damit eine Komplexität von $O(n^2)$ (obwohl wie vorne beschrieben $O(n \log n)$ problemlos möglich ist). Die Autoren geben leider auch nicht an, ob ihre tatsächliche Implementierung anders aussieht und welche Komplexität sie besitzt. Auf die Wahl ihrer Kostenfunktion

$$\kappa_m = \frac{\beta \cdot |a| \cdot |b|}{|a| + |b|} = \frac{(180^\circ - \alpha) \cdot |a| \cdot |b|}{|a| + |b|}$$

gehen sie im selben Text dagegen genauer ein. Die Formel beruht auf der einfachen Idee, daß ein Punkt um so wichtiger für die Kurve ist, um so mehr Aufwand nötig ist, ihn „flachzudrücken“: β ist der Winkel, um den b gegenüber a verdreht ist. Die Strecken a und b müssen also um β_a und β_b (mit $\beta_a + \beta_b = \beta$) gedreht und um $\frac{|c|}{|a|+|b|}$ gestaucht werden, damit $v'_m \in [v_{m-1} v_{m+1}]$.

Schließlich normieren sie die Längen der Strecken a und b noch bezüglich der Länge l des Gesamtpolygonzuges. Damit bekommt man

$$\kappa_m = \frac{\beta \cdot \frac{|a|}{l} \cdot \frac{|b|}{l}}{\frac{|a|}{l} + \frac{|b|}{l}} = \frac{1}{l} \cdot \frac{\beta \cdot |a| \cdot |b|}{|a| + |b|}.$$

Für den Vergleich der Gewichte zweier Punkte des selben Polygonzuges ändert dies also nichts, da der Faktor $\frac{1}{l}$ in allen Fällen gleich ist. Allerdings ändern sich die Gewichte aller Punkte ständig bei jeder Löschung eines Punktes aus dem Gesamtpolygonzug. Dies bedeutet nicht, daß man ständig alle Gewichte neu berechnen muß, da die Sortierung ja gleich bleibt, d. h. man berechnet κ_i ohne den Faktor $\frac{1}{l}$ und fügt ihn lediglich zum Vergleich (Algorithmus 14, Zeile 4) ein. Auch die Bestimmung der aktuellen Länge l geht in konstanter Zeit: $l \leftarrow l - |a| - |b| + |c|$.

Es scheint unklar, wozu Latecki und Lakämper diese Normierung benötigen, die lediglich das Erreichen des Schwellwertes beeinflußt. Sie arbeiten überhaupt nicht mit einem Schwellwert, und führen auch sonst keine Berechnungen durch, die von einem konstanten Faktor $\frac{1}{l}$ beeinflußt würden.[5] Sie benötigen die Normierung später, um unterschiedliche Kurven zu vergleichen, aus meiner Sicht wäre es allerdings geschickter, diese Normierung erst am Ende vorzunehmen.

[4] [BLR00, LL00, LL99b, LL99a]

[5] der lediglich zusätzlichen Aufwand erzeugt

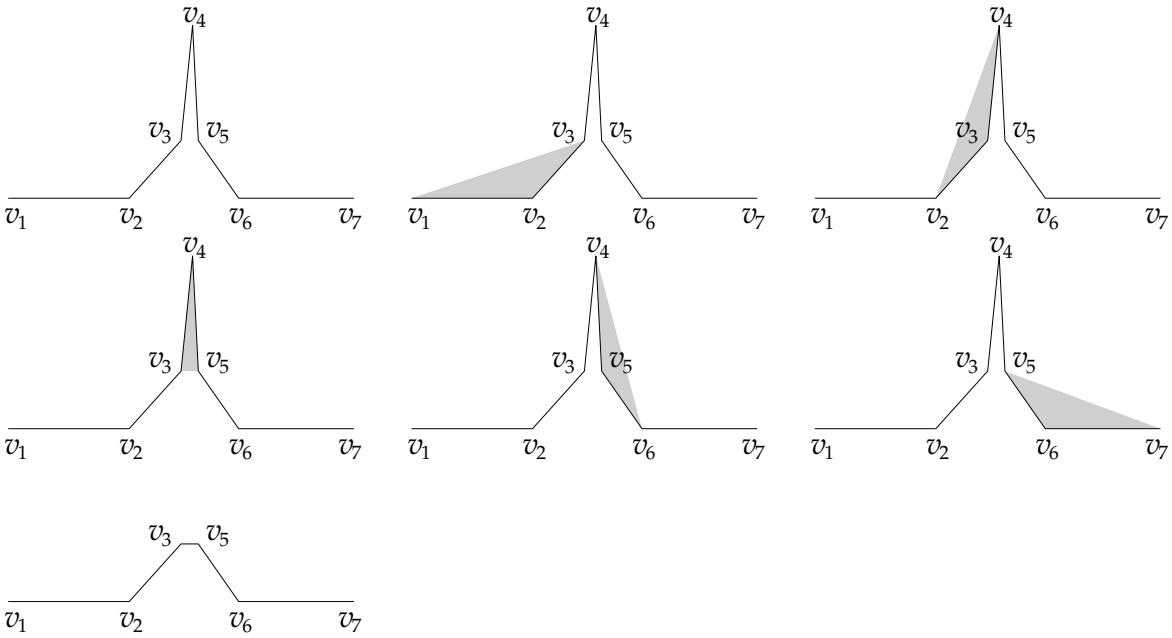


Abbildung 10.3.: weightgen mit $\kappa_m = A_{\triangle v_{m-1} v_m v_{m+1}}$

10.2.2. Visvalingam

Mahes Visvalingam beschreibt in [VW93] einen Generalisierungsalgorithmus für Linienzüge auf geographischen Karten, der seiner Struktur nach Alg. 14 entspricht. Als Gewichtsfunktion benutzt sie die Fläche unter dem Dreieck $\kappa_m = A_{\triangle v_{m-1} v_m v_{m+1}}$. Sie argumentiert, daß die Douglas/Peucker-Generalisierung als „der Generalisierungsalgorithmus der Geographie“ zwar gut geeignet ist, die Feinstruktur einer Bewegung zu unterdrücken, hingegen die *wichtigen* Punkte einer Bewegung nur schlecht selektiert.

Visvalingam und Latecki/Lakämper scheinen gegenseitig nichts von ihren bis auf die Gewichtsfunktion ähnlichen Ansätzen zu wissen, zumindest zitieren sie sich nicht untereinander.

10.2.3. Auswahl der Gewichtsfunktion

Die Wahl der Gewichtsfunktion hängt davon ab, welche Effekte erreicht werden sollen. Daher kann es kein „bestes“ Maß geben, lediglich ein gutes oder schlechtes in Hinblick auf ein bestimmtes Kriterium. Dies zeigt sich schon darin, daß das von Visvalingam vorgeschlagene Flächenmaß zwar für die geographische Anwendung gut funktioniert, für die Generalisierung von Bewegungsspuren aber wenig brauchbar ist. Das Argument ist hier im Grunde das gleiche wie schon in Abschnitt 5.6.2: Bei Betrachtung einer Bewegungsspur interessiert, wo sich das be-

trachtete Objekt befand, und nicht, welche Fläche dabei umfahren wurde.

Gleichzeitig zeigt das Flächenmaß aber auch, daß man die Erkenntnisse aus dem Vergleich der Flächenmaße aus Abschnitt 5.6 nicht ungeprüft übernehmen kann. Die Dreiecksfläche $\triangle v_{m-1}v_m v_{m+1}$ sagt wesentlich mehr über die Lage der drei Punkte zueinander aus als die Fläche zwischen einer Strecke $[v_1 v_n]$ und dem Polygonzug $v_1 v_2 \ldots v_n$ über die Lage der Punkte $v_2, v_3, \ldots, v_{n-1}$.

Abbildung 10.3 zeigt die Generalisierung einer Bewegungsspur mittels weightgen und Flächenmaß: Der doch recht deutliche Abstecher im Punkt v_4 wird als erstes weggeneralisiert. Dieser wie gesagt zur Generalisierung von Küstenlinien durchaus gewünschte Effekt disqualifiziert dieses Maß für die Bewegungsgeneralisierung.

Bessere Ergebnisse liefern der Drehwinkel, das Längenverhältnis, der Abstand zur Strecke (ε-Umgebung) und auch das DCE-Maß (Abschnitt 10.2.1). Beim Douglas/Peucker-Algorithmus hatte das ε-Maß den Vorteil, daß die resultierende Generalisierung q gegenüber dem Originalpolygonzug p das ε-Kriterium erfüllt. Für diesen Algorithmus ist das nicht der Fall, womit das ε-Maß hier keine Sonderstellung besitzt. Der Drehwinkel hat wie schon bei der Douglas/Peucker-Generalisierung den Nachteil, daß sehr kleine (aber dafür spitze) Rückläufigkeiten als sehr wichtig angesehen werden.

Die Maße unterscheiden sich neben der Form ihrer Akzeptanzbereiche vor allem darin, wie sie skalieren. Während beim ε-Maß die Änderung der Länge der Grundlinie c keine Auswirkungen auf die Entfernung ε hat, verändert sich beim Längenverhältnis und beim Drehwinkel diese Entfernung linear: Ist die Grundlinie doppelt so lang, hat ein Punkt, der doppelt so weit entfernt liegt, das gleiche Gewicht.[6] Das DCE Maß liegt dazwischen, für kleine Drehwinkel verhält es sich annähernd linear, für große nicht mehr, die Auswirkung des Winkels β wird durch die Tangens-Funktion bestimmt.

Wie schon oben erwähnt, sollte ein geeignetes Maß irgendwo zwischen konstant und linear skalierend liegen, das ε-Maß und das Längenverhältnis liefern also gewissermaßen die „Ränder“ dieses Bereiches. Neben DCE lassen sich leicht noch weitere Maße finden, so z. B. $\kappa_m = \frac{\mathrm{dist}(v_m, [v_{m-1}v_{m+1}])}{\sqrt{|v_{m-1}v_{m+1}|}}$, das Prinzip sollte klar sein.

Abbildung 10.4 zeigt die Generalisierung einer Bewegungsspur mit fünf unterschiedlichen Maßen: Winkel, Abstand, Ellipse, DCE und Fläche. Um die Generalisierungen vergleichen zu können, wurde nicht bei Erreichen eines vorgegebenen Schwellwertes sondern bei einer vorgegebenen Punktzahl abgebrochen (die Originalkurve besteht aus 61 Punkten, die Generalisierung jeweils aus 13 Punkten (ca. 20%)). Die Abbildung liefert einen ersten Eindruck, wie unterschiedlich die Generalisierung mit unterschiedlichen Maßen ausfallen kann, speziell fällt auf, daß die Generalisierung mittels Flächenmaß (Visvalingam) die Schleife am Ende der

[6] wobei der Punkt natürlich relativ zur Grundlinie an der gleichen Stelle liegen muß.

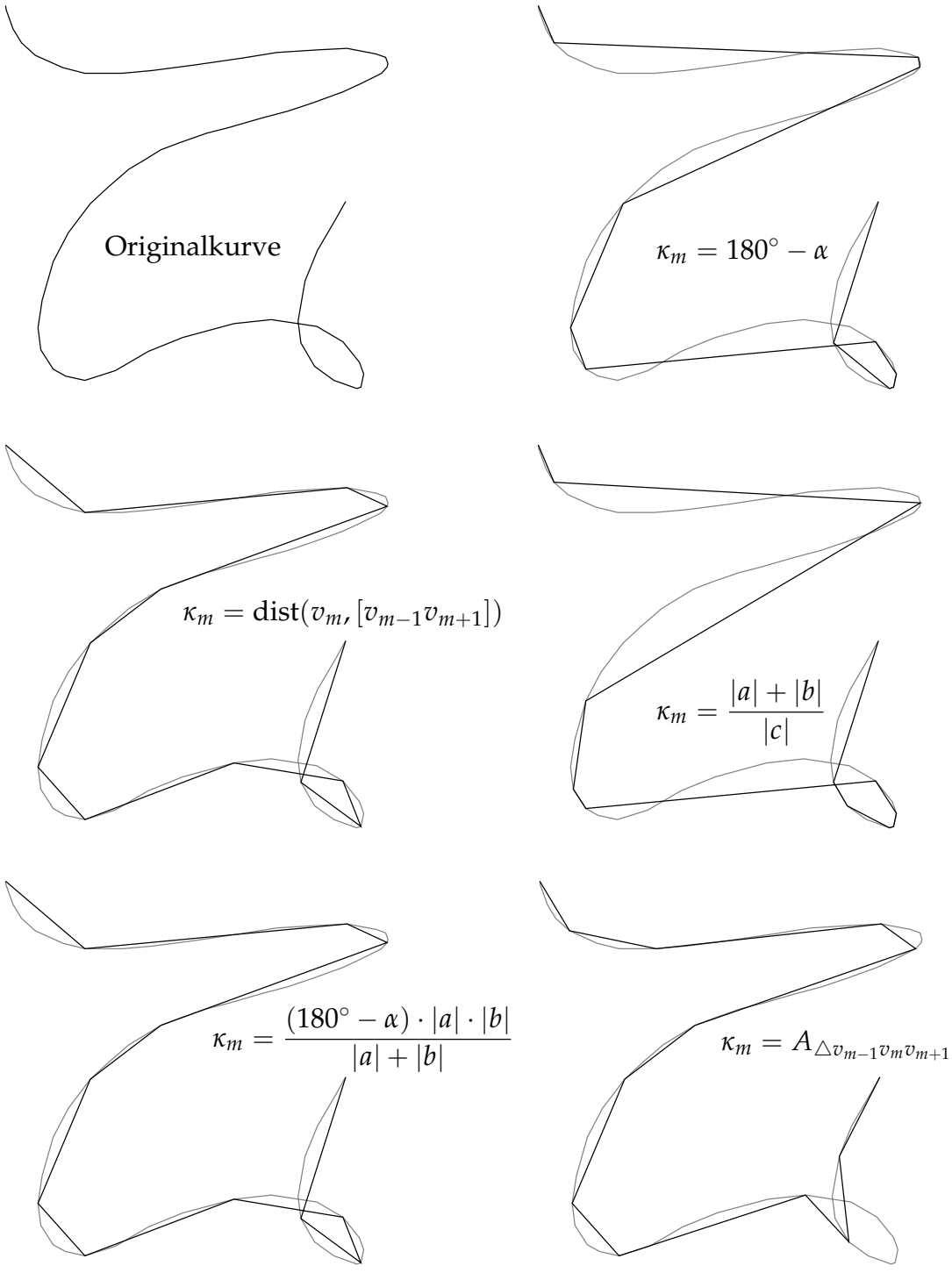


Abbildung 10.4.: Vergleich unterschiedlicher κ_m

Spur nur unzureichend repräsentiert. Das Abstandsmaß sowie DCE geben hier die Form der Spur am besten wieder, während sowohl das Winkel- als auch das elliptische Maß teilweise stark vom Kurvenverlauf abweichen.

Welches Maß letztlich das „beste“ ist, hängt dabei sowohl von der Form der Polygonzüge als auch den Anforderungen an die Generalisierung ab: Wie man auch in der Abbildung sieht, werden geschwungene Formen sowohl vom Winkel- als auch von elliptischen Maß eher schlecht wiedergegeben, wogegen spitze Kehren gut erkannt werden. In der Verarbeitung geschwungener Bewegungsverläufe liefern also unter Umständen andere Maße gute Ergebnisse als in der Verarbeitung eckiger Bewegungen mit scharfen Kurven.

10.3. Geraderücken bei gleichbleibender Eckenzahl

Um aus einem egozentrisch gemessenen Polygonzug eine QMV-Sequenz zu erzeugen, wurden in Abschnitt 3.3.2 unterschiedliche Ansätze vorgestellt. Um die dort beschriebenen Treppeneffekte zu vermeiden, bietet sich eine Vorgeneralisierung an, die dafür sorgt, daß nur leicht gegeneinander verdrehte aufeinanderfolgende Vektoren nicht mehr auftreten können. Wählt man nun $\kappa_m = 180^\circ - \alpha$ und Schwellwert $\kappa = 10^\circ$, so treten im generalisierten Polygonzug nur Drehwinkel größer 10° zwischen aufeinanderfolgenden Vektoren auf.

Eine mittels dieser Vorgeneralisierung erzeugte QMV-Sequenz repräsentiert den Originalpolygonzug jedoch nur unzureichend, da hier nicht ein numerischer MV der Originalsequenz auf einen QMV abgebildet wird, die Vorgeneralisierung faßt ja schon mehrere MV zusammen.

Gesucht ist also ein Verfahren, das aufeinanderfolgende Vektoren, deren Drehwinkel kleinergleich κ ist, geraderückt, ohne dabei Punkte zu entfernen. Hierzu wird einfach wie gehabt generalisiert, die dabei weggeneralisierten Punkte werden dabei aber nicht gelöscht sondern weiter mitgeführt. Die folgende Beschreibung arbeitet (um konsistent zu bleiben) wie die bisherigen Beschreibungen auf einem als Punktfolge gegebenen Polygonzug. Eine egozentrisch gemessene MV-Sequenz ist als Vektorfolge gegeben, auf selbiger sind die folgenden Berechnungen noch einfacher und mit weniger Aufwand möglich.

10.3.1. Schrittweises Vorgehen

Sei κ_k das kleinste Gewicht ($\kappa_m \leq \kappa$), v_k also der „unwichtigste“ Punkt der Bewegungsspur, die Folge $v_{k-1}v_kv_{k+1}$ wird zu $v_{k-1}v_{k+1}$ begradigt. Statt nun v_k einfach zu löschen, wird v_k in die Strecke $[v_{k-1}v_{k+1}]$ eingebettet. Die Lage des eingebetteten Punktes v'_k auf der Strecke ergibt sich dabei aus der Länge der Strecken $[v_{k-1}v_k]$

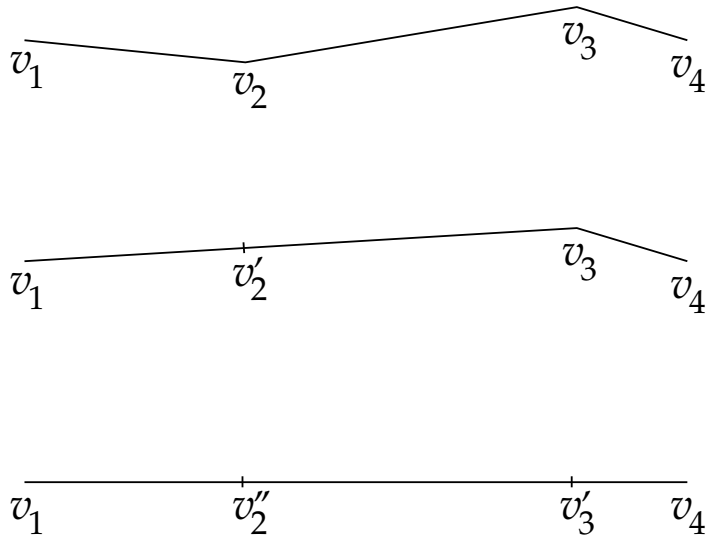


Abbildung 10.5.: Geradeziehen des Polygonzuges $v_1v_2v_3v_4$

und $[v_kv_{k+1}]$:[7]

$$\delta_k = \frac{|v_{k-1}v_k|}{|v_{k-1}v_k| + |v_kv_{k+1}|} = \frac{|v_{k-1}v'_k|}{|v_{k-1}v_{k+1}|} \qquad \text{und} \qquad \frac{|v_kv_{k+1}|}{|v_{k-1}v_k| + |v_kv_{k+1}|} = \frac{|v'_kv_{k+1}|}{|v_{k-1}v_{k+1}|}$$

v'_k geht nicht mehr in die Generalisierung ein sondern wird nur weiter mitgeführt. Nun kommt es vor, daß in einem späteren Schritt der Punkt v_{k-1} entfernt wird. v'_{k-1} berechnet sich wie gezeigt, zusätzlich muß nun aber auch v'_k erneut angepaßt werden: Die Strecke $[v_{k-1}v_{k+1}]$ wird zu $[v'_{k-1}v_{k+1}]$ verkürzt, also muß nun v'_k erneut verschoben werden usw. Da solche Verschiebungen häufiger auftreten, wird nicht in jedem Schritt die neue Position bestimmt, es wird lediglich das Längenverhältnis berechnet.

Abbildung 10.5 zeigt den Polygonzug $v_1v_2v_3v_4$. Im ersten Schritt wird der Punkt v_2 begradigt. Das Seitenverhältnis beträgt $\delta_2 = \frac{|v_1v_2|}{|v_1v_2|+|v_2v_3|} = 0.398$, und dieser Wert wird zunächst gespeichert. Im nächsten Schritt wird der Punkt v_3 begradigt: $\delta_3 = \frac{|v_1v_3|}{|v_1v_3|+|v_3v_4|} = 0.828$. Damit kann nun auch δ_2 angepaßt werden: $\delta_2 \leftarrow \delta_2\delta_3 = 0.398 \cdot 0.828 = 0.330$.

Das allgemeine Schema lautet also: Wird die Punktfolge $v_av_bv_c$ zu v_av_c begradigt, so wird zunächst δ_b aus dem Längenverhältnis der Strecken $[v_av_b]$ und $[v_bv_c]$ bestimmt und dann die δ_i aller Punkte zwischen v_a und v_b mittels $\delta_i \leftarrow \delta_i \cdot \delta_b$ und die δ_j der Punkte zwischen v_b und v_c mittels $\delta_j \leftarrow \delta_j \cdot (1 - \delta_b)$ neu berechnet.

Schließlich werden am Ende der Generalisierung die Positionen der begradigten Punkte bestimmt und man bekommt einen begradigten Polygonzug mit ebenso

[7] Natürlich ist eine der beiden Gleichungen ausreichend. Im Fall $v_{k-1} = v_{k+1}$ wird $v'_k = v_{k-1}$ gewählt.

vielen Punkten wie der Originalpolygonzug. Da das Verfahren wie oben beschrieben nur ein Zwischenschritt zu einer egozentrischen QMV-Darstellung ist, kann man sich die Erzeugung dieses Polygonzuges ersparen und die Umwandlung direkt vornehmen. Um aus einem numerischen MV einen QMV zu bestimmen, müssen Länge und Richtung bestimmt werden. Seien nun v_a und v_b zwei aufeinanderfolgende Punkte der Generalisierung und δ_{a+1} mit δ_{b-1} die angepaßten Längenverhältnisse der dazwischenliegenden begradigten Punkte. Alle Vektoren $v_{a+1} - v_a$ mit $v_b - v_{b-1}$ besitzen die gleiche Richtung, die Länge des Vektors $v_{i+1} - v_i$ bestimmt sich aus $|v_a v_b| \cdot (\delta_{i+1} - \delta_i)$.

Rechnet man direkt auf Vektorsequenzen (der Normalfall bei der Verarbeitung egozentrisch gemessener Bewegungsspuren), sind eine ganze Reihe von Umrechnungen und Längenbestimmungen nicht mehr nötig, was das Verfahren vereinfacht.

10.3.2. Nachträgliches Einfügen

Im oben beschriebenen schrittweisen Geraderücken eines Polygonzuges werden die Längenverhältnisse der einzelnen Punkte ständig neu berechnet. Effizienter ist es, zunächst die Generalisierung eines Polygonzuges zu bestimmen und dann die übrigen Punkte wieder einzufügen. Für *alle* eckentreuen Generalisierungen liefert das folgende Vorgehen einen „geradegerückten“ Polygonzug:

Sei $p = v_1 v_2 \dots v_n$ der Originalpolygonzug und $g(p) = v_1 v_{a_2} v_{a_3} \dots v_{a_m} v_n$ (mit $a_i \in \{1, 2, \dots, n\}$ und $a_i < a_{i+1}$) eine (eckentreue) Generalisierung, so bilden die Polygonzüge $p_1 = v_1 v_2 \dots v_{a_2}$, $p_2 = v_{a_2} v_{a_2+1} \dots v_{a_3}$, $\dots$, $p_m = v_{a_m} v_{a_m+1} \dots v_n$ eine (jeweils in den Endpunkten überlappende) Segmentierung von p. Das Segment p_1 wird dabei in der Generalisierung durch die Strecke $s_1 = [v_1 v_{a_2}]$ repräsentiert, das Segment p_2 durch die Strecke $s_2 = [v_{a_2} v_{a_3}]$ usw.

Für jedes Segment werden nun die Punkte $v_{a_i+1}, v_{a_i+2} \dots, v_{a_{i+1}-1}$ in die Strecke $s_i = [v_{a_i} v_{a_{i+1}}]$ eingebettet, wobei ihre neue Lage (wie schon oben in Abschnitt 10.3.1) auf s_i von ihrer relativen Lage auf p_i abhängt. Hierzu wird zunächst die Gesamtlänge l_i des Polygonzuges p_i und dazu die (zu l_i) relative Länge ρ_j aller Teilstrecken $[v_j v_{j+1}]$ bestimmt:

$$l_i = \sum_{j=a_i}^{a_{i+1}-1} |v_j v_{j+1}|, \qquad \rho_j = \frac{|v_j v_{j+1}|}{l_i}.$$

Damit ist das Längenverhältnis δ_j leicht zu bestimmen:

$$\delta_{a_i+1} = \rho_{a_i+1}, \quad \delta_{a_i+2} = \delta_{a_i+1} + \rho_{a_i+2}, \quad \delta_{a_i+3} = \delta_{a_i+2} + \rho_{a_i+3}, \quad \dots$$

Nun müssen genau wie oben (Abschnitt 10.3.1) noch die Positionen dieser Punkte auf s_i bestimmt werden, dies für alle Segmente p_i. Die Ergebnisse beider Verfahren sind identisch.

11. Puffer-Generalisierung

Der in diesem Kapitel vorgestellte Ansatz ist von einem Modell der spatiotemporalen Wahrnehmung beim Menschen inspiriert, das Kerstin Schill und Christoph Zetsche in [SZ95] entwickeln und basiert auf der Idee eines Puffers fester Größe, in dem die Verarbeitung stattfindet.

11.1. Ein Modell spatiotemporaler Wahrnehmung

Die oben zitierte Arbeit beschäftigt sich mit der Frage, wie raumzeitliche Wahrnehmung und damit die Wahrnehmung von Bewegungen im Gehirn überhaupt funktionieren kann. Es scheint inzwischen in der Wahrnehmungspsychologie unstrittig, daß das Auge die Umwelt nicht kontinuierlich sondern zeitlich gerastert wahrnimmt, d. h. eine Bewegung besteht aus einer Folge von Einzelbildern[1] im zeitlichen Abstand von 30 bis 50 ms. Schill und Zetsche schlagen nun (gestützt durch experimentelle Befunde) vor, daß eine Folge solcher Einzelbilder in einem Puffer fester Größe vorliegt, auf den ein paralleler Zugriff möglich ist. Jedes neu hinzukommende Bild verdrängt dabei ein altes aus diesem Puffer.

Der parallele Zugriff auf eine Folge von Bildern erlaubt es nun sehr einfach, die Bewegung eines Objekts als Folge seiner Positionen in den einzelnen Bildern zu erfassen und weiterzuverarbeiten.

In Wahrheit ist die Situation natürlich nicht so einfach, das betrachtete Objekt muß auf jedem Bild identifiziert werden, Relativbewegungen müssen herausgerechnet werden, Bewegungen in Richtung der z-Achse (also auf den Beobachter zu oder von ihm weg) sind gesondert zu behandeln usw. Ich verzichte an dieser Stelle darauf, auf diese Vorverarbeitung genauer einzugehen und beschränke mich auf die Feststellung, daß eine Bewegung auch hier wieder als Folge von Positionen, also stark vereinfacht als Polygonzug vorliegt.[2]

[1] Aus diesem Grund funktioniert auch ein Kinofilm, der bei 24 Bildern pro Sekunde ja ebenfalls Einzelbilder im zeitlichen Abstand von gut 40 ms zeigt. Für eine Biene, deren visuelles System eine weit höhere zeitliche Auflösung besitzt, wäre ein Kinofilm eine Folge stehender Bilder.

[2] Es gibt Hinweise, daß schon in dieser Vorverarbeitung eine Bewegung nicht als Folge von Punkten sondern als Folge von Richtungsvektoren vorliegt (schon aus der Kenntnis des jeweils benachbarten Bildes läßt sich ein Richtungsvektor bestimmen). Dies verändert die Grundidee jedoch nicht.

Auf dieser Idee eines Puffers fester Größe zur Verarbeitung einer Bewegungsspur basieren die folgenden Algorithmen. Dies soll *nicht* andeuten, daß diese Mechanismen in der spatiotemporalen Wahrnehmung des Menschen ebenfalls ablaufen, sie bedienen sich lediglich diese Grundstruktur.

11.2. Sliding Window

Übertragen auf einen Algorithmus sieht das oben beschriebene Modell wie folgt aus: Eine Bewegungsspur, gegeben als Polygonzug $p = v_1 v_2 \dots v_n$ wird sequentiell verarbeitet, indem jeweils eine konstante Anzahl aufeinanderfolgender Punkte $v_k, v_{k+1}, \dots v_{k+m}$ parallel betrachtet wird. Eine Funktion $f(v_k, v_{k+1}, \dots v_{k+m})$ bearbeitet genau diesen Ausschnitt (dieses Fenster der Breite $m+1$), danach wird das Fenster um eine Position verschoben und $f(v_{k+1}, v_{k+2}, \dots v_{k+m+1})$ bestimmt usw. Technisch gesehen arbeitet f also auf einem FIFO-Puffer fester Größe.

Damit lassen sich sehr leicht Algorithmen zur Glättung der Bewegungsspur konstruieren.

11.2.1. Lineare Glättung über mehrere Punkte

Bestimmt sich $w_t = \overline{\text{pos}}(v_k, v_{k+1}, \dots v_{k+m})$ (mit m gerade, $t = k + \frac{m}{2}$) mittels der Glättungsfunktion $\overline{\text{pos}}$, die die mittlere Position aller Punkte liefert, so stellt die Punktfolge w_i eine geglättete Version des Originalpolygonzuges dar.[3]

$$w_t = \frac{1}{m+1} \sum_{i=k}^{k+m} v_i, \qquad \text{also mit } v_i = (x_i, y_i) :$$

$$\bar{x}_t = \frac{1}{m+1} \sum_{i=k}^{k+m} x_i, \qquad \bar{y}_t = \frac{1}{m+1} \sum_{i=k}^{k+m} y_i, \qquad w_t = (\bar{x}_t, \bar{y}_t)$$

Hierbei geht jeder der beteiligten Punkte gleich stark in die Berechnung von w_t ein. Wie stark geglättet wird, wird dabei ausschließlich von der Größe des Fensters beeinflußt.

11.2.2. Parametrisierte Glättung über mehrere Punkte

Allerdings ist nicht einsichtig, wieso zur Bestimmung des Punktes w_t genau die Punkte $v_{t-\frac{m}{2}}$ mit $v_{t+\frac{m}{2}}$ mit gleicher Stärke und alle außerhalb liegenden Punkte gar nicht eingehen. Ebenso denkbar wäre beispielsweise ein Maß, bei dem die 3 zentralen Punkte doppelt so stark eingehen (hier mit einem Puffer der Größe 5):

$$w_t = \frac{1}{8} \left(v_{t-2} + 2v_{t-1} + 2v_t + 2v_{t+1} + v_{t+2} \right),$$

[3] Zur Behandlung von Start und Ende des Polygonzuges siehe Abschnitt 11.2.4

oder allgemein (mit Gewichten α_i):

$$w_t = \frac{1}{\sum_{i=0}^{m} \alpha_i} \sum_{i=0}^{m} \alpha_i v_{k+i}$$

11.2.3. Wahl der α_i

Durch die Wahl der α_i kann die Art der Glättung fast beliebig gesteuert werden. Die Breite des Fensters legt fest, wie viele benachbarte Punkte an der Glättung eines Punktes v_k beteiligt sind, die Wahl der α_i legt fest, wie stark sie dies sind. $\forall i \in \{0, 1, \dots m\} : \alpha_i = 1$ liefert den Spezialfall der oben beschriebenen linearen Glättung.

Üblicherweise wird man die Gewichte α_i symmetrisch wählen, wobei die Gewichte nach außen hin abfallen. Für große Fensterbreiten bietet es sich an, die α_i nicht von Hand festzulegen, sondern mittels einer passenden Funktion (Gaußglocke, Dreiecksfunktion, ...) zu bestimmen.

11.2.4. Start und Ende

Wird auf diese Art ein Polygonzug $p = v_1 v_2 \dots v_n$ bearbeitet, so funktioniert dies für die meisten Punkte v_i wunderbar, lediglich die Punkte am Anfang und Ende von p lassen sich nicht so einfach bearbeiten. Für eine Glättung mit einem Puffer der Größe 5 bestimmt sich

$$w_3 = \frac{1}{\alpha_0 + \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4} \left(\alpha_0 v_1 + \alpha_1 v_2 + \alpha_2 v_3 + \alpha_3 v_4 + \alpha_4 v_5\right).$$

Zur Bestimmung von w_1 und w_2 fehlen jedoch Punkte v_0 und v_{-1}, daher ist hier eine Modifikation nötig:

- Eine Möglichkeit besteht darin, das Fenster am Rand kleiner zu wählen, so daß
$$w_1 = v_1, \qquad w_2 = \frac{1}{\alpha_1 + \alpha_2 + \alpha_3} \left(\alpha_1 v_1 + \alpha_2 v_2 + \alpha_3 v_3\right).$$
Hier werden die Ränder einfach hart abgeschnitten, wodurch sich die relative Gewichtung natürlich ändert. Es bietet sich daher an, für die verkürzten Fenster die Gewichte gesondert festzulegen. Insbesondere, wenn die Gewichte mittels einer Funktion bestimmt werden, ist dies problemlos möglich. Gerade wenn die Gewichte beispielsweise eine Gaußglocke bilden, können die Gewichte im verkleinerten Fenster so angepaßt werden, daß sie eine schmälere (statt einer abgeschnittenen) Gaußglocke bilden.

- Die zweite Möglichkeit besteht darin, $\forall i < 1 : v_i := v_1$ und $\forall i > n : v_i := v_n$ zu setzen. Damit ergibt sich

$$w_1 = \frac{1}{\alpha_0 + \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4} \left((\alpha_0 + \alpha_1 + \alpha_2) v_1 + \alpha_3 v_2 + \alpha_4 v_3 \right).$$

  Allerdings findet hierdurch eine (leichte) Verzerrung des Startpunktes statt.

- Wählt man hingegen für v_0 den an v_1 gespiegelten Punkt v_2: $v_0 := 2v_1 - v_2$, ebenso $v_{-1} := 2v_1 - v_3$ usw. (analog für v_{n+1} etc.), so gilt $w_1 = v_1$, und w_2 wird geglättet aber nicht verzerrt.

11.2.5. Lineare Glättung über Richtungsvektoren

Wie schon in Kapitel 2 erwähnt, läßt sich eine Bewegungsspur nicht nur absolut als Folge von Koordinaten sondern auch relativ als Folge von Bewegungsvektoren darstellen, und auch das oben erwähnte Wahrnehmungsmodell geht davon aus, daß die Bewegung nach der Vorverarbeitung nicht unbedingt als Folge von Koordinaten sondern als Folge von Richtungsvektoren vorliegt.

Liegt die Bewegung nun als Folge von Richtungsvektoren $a_1 a_2 \ldots a_n$ vor (diese Folge kann auch aus einem Polygonzug $p = v_1 v_2 \ldots v_{n+1}$ mittels $a_i := v_{i+1} - v_i$ generiert werden, siehe auch Kapitel 2), so läßt sich über diese Vektoren analog glätten. Hierbei wird nicht die Position eines Punktes sondern die Richtung und Länge eines Richtungsvektors verändert (wie oben: m gerade, $t = k + \frac{m}{2}$):

$$b_t = \frac{1}{m+1} \sum_{i=k}^{k+m} a_i, \quad \text{oder gleichwertig:} \quad b_t = \frac{v_{k+m+1} - v_k}{m+1}.$$

11.2.6. Verschränkte Glättung über Richtungsvektoren

In seiner Diplomarbeit[4] [Rai99] schlägt Hans Raith als Generalisierungsverfahren eine sehr ähnliche Glättung vor. Ausgehend von einer Fenstergröße von l Richtungsvektoren a_k bis a_{k+l-1} bestimmt er den Summenvektor

$$a'_{k,k+l-1} := \sum_{i=k}^{k+l-1} a_i.$$

Ein Richtungsvektor a_t ist dabei Teil der Vektoren $a'_{t-l+1,t}, a'_{t-l+2,t+1}, \ldots a'_{t,t+l-1}$, daher bestimmt er den geglätteten Vektor b_t aus der Überlagerung dieser l Summenvektoren. Da an jedem Summenvektor l Richtungsvektoren beteiligt sind, und l

[4] Eine Arbeit im Rahmen unseres Raumkognitionsprojektes in Zusammenarbeit mit Kerstin Schill vom Institut für medizinische Psychologie an der Uni München und basierend auf der vorne beschriebenen QMV-Repräsentation.

Summenvektoren überlagert werden, bestimmt sich

$$b_t = \frac{1}{l^2} \sum_{i=t-l+1}^{t} a'_{i,i+l-1}.$$

11.2.7. Parametrisierte Glättung über Richtungsvektoren

Betrachtet man diesen Algorithmus genauer, sieht man, daß es sich im Grunde um zwei sich überlagernde Fenster handelt. Ein Summenvektor $a'_{k,k+l-1}$ wird über l Richtungsvektoren gebildet, und $l-1$ Summenvektoren überdecken den gleichen Vektor a_t. Für die Berechnung des geglätteten Vektors b_t gilt:

$$\begin{aligned} b_t &= \frac{1}{l^2} \sum_{i=t-l+1}^{t} a'_{i,i+l-1} = \frac{1}{l^2} \sum_{i=t-l+1}^{t} \sum_{j=i}^{i+l-1} a_j \\ &= \frac{1}{l^2} \left(a_{t-l+1} + 2a_{t-l+2} + 3a_{t-l+3} + \ldots + la_t + (l-1)a_{t+1} + \ldots + a_{t+l-i} \right). \end{aligned}$$

Dies ist aber gleichwertig mit einer gewichteten Glättung über ein Fenster der Breite $s = 2l - 1$. Wir bekommen (mit ungeradem s)

$$b_t = \frac{1}{l^2} \sum_{i=t-l+1}^{t+l-1} (l - |i-t|)a_i = \frac{4}{(s+1)^2} \sum_{i=t-\frac{s-1}{2}}^{t+\frac{s-1}{2}} (\frac{s+1}{2} - |i-t|)a_i.$$

Und dies ist nichts anderes als ein Spezialfall einer parametrisierten Glättung. Analog zu vorne (mit einem Fenster von a_k bis a_{k+m}, m gerade, $t = k + \frac{m}{2}$):

$$b_t = \frac{1}{\sum_{i=0}^{m} \beta_i} \sum_{i=0}^{m} \beta_i a_{k+i}.$$

Liegt die Bewegungsspur als Punktfolge $v_1 v_2 \ldots v_n$ vor, so kann man auch direkt auf den Koordinaten rechnen. $a_i = v_{i+1} - v_i$, somit gilt:

$$\begin{aligned} b_t &= \frac{1}{\sum_{i=0}^{m} \beta_i} \sum_{i=0}^{m} \beta_i a_{k+i} = \frac{1}{\sum_{i=0}^{m} \beta_i} \sum_{i=0}^{m} \beta_i (v_{k+i+1} - v_{k+i}) \\ &= \frac{1}{\sum_{i=0}^{m} \beta_i} \left(\sum_{i=1}^{m+1} \beta_{i-1} v_{k+i} + \sum_{i=0}^{m} -\beta_i v_{k+i} \right) \\ &= \frac{1}{\sum_{i=0}^{m} \beta_i} \sum_{i=0}^{m+1} (\beta_{i-1} - \beta_i) v_{k+i} \qquad (\text{mit} \quad \beta_{-1} := 0, \quad \beta_{m+1} := 0), \end{aligned}$$

was eine gleichzeitige Umwandlung eines (als Punktfolge gegebenen) Polygonzuges in eine Sequenz relativer Vektoren und eine Glättung derselben liefert.

11.2.8. Start und Ende

Für die Ränder der Bewegungsspur gilt das oben gesagte analog.

11.2.9. Beispiele

Abbildung 11.1 zeigt die Glättung einer Bewegungsspur mit einem Fenster über drei ($\beta_0 = \beta_1 = \beta_2 = 1$) bzw. fünf ($\beta_0 = \beta_1 = \beta_2 = \beta_3 = \beta_4 = 1$) gleichgewichtete Vektoren[5] sowie einer stärker gewichteten Mitte ($\beta_0 = \beta_2 = 1, \beta_1 = 2$) bzw. ($\beta_0 = \beta_4 = 1, \beta_1 = \beta_3 = 2, \beta_2 = 3$).

11.2.10. Ein Unterschied?

Auf den ersten Blick scheint die Glättung auf Punktfolgen und auf Vektorsequenzen zwar strukturell ähnlich aber im Ergebnis unterschiedlich zu sein. Eine genauere Betrachtung zeigt, daß beide Ansätze (fast) gleichwertig sind.

Auf einer Punktfolge $v_1 v_2 \dots v_n$ bestimmt sich die Vektorfolge $a_1 a_2 \dots a_{n-1}$ durch $a_i = v_{i+1} - v_i$. Weiterhin berechnen sich

$$w_k = \frac{1}{\sum_{i=0}^{m} \alpha_i} \sum_{i=0}^{m} \alpha_i v_{k+i}, \qquad w_{k+1} = \frac{1}{\sum_{i=0}^{m} \alpha_i} \sum_{i=0}^{m} \alpha_i v_{k+i+1}.$$

Nun gilt:

$$\begin{aligned} w_{k+1} - w_k &= \frac{1}{\sum_{i=0}^{m} \alpha_i} \sum_{i=0}^{m} \alpha_i v_{k+i+1} - \frac{1}{\sum_{i=0}^{m} \alpha_i} \sum_{i=0}^{m} \alpha_i v_{k+i} \\ &= \frac{1}{\sum_{i=0}^{m} \alpha_i} \left(\sum_{i=0}^{m} \alpha_i v_{k+i+1} - \sum_{i=0}^{m} \alpha_i v_{k+i} \right) \\ &= \frac{1}{\sum_{i=0}^{m} \alpha_i} \sum_{i=0}^{m} \alpha_i (v_{k+i+1} - v_{k+i}) \\ &= \frac{1}{\sum_{i=0}^{m} \alpha_i} \sum_{i=0}^{m} \alpha_i a_{k+i} \end{aligned}$$

[5] oder (mit $\alpha_i = \beta_i$) drei bzw. fünf Punkten, wie oben gezeigt sind beide Ansätze gleichwertig

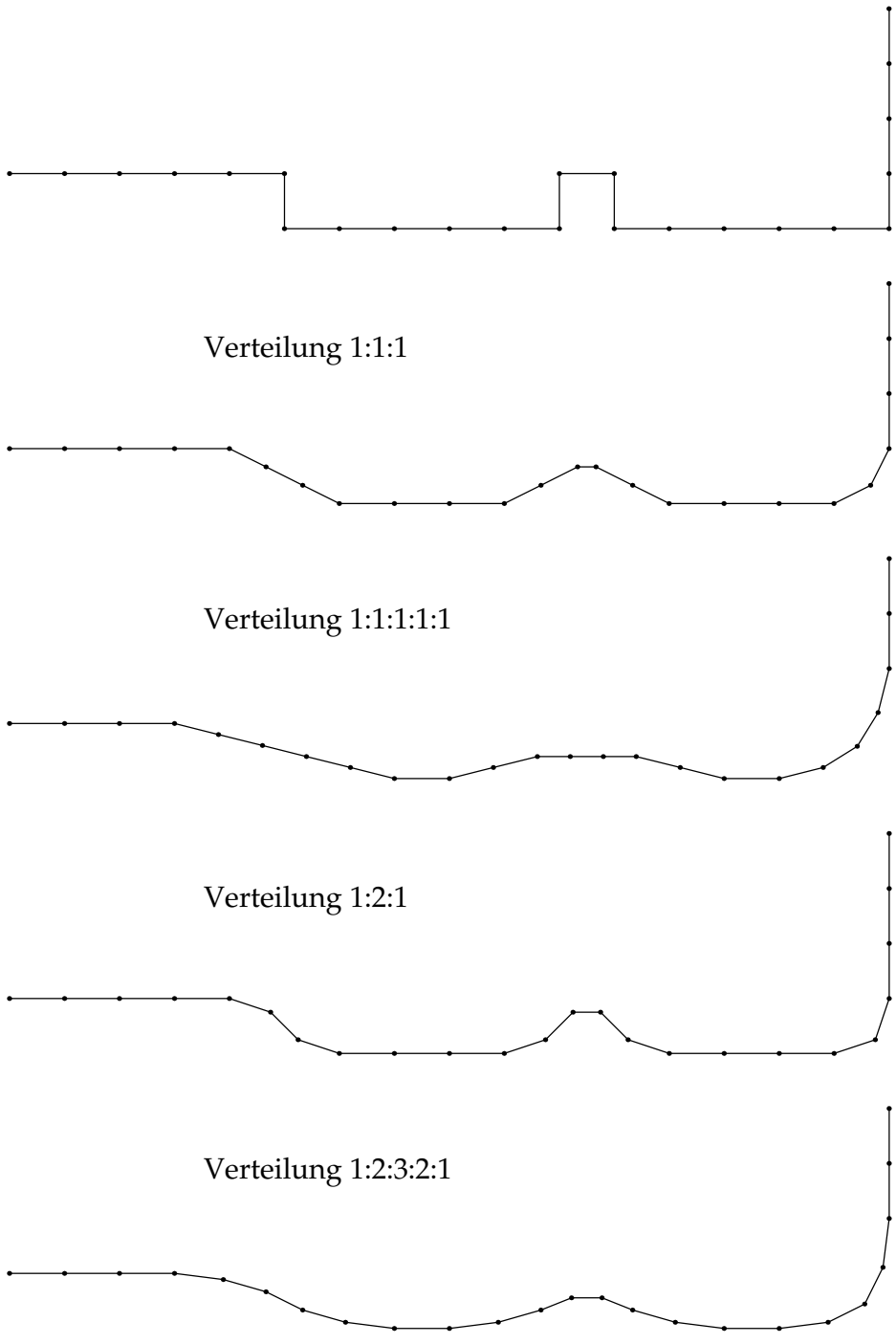


Abbildung 11.1.: Sliding Window mit unterschiedlichen Gewichtungen

Mit $\beta_i = \alpha_i$ und $b_k = w_{k+1} - w_k$ liefert dies die parametrisierte Glättung über Richtungsvektoren, beide Ansätze sind hier also gleichwertig. Der einzige Unterschied besteht in der Behandlung der Randbereiche durch Verkürzung des Fensters. Hier liefern die beiden Ansätze leicht unterschiedliche Ergebnisse.

11.2.11. Komplexität

Für jeden Punkt v_i des Polygonzuges müssen bei einem Puffer der Breite s (unter Vernachlässigung der gesonderten Behandlung von Start und Ende) s Werte addiert werden, der Algorithmus ist also linear in der Länge des Polygonzuges (und in der Breite des Puffers).

11.2.12. „Fensterbreite“

Der beschriebene Algorithmus arbeitet auf einem Puffer fester Größe, d. h. mit einer festen Anzahl von Punkten bzw. Vektoren. Wurde die zu bearbeitende Bewegung mit einem festen Zeitraster gemessen, bedeutet dies, daß der Puffer einen festen Zeitraum umfaßt. Damit ist dieses Verfahren – der Grad der Glättung – allerdings stark abhängig von der Frequenz der Messung bzw. der gefahrenen Geschwindigkeit: Je genauer gemessen (bzw. je langsamer gefahren) wird, desto schwächer ist die Glättung.

Wie schon mehrfach in dieser Arbeit besteht der Ansatz zur Lösung auch hier in einem Wechsel von einem frequenzabhängigen (zeitabhängigen) in ein entfernungsabhängiges Maß. In die Glättung eines Punktes v_k gehen alle Punkte v_{k-a} mit v_{k+b} ein, die entlang des Polygonzuges eine Entfernung kleinergleich eines festgelegten Abstands d besitzen, also

$$a = \max\Big\{x \in \mathcal{N} \mid \sum_{i=1}^{x} |v_{k-i} - v_{k-i+1}| \leq d\Big\},$$

$$b = \max\Big\{y \in \mathcal{N} \mid \sum_{i=1}^{y} |v_{k+i} - v_{k+i-1}| \leq d\Big\}.$$

Nun müssen noch die Gewichte $\alpha_{i,k}$ für die beteiligten Punkte bestimmt werden. Da ja nicht von vornherein klar ist, wie viele Punkte in die Glättung mit eingehen, sollte die Gewichtbestimmung automatisch erfolgen. Nun kann es leicht passieren, daß auf der einen Seite von v_k mehr Punkte innerhalb der Entfernung d liegen als auf der anderen Seite (oBdA: $a > b$). Dadurch bekommt diese Seite ein stärkeres Gewicht, was zu einer unerwünschten Verzerrung führt: Während beim oben beschriebenen Algorithmus die Punkte so verschoben werden, daß auch die Dynamik der Bewegung (also Geschwindigkeitsänderungen) geglättet wird, führt die Verzerrung jetzt zu einer unerwünschten Verstärkung von Beschleunigungseffekten, sie findet also gewissermaßen in die falsche Richtung statt.

Während im ursprünglichen Ansatz auf beiden Seiten gleich viele Punkte in die Glättung eingehen und damit die Seite stärkeres Gewicht bekommt, auf der die Entfernungen zwischen den Punkten größer sind, ist nun die Entfernung auf beiden Seiten ungefähr gleich groß und die Seite, die mehr Punkte enthält (wo also langsamer gefahren wurde), geht stärker in die Berechnung ein.

Um dies zu beheben, werden die Gewichte $\alpha_{i,k}$ wie folgt angepaßt: Zunächst bestimmt sich das Gewicht $\alpha'_{i,k}$ jedes Punktes $v_i \in \{v_{k-a}, \ldots, v_{k+b}\}$ abhängig von seinem Abstand

$$
\begin{aligned}
d_{i,k} &= \sum_{j=i}^{k-1} |v_{j+1} - v_j| \quad (\text{für } i < k) \\
d_{k,k} &= 0 \\
d_{i,k} &= \sum_{j=k}^{i-1} |v_{j+1} - v_j| \quad (\text{für } i > k)
\end{aligned}
$$

mittels einer entsprechenden Funktion[6] bestimmt: $\alpha'_{i,k} = \text{gewicht}(d_{i,k})$. Nun werden abhängig von der Anzahl der Punkte auf der linken (a) und rechten (b) Seite von v_k die Gewichtungen angepaßt:

$$
\alpha_{i,k} = \begin{cases} 2b\alpha'_{i,k} & \text{für } i < k \\ (a+b)\alpha'_{k,k} & \text{für } i = k \\ 2a\alpha'_{i,k} & \text{für } i > k \end{cases}
$$

Das so gewonnene Verfahren ist hinreichend, aber nicht völlig geschwindigkeitsunabhängig: Es berücksichtigt zwar, daß auf der linken und rechten Seite von v_k unterschiedlich viele Punkte an der Glättung beteiligt sind, aber nicht, daß die Abstände zwischen den Punkten einer Seite ja nicht gleich sein müssen, Verschiebungen wirken sich hier leicht in der Glättung aus. Will man auch diese Verschiebungen vermeiden, so muß man ein Gewicht $\alpha_{i,k}$ zusätzlich abhängig vom Abstand des Punktes v_i von seinen Nachbarn wählen. Je weiter ein Punkt von seinen Nachbarn entfernt liegt, desto stärker ist er zu gewichten.

11.2.13. Eigenschaften

Die oben vorgestellten Verfahren verarbeiten zwar Bewegungsspuren, sind aber keine Generalisierungsalgorithmen nach unser Definition,[7] da der Polygonzug nach der Bearbeitung noch genauso viele Punkte besitzt wie zuvor, die Bewegung

[6] Wie oben schon erwähnt, kann sie von der konstanten Funktion bis zur Gaußglocke beliebig gewählt werden.

[7] wenn auch Hans Raith in seiner Arbeit von einer Generalisierung spricht.

wird lediglich geglättet. Damit ist Sliding Window für sich genommen kein Generalisierungsalgorithmus, bildet aber eine geeignete Vorstufe für Algorithmen wie die Σ-Generalisierung (Abschnitt 9.2). Sliding Window glättet die Bewegungsspur, dadurch wird die Feinstruktur unterdrückt und somit stärker generalisiert. Wird direkt auf QMV-Sequenzen gearbeitet, treten hier schon durch die Zusammenfassung gleichgerichteter QMV generalisierende Effekte auf.

Das Verfahren wirkt als „Weichzeichner“ über den Bewegungsverlauf, wobei die Art der Glättung durch Wahl der Fensterbreite und der Gewichte steuerbar ist, wie Abbildung 11.1 zeigt. Ein Nebeneffekt dabei ist, daß Kurven im Bewegungsverlauf durch die Glättung nach „innen“ gezogen werden. Dies ist prinzipbedingt und damit unvermeidbar.

11.3. Sliding Window Generalisierung

Um eine echte Generalisierung zu erreichen, muß in der Verarbeitung eine Punktreduktion erreicht werden. Im oben beschriebenen Verfahren wird die neue Position eines Punktes w_t durch das gewichtete Mittel der $m+1$ ihn umgebenden Punkte bestimmt:

$$w_t = \frac{1}{\sum_{i=0}^{m} \alpha_i} \sum_{i=0}^{m} \alpha_i v_{t-k+i}$$

Damit gibt w_t indirekt die Lage aller dieser Punkte wieder, sie werden durch w_t „repräsentiert“.

11.3.1. Vorgehen

Das legt folgenden Algorithmus nahe: Jeweils $m+1$ Punkte des Originalpolygonzuges werden zu einem Punkt der Generalisierung zusammengefaßt, das Mittel dieser $m+1$ Punkte liefert den korrespondierenden Punkt der Generalisierung. Der erste Punkt der Generalisierung benötigt auch hier eine Sonderbehandlung (ebenso wie der letzte). Wählt man als ersten Punkt der Generalisierung das Mittel des Intervalls v_1 mit v_{m+1}, so kann der Startpunkt der Generalisierung stark vom Startpunkt des Originalpolygonzuges abweichen. Daher wird $w_1 = v_1$ als Startpunkt der Generalisierung gewählt. Die Punkte v_2 mit $v_{1+\frac{m}{2}}$ gehören zum ersten Bereich, werden daher übersprungen.[8] Wir bekommen

$$w_1 = v_1, \quad w_2 = \frac{1}{\sum_{i=0}^{m} \alpha_i} \sum_{i=0}^{m} \alpha_i v_{2+\frac{m}{2}+i}, \quad w_3 = \frac{1}{\sum_{i=0}^{m} \alpha_i} \sum_{i=0}^{m} \alpha_i v_{3+\frac{3m}{2}+i}, \quad \dots$$

[8] Dies entspricht der dritten vorne beschriebenen Möglichkeit (Einführung von „virtuellen“ Punkten $v_0, v_{-1}, \dots$) zur Behandlung von Start und Ende.

Der letzte Punkt des Originalpolygons ist auch letzter Punkt der Generalisierung. Wie schon bei der k-Step-Generalisierung (Abschnitt 9.1) passiert es auch hier meistens, daß das letzte Segment zu kurz ist (die letzten $\frac{m}{2}+1$ Punkte gehören zum Endpunkt). Verzichtet man auf eine inkrementelle Generalisierung, kann man wie bei k-Step die überzähligen Punkte verteilen. Bei inkrementeller Verarbeitung werden entweder die überzähligen Punkte ignoriert (wenn es weniger als $\frac{m+1}{2}$ Punkte sind) oder aus den übrigen Punkten ein verkürztes Segment gebildet (wenn es mehr Punkte sind).

11.3.2. Wahl der α_i

Da jeder Punkt der Generalisierung Repräsentant von $m+1$ Punkten des Originalpolygons ist, sollte auch jeder diese Punkte mit gleichem Gewicht eingehen, damit $\forall \alpha_i : \alpha_i = 1$.

11.3.3. Überlappende Bereiche

Bei der Sliding Window Glättung überlappen die Bereiche maximal, wodurch jeder Punkt des Originalpolygonzuges durch einen Punkt des geglätteten Polygonzuges repräsentiert wird. Bei der gerade beschriebenen Sliding Window Generalisierung überlappen die Bereiche überhaupt nicht, jeweils $m+1$ Punkte des Originalpolygonzuges werden durch einen Punkt in der Generalisierung repräsentiert. Dazwischen sind beliebige Zwischenformen möglich.

Bei einer Überlappung um einen Punkt ist der letzte Punkt des einen Bereiches gleichzeitig erster Punkt des nächsten Bereiches. Da dieser Randpunkt nun durch zwei Punkte der Generalisierung repräsentiert wird, soll er jeweils nur mit halbem Gewicht eingehen, damit gilt:

$$\alpha_0 = \alpha_m = 0.5, \qquad \forall i \in \{1, 2, \ldots, m-1\} : a_i = 1.$$

Bei einer Überlappung um zwei Punkte muß demzufolge gelten:

$$\alpha_0 + \alpha_{m-1} = 1, \qquad \alpha_0 = \alpha_m, \quad \alpha_1 = \alpha_{m-1}, \qquad \forall i \in \{2, 3, \ldots, m-2\} : a_i = 1.$$

Allgemein soll jeder Punkt insgesamt mit Gewicht $\alpha_i = 1$ in die Generalisierung eingehen, geht er in die Bestimmung eines Punktes stärker ein, dann in die Bestimmung des anderen entsprechend schwächer. Die Stärke der Gewichtung ist dabei freigestellt, üblicherweise wird ein Punkt um so stärker eingehen, je weiter er in der Mitte des Intervalls liegt.

Je stärker die Intervalle überlappen, desto stärker glättet und desto weniger generalisiert das Verfahren.

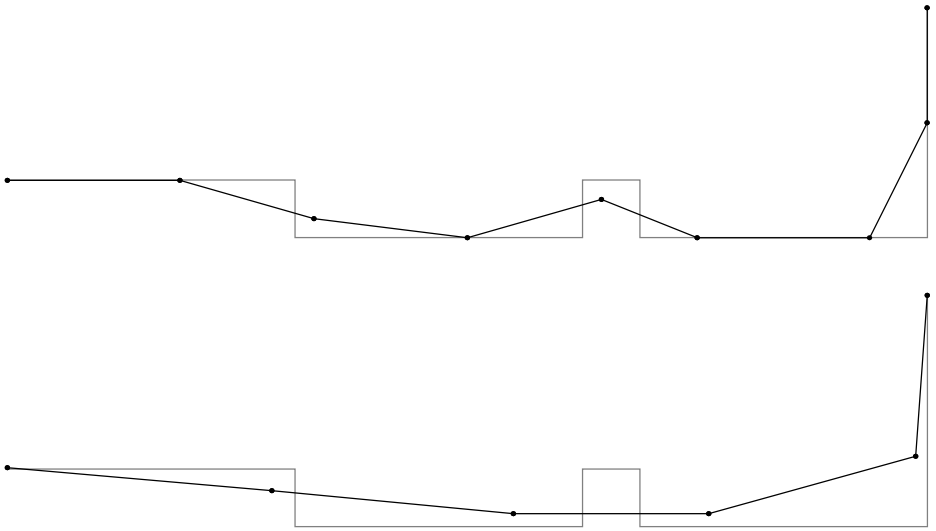


Abbildung 11.2.: Sliding Window Generalisierung mit Fensterbreite 3 und 5

11.3.4. Eigenschaften

Abbildung 11.2 zeigt die Generalisierung des schon aus Abbildung 11.1 bekannten Polygonzuges mit nicht überlappenden Fenstern der Größe 3 und 5. Das Verfahren glättet und generalisiert gleichzeitig. Es ist nicht eckentreu, ähnlich wie bei Convex Hull überlagert die Generalisierung das Originalpolygon. Wie schon die Sliding Window Glättung ist auch die Generalisierung linear (im Vergleich zur Glättung sogar um einen konstanten Faktor schneller).

11.4. Fixed-Size-Buffer-Generalisierung

Die Idee eines Puffers fester Größe bietet sich auch an, um andere, ihrer Struktur nach nicht inkrementelle Algorithmen inkrementell zu machen.

11.4.1. Lokale Gewichtung

Gegeben ist ein Puffer der Größe s, der mit dem Punkten v_1 bis v_s des Polygonzuges $p = v_1 v_2 \dots v_n$ belegt ist. Für jeden Punkt v_i im Puffer wird (analog zur Generalisierung mit lokaler Gewichtung, Kapitel 10) ein Maß κ_i bestimmt, das angibt, wie „wichtig“ der jeweilige Punkt ist (für die Randpunkte v_a und v_e gilt $\kappa_a = \kappa_e = \infty$, da die Gewichte ja aufgrund der Lage eines Punktes zu seinem linken und rechten Nachbarn bestimmt wird und diese Punkte auf einer Seite keinen Nachbarn besitzen).

Nun wird der „unwichtigste“ Punkt v_j im Puffer gelöscht, sofern κ_j unterhalb

eines Schwellwertes κ liegt, die Gewichte der Nachbarn neu bestimmt und der nächste Punkt v_{s+1} eingelesen. Es werden so lange Punkte gelöscht und neue eingelesen, bis alle Punkte im Puffer „wichtiger“ als κ sind. Da jetzt kein Punkt mehr entfernt werden kann, wird der erste Punkt aus dem Puffer geschoben und ein neuer Punkt nachgeholt. Der neue Punkt ist nun Randpunkt und bekommt damit das Gewicht ∞ und nun kann das Gewicht des alten Randpunktes berechnet werden.

Damit gilt folgender Ablauf:

- Der Puffer wird mit den ersten s Punkten des Originalpolygonzuges gefüllt.
- Im Puffer wird generalisiert, in jedem Generalisierungsschritt wird ein Punkt aus dem Puffer gelöscht und in den freiwerdenden Platz ein neuer Punkt des Originalpolygonzuges nachgeholt.
- Kann im Puffer nicht mehr generalisiert werden (alle Gewichte liegen über einen Schwellwert), wird der erste Punkt aus dem Puffer geschoben und ein neuer nachgeholt.

Dieses Vorgehen liefert eine inkrementell arbeitende Generalisierung, die allerdings das in 4.2 vorgestellte Kriterium nicht ganz erfüllt, da sie nicht segmentweise arbeitet.

Betrachtet man den Ablauf der Generalisierung genauer, bemerkt man, daß die Größe des Puffers wenig Einfluß auf sie hat. Zu Beginn liegen die ersten s Punkte des Originalpolygonzuges im Puffer und werden gleichmäßig generalisiert, wobei immer neue Punkte nachgeholt werden.[9] Zu dem Zeitpunkt, zu dem der erste Punkt aus dem Puffer herausgeschoben wird, besitzen alle Punkte im Puffer Gewichte, die über dem Schwellwert liegen, können also nicht generalisiert werden (das ist ja genau das Kriterium, das angibt, daß nun der erste Punkt aus dem Puffer geschoben wird.

Der neu nachgeholte Punkt ist nun Randpunkt, damit kann nun allenfalls der vorletzte Punkt im Puffer generalisiert werden. Wird er entfernt, so berechnet sich das Gewicht des vorvorletzten Punktes neu, theoretisch können nun also nach und nach wieder Punkte in der Mitte des Puffers Gewichte unterhalb des Schwellwertes erhalten, dies ist aber unwahrscheinlich. In der Praxis findet somit Generalisierung unabhängig von der Puffergröße fast ausschließlich am Ende des Puffers statt, große Puffergrößen ($s > 5$) sind somit sinnlos und verzögern nur die inkrementelle Verarbeitung.

Möchte man den Puffer gleichmäßiger nutzen, müssen also früher neue Punkte nachkommen. Eine Möglichkeit, dies zu erreichen ist, den ersten Punkt aus dem Puffer zu schieben, sobald das Gewicht des zweiten Punktes im Puffer über dem Schwellwert liegt (dieser zweite Punkt wird damit neuer Randpunkt und kann

[9] Hier wirkt sich die Größe von s noch aus.

nicht mehr entfernt werden). Die Tatsache, daß das Gewicht des zweiten Punktes zu einem bestimmten Zeitpunkt über dem Schwellwert liegt, bedeutet jedoch nicht, daß dies durchgehend so bleiben muß. Wird irgendwann der dritte Punkt entfernt, berechnet sich das Gewicht des zweiten Punktes ja neu und kann dabei auch kleiner werden (wenn dies auch – abhängig vom gewählten Maß – nicht unbedingt sehr wahrscheinlich ist). Wurde der zweite Punkt nun zu schnell an den Rand geschoben, kann er nicht mehr entfernt werden, die Generalisierung wird schlechter.

Möchte man diesen Effekt verringern, wird der erste Punkt erst dann aus dem Puffer geschoben, wenn der zweite *und* dritte Punkt Gewichte über dem Schwellwert haben, womit die Wahrscheinlichkeit sinkt, daß ein Punkt den Puffer zu früh verläßt. Ebenso kann man verlangen, daß die Gewichte von drei benachbarten Punkten über dem Schwellwert liegen müssen usw. Je mehr Punkte man hier wählt, desto „träger“ wird der Algorithmus, die Anzahl muß sowohl abhängig von der Puffergröße als selbstverständlich auch vom genutzten Maß gewählt werden.

11.4.2. Partielle Douglas/Peucker-Generalisierung

Die in 5.9 beschriebene partielle Generalisierung gehört ebenfalls zu den Puffer-Generalisierungen, da sie auch auf einem Puffer beschränkter Länge arbeitet, sie wurde lediglich aufgrund ihrer Ähnlichkeit mit Douglas/Peucker vorne beschrieben.

12. Zoom und Feinstruktur

12.1. Feinstruktur einer Bewegung

Abbildung 12.1 zeigt oben links einen Bewegungsverlauf und oben rechts dessen Generalisierung. Die Generalisierung gibt die Form der Bewegung im Großen sehr gut wieder, und unterdrückt die (in dieser Sequenz sehr starken) Hin- und Herbewegungen im Kleinen, genau dies ist ja Ziel der Generalisierung. „Subtrahiert" man nun die Generalisierung $g(p)$ vom Originalpolygonzug p, so erhält man als Ergebnis (in der Abbildung unten) die *Feinstruktur* der Bewegung. Sie liefert Informationen über den Bewegungsverlauf im Kleinen, ob sich das beobachtete Objekt eher glatt oder eher im Zickzack oder in Schlangenlinien bewegt hat oder auch, an welchen Stellen rangiert wurde.

Die Berechnung der Feinstruktur ist vor allem bei eckentreuen Generalisierungsalgorithmen sehr einfach. Der Polygonzug $p = v_1 v_2 \dots v_n$ wird zu $g(p) = v_1 v_{a_2} v_{a_3} \dots v_{a_{m-1}} v_n$ generalisiert, was eine Segmentierung von p in die Teilstücke $v_1 v_2 \dots v_{a_1}$, $v_{a_1} v_{a_1+1} \dots v_{a_2}, \dots$ liefert. Nun werden die einzelnen Segmente alle gleich ausgerichtet und hintereinandergehängt, d. h. der Polygonzug $v_{a_1} v_{a_1+1} \dots v_{a_2}$ wird (um den Punkt v_{a_1} so gedreht, daß die Strecken $[v_1 v_{a_1}]$ und $[v_{a_1} v'_{a_2}]$ hintereinander auf einer Geraden liegen, analog für die anderen Segmente (Abb. 12.1 unten).

Damit ermöglicht die Feinstruktur den Vergleich von Bewegungen, die im Großen unterschiedliche Strecken gefahren sind.

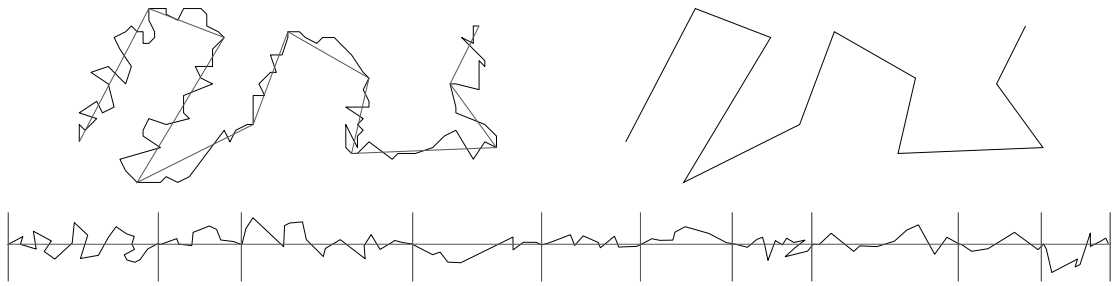


Abbildung 12.1.: Feinstruktur eines Bewegungsverlaufes

12.2. Generalisierung variabler Stärke

Alle bislang beschriebenen Generalisierungsalgorithmen liefern für einen konkreten Satz an Parametern und einen Originalpolygonzug genau eine Generalisierung. Um Generalisierungen unterschiedlicher Stärke zu bekommen, wird der entsprechende Algorithmus mit neuen Parametern erneut aufgerufen. Die Struktur der vorgestellten nicht inkrementellen Generalisierungen (sowohl die Divide-and-Conquer-Verfahren wie Douglas/Peucker als auch der Ansatz mit lokalen Gewichten) ermöglicht jedoch die Generierung aller möglichen Generalisierungen mit nur einem Durchlauf.[1]

Hierzu wird durch den entsprechenden Algorithmus auf den Punkten des Originalpolygonzuges eine Ordnung induziert, die angibt, für welche Parameter (z. B. für welches ε) ein Punkt noch Teil der Generalisierung ist. Die folgenden zwei Abschnitte zeigen den Aufbau der Generalisierung für Divide-and-Conquer-Verfahren und für den Ansatz mit lokaler Gewichtung.

12.2.1. Lokale Gewichte

Das in Kapitel 10 vorgestellte Verfahren der Generalisierung mit lokaler Gewichtung basiert darauf, daß für jeden Punkt v_i des Originalpolygonzuges $p = v_1 v_2 \dots v_n$ ein Maß κ_i bestimmt wird, daß sich aus der Lage des Punktes zu seinen Nachbarn v_{i-1} und v_{i+1} errechnet. Nun wird der „unwichtigste“ Punkt (der Punkt mit kleinstem κ_i) gelöscht und die Gewichte seiner Nachbarn neu berechnet, dieses so lange, bis alle Punkte „wichtiger“ als ein Schwellwert κ sind.

Macht man den Schwellwert nun sehr groß ($\kappa = \infty$), werden nach und nach alle Punkte aus p entfernt, bis schließlich nur noch die Randpunkte v_1 und v_n bleiben. Die Reihenfolge, in der die einzelnen Punkte entfernt wurden, liefert eine Sortierung auf ihnen,[2] so bilden die k zuletzt entfernten Punkte eine gültige Generalisierung des Originalpolygonzuges. Diese Generalisierung kann in linearer Zeit erzeugt werden, indem der Originalpolygonzug durchlaufen wird und dabei die k Punkte ausgewählt werden, die als letzte entfernt wurden.

Wurde diese Sortierung auf einem Polygonzug für ein bestimmtes Maß einmal durchgeführt (in $O(n \log n)$), können nun Generalisierungen mit beliebig gewählter Punktzahl in linearer Zeit abgelesen werden. Um die Generalisierungen nicht nur anhand ihrer Punktzahl sondern auch anhand von Schwellwerten auswählen zu können, muß lediglich in der Sortierung gespeichert werden, bei welchem Schwellwert die Generalisierung an welcher Stelle abgebrochen würde.

Sei v_a der „unwichtigste“ Punkt, κ_a also das kleinste Gewicht. Dann wird v_a

[1] In [Cro91] wird eine entsprechende Baumstruktur für Divide-and-Conquer-Verfahren vorgestellt.

[2] Dabei stört es nicht, daß mehrere Punkte das gleiche Gewicht besitzen können, der Algorithmus wählt einen von ihnen aus.

als erster Punkt entfernt, wenn für den Schwellwert κ gilt: $\kappa \geq \kappa_a$. Sei nun im um v_a verkürzten Polygonzug v_b der Punkt mit dem kleinsten Gewicht κ_b. v_b wird bei einer Generalisierung also genau dann entfernt, wenn $\kappa \geq \kappa_a$ und $\kappa \geq \kappa_b$. Gleiches gilt für den nächsten Punkt v_c usw.

Zum Punkt v_a wird das Gewicht κ_a abgespeichert, zum Punkt v_b das Gewicht $\max\{\kappa_a, \kappa_b\}$, zum Punkt v_c das Gewicht $\max\{\kappa_a, \kappa_b, \kappa_c\}$ usw. Einfach die jeweiligen Gewichte κ_a, κ_b, κ_c, ... abzuspeichern, reicht nicht aus, da sich durch Entfernen eines Punktes die Gewichte seiner Nachbarpunkte verändern, wobei sie auch kleiner werden können.

Die so gewonnene Liste enthält also alle Punkte des Originalpolygonzuges in der Reihenfolge, in der sie bei der Generalisierung entfernt werden und mit dem Schwellwert, ab dem sie entfernt werden. Um nun eine Generalisierung für einen Schwellwert κ zu erhalten, werden beim Durchlaufen des Originalpolygonzuges nun einfach alle Punkte ausgewählt, zu denen ein höherer Schwellwert gespeichert ist.

12.2.2. Divide-and-Conquer-Verfahren

Eine ähnliche Sortierung wie oben läßt sich auch für Divide-and-Conquer-Verfahren aufbauen. Ich zeige die Grundstruktur am Beispiel von Douglas/Peucker mit ε-Ähnlichkeitsmaß, die Adaption auf andere Verfahren sollte klar sein.

Wird ein Polygonzug $v_1 v_2 \ldots v_n$ mittels Douglas/Peucker generalisiert, so beginnt die Generalisierung mit $v_1 v_n$. Im ersten Schritt der Generalisierung wird nun der zu $[v_1 v_n]$ am weitesten entfernt liegende Punkt v_k ausgewählt, im zweiten Schritt der zu $[v_1 v_k]$ am weitesten entfernt liegende Punkt v_l ($l < k$) und der zu $[v_k v_n]$ am weitesten entfernt liegende Punkt v_r ($r > k$) usw.

Bricht man nun nicht an einem Schwellwert ε ab, sondern generalisiert weiter, bis jedes Segment nur noch aus 2 Punkten besteht (also nicht mehr weitergeneralisiert werden kann), so erhält man eine Baumstruktur: Erster Knoten ist v_k mit den Kindern v_l und v_r, diese haben weitere Kinder usw. In diesem Baum ist eine Sortierung der Punkte kodiert. Allerdings ist die Kodierung nicht ganz so offensichtlich wie oben. Startpunkt v_1 und Endpunkt v_n sind Teil jeder Generalisierung. v_k ist Teil aller Generalisierungen bis auf die triviale $v_1 v_n$, also Teil aller Generalisierungen mit einem Schwellwert $\varepsilon \leq d_k = \text{dist}([v_1 v_n], v_k)$.

Im nächsten Schritt werden die Punkte v_l (mit Abstand $d_l = \text{dist}([v_1 v_k], v_l)$) und v_r (mit Abstand $d_r = \text{dist}([v_k v_n], v_r)$) betrachtet. Dabei kann es sein, daß $d_l > d_k$ (ebenso für d_r), da ja der Abstand zu unterschiedlichen Strecken berechnet wird. Der Punkt v_l ist genau dann Teil einer Generalisierung mit Schwellwert ε, wenn $d_k \geq \varepsilon$ und $d_l \geq \varepsilon$. Analog zu oben reicht es also auch hier nicht, die Distanz des Punktes zu bestimmen, vielmehr muß hier das Minimum der Distanzen aller

Elternknoten gespeichert werden. Zum Punkt v_k wird also die Distanz d_k gespeichert, zu v_l die Distanz $\min\{d_k, d_l\}$, zu den Kindern v_{lr} und v_{ll} von v_l die Distanzen $\min\{d_k, d_l, d_{lr}\}$ bzw. $\min\{d_k, d_l, d_{ll}\}$ usw.

Damit kann man nun in linearer Zeit eine Generalisierung mit beliebigem Schwellwert ε ablesen, indem aus dem Originalpolygon diejenigen Punkte gewählt werden, deren Distanzen über diesem Schwellwert liegen. Ebenso kann man die Punkte nach ihren Distanzen sortieren und erhält wie oben eine sortierte Liste. Durch die Minimumbildung können viele Punkte mit gleichen Werten auftreten. Da kein Punkt Teil einer Generalisierung sein kann, wenn sein Elternknoten dies nicht ist,[3] werden Punkte gleicher Distanz zusätzlich nach ihrer Tiefe im Baum sortiert, je näher ein Knoten an der Wurzel liegt, desto wichtiger ist er. Aus der so erzeugten Liste können nun wie oben Generalisierungen mit beliebig gewählter Punktzahl abgelesen werden.

12.2.3. Zoom

In der Geographie sind Generalisierungen unterschiedlicher Stärke vor allem dann interessant, wenn beispielsweise der Benutzer eines Geoinformationssystems in eine Karte hineinzoomt. Küstenlinien und andere Strukturen können dann abhängig von der gewählten Auflösung mehr oder weniger stark generalisiert werden, um eine optimale Darstellung zu gewährleisten. Dieser Aspekt ist für die Verarbeitung von Bewegungsverläufen nur insofern von Nutzen, wie Bewegungsspuren graphisch dargestellt – z. B. in eine Karte eingezeichnet – werden. Die durch die obigen Verfahren gewonnene Gewichtsverteilung liefert jedoch noch weitere Informationen, wie der folgende Abschnitt zeigt.

12.3. Gewichtsverteilung

Abbildung 12.2 zeigt die Spuren A, B, C und D von vier Bewegungsverläufen. B beschreibt dabei im Großen die gleiche Form wie A, besitzt aber eine weit weniger ausgeprägte Feinstruktur, gleiches gilt analog für C und D. Alle vier Polygonzüge bestehen aus 63 Eckpunkten. Nun wurde für jeden Polygonzug mittels der Douglas/Peucker-Generalisierung die „Gewichtsverteilung“ für alle Punkte berechnet (Abschnitt 12.2.2), d. h. bestimmt, bei welchem Wert von ε welcher Punkt weggeneralisiert wird. Die Kurven a, b, c und d stellen die Gewichtsverteilung der Polygonzüge A, B, C und D graphisch dar: Auf der waagerechten Achse ist der Schwellwert ε angegeben, auf der senkrechten Achse die Anzahl Eckpunkte, die der mit Schwellwert ε generalisierte Polygonzug besitzt. Wird der Polygonzug A oder

[3] Dies würde die Generalisierung verfälschen, wie man sich leicht überlegt.

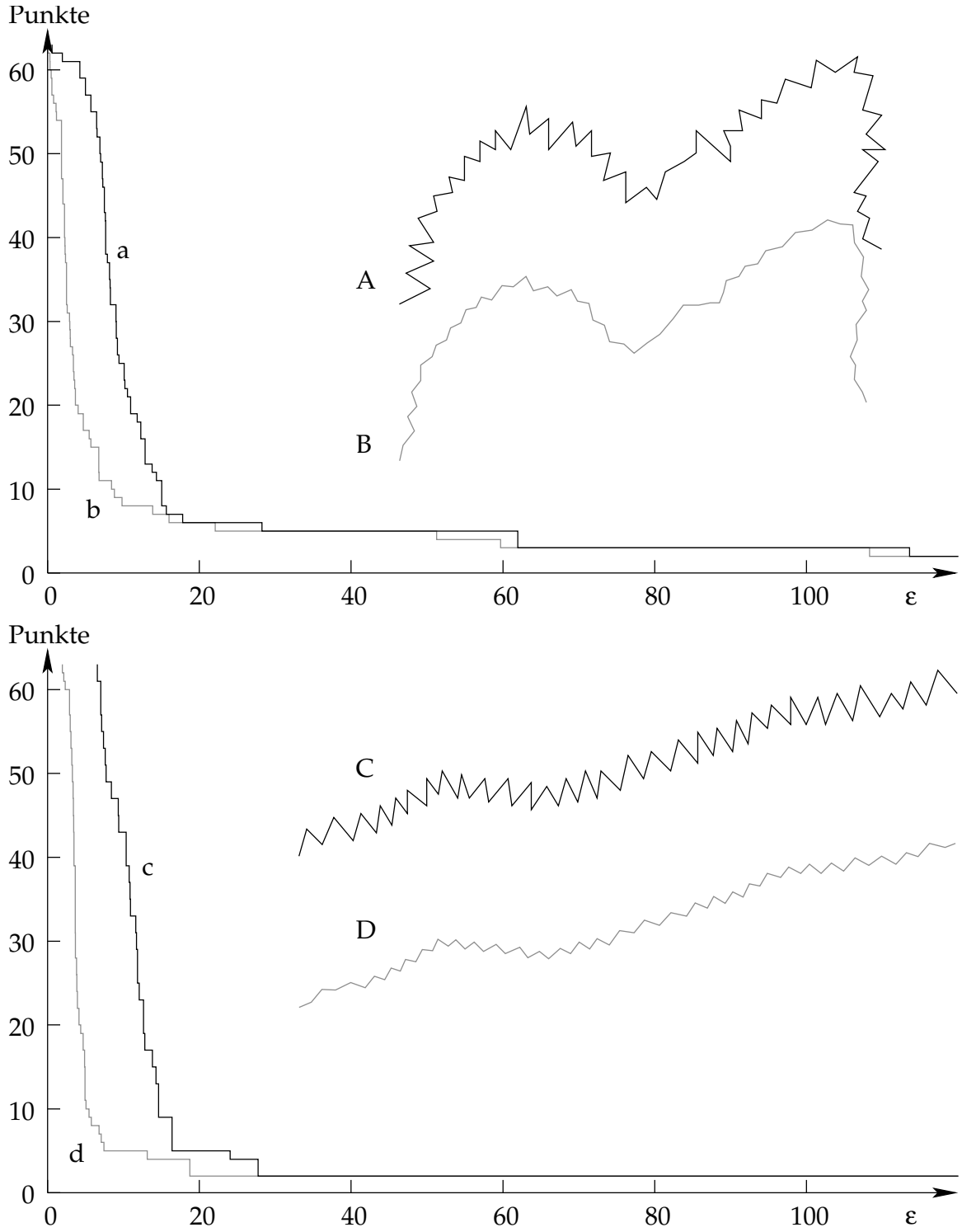


Abbildung 12.2.: Gewichtsverteilungen von Bewegungsverläufen unterschiedlicher Feinstruktur

B mit $\varepsilon = 40$ generalisiert, besteht der generalisierte Polygonzug also aus 5 Punkten, für $\varepsilon = 10$ besitzt die Generalisierung von A 25 Punkte, die Generalisierung von B nur 10 Punkte.

Für kleine ε weichen die Gewichtsverteilungen der beiden Polygonzüge A und B deutlich voneinander ab, die Kurven a und b geben die unterschiedlich ausgeprägte Feinstruktur also sehr schön wieder. Gleiches gilt für die Polygonzüge C und D. Vergleicht man die Kurven a und c bzw. b und d, so ist die ähnliche Feinstruktur deutlich sichtbar: Sowohl in a als auch in c findet der deutliche Abfall in der Eckpunktzahl zwischen $\varepsilon = 7$ und $\varepsilon = 15$ statt, während er für die Kurven b und d zwischen $\varepsilon = 0$ und $\varepsilon = 7$ liegt.

13. Einordnung der Algorithmen

Die in den letzten Kapiteln vorgestellten Generalisierungsalgorithmen unterscheiden sich in einer ganzen Reihe von Eigenschaften voneinander, sowohl hinsichtlich des Ähnlichkeitsmaßes, das sie erfüllen, als auch hinsichtlich der Art der Verarbeitung des Bewegungsverlaufes. Daher ist es nicht möglich, einen *besten* Generalisierungsalgorithmus anzugeben, vielmehr sind die unterschiedlichen Ansätze für konkrete Anwendungen besser oder schlechter geeignet. Dieses Kapitel liefert daher einen Überblick über die unterschiedlichen Eigenschaften der vorgestellten Algorithmen.

13.1. Klassifizierung

In seiner Übersicht über Generalisierungsalgorithmen in der Kartographie [McM87] schlägt Robert Brainerd Mc Master ein Klassifizierungsschema für Generalisierungsalgorithmen vor, das die Algorithmen in erster Linie danach einteilt, ob sie einen lokalen (der Algorithmus betrachtet einzelne Punkte und deren Nachbarn) oder globalen (der Algorithmus betrachtet den gesamten Polygonzug) Ansatz verfolgen. Zur Einordnung der hier vorgestellten Algorithmen ist diese Klassifizierung wenig geeignet, weil sie die besonderen Merkmale der Algorithmen nicht hervorhebt, so werden z. B. inkrementelle Verfahren gar nicht betrachtet. Statt zu versuchen, alle Algorithmen in ein festes Schema einzuordnen, gebe ich im Folgenden nochmals einen Überblick über konkrete Eigenschaften, die die Wahl eines geeigneten Algorithmus für ein bestimmtes Problem erleichtern.

13.2. Inkrementelle Algorithmen

Die Messung der Bewegungsspur eines mobilen Roboters (oder auch eines Autos etc.) kann dazu dienen, den weiteren Verlauf der Bewegung zu planen. Beispielsweise wird das Verhalten des (bereits in Kapitel 7 erwähnten) selbstfahrenden Rollstuhls Rolland durch Vergleich der bereits gefahrenen Strecke mit früher gefahrenen Routen gesteuert, ebenso ist damit eine Selbstlokalisation des Rollstuhls

möglich.[1] Wie schon mehrfach erwähnt, sind für die Weiterverarbeitung eines Bewegungsverlaufes während der Messung inkrementell arbeitende Algorithmen am geeignetesten. Zwar sind heutige Computer schnell genug, um einen Bewegungsverlauf für jeden neu hinzukommenden Punkt neu zu generalisieren, dies kann aber bei nicht inkrementellen Algorithmen dazu führen, daß sich durch Hinzufügen von Punkten am Ende des Polygonzuges die Generalisierung des gesamten Polygonzuges ändert, was die Weiterverarbeitung erschwert.

Ist der Algorithmus darüberhinaus historiefrei inkrementell, so besitzt er bezüglich der Länge der Gesamtsequenz (also bezogen auf die Anzahl Segmente, in die generalisiert wird) lineare Komplexität.

Algorithmen, die eine Vorausschau betreiben (z. B. WεG), sind von ihrer Struktur her inkrementell, sie arbeiten die einzelnen Segmente der Generalisierung der Reihe nach ab, erfüllen aber nicht zwingend die vorne gegebene Definition für inkrementelle Algorithmen. Inwieweit sie für eine Bewegungsverarbeitung „on the fly“ geeignet sind, hängt zum einen davon ab, wie stark sich diese Vorausschau auswirkt, also wie viele Punkte sie umfaßt, und zum anderen davon, eine wie starke Verzögerung die konkrete Anwendung erlaubt, wie weit also die Generalisierung hinter der aktuellen Position hinterherhinkt. Wird – wie beim autonom fahrenden Rollstuhl Rolland – beispielsweise mittels der generalisierten Bewegungsspur getestet, ob ein mobiler Roboter sich noch auf dem richtigen Weg befindet, kann schon eine Vorausschau um wenige Schritte zu viel sein, ein falsches Abbiegen soll ja nicht erst bemerkt werden, wenn er schon einige Meter in den falschen Gang hineingefahren ist.[2]

Diese Effekte müssen auch bei Einsatz einer Eckensuche (Abschnitt 6.4) berücksichtigt werden. Die Eckensuche hat sich als wichtig erwiesen, um möglichst gut den Punkt bestimmen zu können, an dem Rolland von einem Gang in einen anderen biegt, sie verbessert also das Ergebnis der Generalisierung. Sie darf dabei aber nicht zu einer großen Verzögerung der Generalisierung führen.

13.3. Generalisierung auf der vollständigen Spur

Sowohl die Divide-und-Conquer-Verfahren als auch die Generalisierung mit lokaler Gewichtung arbeiten auf dem vollständigen Polygonzug p. McMaster und an-

[1] Eine ausführliche Darstellung findet sich in [Lan02], siehe aber auch [RL98, Röf99] und die gemeinsamen Arbeiten [MSER99, MRL+00]. Eine knappe Darstellung der Navigation mittels aufgezeichneter Eigenbewegungen findet sich auch in [Mus00, Kapitel 8]. Dort ist auch ein Verfahren zur Navigation auf Basis qualitativer Routenbeschreibungen beschrieben, das in Kooperation zwischen dem Bremer Rolland-Projekt und uns (Alexandra Musto und mir) entwickelt wurde.

[2] Zwar liefert die Keilberechnung von WεG eine Schätzung der möglichen Bewegungsrichtung, doch ist diese aufwendiger zu handhaben als ein schlichter Vergleich von Polygonzügen.

dere nennen die Douglas/Peucker-Generalisierung ein *globales* und die Generalisierung mit lokaler Gewichtung ein *lokales* Verfahren, weil ersteres ausgehend vom gesamten Polygonzug einen Eckpunkt selektiert, wogegen letzteres die Gewichte der einzelnen Punkte bezüglich der direkten Nachbarn bestimmt. Diese Bezeichnungen sind zwar anschaulich, aber auch teilweise irreführend: Im ersten Schritt der Douglas/Peucker-Generalisierung wird zwar ein Punkt v_k aus allen Punkten des Polygonzuges ausgewählt, was das Attribut „global“ gerechtfertigt erscheinen läßt, schon in die Auswahl der nächsten Punkte geht jedoch nur noch der Teilpolygonzug $v_1v_2 \ldots v_k$ bzw. $v_kv_{k+1} \ldots v_n$ ein, und diese Unterteilung schreitet immer weiter fort. Bei der Generalisierung mit lokaler Gewichtung werden dagegen in jedem Schritt die Gewichte aller (noch vorhandenen) Punkte miteinander verglichen. Die Gewichte werden also zwar lokal bestimmt, aber global bewertet.

13.4. Gewichtsverteilung

Sowohl die Divide-and-Conquer-Verfahren als auch die Generalisierung mit lokaler Gewichtung sind geeignet, wenn ein Bewegungsverlauf im Nachhinein untersucht werden soll. Sie erlauben mit relativ geringem Aufwand die Ermittlung der Gewichtsverteilung eines Bewegungsverlaufes (Kapitel 12), die wiederum Informationen über Grob- und Feinstruktur der Bewegung liefert.

Die Bestimmung dieser Verteilung ist mit inkrementellen Verfahren nur mit sehr großem Aufwand möglich, da hier jede Änderung des Generalisierungsparameters zu völlig anderen Segmentierungen führen kann.

13.5. Korridorgeneralisierungen

Eine ganze Reihe der vorgestellten Generalisierungen (angefangen von der Douglas/Peucker- und Persson/Jungert/Hernández-Generalisierung über simpleinc$_\varepsilon$, WεG und die Rolland-Generalisierung bis zu ConvexHull) sind Korridor-Generalisierungen, d. h. die Generalisierung beschreibt einen Korridor, in dem der Originalpolygonzug vollständig liegt. Korridorgeneralisierungen sind vor allem dann interessant, wenn Bewegungen autonomer Roboter in einem Gangsystem verarbeitet werden sollen, da sie einen maximalen Abstand des Originalpolygonzuges von der Generalisierung garantieren.

Dabei ist es nicht zwingend nötig, daß die Generalisierung das ε-Kriterium (mit ε gleich halber Korridorbreite) erfüllt, wie die Rolland-Generalisierung zeigt. Bei Bedarf kann zu jedem Segment der Rolland-Generalisierung angegeben werden, wie weit der Originalpolygonzug auf welcher Seite von der Generalisierung abweicht, womit die genaue Lage des Korridors, innerhalb dessen die Bewegung

stattfand, klar ist.[3] Dies kann dann hilfreich sein, wenn die gefahrene Strecke beispielsweise mit einer gegebenen Karte gematcht werden soll.

13.6. Ähnlichkeitsmaße

Erfüllt eine Generalisierung g_ε das ε-Kriterium, so weicht ein Polygonzug $p = v_1 v_2 \dots v_n$ von seiner Generalisierung $g_\varepsilon(p)$ an keiner Stelle weiter als ε ab. Der Polygonzug $g_\varepsilon(p)$ macht also eine Aussage über die Lage aller Punkte von p: Sie liegen alle in einem bestimmten Bereich. Gleiches gilt auch für die anderen vorgestellten Ähnlichkeitsmaße: Erfüllt eine Generalisierung g_γ ein Winkel-Ähnlichkeitsmaß, so gilt für $g_\gamma(p) = v_1 v_{a_2} v_{a_3} \dots v_n$, daß alle Punkte $v_i \in \{v_{a_k+1}, v_{a_k+2}, \dots, v_{a_{k+1}-1}\}$ in einer zwischen den Punkten v_{a_k} und $v_{a_{k+1}}$ aufgespannten Linse liegen. In allen Fällen liefert also die Generalisierung $g(p)$ sichere Informationen über die mögliche Lage aller Punkte von p.

Für die Generalisierung mit lokaler Gewichtung gilt dies nicht. Obwohl diese Generalisierung die Form des Originalpolygonzuges im großen gut wiedergibt, läßt sich kein konkretes Ähnlichkeitsmaß angeben. Dies liegt an der lokalen Gewichtsbestimmung: Für das Gewicht κ_i eines Punktes v_i ist nur die Lage zu seinen benachbarten Punkten entscheidend, bezüglich dieser kann also ein bestimmtes Ähnlichkeitsmaß erfüllt sein ($\mathrm{dist}([v_{i-1}v_{i+1}], v_i) < \varepsilon$ oder ähnliches). Wird nun in einem Schritt der Punkt v_i und in einem weiteren der Punkt v_{i+1} entfernt, so ist nicht mehr sichergestellt, daß v_i von der von den beiden noch in der Generalisierung vorhandenen Punkten v_{i-1} und v_{i+2} aufgespannten Strecke $[v_{i-1}v_{i+2}]$ nicht weiter als ε abweicht.

13.7. Eckentreue Algorithmen

Bei einem eckentreuen Algorithmus g sind alle Eckpunkte der Generalisierung $g(p) = v_1 v_{a_2} v_{a_3} \dots v_n$ auch Eckpunkte des Originalpolygonzuges $p = v_1 v_2 \dots v_n$. Die Generalisierung wählt also aus p bestimmte (wichtige) Punkte aus bzw. entfernt bestimmte (unwichtige) Punkte. Eckentreue Generalisierungen besitzen eine Reihe von technischen Vorteilen: Jedes Teilstück $v_{a_i} v_{a_i+1} \dots v_{a_{i+1}}$ des Originalpolygonzuges p wird dabei durch die Strecke $[v_{a_i} v_{a_{i+1}}]$ repräsentiert. Diese einfache Zuordnung erlaubt z. B. eine einfache Bestimmung der Feinstruktur (Abschnitt 12.1) oder auch den Einsatz der Generalisierung zum „Geraderücken“ eines Polygonzuges (Abschnitt 10.3.2).

[3] nicht zu verwechseln mit dem Gang eines Gebäudes, durch den der Roboter an dieser Stelle gefahren ist. Der Roboter muß für seine Bewegung ja nicht unbedingt die gesamte Gangbreite genutzt haben.

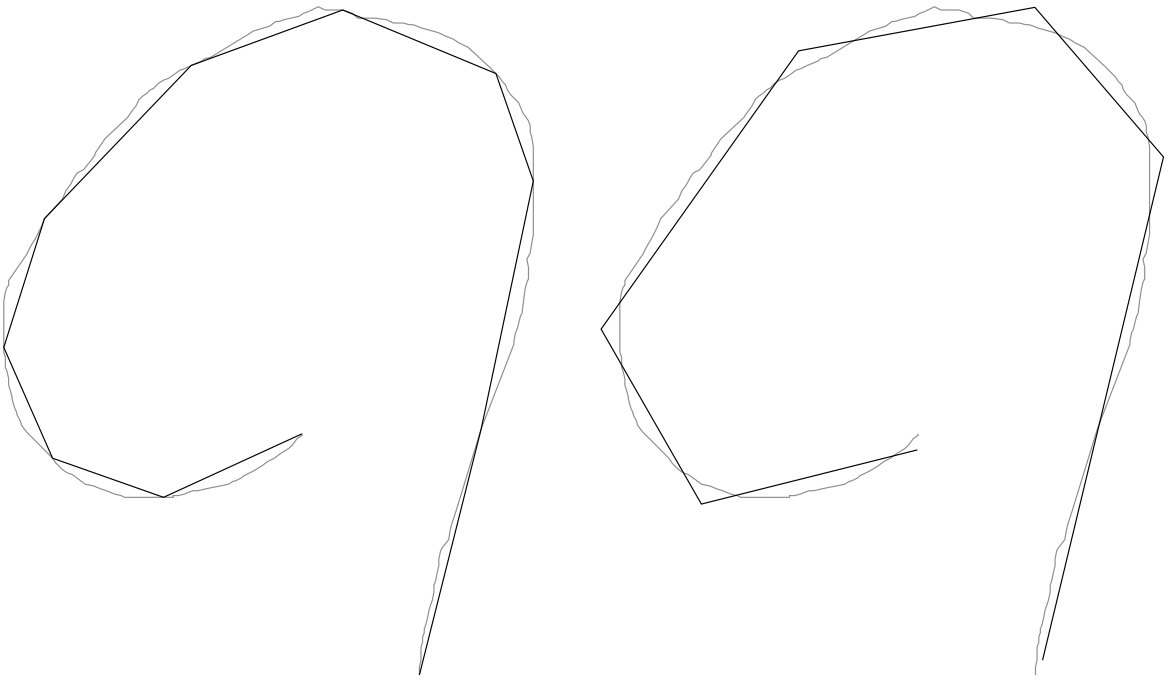


Abbildung 13.1.: Generalisierung mittels WεG und ConvexHull

Grundsätzlich ist auch bei nicht eckentreuen Generalisierungen diese Zuordnung einer Teilstrecke der Generalisierung zu einem Teilpolygonzug der Originalkurve möglich, allerdings ist dies technisch etwas aufwendiger.

Ein Effekt der eckentreuen Generalisierung besteht darin, daß sie Kurven stets verkürzt. Abbildung 13.1 zeigt die Generalisierung eines Bewegungsverlaufes mittels WεG (links) und ConvexHull (rechts), in beiden Fällen mit gleichem Wert für ε. Der mit WεG generalisierte Polygonzug liegt vollständig innerhalb des Kurvenverlaufes, die Kurve wird enger. Die Convex Hull Generalisierung hingegen „überdeckt“ den Kurvenverlauf gewissermaßen.

13.8. Auswahl markanter Ecken

Der Douglas/Peucker-Algorithmus (ebenso wie der Persson/Jungert/Hernández-Algorithmus) liefert Generalisierungen, die die Feinstruktur einer Bewegung gut unterdrücken. Er wählt aber nicht unbedingt die markantesten Punkte des Originalpolygonzuges als Eckpunkte aus (siehe auch Abschnitt 5.7, Abbildung 5.19). Die Wahl eines Eckpunktes hängt an seiner Lage bezüglich des Start- und Endpunktes des in einer bestimmten Rekursionstiefe betrachteten Teilpolygonzuges. Dabei ist die Wahrscheinlichkeit, einen Eckpunkt zu erwischen, der exponiert liegt, zwar höher als für andere Punkte, es können aber auch sehr unscheinbare Punkte ausge-

wählt werden, wie das Beispiel in der oben genannten Abbildung zeigt.

Noch extremer ist die Situation bei den inkrementellen Verfahren, da hier der Punkt als Eckpunkt ausgewählt wird, an dem das gerade betrachtete Segment seitlich verlassen wird, und dieser Punkt ist in den meisten Fällen kein markanter Eckpunkt. Daher ist eine Eckensuche (Kapitel 6.4) bei diesen Algorithmen unverzichtbar. Die kumulativen Algorithmen liefern hier erwartungsgemäß die schlechtesten Ergebnisse, wenn auch die Qualität der Σ-Generalisierung durch Vorschaltung der Streckenkonkatenation etwas verbessert werden kann.

Durch die Generalisierung mit lokaler Gewichtung werden markante Eckpunkte hingegen gut gefunden. Dadurch, daß „unwichtige“ Punkte entfernt werden, bleiben im Ergebnis genau die Punkte als Ecken übrig, die bezüglich ihrer näheren Umgebung (und bezüglich der gewählten Gewichtsfunktion κ) besonders markant sind.

13.9. Schleifen und Rückläufigkeiten

Die Erkennung von internen Schleifen (Abschnitt 5.3 und 6.3) ist eine Anforderung, die über die einfache Forderung hinausgeht, ein bestimmtes Ähnlichkeitsmaß (z. B. das ε-Kriterium) zu erfüllen. Sie beruht auf dem Gedanken, daß es nicht ausreicht, wenn ein Polygonzug p in einer bestimmten Umgebung um einen Polygonzug q liegt, damit beide Polygonzüge einander hinreichen ähnlich sind: p und q sollen darüberhinaus die gleiche Bewegungsrichtung besitzen: Verläuft die Bewegung in einem Teilstück von p weiter als eine bestimmte Toleranz ε' in die zum Verlauf von q entgegengesetzte Richtung, so liegt p zwar noch in der ε-Umgebung von q, die Polygonzüge sind sich aber nicht mehr hinreichend ähnlich. Genau in diesen Situationen greift die Erkennung von Schleifen und Rückläufigkeiten.

Für die Algorithmen, die nicht auf Akzeptanzbereichen beruhen, ist aufgrund der völlig anderen Struktur eine Schleifenerkennung in diesem Sinne nicht nötig und/oder möglich.

13.10. Teilgeneralisierungen

Werden ein Polygonzug $p = v_1 v_2 \dots v_n$ und ein Polygonzug $p' = v_a v_{a+1} \dots v_b$ mit $1 \leq a \leq b \leq n$ (p' ist also Teil von p) generalisiert, so unterscheidet sich die Generalisierung $g(p)$ im Bereich v_a mit v_b (also im Teilstück von $g(p)$, das diesen Bereich repräsentiert) im Allgemeinen von $g(p')$. Die Polygonzüge $g(p)$ und $g(p')$ haben also teilweise unterschiedliche Eckpunkte. Die Stärke dieser Unterschiede ist abhängig vom gewählten Algorithmus.

Die Divide-and-Conquer-Generalisierungen sind stark von der Lage des Start- und Endpunktes des betrachteten Polygonzuges abhängig, liefern also für gewöhn-

lich sehr unterschiedliche Generalisierungen auf Teilpolygonzügen. Besitzen p und p' sehr exponierte Ecken, so ist die Wahrscheinlichkeit groß, daß diese sowohl von $g(p)$ als auch von $g(p')$ gefunden werden und beide Generalisierungen daher mehrere Punkte gemeinsam haben, ist p dagegen eher rund, ist dies eher nicht der Fall.

Inkrementelle Verfahren sind dagegen zwar abhängig vom Startpunkt der Generalisierung, bei gleichem Startwert ($a = 1$) sind dagegen nach Definition auch die restlichen Punkte der Generalisierungen abgesehen vom letzten identisch (wobei $g(p)$ darüberhinaus noch zusätzliche Punkte haben kann). Sind die Startwerte hingegen nicht gleich, ist auch hier die Form der Bewegung entscheidend: Wird die Bewegungsspur eines mobilen Roboters durch die Gänge eines Bürogebäudes z. B. mit WεG generalisiert, so werden durch die Generalisierung die Punkte erkannt, an denen er von einem Gang in den anderen biegt. Kommt hier noch eine Eckensuche hinzu, so wird auch bei unterschiedlichen Startpunkten in der Regel schon nach wenigen Segmenten ein gleicher Eckpunkt gewählt, und die folgenden Eckpunkte sind dann ebenfalls identisch.

Die Generalisierung mit lokaler Gewichtung arbeitet weitgehend unabhängig von Start- und Endpunkt. Die Gewichte der einzelnen Punkte werden in Bezug auf ihre direkten Nachbarn bestimmt und sind für p und p' im Bereich von v_a mit v_e zunächst identisch (abgesehen von v_a und v_e selbst, die in p' die Gewichte $\kappa_a = \kappa_e = \infty$ besitzen). Werden nun nach und nach Punkte entfernt, so verläuft dies in beiden Polygonzügen weitgehend identisch, Unterschiede treten nur an den Rändern auf, wo v_a und v_e in p jederzeit weggeneralisiert werden können, während sie in p' fest bleiben. Damit unterscheidet sich $g(p')$ nur in seinen ersten und letzten Punkten von $g(p)$.

13.11. Geschwindigkeit und Dynamik

In einem mit festem Zeitraster gemessenen Bewegungsverlauf ist die Dynamik der Bewegung implizit im Abstand aufeinanderfolgender Punkte des Polygonzuges kodiert: Je weiter zwei aufeinanderfolgende Punkte des Polygonzuges voneinander entfernt liegen, desto schneller ist das beobachtete Objekt an dieser Stelle gefahren. Diese Geschwindigkeitsinformation geht bei der Generalisierung verloren, da die Eckpunkte des generalisierten Polygonzuges keinem festen Zeitraster mehr entsprechen.

Soll die Geschwindigkeitsinformation bei der Generalisierung erhalten bleiben, so kann jedes Segment der Generalisierung mit der in diesem Bereich gefahrenen Durchschnittsgeschwindigkeit attributiert werden.

Dabei treten zwei unterschiedliche Arten von „Geschwindigkeit“ auf: Abbildung 13.2 zeigt drei (unterschiedliche) Bewegungsverläufe A, B und C, die alle identisch (zu einer einzigen Strecke) generalisiert werden. Betrachtet man nun die Zeit, die jeweils für den Weg vom Start- zum Endpunkt benötigt wurde, so sind dies

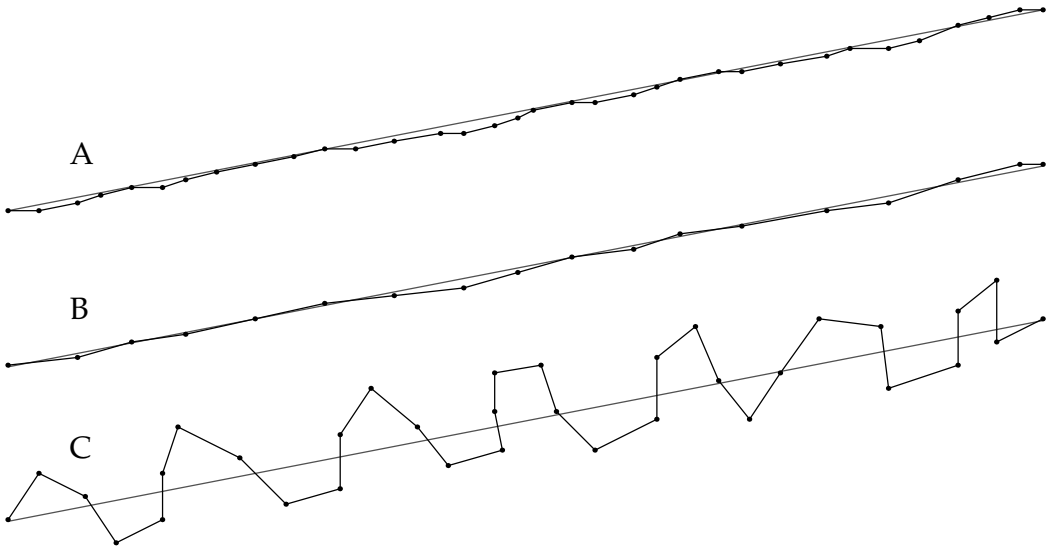


Abbildung 13.2.: Unterschiedliche Geschwindigkeiten

sowohl in A als auch in C $33\Delta t$. In B wurde die Strecke vom Start- zum Endpunkt dagegen in nur $17\Delta t$, also annähernd doppelt so schnell zurückgelegt.

Betrachtet man nun nur die Generalisierung (also die direkte Strecke von Start- zu Endpunkt) und die Zeit, in der diese Strecke zurückgelegt wurde, so war das betrachtete Objekt in B doppelt so schnell wie in A und C (es hat die Entfernung zwischen Start- und Endpunkt in halber Zeit bewältigt).

Betrachtet man nun aber die Punktabstände in den Originalpolygonzügen, ergibt sich ein anderes Bild: Die in einem Zeitschritt Δt zurückgelegte Entfernung ist in B und C ungefähr gleich, während sie in A im Schnitt halb so groß ist. Während das beobachtete Objekt also in B und C ungefähr gleich schnell gefahren ist, war es in A nur halb so schnell.

Die Unterschiede zwischen beiden Fällen entstehen dadurch, daß die Zeit, die benötigt wird, um die Entfernung von Start- zu Endpunkt zurückzulegen, im ersten Fall auf die generalisierte Strecke und im zweiten Fall auf die Länge des Originalpolygonzuges bezogen wird. Zu jedem Segment der Generalisierung muß also nicht nur die Zeitspanne gespeichert werden, in der es zurückgelegt wurde, sondern auch die Länge des Weges, der dabei zurückgelegt wurde (also die Länge des Originalpolygonzuges im entsprechenden Segment).

13.12. Minimale Generalisierungen

Mit den in Abschnitten 6.7 (WεG) und 8.4 (Convex Hull) angegebenen Verfahren läßt sich für jeden Polygonzug eine aus einer minimalen Anzahl Segmente

bestehende Segmentierung erzeugen, wobei das erste (WεG-basierte) Verfahren die Menge aller minimalen eckentreuen und das zweite (Convex-Hull-basierte) Verfahren die Menge aller minimalen nicht eckentreuen Segmentierungen liefert. Während das eckentreue Verfahren minimale Generalisierungen erzeugt, liefert das Convex-Hull-Verfahren nur minimale Segmentierungen, also eine Sequenz von Strecken, deren ε-Umgebungen den Originalpolygonzug p vollständig überdecken. Ist dies nicht ausreichend und ein Polygonzug als Ergebnis der Generalisierung gewünscht, so müssen die einzelnen Strecken zusammengebaut werden, was wie schon erwähnt auf Kosten der Minimalität gehen kann.

Das WεG-basierte Verfahren liefert dagegen die Menge aller minimalen eckentreuen Generalisierungen. Dabei bedeutet minimal nicht unbedingt gut, da durch den Algorithmus ja nicht unbedingt die markantesten Eckpunkte ausgewählt werden. Um nun aus der Menge der minimalen Generalisierungen diejenige auszuwählen, die die markantesten Eckpunkte besitzt, läßt sich das Verfahren mit der Generalisierung mit lokaler Gewichtung kombinieren: Für den zu generalisierenden Originalpolygonzug p werden mit dem in Abschnitt 12.2.1 vorgestellten Verfahren die effektiven Gewichte aller Punkte v_i bestimmt. Nun kann aus der Menge aller minimalen Generalisierungen diejenige ausgewählt werden, deren Eckpunkte die höchsten Gewichte besitzen.

Die Menge der minimalen Generalisierungen liegt ja wie beschrieben als Graph vor. Nun sind zwei Herangehensweisen möglich:

- Aus dem Graphen werden so lange die Eckpunkte mit den kleinsten Gewichten gestrichen (ein Punkt darf nicht gestrichen werden, wenn der Graph ohne ihn nicht mehr zusammenhängend wäre), bis der Graph nur noch aus einem Pfad besteht, der dann die Generalisierung liefert.

- Aus dem Graphen wird der Punkt v_i ($1 < i < n$) mit dem größten Gewicht κ_i ausgewählt und alle Punkte, die nicht auf einem Pfad von v_1 nach v_i oder von v_i nach v_n liegen, entfernt. Im nächsten Schritt wird aus dem resultierenden Graphen der Punkt mit dem nächstgrößten Gewicht ausgewählt usw.

Das erste Verfahren sorgt dafür, daß Punkte mit kleinen Gewichten möglichst nicht in der Generalisierung auftauchen, das zweite Verfahren wählt dagegen gezielt Punkte mit großen Gewichten aus. Dies liefert häufig unterschiedliche Generalisierungen, da das Entfernen eines Punktes mit kleinem Gewicht dazu führen kann, daß ein anderer Punkt mit sehr großem Gewicht ebenfalls nicht Teil der Generalisierung sein kann und umgekehrt kann die Wahl eines Punktes mit sehr großem Gewicht dazu führen, daß ein anderer Punkt mit sehr kleinem Gewicht zwangsläufig auch Teil der Generalisierung ist.

14. Schlußbemerkungen

14.1. Zusammenfassung

In dieser Arbeit wurden Verfahren zur qualitativen Repräsentation und Algorithmen zur Generalisierung von Bewegungsverläufen entwickelt und hinsichtlich ihrer Eigenschaften untersucht.

Auf Basis der in Kapitel 2 beschriebenen Repräsentation von Bewegungsverläufen durch (diskrete) Punkt- bzw. Vektorsequenzen in unterschiedlichen Referenzrahmen wurden zunächst qualitative Repräsentationen vorgestellt (Kapitel 3).

Die dort beschriebene qualitative Architektur umfaßt dabei sowohl die direkt aus einer numerischen Darstellung hervorgehende QMV-Repräsentation in unterschiedlichen Referenzrahmen (allozentrisch und egozentrisch), als auch die formenbasierte (einer natürlichsprachlichen Beschreibung näherkommende) propositionale Repräsentation.

Sie eignet sich damit sowohl als Vorstufe zur Erzeugung „natürlichsprachlicher“ (oder zumindest einem Laien verständlicher) Beschreibungen von Bewegungsverläufen, z. B. zur automatischen Generierung von Wegbeschreibungen, als auch zur Erzeugung qualitativer Routenbeschreibungen zur Steuerung autonomer mobiler Systeme (Kooperation mit dem Bremer Rolland-Projekt).

Die beschriebene QMV-Algebra und der QMV-Vektorraum erlauben die Anwendung des umfangreichen Instrumentariums der Geometrie zur Verarbeitung von QMV-Sequenzen, wobei allerdings aufgrund der bei diesen Berechnungen unter Umständen auftretenden Verzerrungen der Bewegungsspur die Aussagekraft der Ergebnisse sorgfältig geprüft werden muß. Insbesondere können auf Grundlage des QMV-Vektorraums die im Hauptteil der Arbeit beschriebenen Generalisierungsalgorithmen nicht nur auf numerische sondern auch auf QMV-Sequenzen angewandt werden.

Ein wesentlicher Schritt in der Verarbeitung von (gemessenen) Bewegungsverläufen ist ihre Vereinfachung mittels Generalisierung. Nach einer Einführung der wesentlichen Grundbegriffe wurden unterschiedliche Generalisierungsalgorithmen vorgestellt und schließlich hinsichtlich ihrer Eigenschaften untersucht. Eine Reihe dieser Algorithmen stammt ursprünglich aus der Geographie, wo sie zur Generalisierung von Küstenlinien und ähnlichen Strukturen eingesetzt werden. An

die Vereinfachung von Bewegungsverläufen werden allerdings andere Anforderungen gestellt als an die Vereinfachung von geographischen Formationen, so daß zum einen Algorithmen angepaßt und erweitert, zum anderen aber auch eine Reihe neuer Algorithmen (in erster Linie die inkrementell arbeitenden Algorithmen) speziell für die Bewegungsverarbeitung entwickelt wurden.

Die Zusammenstellung dieser Algorithmen liefert einen Werkzeugkasten zur Bewegungsvereinfachung, in dem es nun nicht ein *bestes* Werkzeug zur Generalisierung von Bewegungsverläufen, sondern vielmehr viele Werkzeuge mit unterschiedlichen Eigenschaften (Kapitel 13) gibt, aus denen für eine spezielle Anwendung das geeignete gewählt werden kann.

Und schließlich eignen sich Generalisierungsverfahren über die schlichte Vereinfachung von Bewegungsverläufen hinaus auch zur Analyse der Feinstruktur eines Bewegungsverlaufes (Kapitel 12).

14.2. Ausblick

Die in dieser Arbeit vorgestellten Werkzeuge und Verfahren bilden nur einen – wenn auch wesentlichen – Schritt in der Verarbeitung von Bewegungsverläufen. Hieraus ergeben sich weitere Fragestellungen:

Vergleich von Bewegungsverläufen. Ist man in der Lage, einen Bewegungsverlauf p mit einer vorher gemessenen Referenzspur p' zu vergleichen, so kann dies beispielsweise zur Steuerung autonomer mobiler Systeme eingesetzt werden.[1]

Gesucht ist ein Maß, das bestimmt, wann zwei Spuren einander „im Großen“ hinreichend ähnlich sind.[2] Im allozentrischen Fall ist dies relativ einfach, da sich Fehler nicht aufsummieren. Liegt p in einer ε-Umgebung um p' (oder, um den Rechenaufwand geringer zu halten, um $g(p')$ mit einer ε-Generalisierung g), und liegen ihre Start- und Endpunkte ebenfalls weniger als ε voneinander entfernt, so ist die Form beider Spuren „im Großen“ ähnlich.[3]

Der egozentrische Fall ist hingegen schwieriger zu fassen, da hier beide Bewegungsspuren aufgrund der akkumulierten Meßfehler durchaus stark voneinander abweichen können, obgleich beide Male die gleiche Strecke gefahren wurde. Der

[1] Der schon mehrfach erwähnte Bremer Rollstuhl Rolland nutzt dies zur Navigation. Der Vergleich des aktuellen Bewegungsverlaufes mit einer Referenzroute liefert die Information, welches Verhalten der Rollstuhl zu einem bestimmten Zeitpunkt zeigen soll (sich rechts halten, links halten, geradeaus fahren oder anhalten) und ob er sich noch auf dem richtigen Weg befindet.

[2] wobei noch hinzukommt, daß während der Fahrt natürlich nur die bislang zurückgelegte Strecke zum Vergleich zur Verfügung steht. Daher muß geprüft werden, ob das gefahrene Teilstück mit dem Anfang von p' übereinstimmt.

[3] Ist zusätzlich eine Schleifenerkennung gewünscht, so wird die Sache etwas aufwendiger.

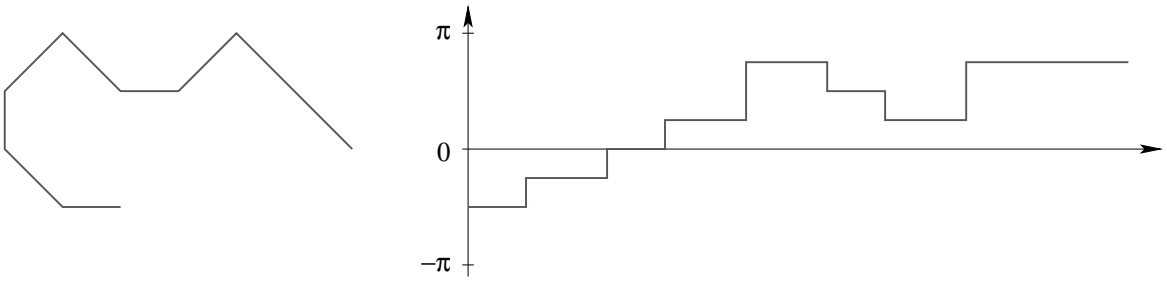


Abbildung 14.1.: Tangentenraumdarstellung eines Polygonzuges

im Bremer Rolland-Projekt verfolgte Ansatz, die Generalisierungen beider Bewegungsspuren zu vergleichen, liefert für die bei Fahrten durch die Gänge eines Bürogebäudes auftretenden Polygonzüge mit langen Geradenstücken und scharfen Kurven gute Ergebnisse.[4]

Sollen dagegen egozentrische Bewegungsspuren verglichen werden, die nicht die eckige Form einer Fahrt durch die Gänge eines Bürogebäudes besitzen, sondern eher runde, geschwungene Verläufe beschreiben, werden die Ergebnisse schlechter: Da weniger markante Ecken auftreten, können sich die Generalisierungen der beiden Spuren p und p' weit stärker voneinander unterscheiden, das Matching beider Spuren wird schwieriger.

In diesen Fällen erscheinen mir der folgende Ansatz erfolgversprechender: Latecki und Lakämper benutzen in [LL00] die Tangentenraumdarstellung von Polygonen zum Vergleich der Umrißlinien von Objekten. Das dort beschriebene Verfahren läßt sich mit einigen Modifikationen auch auf Bewegungsspuren anwenden.

In der Tangentenraumdarstellung (Abbildung 14.1) eines Polygonzuges wird in x-Richtung die Kurvenlänge und in y-Richtung der Winkel an der entsprechenden Stelle angetragen. Berechnet man nun die vorzeichenbehaftete Fläche A zwischen zwei Polygonzügen in Tangentenraumdarstellung, so schwankt diese um 0, wenn beide im Großen die gleiche Form beschreiben. Damit dies klappt müssen die Längen der Polygonzüge vorher geeignet normiert werden.

Weichen beide Polygonzüge nun aufgrund der Akkumulation von Meßfehlern nach und nach immer weiter voneinander ab, so drückt sich das in einem langsamen Wachstum der Fläche A aus, unterscheiden sie sich hingegen an einer bestimmten Stelle fundamental in ihrer Form, so führt dies zu einem raschen Flä-

[4] Werden die Ecken im Bewegungsverlauf durch die Generalisierung robust erkannt (an diesen Stellen ist der Roboter abgebogen), so liefern ähnliche Bewegungsverläufe auch ähnliche Generalisierungen. Zur Kompensation der Meßfehler müssen beim Vergleich allerdings starke Winkelabweichungen an den Eckpunkten der Generalisierungen toleriert werden. Bei Fahrten durch die Gänge eines Bürogebäudes ist dieser völlig ausreichend, da die möglichen Routen hier schon dadurch stark eingeschränkt sind, daß der Roboter nicht durch die Wand fahren kann. Falsches Abbiegen führt hier in den meisten Fällen umgehend zu so starken Abweichungen in der Bewegungsspur, daß es schnell erkannt wird.

chenanstieg.

Das Verfahren kann selbstverständlich nicht entscheiden, ob leichte Abweichungen Folge real unterschiedlicher Bewegungsverläufe oder Folge von Meßfehlern sind (wie auch), aber es kann feststellen, ob zwei Bewegungsverläufe einander ähnlich genug sind, daß auftretende Abweichungen Folge von Meßfehlern sein könnten oder nicht.

Offen ist hier zum einen die Frage der geeigneten Normierung der Polygonzuglängen, die sehr wesentlich für die Qualität des Verfahrens ist. Besitzt einer der Polygonzüge beispielsweise in einem Teilstück eine ausgeprägte Feinstruktur, z. B. eine Folge von Zickzackbewegungen aufgrund eines Rangiermanövers, so ist der Polygonzug an dieser Stelle länger, was zu einer Verschiebung im Tangentenraum und damit letztlich zu einem fehlerhaften Matching führt. Dieses Problem sollte sich durch die Wahl einer geeigneten Vorgeneralisierung (oder Glättung) lösen lassen.

Zum anderen ist bislang noch unklar, wie genau ein „langsamer“ Anstieg der Fläche A zu bewerten ist, also welche Schwellwerte und Parameter hier wie gewählt werden müssen.

Wegbeschreibungen. Die Verarbeitung einer (numerischen) Bewegungsspur endet natürlich nicht mit ihrer Vereinfachung oder der Umwandlung in eine qualitative Darstellung. Ein gerade in Bezug auf das Rolland-Projekt bzw. allgemein beim Einsatz mobiler autonomer Roboter interessanter Ansatz ist die Erzeugung von „natürlichsprachlichen“ Wegbeschreibungen[5] aus den Bewegungsspuren des Roboters und umgekehrt die Steuerung des Roboters mittels (semi-) „natürlichsprachlicher“ Wegbeschreibungen.

Ein erster Schritt hierzu ist ein in Zusammenarbeit mit dem Bremer Rolland-Projekt entwickelter Ansatz zur Navigation mittels qualitativer Routenbeschreibungen[6], der auf einer modifizierten QMV-Darstellung beruht. Ein wichtiges Element ist hierbei die Einbeziehung externer Landmarken in die Bewegungsbeschreibung. Eine Ausweitung dieses Ansatzes scheint lohnenswert.

Werden nun noch Verfahren entwickelt, die aus einer qualitativen propositionalen Darstellung eines Bewegungsverlaufes natürlichsprachliche Wegbeschreibungen generieren und umgekehrt eine natürlichsprachliche Wegbeschreibung parsen und in eine qualitative Bewegungssequenz umwandeln können (beides unter Einbeziehung von Landmarken), liefert dies ein nützliches Werkzeug zur Mensch-Maschine-Interaktion in einer ganzen Reihe von Anwendungen.

[5] Siehe z. B. [RR90]

[6] [Mus00, Kapitel 8]

Anhang A.

Implementierung

Die vorgestellte qualitative Architektur, die QMV-Algebra und die Generalisierungsalgorithmen wurden in Java implementiert. Die Implementierung umfaßt

- Klassen zur Repräsentation (und Manipulation) von Bewegungsverläufen in Form von numerischen Punkt- und Vektorsequenzen sowie in Form von QMV-Sequenzen und Shape-Sequenzen, wobei die einzelnen Darstellungen ineinander umgewandelt werden können.
- die QMV-Algebra (inklusive graphischer Darstellung von QMV-Sequenzen).
- eine Bibliothek der vorgestellten Generalisierungsalgorithmen.
- eine graphische Oberfläche (Abbildung A.1).
- ein textbasiertes Interface für den Batchbetrieb.

Die einzelnen Komponenten wurden in einer Testumgebung (MM-Tool) zusammengefaßt.

Die graphische Oberfläche besteht im wesentlichen aus drei Komponenten: dem Whiteboard (in der Abbildung links), dem Bedienfeld (rechts) und einem QMV-Textfeld (nicht dargestellt).

Im „Whiteboard“ werden numerische Bewegungsverläufe sowie graphische Repräsentationen von QMV-Sequenzen dargestellt. Darüberhinaus können hier Mausbewegungen gemessen werden, womit schnell und einfach Testdaten zur Bewegungsverarbeitung erzeugt werden können.

Das Bedienfeld bietet Zugriff auf die unterschiedlichen Generalisierungsalgorithmen, wobei auch mehrere Algorithmen hintereinandergeschaltet werden können.

Im QMV-Textfeld werden die qualitativen Repräsentationen von Bewegungsverläufen in Textform angezeigt und können dort editiert und manipuliert werden.

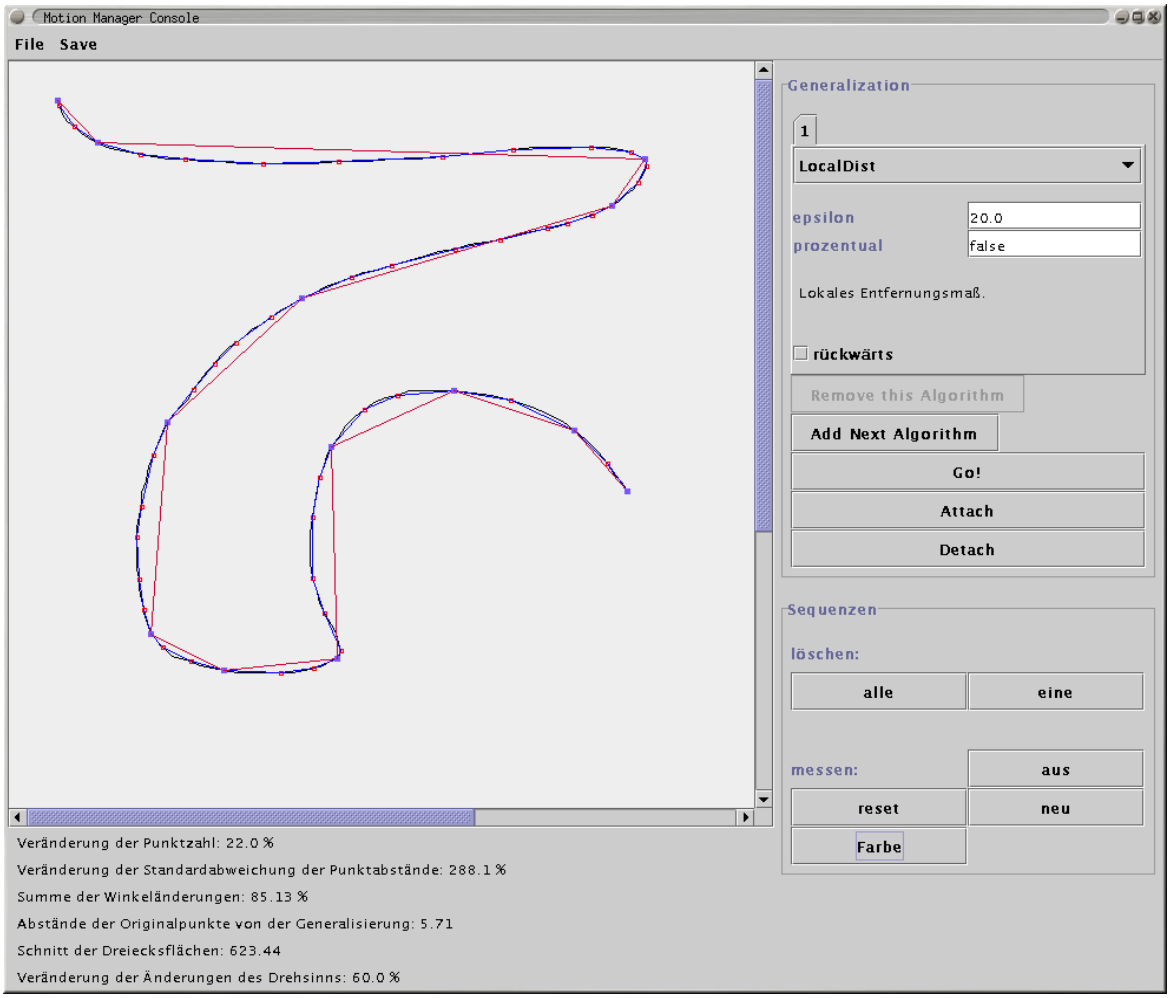


Abbildung A.1.: Die Oberfläche von MM.

Numerische Bewegungsverläufe werden als Polygonzüge im XFig-Format[1] gespeichert und können auch aus XFig-Dateien geladen werden. Dies erlaubt sowohl die einfache Weiterverarbeitung als auch die Umwandlung in und aus Postscript und PDF.[2] QMV- und Shape-Sequenzen werden als reiner ASCII-Text gespeichert und geladen und können somit mit einem beliebigen Editor bearbeitet werden.

Bei der Implementierung wurde auf hohe Modularität Wert gelegt. Die einzelnen Komponenten sind weitgehend unabhängig voneinander und kommunizieren über festgelegte Interfaces. Dadurch lassen sich unterschiedliche Richtungs- und Entfernungsdomänen für die QMV-Repräsentation einfach auswählen und testen,

[1] XFig ist ein beliebtes Vektorgraphikprogramm für UNIX und Linux.
[2] Diese Möglichkeit fand unter anderem auch für die in dieser Arbeit gezeigten Graphiken Anwendung.

ebenso können neue Generalisierungsalgorithmen mit sehr geringem Aufwand hinzugefügt und getestet werden. Ein neuer Generalisierungsalgorithmus meldet sich hierbei selbständig mit Art und Typ der von ihm benötigten Parameter bei der Oberfläche an, wodurch das Hinzufügen neuer Algorithmen keine Veränderungen im bestehenden Code verlangt.

Neben der Anwendung innerhalb der graphischen Oberfläche lassen sich alle Generalisierungsalgorithmen auch im Batchbetrieb von der Kommandozeile aus aufrufen, wodurch ganze Sammlungen von Polygonzügen automatisiert generalisiert werden können.

Literaturverzeichnis

[All83] James F. Allen. Maintaining Knowledge about Temporal Intervals. *Communications of the ACM*, 26(11):832–843, November 1983.

[AS93] Günter Aumann, Klaus Spitzmüller. *Computerorientierte Geometrie*, Band 89 von *Reihe Informatik*. B.I. Wissenschaftsverlag, Mannheim, Wien, Zürich, 1993.

[BDH96] C. Bradford Barber, David P. Dobkin, Hannu Huhdanpaa. The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software (TOMS)*, 22(4):469–483, 1996. ISSN 0098-3500.

[BLR00] Thomas Barkowsky, Longin Jan Latecki, Kai-Florian Richter. Schematizing Maps: Simplification of Geographic Shape by Discrete Curve Evolution. In Freksa et al. [FBHW00], Seiten 41–53.

[CDFH97] Eliseo Clementini, Paolino Di Felice, Daniel Hernández. Qualitative representation of positional information. *Artificial Intelligence*, 95(2):317–356, 1997.

[Cla81] B. L. Clarke. A Calculus of Individuals Based on Connection. *Notre Dame Journal of Formal Logic*, 2(3), 1981.

[Coh97] Anthony G. Cohn. Qualitative Spatial Representation and Reasoning Techniques. In G. Brewka, C. Habel, B. Nebel (Herausgeber), *Proceedings of KI-97*, Nummer 1303 in Lecture Notes in Artificial Intelligence, Seiten 1–30. Springer, Berlin, Heidelberg, New York, 1997.

[Cro91] Robert G. Cromley. Hierarchical methods of line simplification. *Cartography and Geographic Information Systems*, 18(2):125–131, 1991.

[dBvKS95] Marc de Berg, Marc van Kreveld, Stefan Schirra. A new approach to subdivision simplification. In *Twelfth International Symposium on Computer- Assisted Cartography*, Band 4, Seiten 79–88. Charlotte, North Carolina, 1995.

[DP73] David H. Douglas, Thomas K. Peucker. Algorithms for the reduction of the number of points required to represent a line or its caricature. *The Canadian Cartographer*, 10(2):112–122, 1973.

[EMS+97] Andreas Eisenkolb, Alexandra Musto, Kerstin Schill, D. Hernández, Wilfried Brauer. Qualitative Representation of the Course of Motion: Cognitive and Psychophysical Foundations. Forschungsberichte Künstliche Intelligenz FKI-225-97, Institut für Informatik, Technische Universität München, Oktober 1997.

[EMS+98] Andreas Eisenkolb, Alexandra Musto, Kerstin Schill, D. Hernández, Wilfried Brauer. Representational Levels for the Perception of the Courses of Motion. In Freksa et al. [FHW98], Seiten 129–155.

[EMZ+98] Andreas Eisenkolb, Alexandra Musto, Christoph Zetzsche, Wilfried Brauer, Kerstin Schill. Der Einfluß von Interstimulus-Intervall und örtlichem Versatz auf die Richtungsdiskrimination von Punktbahnen. In H. Buelthoff, M. Fahle, K. Gegenfurthner, H. Mallot (Herausgeber), *1. Tübinger Wahrnehmungskonferenz, Visuelle Wahrnehmung*, Seite 45. Knirsch Verlag, 1998.

[ES99] Andreas Eisenkolb, Kerstin Schill. Investigating the processing of occluded spatio-temporal patterns: The "Dynamic Poggendorff" Effect. In *ECVP '99, Perception 28*, Seite 148. 1999.

[ESB+00] Andreas Eisenkolb, Kerstin Schill, Volker Baier, Alexandra Musto, Wilfried Brauer. Visual Processing and Representation of Spatio-Temporal Patterns. In Freksa et al. [FBHW00].

[EZM+98] Andreas Eisenkolb, Christoph Zetzsche, Alexandra Musto, Wilfried Brauer, Kerstin Schill. Analysis and modelling of the visual processing of spatio-temporal information: influence of temporal and spatial separation on the discrimination of trajectories. In *ECVP 98, Perception 27*, Seite 188. 1998.

[FBHW00] Christian Freksa, Wilfried Brauer, Christopher Habel, Karl Friedrich Wender (Herausgeber). *Spatial Cognition II, Integrating Abstract Theories, Empirical Studies, Formal Methods, and Practical Applications*, Band 1849 von *Lecture Notes in Artificial Intelligence*. Springer, 2000. ISBN 3-540-67584-1.

[FHW98] C. Freksa, C. Habel, K. Wender (Herausgeber). *Spatial Cognition. An Interdisciplinary Approach to Representing and Processing Spatial Knowledge*, Band 1404 von *Lecture Notes in Artificial Intelligence*. Springer, 1998.

[FM99] Christian Freksa, David M. Mark (Herausgeber). *Spatial Information Theory. Cognitive and Computational Foundations of Geographic Information Science. International Conference COSIT'99*, Band 1661 von *Lecture Notes in Computer Science*. Springer, Berlin, Heidelberg, New York, August 1999.

[FMB00] Christian Freksa, Reinhard Moratz, Thomas Barkowsky. Schematic Maps for Robot Navigation. In Freksa et al. [FBHW00], Seiten 100–114.

[For83] Kenneth D. Forbus. Qualitative reasoning about space and motion. In Dedre Gentner, Albert L. Stevens (Herausgeber), *Mental Models*, Seiten 53–73. Lawrence Erlbaum, Hillsdale, NJ, 1983.

[Fre91] Christian Freksa. Qualitative Spatial Reasoning. In Mark und Frank [MF91], Seiten 361–372.

[Fre92a] Christian Freksa. Temporal reasoning based on semi-intervals. *Artificial Intelligence*, 54:199–227, 1992.

[Fre92b] Christian Freksa. Using Orientation Information for Qualitative Spatial Reasoning. In Andrew U. Frank, I. Campari, Formentini U. (Herausgeber), *Theories and Methods of Spatio-Temporal Reasoning in Geographic Space*, Nummer 639 in Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, New York, 1992.

[Gal93] Antony Galton. Towards an integrated logic of space, time, and motion. In Ruzena Bajcsy (Herausgeber), *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, Seiten 1550–1555. International Joint Conferences on Artificial Intelligence, Inc., Morgan Kaufmann, San Mateo, CA, August 1993.

[Gal95] Antony Galton. Towards a Qualitative Theory of Movement. In Andrew U. Frank, Werner Kuhn (Herausgeber), *Spatial Information Theory. A Theoretical Basis for GIS. European Conference, COSIT'95*, Band 988 von *Lecture Notes in Computer Science*, Seiten 377–396. Springer, Berlin, Heidelberg, New York, September 1995.

[Gal97] Antony Galton. Continuous Change in Spatial Relations. In Stephen C. Hirtle, Andrew U. Frank (Herausgeber), *Spatial Information Theory. A Theoretical Basis for GIS*, Nummer 1329 in Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, New York, 1997.

[GBH98] Bryan L. Gros, Randolph Blake, Eric Hiris. Anisotropies in visual motion perception: A fresh look. *Journal of the Optical Society of America A*, 15(8):2003–2011, August 1998.

[GE] Roop K. Goyal, Max J. Egenhofer. Cardinal Directions between Extended Spatial Objects. *IEEE Transactions on Knowledge and Data Engineering*, in press.

[Hab99] Christopher Habel. Drehsinn und Reorientierung – Modus und Richtung beim Bewegungswerb drehen. In G. Rickheit (Herausgeber), *Richtung im Raum*. Deutscher Universitäts-Verlag, Wiesbaden, 1999.

[Her91] Daniel Hernández. Relative Representation of Spatial Knowledge: The 2-D Case. In Mark und Frank [MF91], Seiten 373–385.

[Her94] Daniel Hernández. *Qualitative Representation of Spatial Knowledge*, Band 804 von *Lecture Notes in Artificial Intelligence*. Springer, Berlin, Heidelberg, New York, 1994.

[Her96] Daniel Hernández. Repräsentation und Verarbeitung räumlichen Wissens, Wintersemester 1996. Vorlesungsskript, TU München.

[HHM99] Christopher Habel, Bernd Hildebrandt, Reinhard Moratz. Interactive Robot Navigation based on Qualitative Spatial Representations. Seiten 219–224. infix, Sankt Augustin, 1999.

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

[HS97] John Hershberger, Jack Snoeyink. Cartographic Line Simplification and Polygon CSG Formulae and in $O(n \log n)$ Time. In *Workshop on Algorithms and Data Structures*, Seiten 93–103. 1997.

[JH96] Erland Jungert, Daniel Hernández. Qualitative motion of point-like objects, 1996. Unpublished manuscript.

[Joh73] Gunnar Johansson. Visual perception of biological motion and a model for its analysis. *Perception & Psychophysics*, 14(2):201–211, 1973.

[Joh75] Gunnar Johansson. Visual Motion Perception. *Scientific American*, 232(4):76–88, 1975.

[Joh76] Gunnar Johansson. Spatio-Temporal Differentiation and Integration in Visual Motion Perception. *Psychological Research*, (38):379–393, 1976.

[Lan02] Axel Lankenau. *The Bremen Autonomous Wheelchair "Rolland": Self-Localization and Shared Control*. Doktorarbeit, Universität Bremen, 2002.

[Lev96] S. C. Levinson. Frames of reference and Molyneux's question. In Paul Bloom, Mary A. Peterson, Lynn Nadel, Merrill F. Garrett (Herausgeber), *Language and Space*, Seiten 109–169. MIT Press, Cambridge, MA, 1996.

[LL99a] Longin Jan Latecki, Rolf Lakämper. Convexity Rule for Shape Decomposition Based on Discrete Contour Evolution. *Computer Vision and Image Understanding (CVIU)*, 73(3):441–454, 1999.

[LL99b] Longin Jan Latecki, Rolf Lakämper. Polygon Evolution by Vertex Deletion. In Mads Nielsen, Peter Johansen, Ole Fogh Olsen, Joachim Weickert (Herausgeber), *Scale-Space*, Band 1682 von *Lecture Notes in Computer Science*, Seiten 398–409. Springer, 1999. ISBN 3-540-66498-X.

[LL00] Longin Jan Latecki, Rolf Lakämper. Shape Similarity Measure Based on Correspondence of Visual Parts. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 22(10):1185–1190, 2000.

[McM87] Robert B. McMaster. Automated Line Generalization. *Cartographica*, 24(2):74–111, 1987.

[ME98] Alexandra Musto, Andreas Eisenkolb. Eine intermediäre Schicht in der Bewegungswahrnehmung. Forschungsberichte Künstliche Intelligenz FKI-226-98, Institut für Informatik, Technische Universität München, Oktober 1998.

[MF91] David M. Mark, Andrew U. Frank (Herausgeber). *Cognitive and Linguistic Aspects of Geographic Space*. Kluwer, 1991.

[MRL+00] R. Müller, T. Röfer, A. Lankenau, A. Musto, K. Stein, A. Eisenkolb. Coarse Qualitative Descriptions in Robot Navigation. In Freksa et al. [FBHW00], Seiten 265–276.

[MSE+00] Alexandra Musto, Klaus Stein, Andreas Eisenkolb, Kerstin Schill, Thomas Röfer, Wilfried Brauer. From Motion Observation to Qualitative Motion Representation. In Freksa et al. [FBHW00], Seiten 115–126.

[MSER99] Alexandra Musto, Klaus Stein, Andreas Eisenkolb, Thomas Röfer. Qualitative and Quantitative Representations of Locomotion and their Application in Robot Navigation. In Thomas Dean (Herausgeber), *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99)*, Seiten 1067–1073. Morgan Kaufman Publishers, Inc., San Francisco, CA, August 1999.

[Mul98a] P. Muller. A Qualitative Theory of Motion Based on Spatio-temporal Primitives. In A.G. Cohn, L.K. Schubert, S.C. Shapiro (Herausgeber),

Principles of Knowledge Representation and Reasoning: Proceedings of the Sixth International Conference (KR'98). Morgan Kaufmann, 1998.

[Mul98b] P. Muller. Space-Time as a Primitive for Space and Motion. In *Proceedings of the International Conference on Formal Ontology in Information Systems (FOIS98)*, Frontiers in Artificial Intelligence and Applications. IOS Press, 1998.

[Mus00] Alexandra Musto. *Qualitative Repräsentation von Bewegungsverläufen*. Doktorarbeit, TU München, 2000.

[Nun95] Joan Nunes. General Concepts of Space and Time. In Andrew Frank (Herausgeber), *GIS-Material for a Post-Graduate Course*, Band 1, Seiten 7–34. TU Wien, 1995.

[PJ92] Jonas Persson, Erland Jungert. Generation of multi-resolution maps from run-length-encoded data. *International Journal of Geographical Information Systems*, 6(6):497–510, 1992.

[Rai99] Hans Raith. *Methoden zur Analyse von Bewegungsinformationen in technischen Systemen und Implementierung eines biologischen Modells zur Repräsentation spatio-temporaler Informationen*. Diplomarbeit, Institut für Informatik, Technische Universität München, 1999.

[Ram72] Urs Ramer. An iterative apprach for polygonal approximation of planar closed curves. In *Computer Graphics and Image Processing*, Band 1, Seiten 244–256. 1972.

[RCC92] D. A. Randell, Z. Cui, A. G. Cohn. A Spatial Logic Based on Regions and Connection. In *Proceedings 3rd International Conference on Knowledge Representation and Reasoning*, Seiten 165–176. Morgan Kaufmann, San Mateo, CA, 1992.

[RL98] Thomas Röfer, Axel Lankenau. Architecture and Applications of the Bremen Autonomous Wheelchair. In P. P. Wang (Herausgeber), *Proceedings of the Fourth Joint Conference on Information Systems*, Band 1, Seiten 365–368. Association of Intelligent Machinery, 1998.

[Röf99] Thomas Röfer. Route Navigation Using Motion Analysis. In Freksa und Mark [FM99], Seiten 21–36.

[RR90] P. Ruhrberg, H. Rutz. Räumliches Wissen und Semantik im Kontext der Generierung von Wegbeschreibungen. Band 245 von *Informatik-Fachberichte*, Seiten 235–249. Springer, Berlin, Heidelberg, New York, 1990.

[RSB+] Florian Röhrbein, Kerstin Schill, Volker Baier, Klaus Stein, Christoph Zetzsche, Wilfried Brauer. Motion Shapes: Empirical Studies and Neural Modeling. In C. Freksa, C. Habel, K. Wender (Herausgeber), *Spatial Cognition*, Lecture Notes in Artificial Intelligence. Springer.

[Ste98] Klaus Stein. *Generalisierung und Segmentierung von qualitativen Bewegungsdaten*. Diplomarbeit, Institut für Informatik an der TU München, 1998.

[Ste01] Angelika Steger. *Diskrete Strukturen*, Band 1. Springer, Berlin, Heidelberg, New York, 2001.

[Sto97] Oliviero Stock (Herausgeber). *Spatial and Temporal Reasoning*. Kluwer, 1997.

[SZ95] Kerstin Schill, Christoph Zetzsche. A model of visual spatio-temporal memory: The icon revisited. *Psychological Research*, 57:88–102, 1995.

[SZB+98] Kerstin Schill, Christoph Zetzsche, Wilfried Brauer, Andreas Eisenkolb, Alexandra Musto. Visual representation of spatio-temporal structure. In B. E. Rogowitz, T. N. Pappas (Herausgeber), *Human Vision and Electronic Imaging. Proceedings of SPIE*, Seiten 128–137. 1998.

[TL99] Barbara Tversky, Paul U. Lee. Pictorial and verbal tools for conveying routes. In Freksa und Mark [FM99], Seiten 51–64.

[VW93] Maheswari Visvalingam, J. D. Whyatt. Line generalisation by repeated elimination of points. *The Cartographic Journal*, 30(1):46–51, 1993.

[Weh99] Rüdiger Wehner. Large-Scale Navigation: The Insect Case. In Freksa und Mark [FM99], Seiten 1–20.

[WW88] J. D. Whyatt, P. R. Wade. The Douglas-Peucker line simplification algorithm. *Bulletin of the Society of University Cartographers*, 22(1):17–27, 1988.

[ZSB+98] Hubert D. Zimmer, Harry R. Speiser, Jörg Baus, Anselm Blocher, Eva Stopp. The Use of Locative Espressions in Dependence of the Spatial Relation between Target and Reference Object in Two-Dimensional Layouts. In Freksa et al. [FHW98], Seiten 223–240.