Fernuniversität in Hagen - Fakultät für Informatik

Masterarbeit

zur Erlangung des akademischen Grades
Master of Science Informatik

Paarweise Sichtbarkeit in Polygonen

Autor: Klaus Teufel
Hauptweg 24
1170 Wien

Matrikelnummer: 7260466

Abgabetermin: 4.5.2020

1. Betreuer: Prof. Dr. André Schulz
2. Betreuerin: Dr. Lena Schlipf

Zusammenfassung

Der Artikel „On Romeo and Juliet Problems: Minimizing Distance-to-Sight“ [1] von Ahn et al. stellt folgendes Problem der Algorithmischen Geometrie vor: gegeben seien 2 Punkte s und t innerhalb eines einfachen Polygons P mit n Knoten. Was ist dann die minimale Distanz, die sie zurücklegen müssen, um einander sehen zu können? Dieses Problem kann anhand eines von Ahn et al. eingeführten Algorithmus in $\mathcal{O}(n)$ Zeit gelöst werden.

Die vorliegende Masterarbeit beschäftigt sich mit diesem Problem auf zweifache Weise: im ersten Teil der Arbeit wird der Lösungsalgorithmus theoretisch erklärt, d.h. der Inhalt des oben genannten Artikels wird wiedergegeben und teilweise weiterentwickelt. Der zweite Teil der Arbeit besteht in einer Implementierung des Algorithmus in der Programmiersprache C++ unter Zuhilfenahme der Bibliothek CGAL [5] und des Qt-Frameworks [19].

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Aufgabenstellung

Sichtbarkeitsprobleme sind ein wichtiges Themenfeld der Algorithmischen Geometrie. Darunter fallen z.B. das „Art Gallery Problem“, bei dem es darum geht, an wie vielen Punkten innerhalb einer Kunstgalerie Aufpasser positioniert werden müssen, so dass jeder Punkt in der Galerie für mindestens einen Aufpasser sichtbar ist [8]. Ein ähnliches Problem ist das „Watchman Route Problem“. Es besteht darin, in einem Polygon mit n Knoten einen Pfad zu finden, so dass jeder Punkt im Polygoninneren und auf dem Polygonrand von mindestens einem Punkt auf dem Pfad sichtbar ist [4, 7, 22]. Sichtbarkeit zwischen zwei Punkten p und q bedeutet dabei, dass die Strecke zwischen p und q ganz im Inneren des Polygons verläuft.

Die vorliegende Masterarbeit beschäftigt sich mit einer Variante dieser Probleme, dem *paarweisen Sichtbarkeitsproblem*, das von Ahn et al. im Artikel „On Romeo and Juliet Problems: Minimizing Distance-to-Sight“ [1] vorgestellt wurde. Informell geht es bei diesem Problem darum, dass sich zwei Personen - Romeo und Julia - in einem Polygon befinden. Sie wollen die Distanzen[1] minimieren, die sie zurücklegen müssen, um sich gegenseitig sehen zu können. Formal ist das Problem folgendermaßen definiert:

[1] Wann immer in dieser Arbeit von Distanz die Rede ist, ist die euklidische Distanz gemeint.

Paarweises Sichtbarkeitsproblem: *Gegeben seien zwei Punkte s und t in einem einfachen Polygon P mit n Knoten. Gesucht sind Punktepaare $s^*, t^* \in P$ und Pfade von s nach s^* und von t nach t^*, für die gilt:*

(i) *s^* und t^* sind gegenseitig sichtbar*

(ii) *die Pfade von s nach s^* und von t nach t^* verlaufen innerhalb des Polygons und ihre Länge ist minimal*

Für das Problem gibt es 2 Varianten: bei der *min-max* Variante soll die längere der beiden Distanzen minimiert werden. Bei der *min-sum* Variante soll die Summe der beiden Distanzen minimiert werden. Für beide Varianten entwickeln Ahn et al. einen Algorithmus, der in Zeitkomplexität $\mathcal{O}(n)$ eine Lösung berechnet.[2] Die Autoren stellen auch eine Anfrageversion für die *min-max* Variante des Problems vor, die in dieser Masterarbeit aber nicht behandelt wird.

Ich beschäftige mich in dieser Masterarbeit mit dem paarweisen Sichtbarkeitsproblem auf zweifache Weise: erstens gehe ich auf die theoretischen Details des von Ahn et al. vorgestellten Algorithmus ein und verfeinere den Algorithmus, wo es notwendig ist. Zweitens stelle ich eine praktische Implementierung des Algorithmus vor, die ich in der Programmiersprache C++ geschrieben habe.

Die Wahl von C++ ist eher willkürlich, denn der Algorithmus könnte auch in einer anderen höheren Programmiersprache implementiert werden. C++ ist aber insofern eine sinnvolle Wahl, weil es in C++ einfach einzubindende Bibliotheken und Frameworks gibt, die für die Implementierung notwendig sind: zum einen die CGAL Bibliothek [5], die viele geometrische Algorithmen, insbesondere für die Triangulierung eines Polygons bereitstellt, und zum anderen das Qt-Framework [19], das eine Visualisierung der Ergebnisse und Tests ermöglicht.

Während der praktischen Implementierung sind mir einige kleinere Ungenauigkeiten und Probleme in der Formulierung des Algorithmus durch Ahn et al. aufgefallen. Ich habe versucht, diese Ungenauigkeiten im theoretischen

[2]Beide Varianten des paarweisen Sichtbarkeitsproblems sind trivial, wenn es eine Strecke zwischen s und t gibt, die ganz im Polygoninneren verläuft. Um dies zu überprüfen, kann man die Schnittpunkte zwischen der Strecke und den Kanten von P berechnen, was in $\mathcal{O}(n)$ Zeit möglich ist. Da das triviale Problem uninteressant ist, wird im folgenden davon ausgegangen, dass die Strecke zwischen s und t nicht ganz im Polygoninneren verläuft.

Teil der Arbeit auszuräumen und dadurch zu einem besseren Verständnis des paarweisen Sichtbarkeitsproblems beizutragen. Das wiederum hat dabei geholfen, die praktische Implementierung einfacher und robuster zu gestalten.

1.2 Verwandte Arbeiten

Wie schon erwähnt gibt es eine Reihe von Sichtbarkeitsproblemen, mit denen sich die Algorithmische Geometrie beschäftigt und die eine gewisse Ähnlichkeit mit dem paarweisen Sichtbarkeitsproblem haben. Für das „Art Gallery Problem“ bewies Chvátal als erster, dass für eine Lösung maximal $\frac{n}{3}$ Aufpasser notwendig sind [8]. Für das „Watchman Route Problem“ schlugen Carlsson, Jonsson und Nilsson einen Algorithmus vor, der das Problem im schlechtesten Fall in Zeitkomplexität $\mathcal{O}(n^6)$ löst [4], der von Tan entwickelte Algorithmus löst das Problem in $\mathcal{O}(n^5)$ [22].

Wynters und Mitchell beschäftigten sich mit der Frage, welche minimalen Pfade zwei Roboter, die sich in einer Ebene mit polygonalen Hindernissen befinden, zurücklegen müssen, um gegenseitig sichtbar zu sein. Für die *min-sum* Variante des Problems schlugen sie einen Algorithmus mit Zeitkomplexität $\mathcal{O}(nE)$ vor, wobei n die Anzahl der Knoten der polygonalen Hindernisse bezeichnet und E die Anzahl der Kanten der Sichtbarkeitsgraphen dieser Hindernisse. Für die *min-max* Variante des Problems ist die Zeitkomplexität des von ihnen entwickelten Algorithmus $\mathcal{O}(n^3 \log n)$ [26].

Khosravi und Ghodsi haben das Problem untersucht, in einem einfachen Polygon die minimale Distanz von einem Startpunkt zu einem Polygonknoten zu finden, von dem aus ein Anfragepunkt sichtbar ist [14]. Sie entwickelten einen Algorithmus, der ein Polygon mit n Knoten in Zeit $\mathcal{O}(n^2)$ so aufbereitet, dass Anfragen bezüglich der minimalen Distanz in Zeit $\mathcal{O}(\log n)$ beantwortet werden können.[3] Arkin et al. verbesserten die Ergebnisse von Khosravi und Ghodsi und erweiterten den Algorithmus für ein Polygon mit Löchern [3]. Wang schlug einen Algorithmus vor, der die Resultate für den Fall verbessert, dass die Anzahl der Löcher in dem Polygon relativ klein ist [24].

Die genannten Probleme ähneln dem hier vorgestellten Sichtbarkeitspro-

[3]Mir ist aufgefallen, dass die von Khosravi und Ghodsi angegebene Zeitkomplexität zur Aufbereitung des Polygons $\mathcal{O}(n^2)$ ist, während Guibas et al.[13] eine lineare Zeitkomplexität angeben, um ähnliche Anfragen in $\mathcal{O}(\log n)$ beantworten zu können. Da ich mich nicht besonders intensiv mit dem Artikel von Khosravi und Ghodsi beschäftigt habe, kann ich nicht sagen, worauf diese Diskrepanz beruht.

blem. Allerdings ist der Artikel von Ahn et al. [1] meines Wissens die erste und einzige Publikation, die das paarweise Sichtbarkeitsproblem vorstellt und eine Lösung dafür entwickelt.

Der Algorithmus zur Lösung des paarweisen Sichtbarkeitsproblems setzt - ähnlich wie viele andere Sichtbarkeitsprobleme - eine Triangulierung des betrachteten Polygons voraus. Bernard Chazelle bewies anhand eines deterministischen Algorithmus, dass jedes einfache Polygon in Zeitkomplexität $\mathcal{O}(n)$ trianguliert werden kann [6]. Amato et al. schlugen einen als weniger kompliziert geltenden randomisierten Algorithmus vor, der eine erwartete lineare Laufzeit hat [2]. Ich nehme in dieser Masterarbeit an, dass das betrachtete Polygon schon anhand eines linearen Algorithmus trianguliert worden ist.

Für die Lösung des paarweisen Sichtbarkeitsproblems sind außerdem die folgenden Konzepte und Datenstrukturen von Bedeutung: der kürzeste Pfad zwischen s und t innerhalb des Polygons, der *Shortest Path Tree*, der die kürzesten Pfade von einem Ausgangspunkt zu allen Knoten eines Polygons enthält, sowie die *Shortest Path Map*, eine Erweiterung des *Shortest Path Tree*, die es ermöglicht, die Länge des kürzesten Pfads von einem Ausgangspunkt zu einem beliebigen Punkt in einem Polygon in Zeit $\mathcal{O}(\log n)$ zu finden.[4] Lee und Preparata haben einen Algorithmus vorgeschlagen, der den kürzesten Pfad zwischen s und t in einem triangulierten Polygon in linearer Zeit findet [15]. Aufbauend auf diesem Algorithmus haben Guibas et al. lineare Algorithmen für den Aufbau des *Shortest Path Tree* sowie der *Shortest Path Map* entwickelt [13].

Für den praktischen Teil der Arbeit habe ich keine Implementierung eines ähnlichen Problems angeschaut. Ich habe alle notwendigen Algorithmen selbst geschrieben, mit Ausnahme der Polygontriangulierung sowie mehrerer einfacher Algorithmen wie z.B. zum Berechnen des Schnittpunkts zweier Geraden, für die ich die Implementierung von *CGAL* [5] benutze.

1.3 Lösung des paarweisen Sichtbarkeitsproblems - eine Zusammenfassung

Der von Ahn et al. vorgeschlagene Algorithmus zur Lösung des paarweisen Sichtbarkeitsproblems basiert auf einem modifizierten Sweep-Ansatz. Normalerweise wird bei einem zweidimensionalen Sweep-Algorithmus eine soge-

[4] Siehe Kapitel 2.1 für genaue Definitionen

nannte Sweep-Gerade über die Ebene bewegt, bis alle Objekte des Problems besucht und verarbeitet wurden [25, 9]. Beim paarweisen Sichtbarkeitsproblem fegt die Sweep-Gerade nicht über die Ebene hinweg. Statt dessen wird sie an den Knoten des kürzesten Pfads von s nach t gedreht. Es wird auch nicht die ganze Gerade betrachtet, sondern nur der Ausschnitt, der sich innerhalb des Polygons befindet. Dieser Geradenausschnitt wird im folgenden *Sichtbarkeitsgerade* genannt.[5] Jede Lösung des paarweisen Sichtbarkeitsproblems liegt auf einer derartigen Sichtbarkeitsgeraden.

Um alle für eine Lösung relevanten Sichtbarkeitsgeraden zu berücksichtigen, wird der Algorithmus mit einer Sichtbarkeitsgeraden initialisiert, die durch s und den zweiten Knoten des kürzesten Pfads von s nach t verläuft. Diese Gerade wird solange am zweiten Knoten des kürzesten Pfads gedreht, bis sie auf den dritten Knoten des kürzesten Pfads trifft. Dann wird sie am dritten Knoten des kürzesten Pfads gedreht, bis sie auf den vierten Knoten trifft usw. Die Drehung endet, wenn die Sichtbarkeitsgerade, die am vorletzten Knoten des kürzesten Pfads gedreht wird, auf t trifft. Während der Drehung trifft die Sichtbarkeitsgerade auf alle potentiellen Lösungen des paarweisen Sichtbarkeitsproblems.

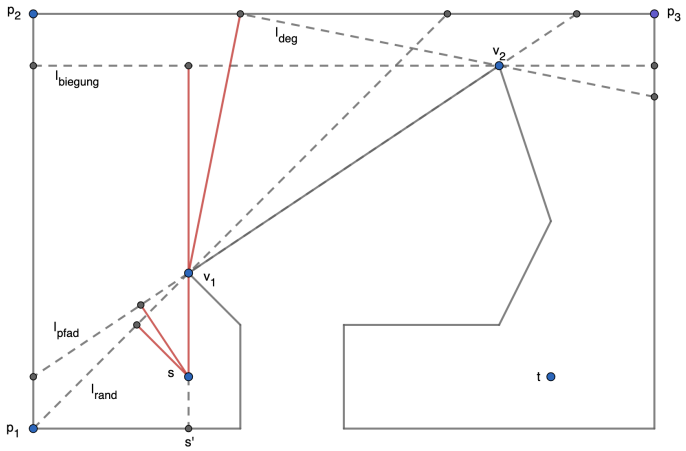

Abbildung 1.1: Pfad-, Rand- und (degenerierte) Biegungsereignisse

Um nun die optimalen Sichtbarkeitsgeraden in dieser unendlichen Menge von potentiellen Lösungen bestimmen zu können, müssen die Ereignisse

[5] Siehe Kapitel 2.2 für eine genaue Definition

identifiziert werden, die eine besondere Bedeutung beim Finden einer Lösung haben. Dazu stellt man sich vor, wie die Sichtbarkeitsgerade durch s und den zweiten Knoten des kürzesten Pfads von s nach t in Richtung t gedreht wird, siehe Abbildung 1.1. Zu Beginn der Drehung verläuft das eine Ende der Sichtbarkeitsgeraden auf dem Kantenstück zwischen s' und p_1. Der kürzeste Pfad von s zur Sichtbarkeitsgeraden besteht aus genau einer Kante, deren Länge durch die Steigung der Sichtbarkeitsgeraden bestimmt ist. Sobald die Sichtbarkeitsgerade auf den Knoten p_1 trifft, ändert sich die Polygonkante, die von ihr bei weiterer Drehung überstreift wird. Da eine Sichtbarkeitsgerade durch ihre Endpunkte auf dem Polygonrand definiert ist, sind derartige Ereignisse beim Finden der Lösung von Bedeutung. Sie werden als *Randereignisse* bezeichnet.

Am Knoten v_1 kann die Sichtbarkeitsgerade nur bis zu l_{pfad} gedreht werden. Bei dieser Sichtbarkeitsgeraden ändert sich der Drehpunkt von v_1 zu v_2. Auch diese Ereignisse sind beim Finden einer Lösung von Bedeutung und werden als *Pfadereignisse* bezeichnet.

Auch nach dem Pfadereignis, besteht der kürzeste Pfad von s zur Sichtbarkeitsgeraden aus genau einer Kante. Dies ändert sich, wenn die Sichtbarkeitsgerade zu $l_{biegung}$ weitergedreht wird.[6] Ab dieser Sichtbarkeitsgeraden besteht der kürzeste Pfad von s zur Sichtbarkeitsgeraden aus 2 Kanten: erstens der Kante zwischen s und v_1 und zweitens aus der Kante von v_1 zur Sichtbarkeitsgeraden. Derartige Ereignisse werden als *Biegungsereignisse* bezeichnet.

Bei den bisher betrachteten Sichtbarkeitsgeraden fiel der Endpunkt des Pfads von s zur Sichtbarkeitsgeraden mit dem Lotfußpunkt auf der Sichtbarkeitsgeraden zusammen. Das ändert sich bei der Sichtbarkeitsgeraden l_{deg}.[7] Bei weiterer Drehung ist der Lotfußpunkt von v_1 aus nicht mehr sichtbar, so dass der Endpunkt des kürzesten Pfads von s zur Sichtbarkeitsgeraden beim Schnittpunkt der Sichtbarkeitsgeraden mit dem Polygonrand liegt. Dies hat Auswirkungen darauf, wie die Länge der letzten Kante des Pfads von s zur Sichtbarkeitsgeraden berechnet wird. Derartige Ereignisse werden als *degenerierte Biegungsereignisse*[8] bezeichnet.

[6] Auf dem Weg dorthin trifft die Sichtbarkeitsgerade auf den Knotenpunkt p_3, wo ein weiteres Randereignis stattfindet, das aber nicht eingezeichnet ist, um die Darstellung übersichtlich zu halten.

[7] Bevor die Sichtbarkeitsgerade auf l_{deg} trifft, tritt bei p_2 ein weiteres Randereignis ein. Auch dieses ist zum Zweck der Übersichtlichkeit nicht eingezeichnet

[8] Der Name *degeneriertes Biegungsereignis* geht auf meine Betreuerin Dr. Schlipf

Man kann also vier Arten von Ereignissen unterscheiden:

- *Pfadereignisse*, bei denen sich der Drehpunkt ändert
- *Randereignisse*, bei denen sich die von der Sichtbarkeitsgeraden überstreifte Polygonkante ändert
- *Biegungsereignisse*, bei denen sich die inneren Knoten von einem der Pfade von s bzw. t zur Sichtbarkeitsgeraden ändern
- *degenerierte Biegungsereignisse*, bei denen sich die Distanzfunktion der letzten Kante des kürzesten Pfads von s zur Sichtbarkeitsgeraden ändert

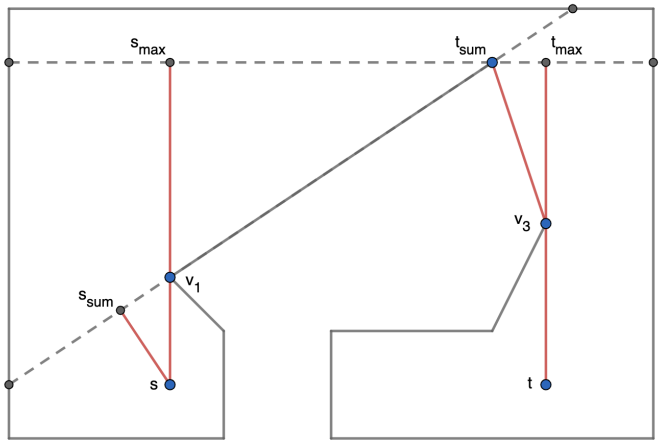

Abbildung 1.2: Die Lösung des min-sum Problems liegt bei s_{sum} und t_{sum}, die Lösung des min-max Problems bei s_{max} und t_{max}.

Alle Pfad-, Rand- und Biegungsereignisse (degeneriert und nicht-degeneriert) können anhand mehrerer Drehungen der Sichtbarkeitsgeraden von s nach t und in entgegengesetzter Richtung berechnet werden. Nach Abschluss der Berechnungen ist für jede Region, die von zwei aufeinanderfolgenden Ereignissen begrenzt wird, folgendes bekannt: der aktuelle Drehpunkt, die Ausschnitte der Polygonkanten, die von der Sichtbarkeitsgeraden innerhalb der Region überstreift werden, die Knoten der kürzesten Pfade von s und t zu den begrenzenden Sichtbarkeitsgeraden und die Distanzfunktion, die in

zurück.

der Region für die letzte Kante des kürzesten Pfads von s bzw. t zur Sichtbarkeitsgeraden verwendet werden muss. Innerhalb einer Region bleiben die kürzesten Pfade von s und t zu den Knoten vor der Sichtbarkeitsgeraden gleich. Lediglich die Längen der letzten Kanten ändern sich aufgrund der Drehung der Sichtbarkeitsgeraden. Um ein lokales Minimum innerhalb einer Region zu bestimmen, muss also nur das Minimum der gemeinsamen Distanzfunktion für die Längen der letzten Kanten gefunden werden. Durch einen Vergleich aller lokaler Lösungen können dann die globalen Lösungen (oder die globale Lösung, wenn es nur eine gibt) bestimmt werden.

Abbildung 1.2 zeigt die globalen Lösungen für das gegebene Sichtbarkeitsproblem. In der min-sum Variante ist die Lösung das Punktepaar (s_{sum}, t_{sum}), bzw. die Pfade von s nach s_{sum} und von t über v_3 nach t_{sum}. Die Summe der Länge dieser Pfade ist minimal. In der min-max Variante ist die Lösung das Punktepaar (s_{max}, t_{max}), bzw. die Pfade von s nach s_{max} und von t nach t_{max}. Dort sind die Längen der beiden Pfade gleich. Bei einer Änderung der Sichtbarkeitsgeraden würde die Länge eines Pfades zunehmen, so dass das Minimum der maximalen Länge beider Pfade genau dort eintritt, wo ihre Längen gleich sind.

1.4 Ergebnisse der Arbeit

Ein Ergebnis dieser Masterarbeit ist ein in C++ geschriebenes Programm, das den Algorithmus zur Lösung des paarweisen Sichtbarkeitsproblems implementiert und es ermöglicht, das Problem und seine Lösung zu visualisieren (siehe Abbildung 1.3 für ein Beispiel). Meine Implementierung hängt natürlich eng mit der theoretischen Begründung zusammen. Insbesondere habe ich die für die Implementierung notwendigen Beweise in Kapitel 2 konstruktiv geführt, so dass ich bei der Implementierung einfach den dort beschriebenen Schritten folgen konnte.

Ich habe das Programm sowohl auf dem Mac als auch mit Ubuntu kompiliert und getestet. Eine Version für andere Betriebssysteme (z.B. für Windows) zu erstellen, sollte ohne große Probleme möglich sein, wenn die von mir verwendeten Bibliotheken und Frameworks eingebunden werden.

Eine Implementierung des Algorithmus ist eine programmiertechnische Herausforderung und allein schon deshalb interessant. Sie trägt aber auch dazu bei, Probleme und kleinere Ungenauigkeiten in der Formulierung des Algorithmus aufzudecken, die ohne eine Implementierung nicht auffallen würden.

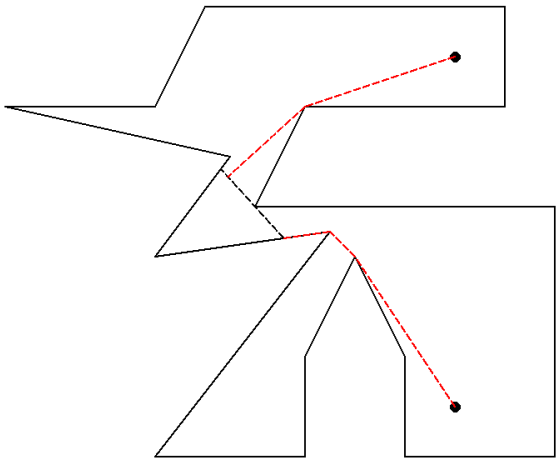

Abbildung 1.3: Visualisierung eines Polygons und der Lösung eines min-max Problems durch das von mir erstellte Programm.

Ich bin bei der Implementierung auf zwei solcher Probleme bzw. Ungenauigkeiten gestoßen. Erstens werden *degenerierte Biegungsereignisse* von Ahn et al. nicht erwähnt. Sie sind für die Berechnung der Minima aber von Bedeutung, weil sich bei ihnen die zu verwendende Distanzfunktion ändert. Ich berücksichtige *degenerierte Biegungsereignisse* sowohl bei der theoretischen Erläuterung des Algorithmus als auch bei seiner Implementierung.

Ein zweites Problem bzw. eine zweite Ungenauigkeit ist mir bei der Berechnung des lokalen *min-sum* Minimums zwischen zwei Ereignissen aufgefallen. Ahn et al. sagen dazu [1, Seite 8]:

> *We can find a minimum in constant time using elementary analysis.*

Damit ist gemeint, dass man die erste Ableitung der Distanzfunktion berechnet und Nullstellen derselben findet. Beim Versuch, dies in C++ zu implementieren, bin ich auf folgende Erkenntnisse gestoßen:

1. Falls keiner der beiden Lotfußpunkte auf der Sichtbarkeitsgeraden von einer Polygonkante versperrt wird, ist es nicht nötig, Ableitungen der Distanzfunktion zu berechnen. Das lokale Minimum liegt in diesem Fall immer bei einem Pfad-, Rand- oder Biegungsereignis.

2. Falls einer oder beide Lotfußpunkte auf der Sichtbarkeitsgeraden von Polygonkanten versperrt werden, fällt das Minimum nicht notwendigerweise mit einem Pfad-, Rand- oder Biegungsereignis zusammen. In diesem Fall ist die erste Ableitung der Distanzfunktion jedoch so kompliziert, dass sie bei einer praktischen Implementierung des Algorithmus nicht genutzt werden kann. Statt dessen ist es notwendig, das lokale Minimum approximativ zu berechnen.

Darüber hinaus behandle ich in dieser Arbeit die Berechnung des *min-max* Minimums, die von Ahn et al. in [1] nur knapp erwähnt wird, so ausführlich, dass eine praktische Implementierung möglich ist. Auch bei der *min-max* Variante des paarweisen Sichtbarkeitsproblems ist die exakte Bestimmung einer Lösung einfach, wenn die Lotfußpunkte auf der Sichtbarkeitsgeraden nicht von Polygonkanten versperrt werden. In diesem Fall muss eine quadratische Gleichung gelöst werden. Andernfalls ist die Berechnung einer exakten Lösung sehr kompliziert, so dass ich die Lösung in der praktischen Implementierung approximativ bestimme.

1.5 Aufbau der Arbeit

Die Masterarbeit gliedert sich entsprechend der Aufgabenstellung in zwei Teile: im ersten Teil (Kapitel 2) wird der Algorithmus zur Berechnung der minimalen Pfade zur paarweisen Sichtbarkeit erläutert. Das Hauptergebnis dieses Teils besteht in dem Nachweis, dass das paarweise Sichtbarkeitsproblem in Zeitkomplexität $\mathcal{O}(n)$ lösbar ist. Im zweiten Teil (Kapitel 3) gehe ich auf die wichtigsten programmiertechnischen Aspekte meiner Implementierung ein. Den Abschluss der Arbeit bildet eine Zusammenfassung und ein Ausblick, welche weiteren Fragestellungen bezüglich des paarweisen Sichtbarkeitsproblems von Interesse sein könnten.

Kapitel 2

Theoretische Aspekte des Algorithmus

2.1 Vorbemerkungen

Zur Erläuterung des Sichtbarkeitsproblems und seiner Lösung werden folgende Definitionen benötigt:

- P ist ein einfaches Polygon mit n Knoten.
- ∂P ist der Rand von P, wobei $\partial P \subset P$.
- $\pi(s,t)$ ist der kürzeste Pfad zwischen zwei Punkten $s,t \in P$, der innerhalb von P verläuft.
- die Knoten von $\pi(s,t)$ ohne $\{s,t\}$ werden als *innere Knoten* von $\pi(s,t)$ bezeichnet. Alle inneren Knoten eines kürzesten Pfads sind Knoten von P [15]
- die Länge eines kürzesten Pfads ist $|\pi(s,t)|$.
- der kürzeste Pfad zwischen einem Punkt $q \in P$ und einer Strecke $l \subset P$ wird ebenso als $\pi(q,l)$ bezeichnet, seine Länge ist $|\pi(q,l)|$.
- $\overline{qr}$ ist die Strecke mit den Endpunkten q und r
- zwei Punkte $q,r \in P$ sind gegenseitig sichtbar, wenn $\overline{qr} \subset P$.

- eine Strecke g tangiert einen Pfad π an einem Knoten v wenn $v \in g \cap \pi$ und sich die Nachbarknoten von v in π in einer geschlossenen Halbebene befinden, die von der Geraden, die g enthält, begrenzt wird (siehe Abbildung 2.1).

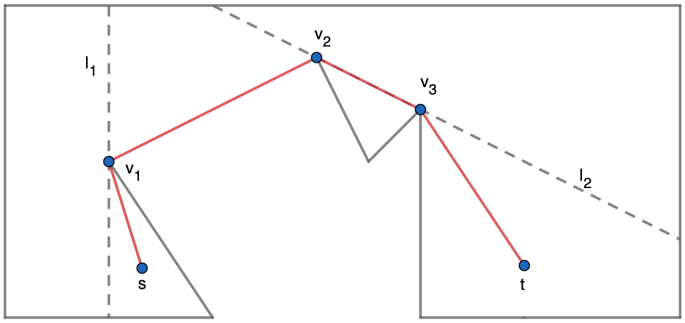

Abbildung 2.1: l_1 tangiert v_1, l_2 tangiert sowohl v_2 als auch v_3

- $\langle v_0, v_1, ..., v_k - 1, v_k \rangle$ bezeichnet die Knoten auf $\pi(s,t)$ mit $s = v_0$ und $t = v_k$

- soweit nicht anders definiert, bezeichnet p einen beliebigen Punkt auf ∂P. p kann dabei ein Knoten sein oder zwischen zwei Knoten liegen.

Das Ziel des Algorithmus besteht darin, für zwei Ausgangspunkte s und t Punktepaare (s^*, t^*) zu finden, so dass s^* und t^* gegenseitig sichtbar sind und $\pi(s, s^*)$ sowie $\pi(t, t^*)$ minimiert werden. In der *min-sum* Variante des Algorithmus ist die Summe $|\pi(s, s^*)| + |\pi(t, t^*)|$ zu minimieren, während in der *min-max* Variante das Maximum der Distanzen $max(|\pi(s, s^*)|, |\pi(t, t^*)|)$ minimiert werden soll. Bei beiden Varianten ist es möglich, dass mehr als eine Lösung existiert, siehe Abbildung 2.2. Wenn $\pi(s,t)$ nur 2 Knoten enthält, sind s und t gegenseitig sichtbar, und das paarweise Sichtbarkeitsproblem ist trivial. Daher wird im folgenden angenommen, dass $\pi(s,t)$ mindestens 3 Knoten enthält.

Bei der Lösung des Sichtbarkeitsproblems geht es also darum, kürzeste Pfade von zwei Anfangspunkten zu anderen Punkten in P zu finden. Zum Finden dieser Pfade können der Shortest Path Tree (SPT_s) und die Shortest Path Map (SPM_s) eines Anfangspunkts s verwendet werden [13]. SPT_s und

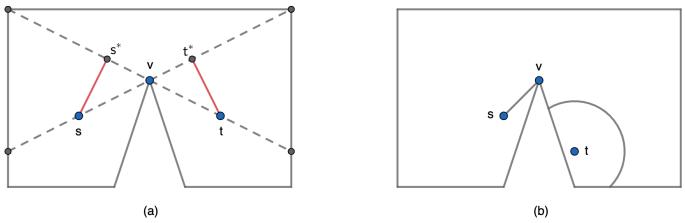

Abbildung 2.2: In Abbildung (a) sind sowohl das Punktepaar (s^*, t) als auch (s, t^*) Lösungen des min-sum Problems. In Abbildung (b) sind alle Punktepaare (v, t^*) mit $|\overline{tt^*}| \leq |\overline{sv}|$ und $t^* \in P$ Lösungen des min-max Problems.

SPM_s werden bei der Erläuterung des Algorithmus immer wieder benötigt und sind folgendermaßen definiert:

Definition 1. *Gegeben sei ein Punkt $s \in P$. p_i seien die Knoten von P. Dann ist der Shortest Path Tree von s: $SPT_s = \cup\pi(s, p_i)$. SPT_s enthält also die kürzesten Pfade von s zu jedem Knoten von P.*

Die Shortest Path Map von s (SPM_s) ist eine Erweiterung von SPT_s. Sie teilt P in $\mathcal{O}(n)$ dreieckige Zellen mit paarweise disjunktem Inneren auf, so dass der kürzeste Pfad von s zu jedem Punkt q innerhalb einer Zelle auf denselben Knoten von P verläuft. Der Vorgänger von q auf $\pi(s, q)$ ist ein Knoten p' von P. Die Zelle, in der sich die Punkte q befinden, die von p' aus sichtbar sind, wird im folgenden als Sichtbarkeitszelle von p' bezeichnet und als $S_{p'}$ abgekürzt. p' wird als Scheitelpunkt dieser Zelle bezeichnet. Ein Knoten $p' \in P$ kann der Scheitelpunkt mehrerer Sichtbarkeitszellen sein.

SPT_s kann aus einer Triangulation von P in $\mathcal{O}(n)$ Zeit berechnet werden und SPM_s kann in Zeit $\mathcal{O}(n)$ aus SPT_s berechnet werden, indem jede Kante von SPT_s erweitert wird, bis sie ∂P erreicht [13]. Für ein Beispiel von SPT_s und SPM_s siehe Abbildung 2.3.

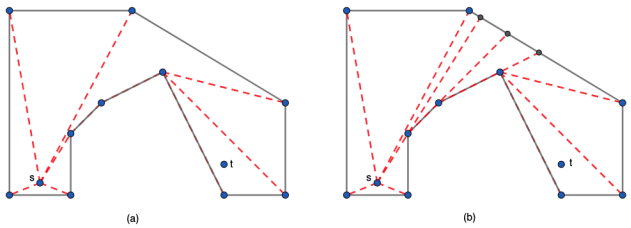

Abbildung 2.3: Abbildung (a) zeigt den Shortest Path Tree von s, Abbildung (b) die Shortest Path Map von s.

2.2 Allgemeine Eigenschaften einer Lösung des Sichtbarkeitsproblems

Im folgenden werden einige notwendige, aber nicht hinreichende Eigenschaften einer Lösung des Sichtbarkeitsproblems erläutert und bewiesen. Dabei sei das Punktepaar (s^*, t^*) zunächst eine optimale Lösung.

Es ist - zumindest in der min-sum Variante des Problems[1] - offensichtlich, dass die Pfade $\pi(s, s^*)$ und $\pi(t, t^*)$ die minimalen Pfade von s nach s^* und von t nach t^* sind. Denn ein nicht minimaler Pfad könnte verkürzt werden, was der angenommenen Optimalität der Lösung widerspricht. Nach Definition von SPM_s gilt, dass der minimale Pfad $\pi(s, s^*)$ von s über Knoten von P bis zum Scheitelpunkt einer Sichtbarkeitszelle führt, von dem aus s^* sichtbar ist. s^* kann dabei mit dem Scheitelpunkt zusammenfallen, oder innerhalb der Sichtbarkeitszelle einschließlich ihres Randes liegen. Ebenso führt $\pi(t, t^*)$ über Knoten von P zum Scheitelpunkt einer Sichtbarkeitszelle von SPM_t, von dem aus t^* sichtbar ist. Die inneren Knoten von $\pi(s, s^*)$ und von $\pi(t, t^*)$ sind also auf jeden Fall Knoten von P.

Um weitere Eigenschaften einer optimalen Lösung beweisen zu können, werden die folgenden zwei Definitionen benötigt:

Definition 2. *Eine Strecke l ist eine* ***Sichtbarkeitsgerade*** *wenn (i) l eine maximale Strecke innerhalb von P ist und (ii) l den Pfad $\pi(s, t)$ an mindestens einem Knoten $v \in \pi(s, t)$ tangiert.*

[1] In der min-max Variante des Problems muss unter bestimmten Voraussetzungen nur ein Pfad minimal sein, der andere Pfad kann minimal sein, muss aber nicht. Wann genau dies der Fall ist, wird im folgenden erklärt.

Jede Sichtbarkeitsgerade l, die einen Knoten v tangiert, teilt das Polygon P in eine Reihe von Subpolygonen auf. P^- sei das Subpolygon, das s enthält und P^+ dasjenige, das t enthält. Dann kann l als die Vereinigung von zwei Strecken l^- und l^+ aufgefasst werden, so dass $l^- \cap l^+ = v$, $l^- \cap P^- \neq \emptyset$ und $l^+ \cap P^+ \neq \emptyset$ (siehe Abbilung 2.4).

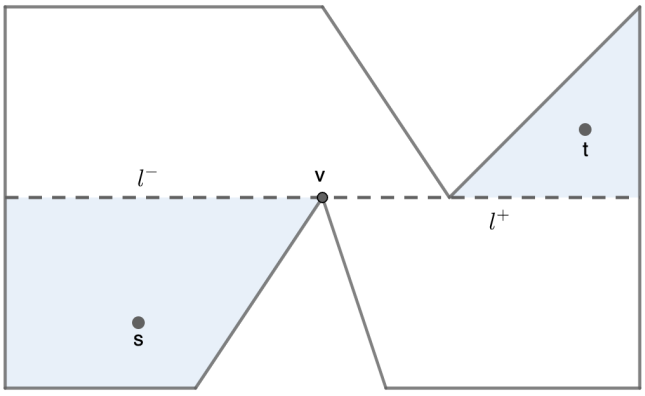

Abbildung 2.4: Einteilung von l in l^- und l^+

Definition 3. *g sei die Gerade, die eine Sichtbarkeitsgerade l enthält. Für den Punkt q gelte: $q \in P, q \notin g$. g_o sei eine auf g orthogonale Gerade, die q enthält. Dann ist der **Lotfußpunkt von q auf l** der Schnittpunkt von g und g_o. Er wird als f_q^l geschrieben.*

Die Strecke $\overline{qf_q^l}$ ist der minimale Weg zwischen q und g. Da eine Sichtbarkeitsgerade als Strecke innerhalb von P definiert wurde, ist zu beachten, dass f_q^l auch außerhalb des Polygons liegen kann.

Anhand dieser Definitionen können weitere Eigenschaften einer optimalen Lösung beschrieben und bewiesen werden. Das erste Lemma hierzu lautet:

Lemma 1. *Jede optimale Lösung des Sichtbarkeitsproblems (s^*, t^*) liegt auf einer Sichtbarkeitsgeraden.*

Beweis. Falls $s = s^*$ oder $t = t^*$, gilt das Lemma auf jeden Fall, da s und t Endknoten von $\pi(s, t)$ sind, d.h. s^* und t^* liegen auf einer Strecke, die s oder t tangiert. Daher wird im folgenden angenommen, dass dies nicht der Fall ist.

Ohne Einschränkung der Allgemeinheit sei $\overline{s^*t^*}$ horizontal, mit s^* links von t^*. l sei die maximale in P enthaltene Strecke, die $\overline{s^*t^*}$ enthält. s' sei derjenige Knoten von P, der vor s^* auf $\pi(s, s^*)$ liegt, und t' derjenige Knoten von P, der vor t^* auf $\pi(t, t^*)$ liegt. Dann liegen s' und t' entweder auf derselben Seite oder auf unterschiedlichen Seiten von l.

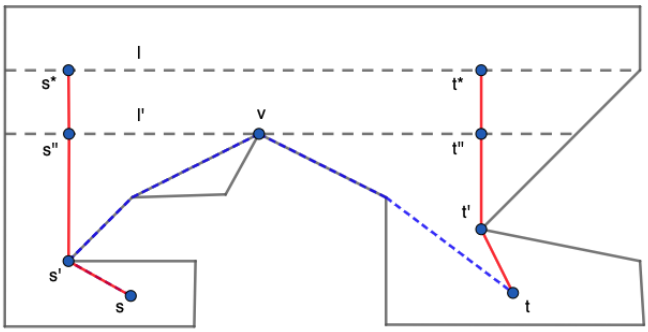

Abbildung 2.5: Wenn s' und t' unter l liegen, muss die Strecke zwischen einem optimalen Punktepaar v tangieren.

Zunächst betrachte ich den Fall, dass s' und t' unterhalb von l liegen. Angenommen l tangiere keinen Knoten von P. Dann können die Distanzen $\overline{s's^*}$ sowie $\overline{t't^*}$ durch eine Parallelverschiebung von l nach unten verkürzt werden (siehe Abbildung 2.5). Das ist aber ein Widerspruch zur Optimalität von (s^*, t^*), so dass l einen Knoten v von P tangieren muss. Da s unterhalb von l^- liegt und t unterhalb von l^+, muss der Pfad $\pi(s, t)$ die Strecke l schneiden. Angenommen der Schnittpunkt mit l^- sei y, und der mit l^+ y'. Dann kann $|\pi(s, t)|$ verkürzt werden, wenn der direkte Weg $\overline{yy'}$ gewählt wird. Das widerspricht aber der Optimalität von $\pi(s, t)$. Daraus folgt, dass $\pi(s, t)$ l nur in v schneidet und $v \in \pi(s, t)$. Der Fall, dass s' und t' oberhalb von l liegen, kann analog bewiesen werden.

Somit bleibt noch der Fall, dass s' und t' auf unterschiedlichen Seiten von l liegen. Ohne Einschränkung der Allgemeinheit kann angenommen werden, dass s' unterhalb und t' oberhalb von l liegt. Wenn $\overline{s^*t^*}$ keinen Knoten von P tangiert, kann l an einem Punkt auf $\overline{s^*t^*}$ in Richtung s' und t' gedreht werden, bis l einen Knoten v von P tangiert. An diesem Knoten kann l weiter gedreht werden, bis ein weiterer Knoten v' von P tangiert wird. Die Schnittpunkte von l mit $\overline{s's^*}$ bzw. $\overline{t't^*}$ während dieser Drehungen seien s'' und

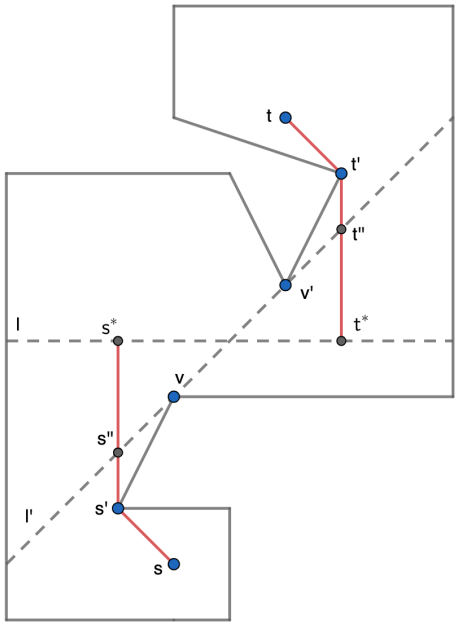

Abbildung 2.6: s' und t' liegen auf unterschiedlichen Seiten von l

t''. Für diese gilt: $\overline{s''t''} \subset P$ und $|\pi(s', s'')| + |\pi(t', t'')| < |\pi(s', s^*)| + |\pi(t', t^*)|$, was der Annahme der Optimalität von s^* und t^* in der min-sum Variante des Sichtbarkeitsproblems widerspricht (siehe Abbildung 2.6). Also muss das Segment $\overline{s^*t^*}$ (wenigstens in der in der min-sum Variante) zwei Knoten von P tangieren.

Nun sei angenommen, dass $\overline{s^*t^*}$ zwei Knoten von P (v und v') tangiert. Desweiteren sei angenommen, dass l keine Kante von $\pi(s, t)$ enthält. Unter diesen Annahmen muss $\pi(s, t)$ l dreimal schneiden: bei einem Punkt y, der auf der Strecke zwischen dem Polygonrand, s^* und v liegt, bei einem Punkt $u \in \overline{vv'}$ und bei einem Punkt y', der auf der Strecke zwischen v', t^* und dem Polygonrand liegt. Dann kann $\pi(s, t)$ verkürzt werden, indem der direkte Weg zwischen y und y' gewählt wird, was der Optimalität von $\pi(s, t)$ widerspricht. Also muss die Kante $\overline{vv'}$ eine Kante von $\pi(s, t)$ sein, so dass $\overline{s^*t^*}$ diese Kante enthält und $\pi(s, t)$ in den Punkten $v, v' \in \pi(s, t)$ tangiert.

Der Vollständigkeit halber ist noch zu erwähnen, dass es in der min-max Variante des Sichtbarkeitsproblems zu einem Sonderfall kommen kann, wenn

s' und t' auf unterschiedlichen Seiten von l liegen und einer der optimalen Endpunkte mit v oder v' zusammenfällt. Angenommen, t^* fällt mit v' zusammen. Dann ist es möglich, dass es Lösungen des min-max Problems bei Sichtbarkeitsgeraden gibt, die v nicht tangieren und bei denen durch eine weitere Drehung der Sichtbarkeitsgeraden in Richtung v die Distanz zu s verringert werden könnte. Abbildung 2.7 illustriert diesen Sonderfall: sowohl l' und das Punktepaar (s^*, v') als auch l und das Punktepaar (v, v') sind Lösungen des min-max Problems, wobei $\overline{ss^*}$ länger ist als $\overline{sv}$. Für das min-sum Problem ist hingegen das Punktepaar (v, v') die eindeutig bestimmte Lösung.

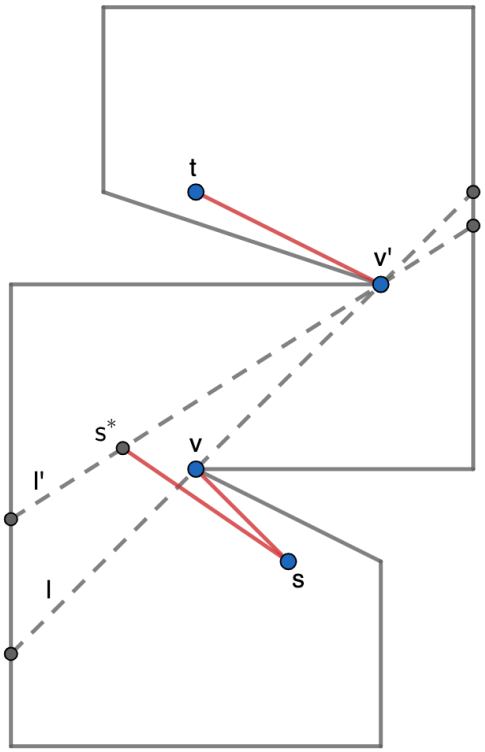

Abbildung 2.7: s und t liegen auf unterschiedlichen Seiten von l, aber nicht jede Sichtbarkeitsgerade, die eine Lösung des min-max Problems enthält, muss sowohl v als auch v′ tangieren

Bei min-max Problemen tangiert die optimale Sichtbarkeitsgerade also

nicht notwendigerweise v und v', wenn s' und t' auf unterschiedlichen Seiten von l liegen. Zumindest ein Knoten von $\pi(s,t)$ wird von einer optimalen Sichtbarkeitsgeraden aber auf jeden Fall tangiert. Denn ansonsten könnte entweder die minimale Distanz bei einem optimalen Punktepaar oder $|\pi(s,t)|$ verringert werden, was entsprechend der bisherigen Argumentation ein Widerspruch ist. Daher gilt das Lemma auch in diesem Sonderfall des min-max Problems.[2]

□

Das folgende Lemma spezifiziert die Lage von s^* und t^* auf der Sichtbarkeitsgeraden noch weiter:

Lemma 2. *Angenommen, die Punkte s^* und t^* liegen auf der Sichtbarkeitsgeraden l. Dann gilt:*

(i) $s^* \in l^-$

(ii) $t^* \in l^+$

Beweis. v sei der von l tangierte Knoten von $\pi(s,t)$. Per Definition teilt l^- das Polygon in Subpolygone auf, von denen eines s enthält. Jeder Pfad von s zu einem Punkt auf $l^+ \setminus v$ muß daher l^- (möglicherweise in v) schneiden. Dieser Schnittpunkt auf l^- ist von s weniger weit entfernt als jeder Punkt auf $l^+ \setminus v$, so dass der Endpunkt von $\pi(s,l)$ auf l^- liegen muss. Durch eine entsprechende Argumentation kann bewiesen werden, dass $\pi(t,l)$ bei einem Punkt auf l^+ endet.

Falls l zwei Knoten v und v' von $\pi(s,t)$ tangiert, kann obige Argumentation auf beide Knoten angewendet werden. v sei der Knoten, der näher bei s liegt, und v' der Knoten, der näher bei t liegt. Dann folgt daraus, dass s^* auf $l^- \setminus \overline{vv'}$ liegt, und t^* auf $l^+ \setminus \overline{vv'}$. □

Die Lage von s^* auf l^- und t^* auf l^+ kann noch weiter eingegrenzt werden. Dafür wird folgendes Hilfslemma benötigt:

[2]Im folgenden wird zunächst angenommen, dass die Lösung der min-max Variante beide Pfade minimiert und damit eindeutig bestimmt ist. In Kapitel 2.5 wird erklärt, wie entschieden werden kann, ob es mehr als eine Lösung gibt und welche Eigenschaften die Menge dieser Lösungen hat.

Lemma 3. *g sei eine Gerade, q ein beliebiger Punkt, der nicht auf g liegt, und f_q^g der Lotfußpunkt von q auf g. Der kürzeste Pfad von q nach g ist die Strecke $\overline{qf_q^g}$. Je weiter ein Punkt $p \in g$ von f_q^g entfernt ist, umso größer wird sein Abstand zu q.*

Beweis. p sei ein beliebiger Punkt auf g. γ sei der Winkel $\angle f_q^g qp$: Dann gilt das Lemma wegen des folgenden trigonometrischen Zusammenhangs:

$$|\overline{qp}| = \frac{|\overline{qf_q^g}|}{\cos(\gamma)} \tag{2.1}$$

Je weiter sich p von f_q^g entfernt, umso mehr nähert sich γ 90° an. Da $|\overline{qf_q^g}|$ konstant bleibt, wird der Abstand $|\overline{qp}|$ dabei immer größer. □

Aufgrund von Lemma 2 liegt s^* auf l^-. Man beachte, dass l^- mehrere Zellen von SPM_s durchqueren kann, siehe Abbildung 2.8. Das folgende Lemma bestimmt sowohl die Sichtbarkeitszelle von SPM_s, in der s^* liegt, als auch die Lage von s^* innerhalb dieser Zelle.

Lemma 4. *Der optimale Punkt s^* liege auf der Sichtbarkeitsgeraden l, die den Knoten $v \in \pi(s,t)$ tangiere. Dann liegt s^* in einer Sichtbarkeitszelle von SPM_s mit Scheitelpunkt p^*, so dass eine der folgenden Bedingungen erfüllt ist:*

(i) $p^* = s^*$

(ii) $s^* = f_{p^*}^{l^-}$, *d.h.* $\overline{p^*s^*}$ *steht orthogonal auf* l^-, *oder*

(iii) $s^* \in \partial P$ *und kein Punkt in* S_{p^*} *ist näher an* $f_{p^*}^{l^-}$ *als* s^*.

Beweis. Für den Beweis betrachtet man als erstes diejenige Sichtbarkeitszelle von SPM_s, die von l^- durchquert wird und deren Scheitelpunkt näher an s liegt als die Scheitelpunkte aller anderen Sichtbarkeitszellen, die l^- durchquert. In Abbildung 2.8 ist das die Sichtbarkeitszelle S_{p_1}. Falls es mehrere solcher Sichtbarkeitszellen mit demselben Scheitelpunkt gibt, dann kann eine von ihnen gewählt werden. In Abbildung 2.9 ist dies der Fall, und es kann eine der Sichtbarkeitszellen mit Scheitelpunkt p_1 gewählt werden.

Falls der Scheitelpunkt p der gewählten Sichtbarkeitszelle auf l^- liegt, dann ist $\pi(s,p)$ der kürzeste Pfad von s zu l^- und das Lemma ist bewiesen.

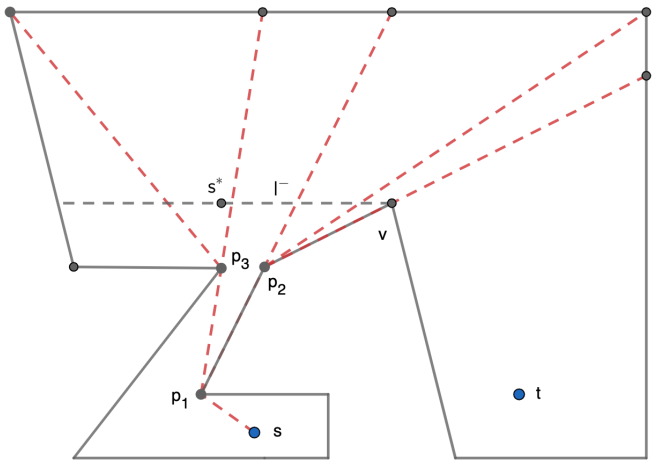

Abbildung 2.8: l^- durchquert mehrere Sichtbarkeitszellen von SPM_s

Im zweiten Fall, also wenn $p \notin l^-$, muß eine Fallunterscheidung bezüglich des Lotfußpunkts $f_p^{l^-}$ getroffen werden: wenn $f_p^{l^-} \in S_p$, dann ist $\overline{pf_p^{l^-}}$ nach Lemma 3 der kürzeste Pfad zwischen p und l^-. Da p am nächsten zu s liegt, besteht $\pi(s, l^-)$ aus $\pi(s, p)$ und $\overline{pf_p^{l^-}}$ und das Lemma ist bewiesen (siehe auch Abbildung 2.8.

Wenn $f_p^{l^-} \notin S_p$, dann betrachtet man die beiden Schnittpunkte von l^- mit dem Rand von S_p. i^* sei derjenige Schnittpunkt, der näher an $f_p^{l^-}$ liegt. Dann ist der minimale Pfad von p zu l^- in S_p aufgrund von Lemma 3 die Strecke $\overline{pi^*}$. Wenn $i^* \in \partial P$ und keine weitere Sichtbarkeitszelle i^* enthält, dann kann die Distanz $|\pi(p, l^-)|$ nicht weiter verkürzt werden. Somit besteht $\pi(s, l^-)$ aus $\pi(s, p)$ und $\overline{pi^*}$ und das Lemma ist bewiesen (siehe Abbildung 2.9).[3]

Andernfalls liegt i^* auf dem Rand einer weiteren Sichtbarkeitszelle von SPM_s mit Scheitelpunkt p'. In diesem Fall wendet man die bisherige Argumentation auf die Sichtbarkeitszelle $S_{p'}$ und $f_{p'}^{l^-}$ an. Dies kann solange wiederholt werden, bis l^- den Scheitelpunkt einer Sichtbarkeitszelle tangiert, der Lotfußpunkt in einer Sichtbarkeitszelle liegt oder der optimale Schnittpunkt Teil des Polygonrandes ist. Die Sichtbarkeitszelle, für die das der Fall ist, habe den Scheitelpunkt p^*. Dann ist $\pi(s, l^-)$ entweder gleich $\pi(s, p^*)$,

[3]Man beachte, dass $v \in \partial P$ und $v \in l^-$. Wenn $i^* = v$, dann liegt der Endpunkt von $\pi(s, l^-)$ bei $v \in l^-$ und Lemma 2 gilt weiterhin.

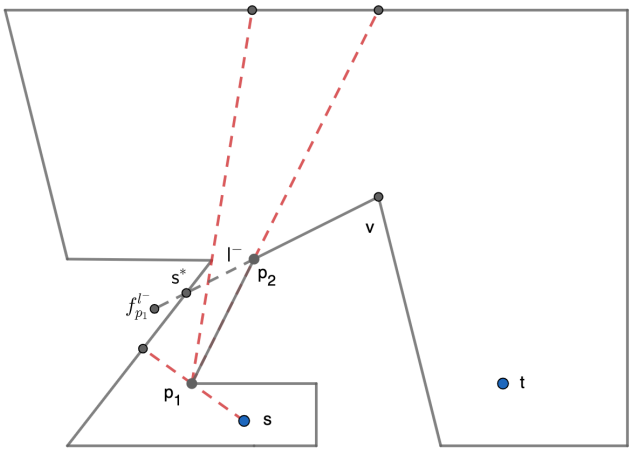

Abbildung 2.9: s^* ist der Schnittpunkt von l^- und der Polygonkante; s^* und $f^{l^-}_{p_1}$ fallen nicht zusammen.

oder besteht aus $\pi(s, p^*)$ und der Strecke $\overline{p^* f^{l^-}_{p^*}}$, oder aus $\pi(s, p^*)$ und der Strecke von p^* zu demjenigen Schnittpunkt von l^- mit ∂P, der am nächsten zu $f^{l^-}_{p^*}$ liegt.

□

Durch Vertauschen von s und t gilt Lemma 4 auch für die optimalen Punkte t^*.

2.3 Die Ereignisse des Sweep-Algorithmus

Wie schon erwähnt, ähnelt der Algorithmus zur Lösung des paarweisen Sichtbarkeitsproblems einem klassischen Sweep-Algorithmus, mit dem Unterschied, dass die Sweep-Gerade nicht über die Ebene hinwegfegt, sondern dass eine Sichtbarkeitsgerade an allen inneren Knoten $\langle v_1, ..., v_k - 1\rangle$ von $\pi(s, t)$ gedreht wird. Der Algorithmus wird mit einer Sichtbarkeitsgeraden, die s und v_1 enthält, initialisiert. Diese Sichtbarkeitsgerade wird an v_1 gedreht (während sie v_1 tangiert), bis sie auf v_2 trifft. Dann wird die Sichtbarkeitsgerade an v_2 gedreht usw., solange bis die Sichtbarkeitsgerade, die an v_{k-1} gedreht wird, auf t trifft. Während dieser Drehungen werden aufgrund von Lemma 1 alle

Sichtbarkeitsgeraden angetroffen, die für eine Lösung des Sichtbarkeitsproblems in Frage kommen.

Wie bei einem klassischen Sweep-Algorithmus müssen die Ereignisse berechnet werden, die eine besondere Relevanz für das Finden der Lösung haben. s^* und t^* seien jetzt und im folgenden die optimalen Endknoten der Pfade zu einer gegebenen Sichtbarkeitsgeraden. Dann treten relevante Ereignisse ein, wenn sich der Drehpunkt der Sichtbarkeitsgeraden ändert, wenn sich die Polygonkante ändert, die von der sich drehenden Sichtbarkeitsgeraden überstreift wird, wenn die Pfade $\pi(s, s^*)$ oder $\pi(t, t^*)$ einen inneren Knoten verlieren oder dazugewinnen, oder wenn sich die Distanzfunktion ändert, die bei der Berechnung der Distanz zwischen s^* bzw. t^* und seinem Vorgänger zu verwenden ist.

Somit können die folgenden Ereignisse unterschieden werden:

1. **Pfadereignisse** sind Sichtbarkeitsgeraden, die zwei aufeinanderfolgende Knoten von $\pi(s, t)$ enthalten. Bei diesen Ereignissen ändert sich der Knoten, an dem die Sichtbarkeitsgerade gedreht wird.

2. **Randereignisse** sind Sichtbarkeitsgeraden, die mindestens einen Knoten p von $P \setminus \pi(s, t)$ enthalten. Bei diesen Ereignissen ändert sich die Kante auf ∂P, die von der Sichtbarkeitsgeraden überstreift wird.

3. **Biegungsereignisse** sind Sichtbarkeitsgeraden, bei denen dem kürzesten Pfad $\pi(s, l)$ oder $\pi(t, l)$ ein Knoten $p \in P$ hinzugefügt oder weggenommen wird.

4. **Degenerierte Biegungsereignisse** sind Sichtbarkeitsgeraden, bei denen sich die zu verwendende Funktion zur Berechnung der Distanz zwischen s^* bzw. t^* und seinem Vorgänger ändert.

Diese Ereignisse werden in einer doppelt verketteten Liste gespeichert, und zwar in der Reihenfolge, in der sie von der sich drehenden Sichtbarkeitsgeraden angetroffen werden. Diese Liste wird im folgenden als *Ereignisliste* bezeichnet. Jedes Element der Liste enthält nach Ende der Berechnungen die folgenden Daten:

- den Drehpunkt der Sichtbarkeitsgeraden
- die Endpunkte des Ereignisses auf dem Polygonrand

- die Knoten von P, aus denen die kürzesten Pfade von s und t zur Sichtbarkeitsgeraden beim Ereignis bestehen
- Informationen darüber, welche Distanzfunktionen während der Drehung der Sichtbarkeitsgeraden bis zum nächsten Ereignis benutzt werden müssen

Wenn alle Ereignisse in der Liste gespeichert sind, kann zwischen jeweils zwei Ereignissen das lokale Minimum berechnet werden. Durch Vergleich aller lokaler Minima kann das globale Minimum bzw. die globalen Minima bestimmt werden.

Die folgenden Beweise zur Berechnung der Pfad-, Rand- und Biegungsereignisse sind konstruktiv. Meine Implementierung des Algorithmus in C++ folgt den hier beschriebenen Schritten. Allerdings benutze ich dabei keine komplizierten Datenstrukturen wie z.B. Finger-Trees, die Guibas et al. bei der linearen Konstruktion der Shortest Path Maps benötigen [13], so dass die Zeitkomplexität des von mir implementierten Algorithmus schlechter als linear ist.

2.3.1 Berechnung der Pfad- und Randereignisse

Im folgenden habe der kürzeste Pfad $\pi(s,t)$ k Knoten. Dann gilt für die Berechnung der Pfad- und Randereignisse das folgende Lemma:

Lemma 5. *Für ein einfaches Polygon P mit n Knoten und Punkten $s,t \in P$ können alle Pfad- und Randereignisse des paarweisen Sichtbarkeitsproblems in der Reihenfolge, in der sie von der sich drehenden Sichtbarkeitsgeraden angetroffen werden, in Zeit $\mathcal{O}(n)$ berechnet und in die Ereignisliste eingefügt werden.*

Beweis. Für die Berechnung der Pfad- und Randereignisse ist folgende Beobachtung entscheidend: wenn eine Sichtbarkeitsgerade l an einem Knoten $v_i \in \pi(s,t)$ mit $0 < i < k$ gedreht wird, überstreift l^+ Sichtbarkeitszellen von SPM_s mit Scheitelpunkt v_i. l^- hingegen überstreift Sichtbarkeitszellen von SPM_t mit Scheitelpunkt v_i. Die Kanten dieser Sichtbarkeitszellen definieren dabei jeweils eine Hälfte eines Pfad- bzw. Randereignisses.

Um diese Beobachtung zu nutzen, wird die Ereignisliste zu Beginn des Algorithmus als eine leere, doppelt verkettete Liste initialisiert. Dann wird $\pi(s,t)$ anhand des in [15] beschriebenen Algorithmus berechnet. Die Knoten

von $\pi(s,t)$ werden dabei als spezielle Datenstrukturen erzeugt, die es erlauben, Kanten von Sichtbarkeitszellen in Listen einzufügen. Außerdem werden die Knoten von $\pi(s,t)$ in P speziell markiert, so dass man im folgenden immer weiß, ob es sich bei einem Knoten von P um einen Knoten des kürzesten Pfads handelt oder nicht.

Anschließend wird anhand des Funnel- oder Trichter-Algorithmus wie in [13] beschrieben SPM_s aufgebaut. Wenn man während dieses Aufbaus Sichtbarkeitszellen konstruiert, die einen Knoten $v_i \in \pi(s,t)$ als Scheitelpunkt haben, dann speichert man die Kanten dieser Sichtbarkeitszellen bei v_i in der richtigen Reihenfolge ab. Danach baut man SPM_t auf und speichert ebenfalls die Kanten von Sichtbarkeitszellen, die $v_i \in \pi(s,t)$ als Scheitelpunkt haben, bei v_i ab. Bei dieser Konstruktion ist zu beachten, in welche Richtung sich l dreht. Dies kann man aber einfach entscheiden, wenn man den Vorgänger und Nachfolger des aktuellen Drehpunkts betrachtet.

Nach dieser Vorbereitung enthält jeder Knoten $v_i \in \pi(s,t)$ mit $0 < i < k$ zwei Listen: in der einen Liste namens $list_i^+$ sind Kanten von SPM_s, die v_i als einen Endpunkt haben, gespeichert; in der anderen Liste namens $list_i^-$ sind Kanten von SPM_t gespeichert, die ebenfalls v_i als einen Endpunkt haben. Dies ist in Abbildung 2.10 illustriert. Man beachte, dass die erste Kante in $list_i^+$ und die letzte Kante in $list_i^-$ gemeinsam das Pfadereignis definieren, bei dem v_i der aktuelle Drehpunkt wird. Die letzte Kante in $list_i^+$ und die erste in $list_i^-$ definieren dasjenige Pfadereignis, bei dem der Nachfolger von v_i der neue Drehpunkt wird. Die anderen Kanten definieren jeweils eine Hälfte eines Randereignisses.

Das erste und letzte Pfadereignis, d.h. die Pfadereignisse, die s bzw. t enthalten, müssen besonders behandelt werden. Denn SPM_s enthält keine Kante, die durch t geht, und SPM_t enthält keine Kante, die durch s geht. $list_1^-$ muss daher so aufbereitet werden, dass die letzte Kante durch v_1 und s geht, und $list_{k-1}^+$ so, dass die letzte Kante durch v_{k-1} und t geht.

Nach dem Aufbau von $list_i^+$ und $list_i^-$ geht man bei jedem Knoten $v_i \in \pi(s,t)$ mit $0 < i < k$ folgendermaßen vor: $list_i^+$ wird von vorne nach hinten durchgegangen und gleichzeitig $list_i^-$ von hinten nach vorne. Der erste Eintrag in $list_i^+$ und der letzte von $list_i^-$ definieren das erste Pfadereignis, das in die Ereignisliste eingefügt wird. In Abbildung 2.10 sind das die Kanten, die bei p_1^+ und p_4^- enden.

Nun muss entschieden werden, welche Kante in $list_i^+$ oder $list_i^-$ das nächste Randereignis definiert. Dazu muss man die Schnittpunkte der Kanten mit dem gegenüberliegenden Rand berechnen und vergleichen. In Abbildung 2.10

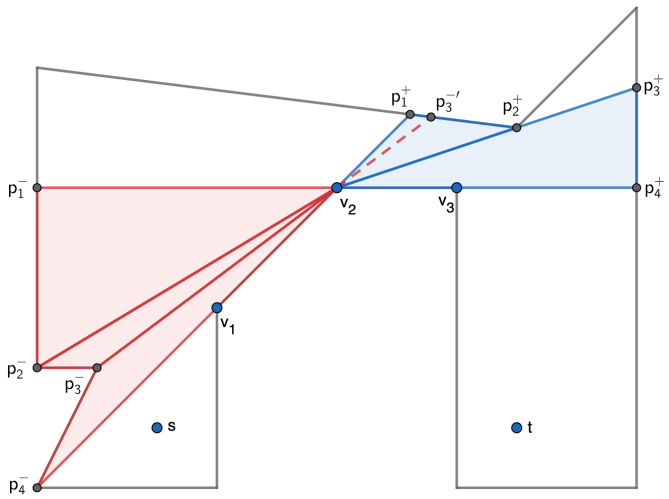

Abbildung 2.10: Kanten von SPM_s und SPM_t definieren eine Hälfte von Pfad- und Randereignissen.

etwa sind die Kanten mit Endpunkten p_3^- und p_2^+ bzw. p_3^+ Kandidaten für das nächste Randereignis. Der Punkt $p_3^{-\prime}$, der p_3^- gegenüberliegt, kann in konstanter Zeit berechnet werden, da die Kante $\overline{p_1^+ p_2^+}$ bekannt ist. Da $p_3^{-\prime}$ von p_1^+ weniger weit entfernt ist als p_2^+ ist die Sichtbarkeitsgerade zwischen p_3^-, v_2 und $p_3^{-\prime}$ das nächste Randereignis und kann in die Ereignisliste eingefügt werden. Nachdem diese Kante behandelt wurde, kann der Zeiger auf $list_i^-$ um eins nach vorne verschoben werden.

Auf diese Art geht man die Listen $list_i^+$ und $list_i^-$ durch, bis man zum Ende von $list_i^+$ und zum Anfang von $list_i^-$ kommt. Dort wird das nächste Pfadereignis in die Ereignisliste eingetragen und der Knoten v_{i+1} kann behandelt werden.

Folgender Sonderfall ist dabei zu beachten: es gibt Knoten von P, die so in das Polygon hereinragen, dass sie von einer Sichtbarkeitsgeraden tangiert werden, aber nicht der Endpunkt des Randereignisses auf dem Polygonrand sind. Das ist z.B. bei Punkt p_2^+ in Abbildung 2.10 der Fall. Diesen Sonderfall muss man bei der Definition der Randereignisse und bei der Suche nach den gegenüberliegenden Schnittpunkten berücksichtigen.

Um die Ereignisliste mit den Pfad- und Randereignissen aufzufüllen, benötigt man $\pi(s,t)$, SPM_s und SPM_t, die in $\mathcal{O}(n)$ aus dem triangulierten

Polygon aufgebaut werden können [15, 13]. Die Listen $list_i^+$ und $list_i^-$ haben insgesamt $\mathcal{O}(n)$ Einträge, die in insgesamt linearer Zeit in der korrekten Reihenfolge eingefügt werden können. Jeder dieser Einträge muss einmal besucht werden und für jeden Eintrag muss ein Ereignis in die Ereignisliste eingetragen werden. Dies geht pro Eintrag in konstanter Zeit. Daher hat der Aufbau der Ereignisliste mit den Pfad- und Randereignissen insgesamt eine lineare Zeitkomplexität. □

2.3.2 Berechnung der Biegungsereignisse

Vorbemerkungen

Nachdem die Pfad- und Randereignisse berechnet und in der Ereignisliste eingetragen sind, können die Biegungsereignisse behandelt werden. Da der kürzeste Pfad von $\pi(s, l)$ nach Lemma 2 auf l^- endet und $\pi(t, l)$ auf l^+, wird im folgenden der Einfachheit halber nur der Pfad $\pi(s, l^-)$ behandelt. Für $\pi(t, l^+)$ gelten alle Ergebnisse durch eine entsprechende Anpassung.

Im folgenden bezeichnet u den Vorgänger von s^* in $\pi(s, l^-)$. Da alle inneren Knoten von $\pi(s, l^-)$ Knoten von P sind, ist auch u ein Knoten von P. Der Vorgänger von u auf $\pi(s, l^-)$ wird als u^- bezeichnet. Falls ein Knoten $p \in P$ hinter u zu $\pi(s, l^-)$ hinzugefügt wird, wird dieser Knoten als u^+ bezeichnet.

Die folgenden Biegungsereignisse können unterschieden werden:

- Ein Knoten $p \in P$ wird zu $\pi(s, l^-)$ hinzugefügt. Solche Biegungsereignisse treten dann ein, wenn $\pi(s, l^-)$ beginnt, in einer Sichtbarkeitszelle von SPM_s zu verlaufen, die einen Scheitelpunkt hat, der bisher nicht Teil von $\pi(s, l^-)$ war.

- Der letzte Knoten u von $\pi(s, l^-)$ vor l^- wird entfernt. Solche Ereignisse treten ein, wenn $\pi(s, l^-)$ beginnt, in einer Sichtbarkeitszelle von SPM_s zu verlaufen, die den Vorgänger u^- von u in $\pi(s, l^-)$ als Scheitelpunkt hat. In diesem Fall wird l^- von u^- aus sichtbar und u kann von $\pi(s, l^-)$ entfernt werden.

- der optimale Punkt s^* auf l^- beginnt bzw. hört auf, mit dem Lotfußpunkt $f_u^{l^-}$ zusammenzufallen. Bei solchen Ereignissen wird dem Pfad zu l^- zwar kein Punkt hinzugefügt bzw. weggenommen, aber die Distanz zu l^- muß unterschiedlich berechnet werden, je nachdem ob s^* mit dem Lotfußpunkt zusammenfällt oder nicht. Diese Biegungsereignisse werden als *degenerierte Biegungsereignisse* bezeichnet.

Biegungsereignisse können zwischen zwei aufeinanderfolgenden Pfad- und Randereignissen stattfinden oder mit diesen zusammenfallen. Je zwei aufeinanderfolgende Pfad- bzw. Randereignisse in der Ereignisliste definieren somit die Regionen, die von l^- überstreift werden und die zur Berechnung der Biegungsereignisse betrachtet werden müssen.

Um alle Biegungsereignisse zu berechnen, beginnt man beim ersten Pfadereignis, d.h. bei der Sichtbarkeitsgeraden l, die durch s und v_1 definiert ist, und iteriert dann über alle Einträge in der Ereignisliste. Zu Beginn der Berechnungen wird der kürzeste Pfad zur Sichtbarkeitsgeraden $\pi(s, l^-)$ mit dem Punkt s als Startpunkt initialisiert. Wenn Knoten von P zu $\pi(s, l^-)$ hinzukommen oder verlorengehen, wird der Pfad aktualisiert und ein Biegungsereignis wird in die Ereignisliste eingefügt. Degenerierte Biegungsereignisse werden auch in die Ereignisliste eingefügt, damit die Berechnung der Distanzen von u zu l^- später korrekt durchgeführt werden kann. Unter der Annahme, dass s im Polygoninneren liegt, fällt der optimale Punkt s^* auf l^- bei Beginn der Drehung von l^- mit dem Lotfußpunkt $f_s^{l^-}$ zusammen.

Im folgenden untersuche ich zunächst, welche Änderungen an den Knoten von $\pi(s, l^-)$ durch ein Pfad- oder Randereignis ausgelöst werden können. Bei solchen Biegungsereignissen ist kein neuer Eintrag in die Ereignisliste nötig, sondern der schon bestehende Eintrag für das Pfad- bzw. Randereignis muss entsprechend angepasst werden. Dann werden die Biegungsereignisse besprochen, die nicht durch Pfad- oder Randereignisse ausgelöst werden. Derartige Biegungsereignisse liegen im Normalfall zwischen aufeinanderfolgenden Pfad- bzw. Randereignissen. Allerdings ist es auch möglich, dass sie mit einem Pfad- oder Randereignis zusammenfallen. Dieser Sonderfall wird im Beweis zu Lemma 11 in Schritt 4 berücksichtigt. Als letztes werden degenerierte Biegungsereignisse untersucht. Es folgt ein Lemma, das beweist, dass alle Biegungsereignisse in Zeitkomplexität $\mathcal{O}(n)$ berechnet und in die Ereignisliste eingefügt werden können.

Biegungsereignisse bei Pfad- und Randereignissen

Unter der Voraussetzung, dass der Lotfußpunkt $f_u^{l^-}$ und der bisherige Drehpunkt v bei einem Pfadereignis nicht zusammenfallen, gilt für Pfadereignisse das folgende Lemma:[4]

[4]Der Fall, dass $f_u^{l^-}$ und v beim Pfadereignis zusammenfallen, wird durch Lemma 8 abgedeckt und in Lemma 11 in Schritt 4 berücksichtigt.

Lemma 6. *Bei einem Pfadereignis ändern sich die inneren Knoten von $\pi(s, l^-)$ nur, wenn sich die Drehrichtung von l ändert. In diesem Fall ist der neue kürzeste Pfad zum Knoten vor l^- der kürzeste Pfad von s zum bisherigen Drehpunkt v. Die Überprüfung, ob sich die Drehrichtung von l ändert, kann in konstanter Zeit durchgeführt werden.*

Beweis. v sei der aktuelle Drehpunkt, v' der Nachfolger von v in $\pi(s, t)$ und l sei die Sichtbarkeitsgerade, die $\overline{vv'}$ enthält. Falls l an v' infinitesimal in die bisherige Richtung weitergedreht wird, dann ändert sich der Weg zu l^- nur infinitesimal. Daher bleibt u der letzte Punkt vor l^- (siehe Abbildung 2.11 (a)).

Wenn l hingegen in die entgegengesetzte Richtung gedreht wird, betrachte man die Kante zwischen den Knoten, die das Pfadereignis auslösen, also z.B. $\overline{vv'}$ in Abbildung 2.11 (b). Diese teilt das Polygon P in zwei Subpolygone mit disjunktem Inneren: P^t, das t enthält, und P^s, das s enthält. Die Zellen von SPM_s, die l^- nach Änderung der Drehrichtung überstreift, liegen alle in P^t. Daraus folgt, dass für jeden kürzesten Pfad $\pi(s, l^-)$ mit $l^- \subset P^t$ gilt: $v \in \pi(s, l^-)$. Da Subpfade von kürzesten Pfaden selbst kürzeste Pfade sind und $v \in \pi(s, t)$, ist der kürzeste Pfad von s nach v der Subpfad $\pi(s, v) \subset \pi(s, t)$. Somit gilt für alle $l^- \in P^t$: $\pi(s, v) \subset \pi(s, l^-)$.

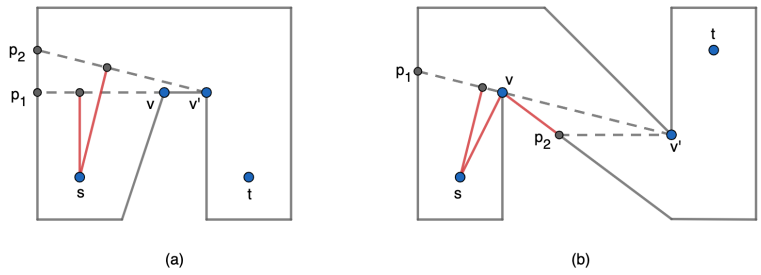

Abbildung 2.11: In Abbildung (a) bleibt s nach dem Pfadereignis der letzte Knoten vor l^-. In Abbildung (b) wird v beim Pfadereignis der letzte Knoten vor l^-.

Um zu überprüfen, ob sich die Drehrichtung von l bei v' ändert, betrachtet man die Halbebenen, die von der Geraden durch v und v' aufgespannt werden. Wenn der Vorgänger von v und der Nachfolger von v' in derselben Halbebene

liegen, ändert sich die Drehrichtung nicht, andernfalls ändert sie sich. Diese Überprüfung kann in konstanter Zeit erfolgen.

□

Für Randereignisse gilt das folgende Lemma:

Lemma 7. *Bei einem Randereignis, das durch einen Knoten $p \in P \setminus \pi(s,t)$ ausgelöst wird, ändern sich die inneren Knoten von $\pi(s,l^-)$ nur, wenn für s^* auf l^- bei weiterer Drehung von l gilt: $s^* \in S_p$. In diesem Fall wird p zu $\pi(s,l^-)$ hinter dem bisherigen letzten Knoten u hinzugefügt. Die Überprüfung, ob p zu $\pi(s,l^-)$ hinzugefügt werden muss, kann in konstanter Zeit durchgeführt werden.*

Beweis. Wenn p nicht der Scheitelpunkt einer Sichtbarkeitszelle von SPM_s ist, dann bleiben die optimalen Punkte s^* bei weiterer Drehung von l von u aus sichtbar. In diesem Fall ändern sich die inneren Knoten von $\pi(s,l^-)$ nicht.

Im folgenden wird daher angenommen, dass p der Scheitelpunkt von $S_p \in SPM_s$ ist. Nach Lemma 4 liegt der optimale Punkt s^* bei weiterer Drehung von l nur dann in S_p, wenn $f_p^{l^-} \in S_p$ oder wenn $f_p^{l^-}$ von der neuen Kante, die l^- nach dem Randereignis überstreift, verdeckt wird. In diesem Fall wird p nach u zu $\pi(s,l^-)$ hinzugefügt, andernfalls ändern sich die inneren Knoten von $\pi(s,l^-)$ nicht, siehe auch Abbildung 2.12.

Um zu überprüfen, ob der Knoten p zu $\pi(s,l^-)$ hinzugefügt werden muss oder nicht, hilft die folgende Überlegung: wenn l infinitesimal von p weggedreht wird, liegt der kürzeste Weg von p nach l auf der Halbgeraden, die bei p beginnt, orthogonal auf der Gerade durch p und v steht und in die derzeitige Drehrichtung zeigt. Jetzt kommt es darauf an, in welcher der geschlossenen Halbebenen, die von der Geraden durch u und p aufgespannt werden, diese Halbgerade liegt. Wenn sie in derselben geschlossenen Halbebene liegt wie der Drehpunkt v, dann muss p nicht zu $\pi(s,l^-)$ hinzugefügt werden. Andernfalls muss p zu $\pi(s,l^-)$ hinzugefügt werden. Diese Überprüfung kann in konstanter Zeit durchgeführt werden. □

Der Vollständigkeit halber sei darauf hingewiesen, dass bei der Entscheidung, ob der Knoten p bei einem Randereignis zum kürzesten Pfad hinzugefügt wird oder nicht, zwei Fälle zu unterscheiden sind:

1. Im ersten Fall verläuft der Pfad von u zu s^* vor dem Randereignis auf der Polygonkante, deren Anfangsknoten u ist. Diese Situation ist in Abbildung 2.12 dargestellt.

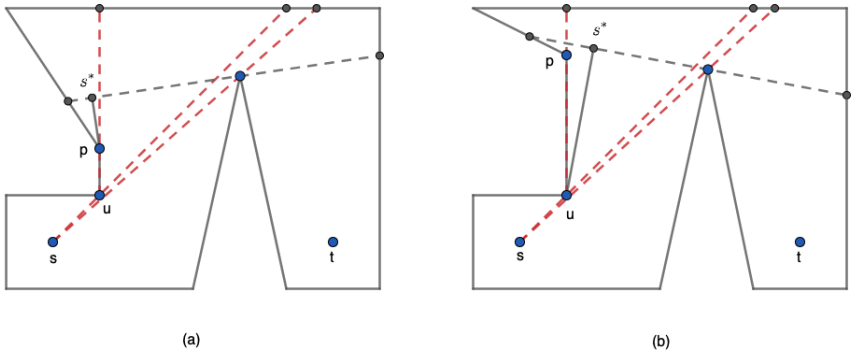

Abbildung 2.12: In Abbildung (a) liegt s^* nach dem Randereignis bei p in S_p und p kommt zu $\pi(s, l^-)$ hinzu. In Abbildung (b) bleibt s^* nach dem Randereignis bei p in S_u und die inneren Knoten von $\pi(s, l^-)$ ändern sich nicht.

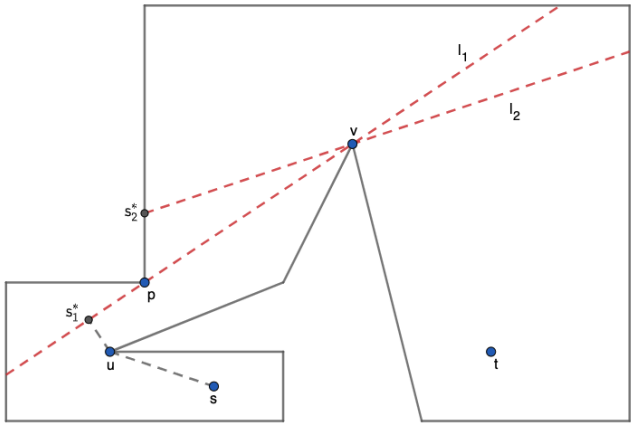

Abbildung 2.13: Bei dem von p ausgelösten Randereignis verläuft $\pi(u, l^-)$ im Polygoninneren von u zu s_1^*. Der Knoten p ragt so in das Polygoninnere hinein, dass nach dem Randereignis der Weg von u nach $f_u^{l^-}$ versperrt ist, so dass p zu $\pi(s, l^-)$ hinzugefügt werden muss.

2. Im zweiten Fall verläuft der Pfad von u zu s^* vor dem Randereignis im Polygoninneren. p wird in diesem Fall zu $\pi(s, l^-)$ nur dann hinzugefügt, wenn der Knoten p so in das Polygon hineinragt, dass er den direkten

Weg zu $f_u^{l^-}$ versperrt, siehe 2.13.

Biegungsereignisse zwischen Pfad- und Randereignissen

Nach der Untersuchung, unter welchen Umständen sich die inneren Knoten von $\pi(s, l^-)$ bei Pfad- und Randereignissen ändern, wird im folgenden untersucht, wann Biegungsereignisse zwischen zwei aufeinanderfolgenden Pfad- bzw. Randereignissen eintreten.

Wie schon erwähnt, ändern sich die inneren Knoten von $\pi(s, l^-)$ nur, wenn s^* bei Drehung von l die Grenze von Sichtbarkeitszellen in SPM_s überquert. Zwischen zwei aufeinanderfolgenden Pfad- bzw. Randereignissen kann dies in zwei Fällen eintreten: im ersten Fall trifft $\pi(u, l^-)$ auf einen Punkt $p \in P$ und s^* liegt im folgenden in S_p, siehe Abbildung 2.14. Im zweiten Fall wird die Kante $\overline{us^*}$ und die Kante zwischen u und seinem Vorgänger u^- auf $\pi(s, l^-)$ aufgrund der Drehung von l parallel. s^* liegt in diesem Fall bei weiterer Drehung von l in S_{u^-} und u kann von $\pi(s, l^-)$ entfernt werden, siehe Abbildung 2.15.

Für den ersten Fall, gilt folgendes Lemma:

Lemma 8. *Wenn $\pi(u, l^-)$ zwischen zwei aufeinanderfolgenden Pfad- oder Grenzereignissen auf einen Knoten $p \in P$ trifft, dann muss dieser Knoten der Nachfolger u^+ von u in $\pi(s, t)$ sein. Dabei tritt ein Biegungsereignis ein, bei dem u^+ als Nachfolger von u zu $\pi(s, l^-)$ hinzugefügt wird.*

Beweis. Der kürzeste Pfad von u zu einer Strecke innerhalb des Polygons liegt in einem sogenannten Trichter (siehe [13]). Der Trichter wird dabei durch den kürzesten Pfad von u zu den beiden Endpunkten der Strecke auf ∂P sowie durch die Strecke selbst begrenzt. Da sich die Sichtbarkeitsgerade l an v dreht, enthält der relevante Trichter v als einen der Streckenendknoten (siehe Abbildung 2.16). Der erste Knoten, den $\overline{us^*}$ bei Drehung von l um v antrifft, muss daher der auf u folgende Punkt u^+ in $\pi(u, v)$ sein. Da $\pi(u, v)$ ein Subpfad von $\pi(s, v)$ ist, ist u^+ auch in $\pi(s, v)$ der auf u folgende Punkt.

Bei weiterer Drehung von l um v liegt s^* in S_{u^+}, so dass u^+ als Nachfolger von u zu $\pi(s, l^-)$ hinzugefügt wird. □

Wenn der Endpunkt (und nicht das Innere) von $\pi(u, l^-)$ auf einen Knoten $p \in P$ trifft, ist folgendes zu beachten:

1. wenn der Endpunkt von $\pi(u, l^-)$ auf einen Knoten $p \notin \pi(s, v)$ trifft, definiert p ein Randereignis, so dass Lemma 7 gilt.

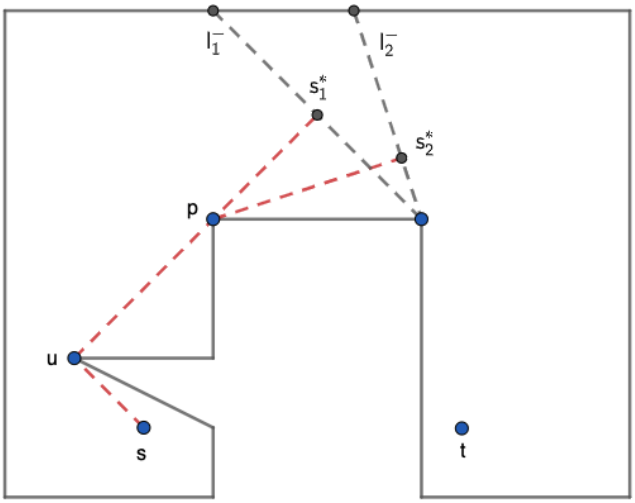

Abbildung 2.14: Bei l_1^- trifft $\overline{us^*}$ auf den Knoten p, der zu $\pi(s, l^-)$ hinzugefügt wird.

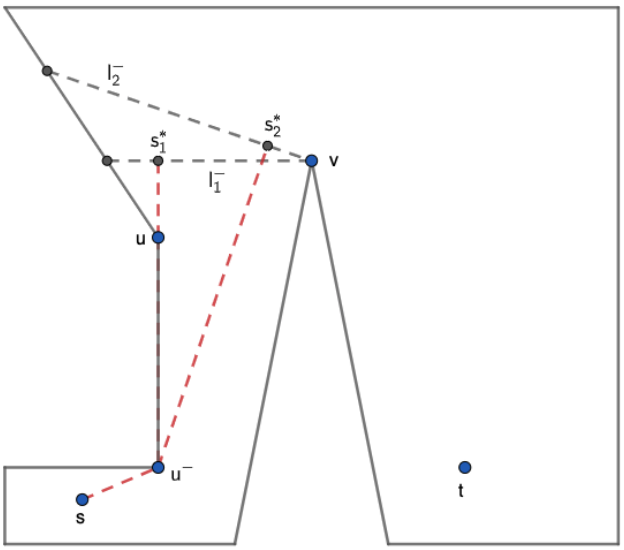

Abbildung 2.15: Bei l_1^- werden $\overline{us^*}$ und $\overline{u^-u}$ parallel und u wird von $\pi(s, l^-)$ entfernt.

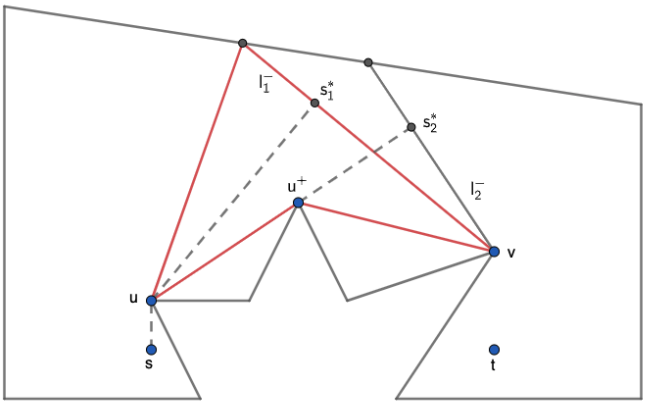

Abbildung 2.16: Der Trichter von u zu l_1^- ist rot eingezeichnet.

2. wenn der Endpunkt von $\pi(u, l^-)$ auf einen Knoten $p \in \pi(s, v)$ trifft, dann muß dieser Knoten nach Lemma 8 der derzeitige Drehpunkt v sein. Bei einem derartigen Biegungsereignis ist zu beachten, dass v bis zum nächsten Pfadereignis der letzte Punkt von $\pi(s, l^-)$ ist, denn v ist ja Teil der sich drehenden Sichtbarkeitsgeraden (siehe Abbildung 2.17).

Der zweite Fall, bei dem sich die inneren Knoten von $\pi(s, l^-)$ ändern, tritt ein, wenn die Kante zwischen u und seinem Vorgänger u^- in $\pi(s, l^-)$ parallel werden und bei weiterer Drehung von l gilt: $s^* \in S_{u^-}$. In diesem Fall sind die folgenden zwei Lemmata von Bedeutung:

Lemma 9. *Ein Knoten $u \in \pi(s, t)$, kann vom Pfad $\pi(s, l^-)$ nicht wieder verschwinden, wenn er einmal hinzugefügt worden ist.*

Beweis. Ein Knoten $u \in \pi(s, t)$ kann nach Lemmata 6 und 8 in zwei Fällen zu $\pi(s, l^-)$ hinzugefügt werden:

1. l^- trifft auf ein Pfadereignis, bei dem sich die Drehrichtung der Sichtbarkeitsgeraden ändert

2. $\pi(u^-, l^-)$ trifft zwischen zwei aufeinanderfolgenden Pfad- oder Grenzereignissen auf einen Knoten $u \in \pi(s, t)$.

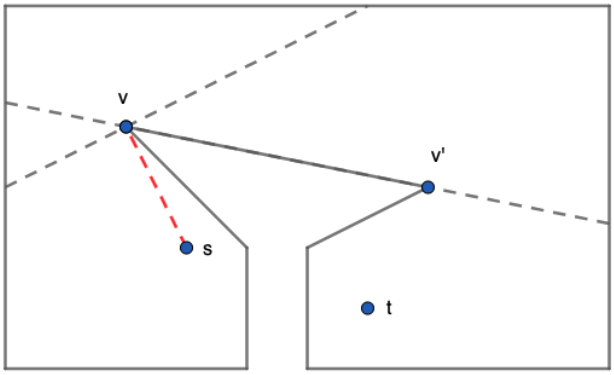

Abbildung 2.17: Der Drehpunkt v ist zwischen den eingezeichneten Sichtbarkeitsgeraden der Endpunkt von $\pi(s, l^-)$.

Im ersten Fall besagt Lemma 6, dass der hinzugefügte Knoten nicht mehr verschwinden kann. Im zweiten Fall gilt nach Lemma 8: wenn der hinzugefügte Knoten u wieder vom kürzesten Pfade verschwindet, muss es einen Trichter zwischen seinem Vorgänger u^- auf $\pi(s, l^-)$ und der Sichtbarkeitsgeraden l^- geben der u nicht enthält. Dies ist aber nach Definition des Trichters nur möglich, wenn u nicht zu $\pi(u^-, v)$ gehört, was ein Widerspruch ist, da u per definitionem Teil von $\pi(s, v)$ und somit von $\pi(u^-, v)$ ist. Daher kann auch im zweiten Fall der hinzugefügte Knoten nicht wieder vom Pfad $\pi(s, l^-)$ verschwinden. □

Lemma 10. *Wenn die Kante $\pi(u, l^-)$ und die Kante zwischen u und seinem Vorgänger u^- in $\pi(s, l^-)$ parallel werden und u daher vom Pfad $\pi(s, l^-)$ verschwindet, wird u während der weiteren Drehung der Sichtbarkeitsgeraden l kein zweites Mal mehr zu $\pi(s, l^-)$ hinzugefügt.*

Beweis. Nach Lemma 9 können Knoten $p \in \pi(s, t)$ nicht mehr von $\pi(s, l^-)$ verschwinden. Also müssen nur die Knoten betrachtet werden, die nicht zu $\pi(s, t)$ gehören und zu $\pi(s, l^-)$ hinzugefügt werden.

Nach Lemma 7 werden Knoten $p \notin \pi(s, t)$ bei Randereignissen zu $\pi(s, l^-)$ hinzugefügt, wenn für s^* bei weiterer Drehung von l gilt: $s^* \in S_p$. Da die von l^- überstreiften Sichtbarkeitszellen paarweise disjunkt sind, können diese Randereignisse bei Drehung der Sichtbarkeitsgeraden nur einmal angetroffen werden. Das heißt, dass ein Knoten $p \notin \pi(s, t)$ nur einmal zu $\pi(s, l^-)$ hinzu-

gefügt werden kann. Wenn er von $\pi(s, l^-)$ wieder verschwindet, ist ein zweites Hinzufügen nicht mehr möglich. □

Degenerierte Biegungsereignisse

Neben den beschriebenen Biegungsereignissen gibt es noch sogenannte *degenerierte Biegungsereignisse*. Bei diesen werden dem kürzesten Pfad zur Sichtbarkeitsgeraden keine Knoten hinzugefügt oder weggenommen, aber eine Polygonkante beginnt bzw. hört auf, den direkten Weg von u zum Lotfußpunkt $f_u^{l^-}$ zu versperren (siehe Abbildung 2.18 als Beispiel). Degenerierte Biegungsereignisse lassen sich folgendermaßen berechnen: die Lotfußpunkte $f_u^{l^-}$ bilden nach dem Thalessatz einen Halbkreis h_{uv} über u und dem Drehpunkt v. Ein degeneriertes Biegungsereignis tritt dort ein, wo sich h_{uv} und die gerade überstreifte Polygonkante schneiden. Maximal kann es für jeden Halbkreis und jede überstreifte Polygonkante zwei solcher Schnittpunkte geben. Bei jedem Schnittpunkt muss entschieden werden, ob die Polygonkante beginnt oder aufhört, den direkten Pfad zu $f_u^{l^-}$ zu versperren. Diese Entscheidung kann in konstanter Zeit durchgeführt werden.[5]

Degenerierte Biegungsereignisse sind für die Berechnung des lokalen Minimums zwischen zwei Ereignissen von Bedeutung. Denn der Abstand zu l^- ist unterschiedlich, je nachdem ob $f_u^{l^-}$ von u aus sichtbar ist oder nicht.

Komplexität der Berechnung aller Biegungsereignisse

Lemma 11. *Alle degenerierten und nicht degenerierten Biegungsereignisse können in der Reihenfolge, in der sie von der sich drehenden Sichtbarkeitsgeraden angetroffen werden, in $\mathcal{O}(n)$ Zeit berechnet und in die Ereignisliste eingefügt werden.*

Beweis. Um alle Biegungsereignisse zu berechnen, werden zwei Iterationen über die Ereignisliste benötigt: in der ersten Iteration werden die Biegungsereignisse, die durch l^- ausgelöst werden, berechnet, in der zweiten Iteration die Biegungsereignisse, die durch l^+ ausgelöst werden.

Bei jedem Durchgang der Iteration betrachtet man zwei aufeinanderfolgende Einträge in der Ereignisliste, die im folgenden als l_1 und l_2 bezeichnet werden. Wenn die durch l^- ausgelösten Biegungsereignisse betrachtet werden, können l_1 und l_2 nur Pfad- oder Randereignisse sein. Bei der zweiten

[5] Dafür muss die Lage einer Halbgeraden bezüglich zweier Halbebenen überprüft werden, wie in Lemma 7 beschrieben.

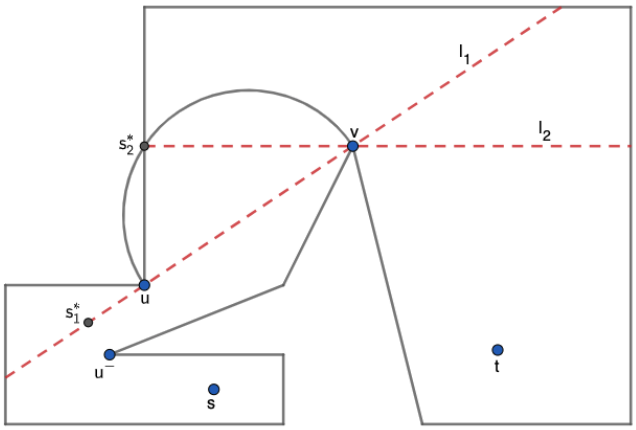

Abbildung 2.18: Ab dem Randereignis, das bei l_1 eintritt, verläuft der kürzeste Pfad von u nach l^- zunächst auf der Polygonkante. Bei l_2 tritt ein degeneriertes Biegungsereignis ein, ab hier ist der Weg von u nach l^- die Strecke $\overline{uf_u^{l^-}}$, die im Polygoninneren verläuft.

Iteration werden hingegen auch die aufgrund von l^- eingefügten Biegungsereignisse berücksichtigt.

Zwischen zwei aufeinanderfolgenden Ereignissen in der Ereignisliste werden immer dieselben Schritte durchgeführt. Diese Schritte werden im folgenden nur für l^- erklärt, sie können aber durch entsprechende Anpassungen auf l^+ angewendet werden.

v sei der Drehpunkt, an dem l zwischen l_1 und l_2 gedreht wird, u sei der zuletzt berechnete Knoten auf $\pi(s, l_1^-)$ vor l_1^- und p_1 und p_2 seien die Schnittpunkte von l_1^- und l_2^- mit dem Polygonrand, siehe Abbildung 2.19. Am Anfang der ersten Iteration wird u mit s initialisiert.

Dann führt man zwischen jeweils zwei Ereignissen l_1 und l_2 die folgenden Schritte durch:

1. Berechne und speichere die degenerierten Biegungsereignisse, die zwischen l_1 und l_2 auftreten können, in temporären Variablen d_1 und d_2.

2. Falls u kein Knoten von $\pi(s, t)$ ist, wiederhole Schritt 2, bis $\pi(s, l^-)$ zwischen l_1 und l_2 keinen Knoten mehr verliert:

   - prüfe, ob u zwischen l_1 und l_2 verloren geht

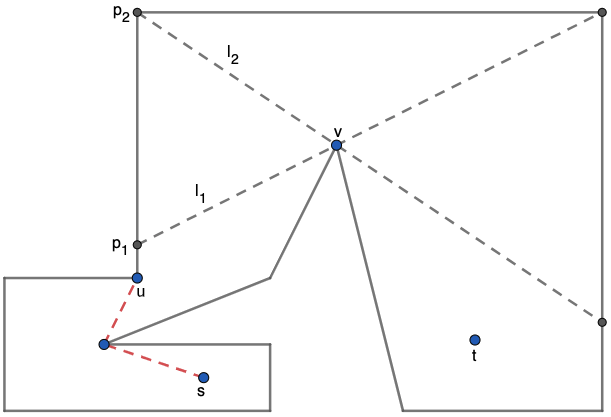

Abbildung 2.19: Die Biegungsereignisse werden zwischen zwei aufeinanderfolgenen Ereignissen in der Ereignisliste (hier zwei Randereignisse) berechnet.

- Wenn ein derartiges Biegungsereignis eintritt, prüfe, ob d_1 und/oder d_2 vor diesem Biegungsereignis liegen bzw. mit ihm zusammenfallen. Je nachdem, ob das der Fall ist, füge d_1 und/oder d_2 in der richtigen Reihenfolge in die Ereignisliste ein bzw. passe die zu verwendende Distanzfunktion beim berechneten Biegungsereignis entsprechend an.
- Prüfe ob das berechnete Biegungsereignis mit l_2 zusammenfällt. Wenn das der Fall ist, gehe direkt zu Schritt 4. Ansonsten füge das Biegungsereignis in die Ereignisliste ein.
- Setze u auf seinen Vorgänger in $\pi(s, l^-)$, passe d_1 und d_2 entsprechend dem neuen Endknoten u an und setze l_1 auf das gerade berechnete Biegungsereignis.

3. Wenn u nach Ende von Schritt 2 ein Knoten von $\pi(s, t)$ ist, wiederhole Schritt 3, bis zwischen l_1 und l_2 kein Knoten von $\pi(s, t)$ mehr zu $\pi(s, l^-)$ hinzukommt:

   - prüfe, ob der Nachfolger von u in $\pi(s, t)$ zwischen l_1 und l_2 zu $\pi(s, l^-)$ hinzugefügt werden muss.
   - Wenn dies der Fall ist, prüfe, ob d_1 und/oder d_2 vor diesem Biegungsereignis liegen bzw. mit ihm zusammenfallen. Je nachdem,

ob das der Fall ist, füge d_1 und/oder d_2 in der richtigen Reihenfolge in die Ereignisliste ein bzw. passe die zu verwendende Distanzfunktion beim berechneten Biegungsereignis entsprechend an.

- Prüfe ob das berechnete Biegungsereignis mit l_2 zusammenfällt. Wenn das der Fall ist, gehe direkt zu Schritt 4. Ansonsten füge das Biegungsereignis in die Ereignisliste ein.
- Setze u auf seinen Nachfolger in $\pi(s,t)$, passe d_1 und d_2 entsprechend dem neuen Endknoten u an und setze l_1 auf das gerade berechnete Biegungsereignis.

4. Wenn nötig, füge d_1 und/oder d_2 vor l_2 in die Ereignisliste ein. Dann führe die folgenden Schritte aus:

   - Falls ein Biegungsereignis aus Schritt 2 oder 3 mit l_2 zusammenfällt, passe den Pfad $\pi(s,l^-)$ bei l_2 entsprechend an.
   - Prüfe, ob l_2 Änderungen an den Knoten von $\pi(s,l^-)$ entsprechend der Lemmata 6 und 7 auslöst. Wenn ja, passe den Pfad $\pi(s,l^-)$ bei l_2 entsprechend an.
   - Prüfe, ob bei l_2 ein degeneriertes Biegungsereignis eintritt. Wenn das der Fall ist, passe die zu verwendende Distanzfunktion bei l_2 entsprechend an.

Die Korrektheit des Algorithmus kann wie folgt bewiesen werden: zwischen zwei Pfad- und Randereignissen kann $\pi(s,l^-)$ nur Knoten $u \notin \pi(s,t)$ verlieren oder Knoten $u^+ \in \pi(s,t)$ dazugewinnen. Nach Lemma 8 kann ein Knoten $u^+ \in \pi(s,t)$ nur zu $\pi(s,l^-)$ hinzukommen, wenn auch sein Vorgänger u ein Knoten von $\pi(s,t)$ ist. Wenn für den letzten Knoten u vor l^- gilt: $u \in \pi(s,t)$, dann müssen auch alle Vorgänger von u auf $\pi(s,l^-)$ zu $\pi(s,t)$ gehören. Das bedeutet dass $\pi(s,l^-)$ zunächst alle nicht zu $\pi(s,t)$ gehörenden Knoten verlieren muss, bevor ein zu $\pi(s,t)$ gehörender Knoten hinzugefügt werden kann. Daher ist die Reihenfolge der Schritte 2, 3 und 4 korrekt. Bei den degenerierten Biegungsereigissen kann man keine Vorhersage über ihre Reihenfolge treffen, so dass sie in den Schritten 2, 3 und 4 jeweils mit berücksichtigt werden müssen.

Um die lineare Zeitkomplexität zu beweisen, betrachtet man die Gesamtkomplexität der einzelnen Schritte. Bezüglich Schritt 4 gilt: ob ein Pfad- bzw. Randereignis Änderungen an den Knoten von $\pi(s,l^-)$ auslöst, kann

nach Lemmata 6 und 7 in konstanter Zeit entschieden werden. Da es $\mathcal{O}(n)$ Pfad- und Randereignisse gibt, kann in $\mathcal{O}(n)$ Zeit entschieden werden, ob alle Pfad- und Randereignisse ein Biegungsereignis auslösen.

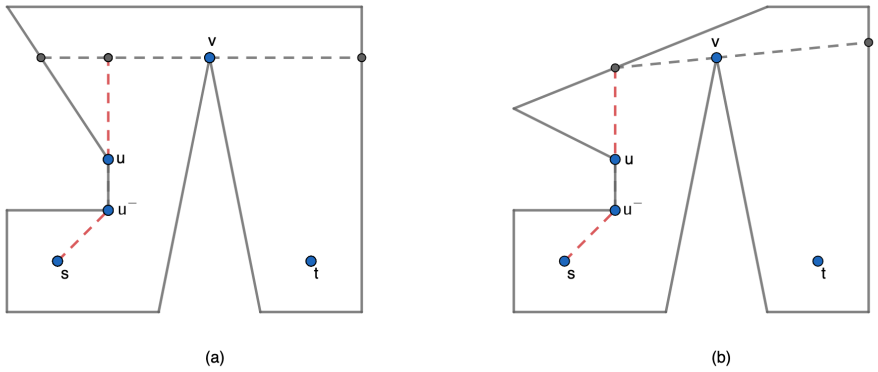

Abbildung 2.20: In Abbildung (a) steht die Gerade durch u^- und u senkrecht auf einer Sichtbarkeitsgeraden, die die gerade überfegte Polygonkante schneidet. In Abbildung (b) schneidet die Gerade durch u^- und u die gerade überfegte Polygonkante. In beiden Fällen geht u verloren.

Um die Biegungsereignisse für Schritt 2 zu berechnen, kann man folgendermaßen vorgehen: wenn u nicht zu $\pi(s,t)$ gehört und einen Vorgänger u^- auf $\pi(s,l^-)$ hat, betrachtet man die Gerade durch u und u^-. Wenn diese Gerade entweder auf einer Sichtbarkeitsgeraden, die die gerade überfegte Polygonkante schneidet, senkrecht steht, oder aber die gerade überfegte Polygonkante schneidet, verliert der Pfad den Knoten u (siehe Abbildung 2.20). Diese Überprüfung kann in konstanter Zeit erfolgen. Ein Knoten p kann nach Lemma 10 nur einmal von $\pi(s,l^-)$ entfernt werden. Insgesamt gibt es daher $\mathcal{O}(n)$ Biegungsereignisse, bei denen ein Knoten verloren gehen kann, und ihre Berechnung hat insgesamt die Zeitkomplexität $\mathcal{O}(n)$.

Auch für Schritt 3 gilt, dass bei der Überprüfung, ob der Nachfolger u^+ von u auf $\pi(s,t)$ zu $\pi(s,l^-)$ hinzukommt, eine Fallunterscheidung gemacht werden muss: hier betrachtet man die Gerade durch u und u^+. Wenn diese Gerade entweder auf einer Sichtbarkeitsgeraden, die die gerade überfegte Polygonkante schneidet, senkrecht steht, oder aber die gerade überfegte Polygonkante schneidet, gewinnt der Pfad den Knoten u^+ hinzu (siehe Abbildung 2.21). Diese Überprüfung kann in konstanter Zeit erfolgen. Nach Lemma 9

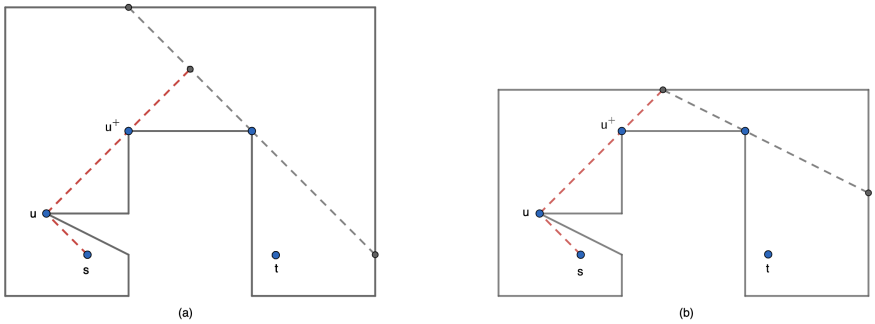

Abbildung 2.21: In Abbildung (a) steht die Gerade durch u und u^+ senkrecht auf einer Sichtbarkeitsgeraden, die die gerade überfegte Polygonkante schneidet. In Abbildung (b) schneidet die Gerade durch u und u^+ die gerade überfegte Polygonkante. In beiden Fällen kommt u^+ zum Pfad dazu.

kann ein Knoten $p \in \pi(s,t)$ nur einmal zu $\pi(s,l^-)$ hinzukommen. Da es $\mathcal{O}(n)$ Knoten von $\pi(s,t)$ gibt, ist die Gesamtkomplexität der Berechnung von Biegungsereignissen, bei denen ein Knoten $p \in \pi(s,t)$ zu $\pi(s,l^-)$ hinzukommt, $\mathcal{O}(n)$.

Für Schritt 1 und die Anpassung der degenerierten Biegungsereignisse in Schritten 2 und 3 müssen Schnittpunkte von Halbkreisen mit Geradenstücken berechnet werden, was in konstanter Zeit erfolgen kann. Degenerierte Biegungsereignisse können zwischen je zwei Pfad-, Grenz- oder Biegungsereignissen auftreten oder mit ihnen zusammenfallen. Da es $\mathcal{O}(n)$ solcher Ereignisse gibt, liegt die Komplexität der Berechnung aller degenerierten Biegungsereignisse in $\mathcal{O}(n)$.

Wenn ein Biegungsereignis erzeugt wird, ist es nötig, den gegenüberliegenden Schnittpunkt mit der Polygonkante zu berechnen. Da die gegenüberliegende Kante, die von l überstreift wird, bekannt ist, (sie ist in den Ereignissen der Ereignisliste gespeichert), kann dies pro Biegungsereignis in konstanter Zeit und somit insgesamt in linearer Zeit erfolgen.

Insgesamt sind zwei Iterationen mit je $\mathcal{O}(n)$ Zeitkomplexität notwendig, so dass die gesamte Zeitkomplexität der Berechnung der Biegungsereignisse und ihrer Einfügung in die Ereignisliste linear in n ist. □

2.4 Lokale Minima zwischen zwei Ereignissen in der min-sum Version des Algorithmus

Im vorherigen Kapitel 2.3 wurde die Berechnung aller relevanten Ereignisse und ihre Speicherung in der Ereignisliste erläutert. Um die globalen Minima für die min-sum Version des Sichtbarkeitsproblems zu finden, müssen zunächst die lokalen Minima, die zwischen 2 aufeinanderfolgenden Ereignissen liegen, gefunden werden. Mit diesem Problem beschäftigt sich dieses Kapitel. Die Berechnung der lokalen Minima ist unterschiedlich, je nachdem ob die Lotfußpunkte auf der Sichtbarkeitsgeraden sichtbar sind oder nicht. Daher gliedert sich dieses Kapitel folgendermaßen: nach ein paar Vorbemerkungen in Abschnitt 2.4.1 wird in Abschnitt 2.4.2 untersucht, wo die lokalen Minima bei freier Sicht auf die Lotfußpunkte liegen. In Abschnitt 2.4.3 wird untersucht, wo sie bei versperrter Sicht liegen. Das Hauptergebnis des Kapitels ist, dass die lokalen Minima zwischen zwei Ereignissen in allen Fällen in konstanter Zeit berechnet werden können.

Bei den Berechnungen der lokalen Minima ist zu beachten, dass der maximale Winkel ρ zwischen zwei Sichtbarkeitsgeraden, die zwei aufeinanderfolgende Ereignisse definieren, 90° ist. Denn spätestens, wenn der Drehwinkel 90° erreicht, tritt ein Biegungsereignis ein, bei dem der aktuelle Drehpunkt Teil des kürzesten Pfads zur Sichtbarkeitsgeraden wird. Im folgenden wird daher - wenn nichts anderes erwähnt ist - $\rho \leq 90°$ angenommen.

Um die Lage von Punkten, Strecken und Geraden bestimmen und Distanzen zwischen ihnen berechnen zu können, bedarf es eines Koordinatensystems. Im folgenden wird ein kartesisches Koordinatensystem angenommen, wobei die x-Werte auf der horizontalen Achse und die y-Werte auf der vertikalen Achse dargestellt werden. Jeder Punkt p ist in diesem Koordinatensystem durch einen x und einen y Wert eindeutig bestimmt. Die Werte der Koordinaten werden als $p.x$ und $p.y$ geschrieben.

2.4.1 Vorbemerkungen

In den folgenden Abschnitten bezeichnen u und u' die letzten Knoten auf den Pfaden $\pi(s, l^-)$ bzw. $\pi(t, l^+)$, die auch Knoten von P sind. v ist der Knotenpunkt von P, an dem die Sichtbarkeitsgerade gedreht wird. l_1 ist die Sichtbarkeitsgerade durch u und v, ihre Steigung ist α_1, l_2 ist die Sichtbarkeitsgerade durch u' und v, ihre Steigung ist α_2. l bezeichnet eine beliebige

Sichtbarkeitsgerade zwischen l_1 und l_2, ihre Steigung wird mit α bezeichnet. Während der Drehung um v ändert sich α und somit der kürzeste Weg von u bzw. u' nach l. Da sich u und u' während der Drehung zwischen zwei Ereignissen nicht ändern, bleiben $|\pi(s,u)|$ und $|\pi(t,u')|$ gleich. Ihre Länge ist die Summe der Länge der Kanten, aus denen die Pfade $\pi(s,u)$ und $\pi(t,u')$ bestehen.

Wenn man eine fixe Sichtbarkeitsgerade l' durch v betrachtet, und wenn die Lotfußpunkte auf l' von u und u' aus sichtbar sind, dann ist der kürzeste Weg von u nach l' die Strecke $\overline{uf_u^{l'}}$. Ebenso ist der kürzeste Weg von u' nach l' die Strecke $\overline{u'f_{u'}^{l'}}$. Die Lotfußpunkte auf den Sichtbarkeitsgeraden zwischen l_1 und l_2 bilden nach dem Satz des Thales einen Halbkreis über u und v bzw. über u' und v und werden als h_{uv} bzw. $h_{u'v}$ bezeichnet.

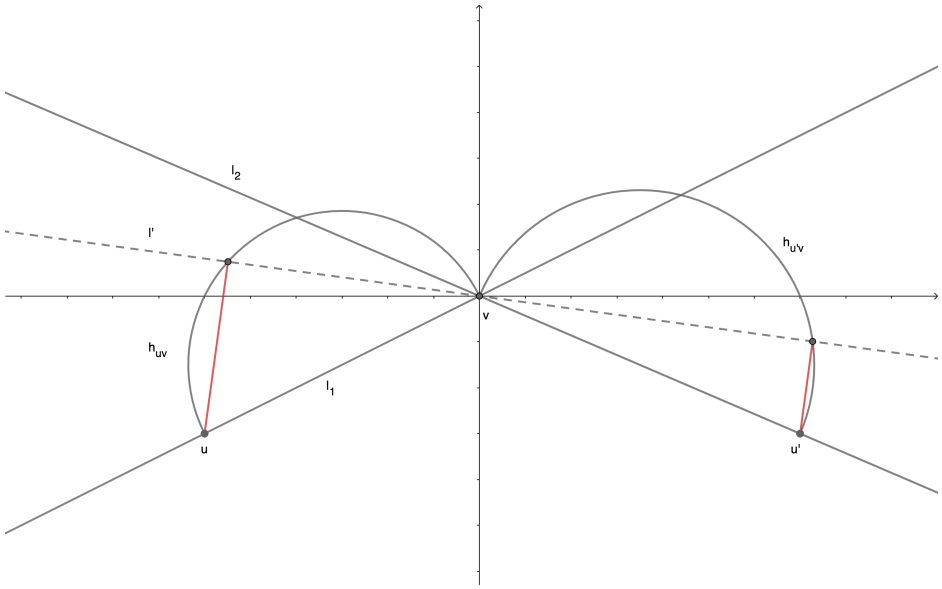

Abbildung 2.22: Die Endpunkte der kürzesten Wege von u (u′) zu den Sichtbarkeitsgeraden durch v liegen auf den Halbkreisen h_{uv} ($h_{u'v}$). Per Annahme liegt v im Ursprung des Koordinatensystems.

Wenn man in einem Polygon beliebige Knoten u, u' und v betrachtet, dann können diese in den unterschiedlichsten Positionen im Koordinatensystem liegen. Allerdings ist es immer möglich das Polygon so zu verschieben und zu drehen, dass v im Ursprung liegt, u in Quadrant 3, u' in Quadrant 4, und dass weder l_1 noch l_2 vertikal verlaufen. Bei dieser Verschiebung und Drehung

ändert sich nichts an der relativen Lage des Minimums und an der Länge der kürzesten Pfade, so dass im folgenden ohne Beschränkung der Allgemeinheit eine solche Lage angenommen wird, auch wenn das Koordinatensystem nicht eingezeichnet ist. Dadurch kann bei der Darstellung der Beweise auf unnötige Fallunterscheidungen verzichtet werden (siehe Abbildung 2.22).

2.4.2 Lokale Minima bei freier Sicht

Dieser Abschnitt beschäftigt sich mit der Frage, welche Eigenschaften die minimale gemeinsame Distanz zwischen zwei Ereignissen aufweist, wenn die Wege von u und u' zu den Lotfußpunkten auf l von keiner Polygonkante versperrt werden. Dies ist dann der Fall wenn jeder Punkt im Halbkreis h_{uv} von u aus sichtbar ist und jeder Punkt im Halbkreis $h_{u'v}$ von u' aus. Zunächst wird dabei angenommen das u auf l_1 liegt und u' auf l_2.

Während der Drehung von l zwischen l_1 und l_2, ändert sich die Steigung α von l, wobei aufgrund der Lage der Sichtbarkeitsgeraden gilt: $\alpha_2 \leq \alpha \leq \alpha_1$. v, u und u' bleiben während dieser Drehung gleich. Daher kann die gemeinsame Distanz von u und u' zu l als Funktion in Abhängigkeit von α beschrieben werden. Diese Distanzfunktion wird im folgenden als $d(\alpha)$ bezeichnet und ist folgendermaßen definiert (siehe Anhang A für eine Herleitung):

$$d(\alpha) = \frac{|(u.x - v.x) \cdot \alpha + v.y - u.y| + |(u'.x - v.x) \cdot \alpha + v.y - u'.y|}{\sqrt{\alpha^2 + 1}} \quad (2.2)$$

Für $d(\alpha)$ gilt im Bereich zwischen α_1 und α_2 das folgenden Lemma:

Lemma 12. *Die Ableitung der Distanzfunktion $d(\alpha)$ hat im Bereich $\alpha_2 \leq \alpha \leq \alpha_1$ maximal einen Nullpunkt. Wenn ein Nullpunkt existiert, handelt es sich um ein Maximum.*

Beweis. Für die Steigungen α_1 und α_2 gilt folgendes:

$$\alpha_1 = \frac{v.y - u.y}{v.x - u.x} \quad (2.3)$$

$$\alpha_2 = \frac{v.y - u'.y}{v.x - u'.x} \quad (2.4)$$

Da per Annahme $u.x < v.x$ und $\alpha \leq \alpha_1$, gilt:

$$\alpha \leq \alpha_1 = \frac{v.y - u.y}{v.x - u.x} \tag{2.5}$$

Daraus folgt:

$$(u.x - v.x) \cdot \alpha + v.y - u.y \geq 0 \tag{2.6}$$

Da per Annahme $u'.x > v.x$ und $\alpha \geq \alpha_2$, gilt außerdem:

$$\alpha \geq \alpha_2 = \frac{v.y - u'.y}{v.x - u'.x} \tag{2.7}$$

Daraus folgt:

$$(u'.x - v.x) \cdot \alpha + v.y - u'.y \geq 0 \tag{2.8}$$

Die gemeinsame Distanz kann daher für den Bereich zwischen l_1 und l_2 ohne Absolutzeichen geschrieben werden. Wenn man in Gleichung 2.2 außerdem konstante Ausdrücke durch Konstantennamen ersetzt, also $a = u.x - v.x$, $b = v.y - u.y$, $c = u'.x - v.x$ und $d = v.y - u'.y$ erhält man:

$$d(\alpha) = \frac{a \cdot \alpha + b + c \cdot \alpha + d}{\sqrt{\alpha^2 + 1}} \tag{2.9}$$

Die erste Ableitung dieser Funktion nach α lautet:

$$d(\alpha)' = \frac{-(b + d) \cdot \alpha + a + c}{(\alpha^2 + 1)^{\frac{3}{2}}} \tag{2.10}$$

Diese Ableitung wird genau dann 0, wenn gilt:

$$\alpha = \frac{a + c}{b + d} \tag{2.11}$$

Falls $\frac{a+c}{b+d}$ zwischen α_1 und α_2 liegt, hat die erste Ableitung der gemeinsamen Distanzfunktion also genau einen Nullpunkt. Wenn dies nicht der Fall ist, hat sie keinen.

Es stellt sich die Frage, ob dieser Nullpunkt ein Maximum, ein Minimum oder ein Wendepunkt ist. Dabei hilt die zweite Ableitung von $d(\alpha)$ (berechnet mithilfe von [20]):

$$d(\alpha)'' = \frac{-b - d}{(\alpha^2 + 1)^{\frac{3}{2}}} + \frac{3 \cdot \alpha \cdot ((b + d) \cdot \alpha - a - c)}{(\alpha^2 + 1)^{\frac{5}{2}}} \tag{2.12}$$

Der zweite Summand in 2.12 wird 0, wenn man die Lösung für den Nullpunkt 2.11 einsetzt. Beim ersten Summanden ist der Nenner auf jeden Fall positiv. Da $b = v.y - u.y$ und $d = v.y - u'.y$ erhält man für den Zähler:

$$-b - d = u.y + u'.y - 2 \cdot v.y \tag{2.13}$$

Per Annahme liegt v über u und u', also gilt $v.y > u.y$ und $v.y > u'.y$. Daher ist der Ausdruck 2.13 negativ und $d(\alpha)''$ insgesamt ist im Nullpunkt von $d(\alpha)'$ negativ. Das heißt: wenn $d(\alpha)$ im Bereich $\alpha_2 \leq \alpha \leq \alpha_1$ einen Nullpunkt der ersten Ableitung hat, handelt es sich um ein Maximum. □

Wenn es also zwischen l_1 und l_2 einen Nullpunkt der ersten Ableitung von $d(\alpha)$ gibt, dann steigt die gemeinsame Distanz sowohl, wenn l von l_1 in Richtung l_2 gedreht wird, als auch, wenn l von l_2 in Richtung l_1 gedreht wird. In diesem Fall liegt das lokale Minimum von $d(\alpha)$ bei l_1 oder l_2. Wenn es keinen Nullpunkt gibt, muss $d(\alpha)$ entweder kontinuierlich steigen oder fallen, wenn l von l_1 nach l_2 gedreht wird. Auch in diesem Fall liegt das lokale Minimum von $d(\alpha)$ bei l_1 oder l_2. Dieser Sachverhalt kann auch anhand der folgenden geometrischen Betrachtungen bewiesen werden:[6]

Lemma 13. *Ohne Beschränkung der Allgemeinheit sei $|\overline{uv}| < |\overline{u'v}|$. Dann gibt es im Bereich der Sichtbarkeitsgeraden l mit Steigung $\alpha : \alpha_2 \leq \alpha \leq \alpha_1$ genau ein lokales Minimum des min-sum Problems und die minimale Distanz d_{min} ist:*

$$\textit{(i)}\ d_{min} = |\pi(s, l_2)| + |\pi(t, u')| \tag{2.14}$$

Wenn $|\overline{uv}| = |\overline{u'v}|$, dann gibt es in diesem Bereich zwei lokale Minima und für d_{min} gilt:

$$\textit{(ii)}\ d_{min} = |\pi(s, l_2)| + |\pi(t, u')| = |\pi(s, u)| + |\pi(t, l_1)| \tag{2.15}$$

Beweis. Da sich $\pi(s, u)$ und $\pi(t, u')$ während der Drehung von l um v nicht ändern, ist es ausreichend, die Distanzen zwischen u, u' und l zu betrachten.

Für den Fall $l = l_1$ gilt folgendes: die Dreiecke $\Delta u v f_u^{l_2}$ und $\Delta u' v f_{u'}^{l_1}$ sind ähnlich, weil beide einen rechten Winkel haben und die gegenüberliegenden Winkel bei v gleich sind. Wenn $\overline{uv} < \overline{u'v}$ folgt daraus $|\overline{u f_u^{l_2}}| < |\overline{u' f_{u'}^{l_1}}|$ und

[6] Lemma 13 ist eigentlich redundant, da es nicht mehr beweist als Lemma 12. Allerdings beleuchtet es den Sachverhalt anschaulich, so dass ich entschlossen habe, es dem Leser nicht vorzuenthalten.

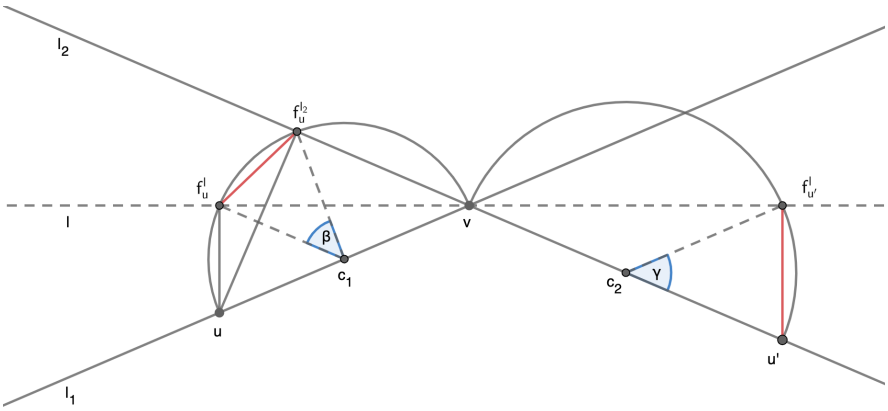

Abbildung 2.23: Die gemeinsame Distanz zu l_2, d.h. $|\overline{uf_u^{l_2}}|$ ist minimal.

$|\pi(s, l_2)| + |\pi(t, u')| < |\pi(s, u)| + |\pi(t, l_1)|$. Wenn $|\overline{uv}| = |\overline{u'v}|$, sind die Dreiecke $\Delta uvf_u^{l_2}$ und $\Delta u'vf_{u'}^{l_1}$ kongruent und es gilt: $|\overline{uf_u^{l_2}}| = |\overline{u'f_{u'}^{l_1}}|$ und daher $|\pi(s, l_2)| + |\pi(t, u')| = |\pi(s, u)| + |\pi(t, l_1)|$.

Für den Fall, dass l zwischen l_1 und l_2 liegt, wird zunächst angenommen, dass die Strecken $\overline{uv}$ und $\overline{u'v}$ gleich lang sind. Außerdem gelten folgende Definitionen (siehe auch Abbildung 2.23):

- c_1 ist der Mittelpunkt des Halbkreises h_{uv}
- c_2 ist der Mittelpunkt des Halbkreises $h_{u'v}$
- β ist der Winkel zwischen $f_u^{l_2}$, c_1 und f_u^l
- γ ist der Winkel zwischen u', c_2 und $f_{u'}^l$
- die Länge der Strecke zwischen f_u^l und $f_u^{l_2}$ ist d_1
- die Länge der Strecke zwischen u' und $f_{u'}^l$ ist d_2

Die Winkel β und γ sind gleich groß. Dies kann anhand der Tatsachen, dass der gesamte Innenwinkel eines Dreiecks 180° beträgt und die Basiswinkel eines gleichschenkligen Dreiecks gleich sind, bewiesen werden. Der genaue (etwas langwierige) Beweis dafür findet sich im Anhang B.

Die gemeinsame Distanz von u und u' zu l ist $|\overline{uf_u^l}| + d_2$. Die gemeinsame Distanz zu l_2 hingegen ist $|\overline{uf_u^{l_2}}|$. Beachte, dass auf jeden Fall gilt: $|\overline{uf_u^l}| + d_1 >$

$|\overline{uf_u^{l_2}}|$. Außerdem gilt $d_1 = d_2$, da die Winkel β und γ und die Kreise mit Mittelpunkt c_1 bzw. c_2 gleich groß sind. Somit gilt: $|\overline{uf_u^{l}}| + d_1 = |\overline{uf_u^{l}}| + d_2 > |\overline{uf_u^{l_2}}|$. Oder in Worten ausgedrückt: die gemeinsame Distanz zu jeder Geraden zwischen l_1 und l_2 ist länger als die gemeinsame Distanz zu l_2.

Für den Fall, dass u' weiter von v entfernt ist als u, sind die Winkel β und γ immer noch gleich, der Kreis mit Mittelpunkt c_2 ist aber größer. Damit ist in diesem Fall d_2 länger als d_1, und die Distanz $|\overline{uf_u^{l_2}}|$ ist wiederum minimal. □

Mithilfe dieser Erkenntnis kann die Annahme fallengelassen werden, dass u auf l_1 und u' auf l_2 liegt, und es gilt das folgende Lemma:

Lemma 14. *Die Halbkreise h_{uv} und $h_{u'v}$ seien von u bzw. u' aus im Bereich zwischen l_1 und l_2 sichtbar. Dann liegt das lokale Minimum auf l_1, l_2 oder es gibt 2 lokale Minima bei l_1 und l_2.*

Formal ausgedrückt gilt für das lokale Minimum d_{min} der gemeinsamen Distanzfunktion $|\pi(s,l)| + |\pi(t,l)|$ im Bereich zwischen l_1 und l_2 eine der folgenden Aussagen:

(i) $d_{min} = |\pi(s, f_u^{l_1})| + |\pi(t, f_{u'}^{l_1})|$

(ii) $d_{min} = |\pi(s, f_u^{l_2})| + |\pi(t, f_{u'}^{l_2})|$

(iii) $d_{min} = |\pi(s, f_u^{l_1})| + |\pi(t, f_{u'}^{l_1})| = |\pi(s, f_u^{l_2})| + |\pi(t, f_{u'}^{l_2})|$

Beweis. Ohne Beschränkung der Allgemeinheit sei $|\overline{uv}| \leq |\overline{u'v}|$. Wenn man sich die Kanten des Polygons wegdenkt, kann man eine Sichtbarkeitsgerade l_1' durch u und v legen und eine andere - l_2' - durch u' und v (siehe Abbildung 2.24). Dann gelten für den Bereich zwischen l_1' und l_2' alle bisher bewiesenen Lemmata.

Wenn es zwischen l_1' und l_2' keinen Nullpunkt der ersten Ableitung der gemeinsamen Distanzfunktion gibt, dann muss die gemeinsame Distanz gemäß Lemma 13 während der gesamten Drehung der Sichtbarkeitsgeraden von l_2' bis l_1' zunehmen. D.h. auch zwischen l_2 und l_1 muss die gemeinsame Distanz zunehmen. Somit ist in diesem Fall $|\pi(s, f_u^{l_2})|+|\pi(t, f_{u'}^{l_2})|$ das lokale Minimum.

Falls zwischen l_1' und l_2' ein Maximum existiert, kann dieses entweder auf einer Geraden zwischen l_1 und l_2 liegen oder außerhalb. Wenn das Maximum außerhalb liegt, nimmt die gemeinsame Distanz zwischen l_1 und l_2 entweder stetig ab oder stetig zu. Das lokale Minimum ist also $|\pi(s, f_u^{l_1})| + |\pi(t, f_{u'}^{l_1})|$

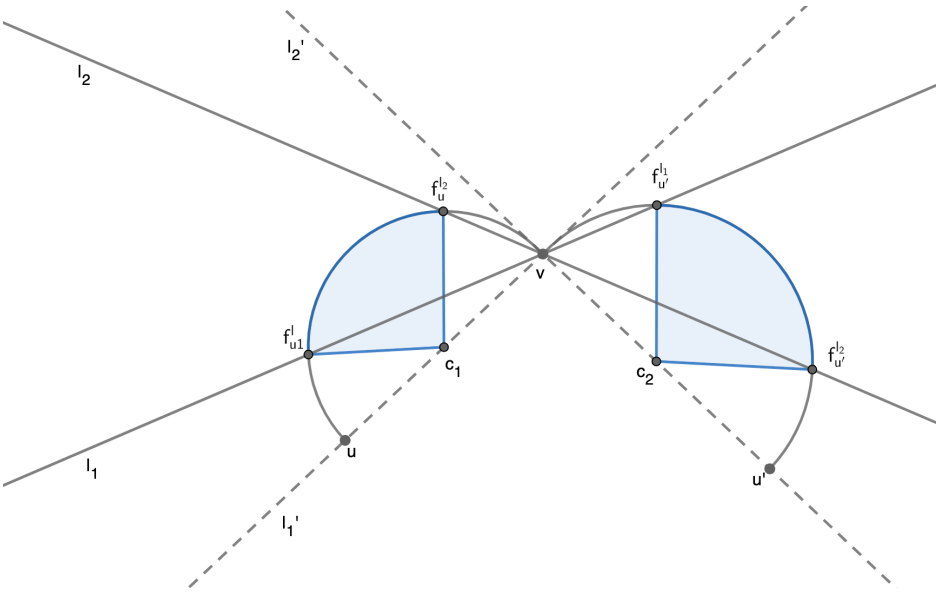

Abbildung 2.24: Die blau markierten Teile der Halbkreise sind von u bzw. u' aus sichtbar.

oder $|\pi(s, f_u^{l_2})|+|\pi(t, f_{u'}^{l_2})|$. Wenn das Maximum zwischen l_1 und l_2 liegt, dann nimmt die gemeinsame Distanz sowohl in Richtung l_1 als auch in Richtung l_2 stetig ab, wenn l vom Maximum weggedreht wird. Daher muss in diesem Fall das lokale Minimum entweder $|\pi(s, f_u^{l_1})|+|\pi(t, f_{u'}^{l_1})|$ oder $|\pi(s, f_u^{l_2})|+|\pi(t, f_{u'}^{l_2})|$ sein, oder die Distanzen sind gleich und beide sind lokale Minima. □

Der Vollständigkeit halber sei noch auf folgenden Spezialfall hingewiesen: angenommen, der Drehwinkel ρ zwischen l_1 und l_2 ist größer als 90°. Wenn l mit l_1 einen Winkel von $\rho = 90°$ bildet, tritt ein Biegungsereignis ein, bei dem der Drehpunkt v zu $\pi(s, l^-)$ hinzugefügt wird. Zwischen l_1 und diesem Biegungsereignis gilt Lemma 13 (siehe Abbildung 2.25). Ab dem Biegungsereignis ist die Länge des kürzesten gemeinsamen Pfads $\overline{\pi(s, v)} + \overline{\pi(t, u')}$, da u' von v aus sichtbar ist. Also gilt auch in diesem Fall, dass das lokale Minimum bei l_1 oder l_2 eintritt oder bei beiden gleich ist.

Wenn also die Sicht von u und u' auf die Halbkreisausschnitte zwischen zwei Ereignissen l_1 und l_2 nicht von Polygonkanten versperrt wird, muss man zur Berechnung des lokalen Minimums keine Sichtbarkeitsgeraden zwischen l_1 und l_2 betrachten. Es reicht, die gemeinsamen Distanzen $|\pi(s, l_1)| + |\pi(t, l_1)|$ und $|\pi(s, l_2)| + |\pi(t, l_2)|$ miteinander zu vergleichen. Wenn diese Distanzen

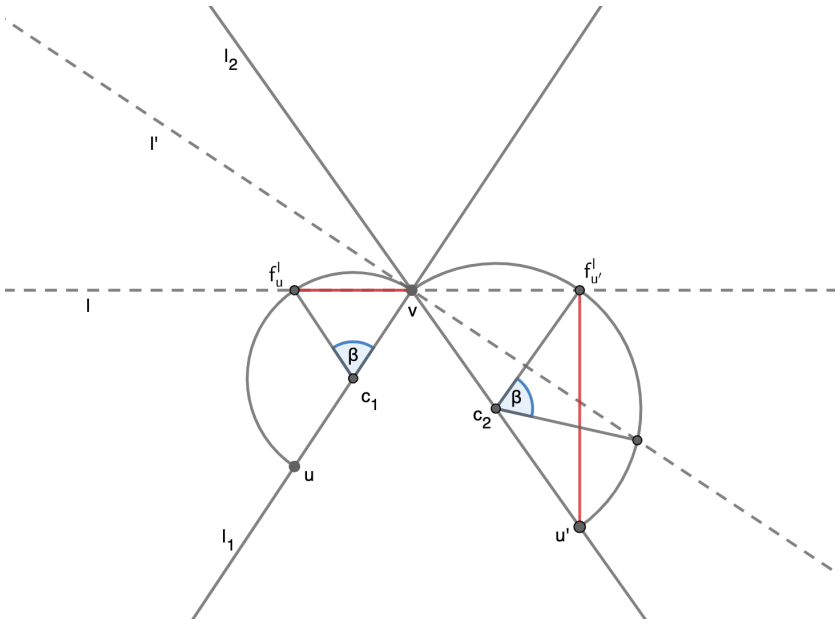

Abbildung 2.25: Bei l' tritt ein Biegungsereignis ein und v wird zu $\pi(s, l^-)$ hinzugefügt. Zwischen l_1 und l' ist $\overline{uv}$ minimal, weil für jedes l zwischen l_1 und l' die Strecke zwischen f_u^l und v kleiner ist als die Strecke zwischen $f_{u'}^l$ und u'.

gleich sind, gibt es 2 lokale Minima, ansonsten nur eines. Damit gilt folgendes Lemma:

Lemma 15. *Wenn die Halbkreise h_{uv} und $h_{u'v}$ zwischen zwei Ereignissen von u und u' aus sichtbar sind, können die lokalen Minima der min-sum Variante des Sichtbarkeitsproblems in konstanter Zeit bestimmt werden.*

2.4.3 Lokale Minima bei versperrter Sicht

In diesem Kapitel wird die Lage der lokalen Minima untersucht, wenn die Sicht von u auf h_{uv} bzw. von u' auf $h_{u'v}$ durch Polygonkanten versperrt ist. In diesem Fall gibt es kein so elegantes Ergebnis wie in Abschnitt 2.4.2, d.h. das lokale Minimum kann auf einer Sichtbarkeitsgeraden zwischen l_1 und l_2 liegen. Im folgenden wird zuerst dargestellt, wann eine Polygonkante Einfluss auf die Lage des lokalen Minimums haben kann. Dann wird die gemeinsame

Distanzfunktion bei versperrenden Polygonkanten eingeführt und auf Probleme bei der Berechnung der lokalen Minima eingegangen.

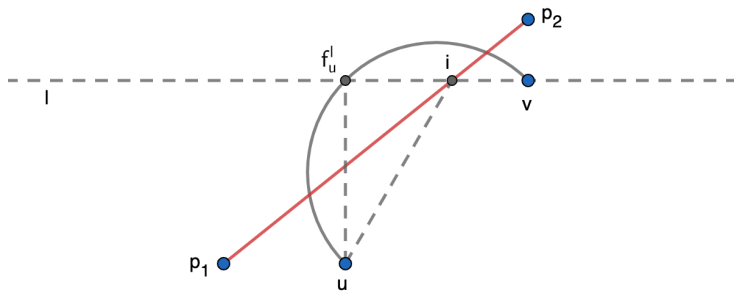

Abbildung 2.26: Der Weg von u zu Schnittpunkten von l mit der Kante $\overline{p_1p_2}$ ist immer länger als der Weg von u zu f_u^l.

Zunächst ist zu beachten, dass die Distanz zum Schnittpunkt einer fixen Sichtbarkeitsgerade mit dem Halbkreis h_{uv} bzw. $h_{u'v}$ auf jeden Fall kleiner ist als die Distanz zum Schnittpunkt derselben Sichtbarkeitsgeraden mit einer im Halbkreis verlaufenden Polygonkante. In Abbildung 2.26 etwa versperrt die rot eingezeichnete Kante $\overline{p_1p_2}$ auf einer Teilstrecke den direkten Weg von u zu den Lotfußpunkten f_u^l auf h_{uv}. i sei der Schnittpunkt von l mit der Polygonkante in diesem Bereich. Dann gilt: $|\overline{uf_u^l}| \leq |\overline{ui}|$, wobei die Gleichheit nur in den zwei Schnittpunkten des Halbkreises mit der Polygonkante gegeben ist.

Nach Lemma 14 liegt das Minimum der gemeinsamen Distanzen entweder auf l_1 oder l_2, wenn die Sicht auf die Halbkreise nicht versperrt ist. Solange die Sicht von u auf $f_u^{l_2}$ und von u' auf $f_{u'}^{l_1}$ frei ist, müssen daher keine Sichtbarkeitsgeraden zwischen l_1 und l_2 auf ein lokales Minimum untersucht werden. Die einzige Konstellation, bei der nicht offensichtlich ist, wo das Minimum liegt, ist somit diejenige, bei der die Polygonkante $f_u^{l_2}$ und/oder $f_{u'}^{l_1}$ verdeckt wie z.B. in Abbildung 2.27.

Um herauszufinden, wo das Minimum in einem derartigen Fall liegt, ist es notwendig, die Distanz zwischen u und u' und den Schnittpunkten von l mit den Polygonkanten in Abhängigkeit der Steigung α von l darzustellen. Angenommen, die Polygonkante e hat die Steigung β und die Gerade g_e, die e enthält, schneidet die y-Achse bei b. Dann wird g_e durch folgende Gleichung beschrieben:

$$y = \beta \cdot x + b \tag{2.16}$$

Die Sichtbarkeitsgerade durch v wird durch folgende Gleichung beschrieben (siehe A.4):

$$y = x \cdot \alpha + v.y - v.x \cdot \alpha \tag{2.17}$$

Die Schnittpunkte i zwischen der Polygonkante und der Sichtbarkeitsgeraden in Abhängigkeit von α sind daher durch folgende Gleichungen bestimmt:

$$i.x(\alpha) = \frac{-v.x \cdot \alpha + v.y - b}{\beta - \alpha} \tag{2.18}$$

$$i.y(\alpha) = \frac{(-b - \beta \cdot v.x) \cdot \alpha + \beta \cdot v.y}{\beta - \alpha} \tag{2.19}$$

Die Distanz zwischen u und i ist dann durch die folgende Gleichung bestimmt, die nur von der Steigung der Sichtbarkeitsgeraden α abhängt:

$$d_{ui}(\alpha) = \sqrt{(u.x - i.x)^2 + (u.y - i.y)^2} \tag{2.20}$$

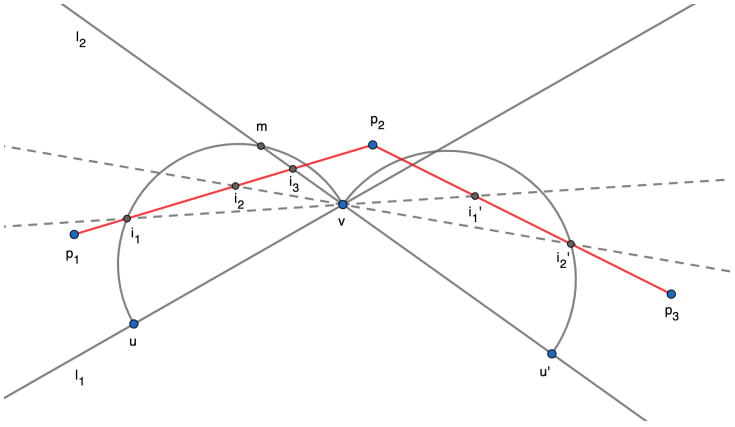

Abbildung 2.27: Die Polygonkanten zwischen p_1, p_2 und p_3 verdecken die Sichtbarkeit der Halbkreise. Das lokale Minimum ohne versperrende Kanten würde bei m und u' liegen.

Um die gemeinsame Distanz zu einer Sichtbarkeitsgerade zu bestimmen, kommt es darauf an, ob sowohl h_{uv} als auch $h_{u'v}$ verdeckt sind, oder nur

einer der beiden Halbkreise. Wenn in Abbildung 2.27 l von l_2 in Richtung l_1 gedreht wird, dann ist zwischen l_2 und der Geraden, die i_2 und i_2' enthält, nur die Sicht von u auf h_{uv} verdeckt. In diesem Bereich ist die gemeinsame Distanz zu l:

$$d(\alpha) = \sqrt{(u.x - i.x)^2 + (u.y - i.y)^2} + \frac{|(u'.x - v.x) \cdot \alpha + (v.y - u'.y)|}{\sqrt{(\alpha^2 + 1)}} \tag{2.21}$$

Zwischen der Geraden, die i_2 und i_2' enthält, und derjenigen, die i_1 und i_1' enthält, ist sowohl die Sicht von u auf h_{uv} als auch die Sicht von u' auf $h_{u'v}$ verdeckt. In diesem Bereich kann der Schnittpunkt von l^+ mit der Polygonkante $\overline{p_2p_3}$ (bezeichnet als i') durch eine entsprechende Anpassung von Gleichung 2.20 in Abhängigkeit von α berechnet werden. Die gemeinsame Distanz zu l ist daher:

$$d(\alpha) = \sqrt{(u.x - i.x)^2 + (u.y - i.y)^2} + \sqrt{(u'.x - i'.x)^2 + (u'.y - i'.y)^2} \tag{2.22}$$

Bei einer weiteren Drehung von l ist nur die Sicht von u' auf $h_{u'v}$ versperrt und Gleichung 2.21 kann entsprechend verwendet werden.

Es stellt sich die Frage, ob die Lage des Minimum bei verdeckenden Polygonkanten besondere Kennzeichen haben muss. Etwa, dass das Minimum nur dort liegen kann, wo eine Polygonkante beginnt oder aufhört, die Sicht auf einen Halbkreis zu verdecken (z.B. bei i_1 oder i_2' in Abbildung 2.27), oder beim Schnittpunkt von l_1 bzw. l_2 mit der Polygonkante (z.B. bei i_3 in 2.27).

Die Antwort ist, dass dies nicht immer der Fall ist. Abbildung 2.28 zeigt eine Konstellation, wo das Minimum zwischen dem Schnittpunkt der Kante mit h_{uv} und dem Schnittpunkt der Kante mit l_1 liegt (für die genauen Werte, die in diesem Beispiel verwendet wurden, siehe Appendix C). Dabei wird angenommen, dass die Sicht von u' auf $h_{u'v}$ frei ist. Bei i_1 ist die Steigung von l ungefähr -0.386, bei i_2 ist sie -0.4. Das Minimum liegt zwischen diesen Punkten, bei einer Steigung von ungefähr -0.397. Der andere Kandidat für das lokale Minimum ist die Sichtbarkeitsgerade durch i_3. Hier ist die gemeinsame Distanz aber etwas höher.

Bei meinen Berechnungen von lokalen Minima in verschiedenen Konstellationen ist mir aufgefallen, dass ein Fall wie in Abbildung 2.28 selten auftritt. Fast immer liegt das lokale Minimum beim Schnittpunkt der Polygonkante mit dem Halbkreis oder mit l_1, oder bei l_2. Minima scheinen nur dann bei

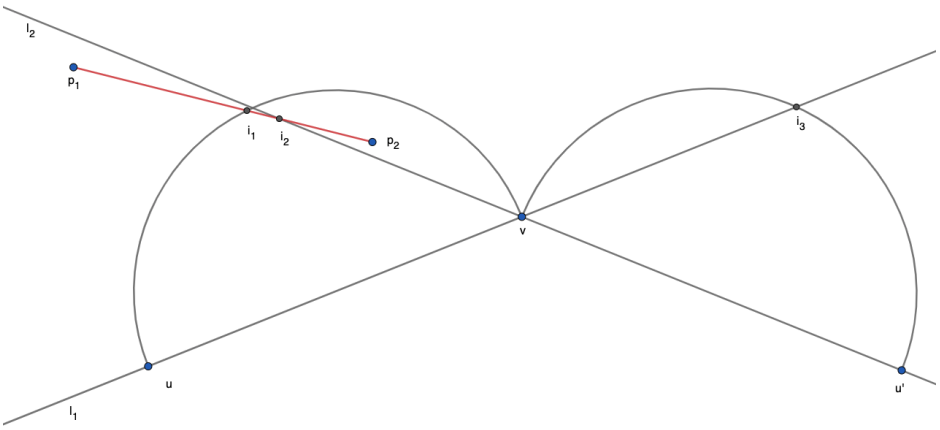

Abbildung 2.28: Das lokale Minimum liegt in dieser Konstellation bei einer Sichtbarkeitsgeraden durch einen Punkt zwischen i_1 und i_2.

anderen Punkten der Polygonkante zu liegen, wenn die Polygonkante sehr nah am optimalen Minimum (also beim Schnittpunkt von h_{uv} mit l_2, wenn $|\overline{uv}| < |\overline{u'v}|$) vorbeiführt. Außerdem sind sie nur geringfügig kleiner als die gemeinsame Distanz beim Schnittpunkt der Polygonkante mit h_{uv} bzw. mit l_2. Ich habe versucht, genauere Abschätzungen dafür zu finden, wie die Polygonkante verlaufen muss, damit das Minimum zwischen i_1 und i_2 liegen kann wie in Abbildung 2.28, bzw. um wie viel das Minimum im Inneren der Polygonkante kleiner sein kann als bei i_1 oder i_2. Dies ist mir aber nicht gelungen, so dass es bei der Berechnung der lokalen Minima immer notwendig ist, auch Sichtbarkeitsgeraden zwischen den Schnittpunkten zu überprüfen.

Wenn es verdeckende Polygonkanten gibt, müsste man zur genauen Bestimmung der lokalen Minima die Distanzfunktionen 2.21 bzw. 2.22 nach α ableiten. Potentielle Kandidaten für lokale Minima wären dann die Nullstellen dieser Ableitungen. Ich habe die Ableitungen anhand von SageMath [20] berechnen lassen. Dabei stellte sich heraus, dass die ersten Ableitungen der Distanzfunktionen und die Ausdrücke für ihre Nullstellen so kompliziert sind, dass sie sich für eine praktische Implementierung des Algorithmus kaum eignen. Ich kann nicht einmal mit Sicherheit sagen, ob das Minimum zwischen l_1 und l_2 eindeutig bestimmt ist, oder ob es mehrere Minima geben kann.[7]

[7]Um dies herauszufinden, müsste man die Distanzfunktionen 2.21 und 2.22 genauer untersuchen, was aber sowohl die mir zur Verfügung stehende Zeit als auch meine mathe-

Daher benutze ich in meiner Implementierung einen approximativen Ansatz, um die lokalen Minima bei versperrenden Polygonkanten zu bestimmen. Der Approximationsalgorithmus ist nicht besonders ausgefeilt[8]: zwischen l_1 und l_2 werden in bestimmten Abständen Sichtbarkeitsgeraden l' gelegt. Dann werden die Distanzen $|\pi(u, l')|$ und $|\pi(t, l')|$ berechnet. Während dieser Berechnungen werden die angetroffenen lokalen Minima abgespeichert. Wenn die lokalen Minima innerhalb eines Bereichs gleich sind, wird das durch den Algorithmus berücksichtigt.

Wenn die Anzahl der Iterationen beim gewählten approximativen Ansatz durch eine konstante Zahl begrenzt ist, gilt für die Bestimmung der lokalen Minima das folgende Lemma:

Lemma 16. *Wenn mindestens einer der Halbkreise h_{uv} und $h_{u'v}$ zwischen zwei Ereignissen von u bzw. u' aus nicht sichtbar ist, können die lokalen Minima der min-sum Variante des Sichtbarkeitsproblems in konstanter Zeit bestimmt werden.*

2.5 Lage des Minimums in der min-max Version des Algorithmus

In diesem Abschnitt wird gezeigt, dass in konstanter Zeit entschieden werden kann, ob die Lösung der min-max Version des Sichtbarkeitsproblems zwischen zwei in der Ereignisliste aufeinanderfolgenden Ereignissen bzw. bei einem dieser Ereignisse liegt. Wenn die Entscheidung positiv ausfällt, kann auch die Berechnung, bei welcher Sichtbarkeitsgeraden die Lösung eintritt, in konstanter Zeit berechnet werden. Dabei ist v wieder der aktuelle Drehpunkt, u der letzte Knoten von $\pi(s, l)$, der ein Knoten von P ist, und u' der letzte Knoten von $\pi(t, l)$, der ein Knoten von P ist. l_1 und l_2 sind zwei aufeinanderfolgende Ereignisse in der Ereignisliste.

matischen Fähigkeiten übersteigt.

[8]Ich bin mir sicher, dass es bessere Algorithmen zur Approximation gibt, allerdings ist mir in der verfügbaren Zeit nichts besseres eingefallen

Bei der Lösung des min-max Problems helfen die folgenden beiden Lemmata:

Lemma 17. *l_1 und l_2 seien 2 aufeinanderfolgende Ereignisse in der Ereignisliste. Dann wächst die Distanz $|\pi(s,l)|$ während der Drehung der Sichtbarkeitsgeraden l von l_1 nach l_2 kontinuierlich, bis der maximale Distanzzuwachs $|\overline{uv}|$ erreicht ist.*

Beweis. Da die Distanz $|\pi(s,u)|$ während der Drehung von l konstant bleibt, ist es nur nötig, die Distanz zwischen u und l zu betrachten. Zunächst beachte man, dass ein Biegungsereignis spätestens dann eintritt, wenn der Drehwinkel ρ zwischen l und l_1 90° erreicht. Ab diesem Biegungsereignis ist der kürzeste Weg von u nach l $\overline{uv}$. Diese Distanz ändert sich auch bei weiterer Drehung von l nicht, da v auf der sich drehenden Sichtbarkeitsgeraden liegt, so dass der maximale Distanzzuwachs $|\overline{uv}|$ ist.

Im Bereich $0 \leq \rho \leq 90°$ muss unterschieden werden, ob die Sicht von u auf den Halbkreis h_{uv} frei ist oder nicht. Falls die Sicht frei ist, gilt für die Distanz zwischen u und dem Lotfußpunkt f_u^l:

$$|\overline{uf_u^l}| = sin(\rho) \cdot |\overline{uv}| \tag{2.23}$$

Da $|\overline{uv}|$ während der Drehung gleicht bleibt, und $sin(\rho)$ für $0 \leq \rho \leq 90°$ kontinuierlich wächst, ist das Lemma für diesen Fall bewiesen.

Jetzt ist noch der Fall zu betrachten, dass die Sicht auf h_{uv} nicht frei ist. Dabei ist zu unterscheiden, ob u der Anfangsknoten der versperrenden Polygonkante ist oder nicht. Falls dies der Fall ist, verläuft der kürzeste Weg $\pi(u,l)$ auf der Polygonkante, und zwar von u zu Punkten auf der Polygonkante, die sich bei Drehung von l immer weiter von u entfernen. In diesem Fall nimmt $\pi(s,l)$ auf jeden Fall zu.[9]

Im anderen Fall ist u nicht der Anfangsknoten der versperrenden Polygonkante. Damit in diesem Fall eine Kante e die Sicht auf h_{uv} versperrt, muss sie bei einem Lotfußpunkt $f_u^{l'}$ von u auf einer Sichtbarkeitsgeraden l' in den Halbkreis eintreten. Wenn man die Halbebenen betrachtet, die durch die Gerade durch v und $f_u^{l'}$ aufgespannt werden, muss außerdem gelten, dass e im weiteren Verlauf in der Halbebene liegt, die u nicht enthält. Denn ansonsten müssten die Sichtbarkeitsgeraden von v zu e im Polygonäußeren verlaufen.

[9]Dieser Fall ist z.B. in Abbildung 2.13 illustriert: bei l_1 tritt ein Biegungsereignis ein, bei dem p der letzte Knoten vor l wird. Bei weiterer Drehung von l in Richtung t verläuft die letzte Kante des Pfads von p zu l auf der Polygonkante, deren Anfangsknoten p ist.

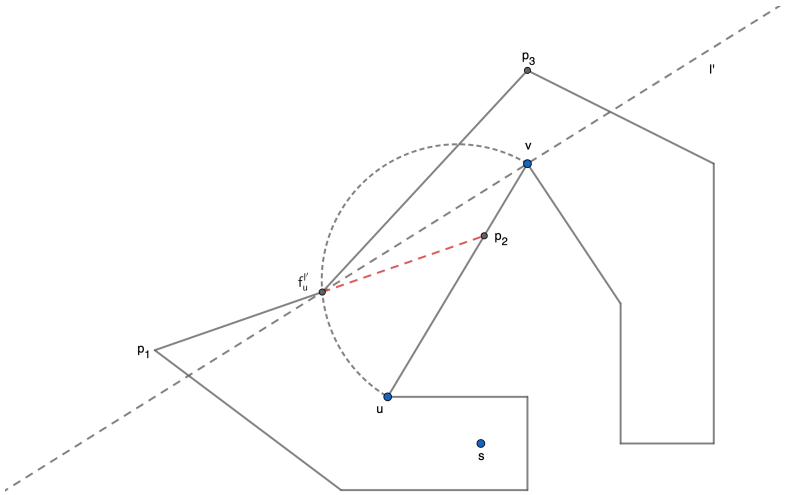

Abbildung 2.29: Nur Kanten, die oberhalb von $\overline{f_u^{l'}v}$ verlaufen, sind für die Berechnung des minimalen Pfades relevant.

Dies ist in Abbildung 2.29 illustriert. Kanten, die mit einer Steigung wie $\overline{p_1p_2}$ verlaufen, spielen bei der Berechnung des Minimums keine Rolle.

Für eine Kante e, die ab Eintritt in h_{uv} in der Halbebene verläuft, die u nicht enthält, gilt, dass der Lotfußpunkt von u auf der Geraden durch e außerhalb von h_{uv} liegt. Das bedeutet, dass sich der Schnittpunkt von l mit e, der gleichzeitig der Endpunkt von $\pi(s, l)$ ist, während der Drehung von l von diesem Lotfußpunkt wegbewegt. Nach Lemma 3 steigt daher die Länge von $\pi(u, l)$ kontinuierlich an.

Wenn e den Halbkreis h_{uv} verläßt, tritt ein degeneriertes Biegungsereignis ein. Das bedeutet, dass die Sicht auf h_{uv} wieder frei ist, und der erste Teil dieses Beweises angewendet werden kann.

□

Lemma 17 gilt auch, wenn s und t sowie die Drehrichtung von l ausgetauscht werden und muß daher für $\pi(t, l)$ nicht wiederholt werden. Mithilfe dieses Lemmas kann das folgende Lemma bewiesen werden:

Lemma 18. *Während der Drehung der Sichtbarkeitsgeraden l von s nach t wächst die Länge des Pfads $\pi(s, l)$, außer in Regionen, in denen der Endpunkt*

von $\pi(s,l)$ der aktuelle Drehpunkt ist. Während solche Regionen überstreift werden, bleibt die Länge des Pfades gleich.

Beweis. Nach Lemma 17 ist der maximale Distanzzuwachs zwischen zwei Ereignissen $|\overline{uv}|$. Bei einer weiteren Drehung der Sichtbarkeitsgeraden endet $\pi(s,l)$ beim aktuellen Drehpunkt und die Länge des Pfades bleibt bis zum nächsten Pfadereignis gleich.

In allen anderen Fällen nimmt $|\pi(s,l)|$ zwischen zwei Ereignissen kontinuierlich zu. Deshalb muss nur noch bewiesen werden, dass die Länge von $\pi(s,l)$ bei einem Ereignis nicht plötzlich kleiner wird. Hier ist zu unterscheiden, ob sich bei einem Ereignis die bisherigen Knoten von $\pi(s,l)$ ändern: Wenn kein neuer Knoten zu $\pi(s,l)$ hinzu- oder wegkommt, ändert sich die Länge von $\pi(s,l)$ bei dem Ereignis nicht. Wenn ein neuer Knoten zu $\pi(s,l)$ hinzukommt, kann die Länge von $\pi(s,l)$ nur größer werden. Wenn ein Knoten u von $\pi(s,l)$ entfernt wird und sein Vorgänger u^- der letzte Knoten vor l wird, ist zu beachten, dass bei dem Ereignis gilt: $u \in \overline{u^-l}$, so dass sich auch bei einem solchen Ereignis die Länge von $\pi(s,l)$ nicht ändert. □

Lemma 18 gilt auch, wenn s und t sowie die Drehrichtung von l ausgetauscht werden. Da der Drehpunkt v nicht der Endknoten von beiden Pfaden sein kann, nimmt entweder $|\pi(s,l)|$ bei Drehung der Sichtbarkeitsgeraden von s nach t zu, während $|\pi(t,l)|$ abnimmt oder gleich bleibt, oder $|\pi(s,l)|$ bleibt gleich, während $|\pi(t,l)|$ abnimmt. Daher liegt eine Lösung des min-max Problems bei der Sichtbarkeitsgeraden l, für die gilt: $|\pi(s,u)| + |\pi(u,l)| = |\pi(t,u')| + |\pi(u',l)|$. Zwischen zwei Ereignissen l_1 und l_2 kann die Gleichheit nur dann eintreten, wenn folgendes gilt:

$$|\pi(s,u)| \leq |\pi(t,u')| + |\pi(u',l_1)| \tag{2.24}$$

$$|\pi(t,u')| \leq |\pi(s,u)| + |\pi(u,l_2)| \tag{2.25}$$

Um die genaue Lage der Sichtbarkeitsgeraden zu bestimmen, bei der die Lösung des min-max Problems liegt, muss eine Fallunterscheidung getroffen werden (siehe Abbildung 2.30): im ersten Fall, endet $\pi(s,l)$ oder $\pi(t,l)$ beim aktuellen Drehpunkt v. Wenn in diesem Fall die Gleichungen 2.24 und 2.25 zutreffen, ist die Lösung des min-max Problems einfach zu bestimmen: angenommen, u fällt mit dem Drehpunkt v zusammen. Dann ist der Lösungspfad, der bei s beginnt, $\pi(s,v)$. Für den Punkt t ist die Lösung nicht eindeutig bestimmt. Die Menge der Lösungen besteht aus den Pfaden, die von t zu u' führen und von dort zu Punkten, von denen aus v sichtbar ist und wo die

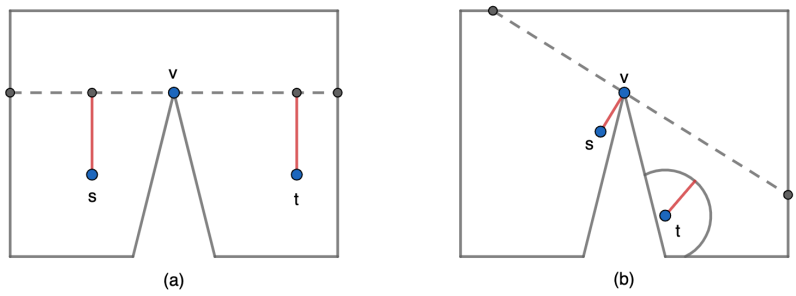

Abbildung 2.30: In Abbildung (a) befindet sich die Lösung des min-max Problems auf der eingezeichneten Sichtbarkeitsgeraden, in Abbildung (b) kann der Pfad, der bei t beginnt, bei jedem Punkt innerhalb des eingezeichneten Kreisausschnitts enden.

Gesamtlänge des Pfades maximal so lang ist wie $|\pi(s,v)|$. Die Menge der Endpunkte t_l dieser Pfade wird als M_l bezeichnet und kann formal folgendermaßen beschrieben werden:

$$M_l = \{t_l | t_l \in P, \overline{t_l v} \in P, \overline{u' t_l} \in P, |\pi(t,u')| + |\overline{u' t_l}| \leq |\pi(s,v)|\} \tag{2.26}$$

Entsprechendes gilt, wenn u' mit v zusammenfällt.

In allen anderen Fällen, d.h. wenn weder u noch u' mit v zusammenfallen, müssen die Distanzfunktionen genauer betrachtet werden. Die Länge von $\pi(s,u)$ bzw. $\pi(t,u')$ ist dabei einfach die Summe der Kantenlängen. Für die Distanzen $|\pi(u,l)|$ und $|\pi(u',l)|$ gilt nach Lemma 17, dass eine der Distanzen bei Änderungen von α kontinuierlich zunimmt, während die andere kontinuierlich abnimmt. Das bedeutet, dass die Steigung α, bei der die Distanzen gleich werden, eindeutig bestimmt ist.

Um die Distanzfunktionen für $|\pi(u,l)|$ und $|\pi(u',l)|$ angeben zu können, muss unterschieden werden, ob die Sicht auf h_{uv} und $h_{u'v}$ frei ist oder nicht. Wenn die Sicht auf beide Halbkreise frei ist, können die Distanzfunktionen für $|\pi(u,l)|$ und $|\pi(u',l)|$ anhand der Erläuterungen in Appendix A leicht hergeleitet werden. In diesem Fall ist folgende Gleichung für α zu lösen:

$$|\pi(s,u)| + \frac{|(u.x - v.x) \cdot \alpha + v.y - u.y|}{\sqrt{\alpha^2 + 1}} = |\pi(t,u')| + \frac{|(u'.x - v.x) \cdot \alpha + v.y - u'.y|}{\sqrt{\alpha^2 + 1}} \quad (2.27)$$

Unter der Annahme, dass l_1 und l_2 so liegen, wie in Kapitel 2.4.1 beschrieben, können mit Lemma 12 die Absolutzeichen in Gleichung 2.27 weggelassen werden. Wenn man außerdem konstante Werte durch Konstantennamen ersetzt, also $a = |\pi(s,u)| - |\pi(t,u')|$, $b = u'x - u.x$ und $c = u.y - u'.y$ kann Gleichung 2.27 in folgende Form gebracht werden:

$$a = \frac{b \cdot \alpha + c}{\sqrt{\alpha^2 + 1}} \quad (2.28)$$

Wenn man beide Seiten quadriert und alle Terme auf die linke Seite bringt, erhält man die folgende quadratische Gleichung:

$$(a^2 - b^2) \cdot \alpha^2 - 2 \cdot b \cdot c \cdot \alpha + a^2 - c^2 = 0 \quad (2.29)$$

Gleichung 2.29 kann anhand der Formel für quadratische Gleichungen nach α gelöst werden. Diejenige der beiden Lösungen, für die die Sichtbarkeitsgerade zwischen l_1 und l_2 liegt, ist die gesuchte Lösung.

In allen anderen Fällen, d.h. wenn die Sicht auf die Halbkreise nicht auf beiden Seiten frei ist, sind die zu lösenden Gleichungen komplizierter. Falls die Sicht auf h_{uv} oder $h_{u'v}$ versperrt ist, muß mit Gleichung 2.20 eine der folgenden Gleichungen für α gelöst werden:

$$|\pi(s,u)| + \frac{|(u.x - v.x) \cdot \alpha + v.y - u.y|}{\sqrt{\alpha^2 + 1}} = |\pi(t,u')| + \sqrt{(u'.x - i'.x)^2 + (u'.y - i'.y)^2} \quad (2.30)$$

$$|\pi(s,u)| + \sqrt{(u.x - i.x)^2 + (u.y - i.y)^2} = |\pi(t,u')| + \frac{|(u'.x - v.x) \cdot \alpha + v.y - u'.y|}{\sqrt{\alpha^2 + 1}} \quad (2.31)$$

Im letzten Fall, d.h. wenn die Sicht auf beide Halbkreise versperrt ist, muß folgende Gleichung für α gelöst werden:

$$|\pi(s,u)| + \sqrt{(u.x - i.x)^2 + (u.y - i.y)^2} = \\ |\pi(t,u')| + \sqrt{(u'.x - i'.x)^2 + (u'.y - i'.y)^2} \quad (2.32)$$

Ähnlich wie bei der min-sum Variante ist die exakte Lösung des min-max Problems bei versperrter Sicht komplizierter als bei freier Sicht und die Gleichungen 2.30, 2.31 und 2.32 können nicht so leicht in eine einfach zu lösende Form gebracht werden. Ich habe mehrere Stunden damit verbracht, zu versuchen, die Gleichungen nach α zu lösen, aber es ist mir nicht gelungen. Daher lasse ich die Frage, ob bzw. wie man eine exakte Lösung in diesen Fällen finden kann, unbeantwortet.

Anstatt eine exakte Lösung zu finden, habe ich bei der praktischen Implementierung des Algorithmus beschlossen, die Lösung in allen Fällen zu approximieren. Da bekannt ist, dass die Lösung eindeutig bestimmt ist, kann man z.B. einen Divide-And-Conquer Algorithmus anwenden: man wählt auf einer Seite den Ausschnitt des Polygonrandes, der zwischen l_1 und l_2 liegt und legt eine Sichtbarkeitsgerade l' durch die Mitte. Dann berechnet man die Distanzen $|\pi(s,l')|$ und $|\pi(t,l')|$. Wenn gilt $|\pi(s,l')| < |\pi(t,l')|$, teilt man als nächstes das Randstück, das von l' und l_2 begrenzt wird, ansonsten dasjenige, das von l' und l_1 begrenzt wird. Dies kann solange durchgeführt werden, bis die Distanzen gleich sind, bzw. sich nur noch um einen kleinen, vorgegebenen Wert unterscheiden. Der Einfachheit halber habe ich in der praktischen Implementierung einen derartigen Algorithmus für alle Gleichungen verwendet.

Bei den bisher beschriebenen Fällen tritt die Gleichheit der Distanzen *zwischen* zwei Ereignissen ein. Es ist allerdings auch möglich, dass die Gleichheit *bei* einem Ereignis eintritt, und zwar, wenn bei dem Ereignis ein Knoten zu $\pi(s,l)$ oder $\pi(t,l)$ hinzukommt bzw. verlorengeht, und einer der Pfade dadurch abrupt länger oder kürzer wird, je nachdem von welcher Seite die Sichtbarkeitsgerade sich dem Ereignis annähert. Eine derartige abrupte Änderung der Länge eines Pfades kann bei zwei Arten von Ereignissen eintreten: bei Randereignissen, die ein Biegungsereignis auslösen, oder bei Pfadereignissen, bei denen sich die Drehrichtung der Sichtbarkeitsgeraden ändert und die dadurch ein Biegungsereignis auslösen.

Abbildung 2.31 illustriert den Fall für Randereignisse: bei l tritt ein Randereignis ein, das ein Biegungsereignis auslöst. Bei weiter Drehung von

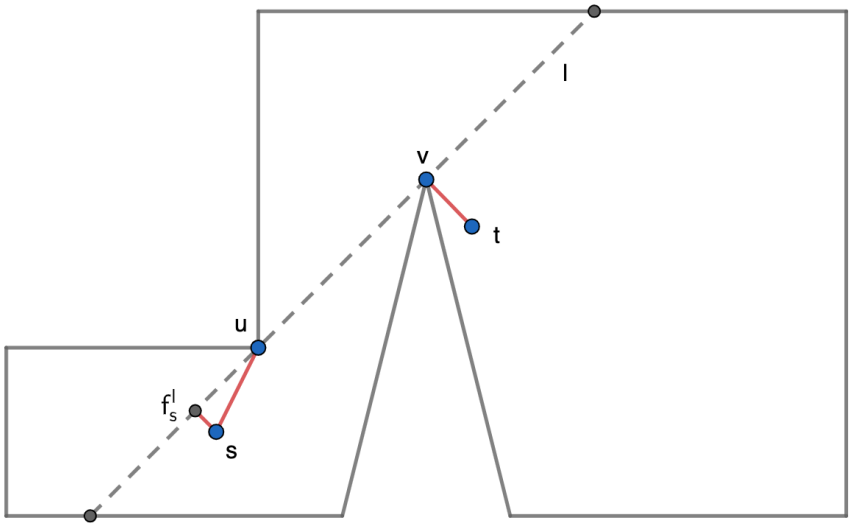

Abbildung 2.31: Lösung des min-max Problems bei einem Ereignis

l in Richtung t wird der Punkt u zu $\pi(s,l)$ hinzugefügt. Das bedeutet, dass die Distanz $|\pi(s,l)|$ bei dem Ereignis abrupt von $|\overline{sf_s^l}|$ auf $|\overline{su}|$ wächst. Es gilt: $|\overline{sf_s^l}| < |\overline{vt}| < |\overline{us}|$, so dass die Gleichheit der Distanzen bei einem Endpunkt von $\pi(s,l)$ eintritt, der auf $\overline{f_s^l u}$ liegt. Bei der Berechnung der min-max Lösungen ist dieser Sonderfall zu berücksichtigen.[10] Auch bei abrupten Änderungen der Distanzen ist es möglich, dass die Lösung des min-max Problems nicht eindeutig definiert ist. Die Menge der Lösungen ist dann wie in 2.26 definiert.

Für die Komplexität der Entscheidung, ob bzw. wo die min-max Lösung zwischen zwei Ereignissen liegt, gilt folgendes Lemma:

Lemma 19. *Die Entscheidung, ob eine Lösung des min-max Problems zwischen zwei in der Ereignisliste aufeinanderfolgenden Ereignissen l_1 und l_2 bzw. bei einem der Ereignisse liegt, kann in konstanter Zeit getroffen werden. Wenn die Entscheidung positiv ausfällt, können die Sichtbarkeitsgerade l, bei der die Lösung liegt, und die Endpunkte von $\pi(s,l)$ und $\pi(t,l)$ in konstanter*

[10] Abbildung 2.11 (b) illustriert den Fall, wo sich die Längen der Pfade bei einem Pfadereignis abrupt ändern. Auch in diesem Fall muss überprüft werden, ob die Gleichheit der Distanzen bei dem Ereignis eintritt.

Zeit gefunden werden.

Die Entscheidung, ob die Lösung eindeutig bestimmt ist, kann in konstanter Zeit getroffen werden. Wenn es eine Menge von Lösungen gibt, kann in konstanter Zeit berechnet werden, wie weit Punkte dieser Menge von u bzw. u' entfernt sein dürfen.

Beweis. Um zu entscheiden, ob die Lösung zwischen l_1 und l_2 liegt, müssen $|\pi(t,u')| + |\pi(u',l_1)|$ und $|\pi(s,u)| + |\pi(u,l_2)|$ berechnet werden. Bei der Berechnung des Wegs von u' zu l_1 bzw. von u nach l_2 ist zu unterscheiden, ob die Sicht auf die Lotfußpunkte $l_{u'}^{l_1}$ und $l_u^{l_2}$ frei ist oder nicht. Das ist nach Berechnung der degenerierten Biegungsereignisse bekannt, so dass die Berechnung der Länge der Pfade in konstanter Zeit möglich ist.

Wenn die Prüfung ergibt, dass die Lösung zwischen l_1 und l_2 liegt, kann anhand der Distanzfunktionen für $\pi(u,l)$ und $\pi(u',l)$ die genaue Lage der Lösung bestimmt werden, d.h. die Sichtbarkeitsgerade bei der gilt: $|\pi(s,u)| + |\pi(u,l)| = |\pi(t,u')| + |\pi(u',l)|$. Wenn der Halbkreis h_{uv} von u und $h_{u'v}$ von u' aus sichtbar ist, muss dazu eine quadratische Gleichung gelöst werden. In allen anderen Fällen sind die Distanzfunktionen komplizierter, so dass ein approximativer Ansatz zur Bestimmung der Lösung verwendet werden kann. Solange die Anzahl der Iterationen dabei durch eine konstante Zahl begrenzt ist, kann die Lösung in konstanter Zeit gefunden werden.

Falls $\pi(s,l)$ in der Region, in der die Lösung liegt, beim Drehpunkt v endet, kann es für $\pi(t,l)$ mehr als eine Lösung geben. Dies kann überprüft werden indem die Differenz $\pi(s,v) - \pi(t,u')$ berechnet wird. Wenn die Differenz positiv ist, kann $\pi(t,l)$ bei Punkten enden, die um den Wert dieser Differenz von u' entfernt liegen und für die v sichtbar ist. Entsprechendes gilt wenn $\pi(t,l)$ bei v endet.[11]

Um zu überprüfen, ob die Lösung *bei* einem Ereignis liegt, müssen die Längen von $\pi(s,l)$ und $\pi(t,l)$ vor und nach dem Gewinn bzw. Verlust von Knoten berechnet und miteinander verglichen werden. Insgesamt müssen also je 2 Pfadlängen für $\pi(s,l)$ und $\pi(t,l)$ berechnet werden, was in konstanter Zeit möglich ist. Auch die Entscheidung, ob die Lösung eindeutig bestimmt

[11] Um die Menge der Lösungen genau zu bestimmen, müsste man eine Scheibe um u bzw. u' legen und diejenigen Ausschnitte der Scheibe berechnen, die innerhalb der Schnittmenge der Sichtbarkeitszellen von v und u' (bzw. u) liegen. Das ist kein triviales Problem, so dass ich auf diese Berechnung nicht genauer eingehe. In meiner Implementierung wird einfach der Radius der Scheibe, innerhalb der die Lösungen liegen können, eingezeichnet. Dabei werden die Ränder des Polygons und der Sichtbarkeitszellen nicht berücksichtigt.

ist, kann in konstanter Zeit getroffen werden: dazu muss lediglich die Differenz der zwei minimalen Pfade bestimmt werden. Wenn die Differenz ungleich 0 ist, gibt es für den kürzeren Pfad nicht nur eine Lösung, sondern eine Menge an Lösungen, die anhand der Differenz berechnet werden kann. □

2.6 Komplexität des paarweisen Sichtbarkeitsproblems

Für die Berechnung der lokalen Minima zwischen zwei Ereignissen l_1 und l_2, die in der vollständigen Ereignisliste aufeinander folgen, gilt folgendes Lemma, das die Lemmata 15, 16 und 19 zusammenfasst:

Lemma 20. *Die lokalen min-sum Minima, die zwischen bzw. auf l_1 und l_2 liegen, können in konstanter Zeit berechnet werden. Ebenso kann in konstanter Zeit entschieden werden, ob eine Lösung des min-max Problems zwischen bzw. auf l_1 und l_2 liegt. Wenn die Entscheidung positiv ausfällt, kann die Lage der Lösung in konstanter Zeit bestimmt werden.*

Beweis. Nach Berechnung aller Ereignisse sind bei jedem Ereignis die Punkte u, u' und v sowie $|\pi(s,u)|$ und $|\pi(t,u')|$ gespeichert. Außerdem weiß man aufgrund der Berechnung der Biegungsereignisse, ob h_{uv} und $h_{u'v}$ von u bzw. u' aus sichtbar sind, und ob $\pi(s,u)$ oder $\pi(t,u')$ bei l_1 oder l_2 Knoten verlieren oder dazugewinnen. Diese Daten sind ausreichend, um zu entscheiden, welche Distanzfunktionen zwischen l_1 und l_2 gelten, bzw. ob eine Lösung bei einem der Ereignisse eintritt. Somit ergibt sich das Lemma, indem entweder Lemma 15, 16 oder 19 angewendet wird. □

Damit kann das theoretische Hauptergebnis dieser Arbeit in folgendem Theorem zusammengefasst werden:

Theorem 1. *Gegeben sei ein einfaches Polygon P mit n Knoten, und Punkte $s,t \in P$. Für ein Punktepaar (s^*,t^*) gelte:*

1. $\overline{s^*t^*} \subset P$
2. $|\pi(s,s^*)| + |\pi(t,t^*)|$ *oder* $max(|\pi(s,s^*)|,|\pi(t,t^*)|)$ *ist minimal*

Dann können alle Punktepaare, bei denen die beiden Bedingungen gelten, und die dazu gehörenden Pfade in $\mathcal{O}(n)$ Zeit berechnet werden.

Beweis. Für die Berechnung der Ereignisse werden der kürzeste Pfad zwischen s und t sowie die Shortest Path Maps von s und t benötigt. Diese können aus der Triangulation von P in $\mathcal{O}(n)$ Zeit berechnet werden. Die Triangulation selbst kann in $\mathcal{O}(n)$ Zeit berechnet werden. In Kapitel 2.3 wurde erläutert, wie alle Ereignisse in der korrekten Reihenfolge in $\mathcal{O}(n)$ berechnet und in der Ereignisliste gespeichert werden können. Nach der Berechnung der Ereignisse muss zwischen je zwei aufeinanderfolgenden Ereignissen eine Lösung gesucht werden, wie in Lemma 20 beschrieben. Da es insgesamt $\mathcal{O}(n)$ Ereignisse gibt und die Lösung zwischen zwei Ereignissen in konstanter Zeit gefunden werden kann, wird dafür insgesamt $\mathcal{O}(n)$ Zeit benötigt. □

Kapitel 3

Implementierung des Algorithmus in C++

3.1 Vorbemerkungen

Ich habe den Algorithmus zur Lösung des Sichtbarkeitsproblems, der in Kapitel 2 beschrieben wird, als ein Qt Projekt [19] unter Zuhilfenahme von CGAL [5] implementiert. Das Qt Projekt hat drei Unterprojekte:

1. **Eine Bibliothek**, die den eigentlichen Algorithmus implementiert. Die Bibliothek benötigt CGAL.
2. **Eine Qt Application**, die das Sichtbarkeitsproblem visualisiert.
3. **Ein Qt Testprojekt**, das die Bibliothek testet.

Das gesamte Projekt ist als öffentliches Repository auf github unter der folgenden url zu finden:

https://github.com/klauste/ShortestPathToVisibility

Ich habe das Programm auf macOS Catalina und Ubuntu 18.04 kompiliert und getestet. Instruktionen zum Kompilieren des Projekts sind im Anhang D und auf github zu finden. Außerdem habe ich eine dmg Datei erzeugt, die auf macOS Catalina und macOS Mojave ausführbar ist. Möglicherweise ist sie auch mit anderen macOS Versionen ausführbar, allerdings habe ich das nicht getestet.

Wenn man den Quellcode kompilieren will, empfehle ich, das auf Ubuntu zu tun, da die Installation von CGAL, Qt und den anderen benötigten Bibliotheken einfacher als auf dem Mac ist. Falls Ubuntu nicht das vorhandene Betriebssystem sein sollte, kann es in einer Virtuellen Maschine installiert werden.

Eine kompilierbare Version des Programms für andere Betriebssysteme zu erstellen, sollte nicht schwierig sein, wenn die notwendigen externen Bibliotheken installiert und in der Qt Projektdatei korrekt eingebunden werden.

Die drei Unterprojekte werden im folgenden nur knapp beschrieben, da alle Klassen und Funktionen im Code selbst kommentiert sind, und - zumindest nach intensiver Beschäftigung mit dem Quellcode - verständlich sein sollten.

3.2 Bibliothek zur Lösung des Sichtbarkeitsproblems

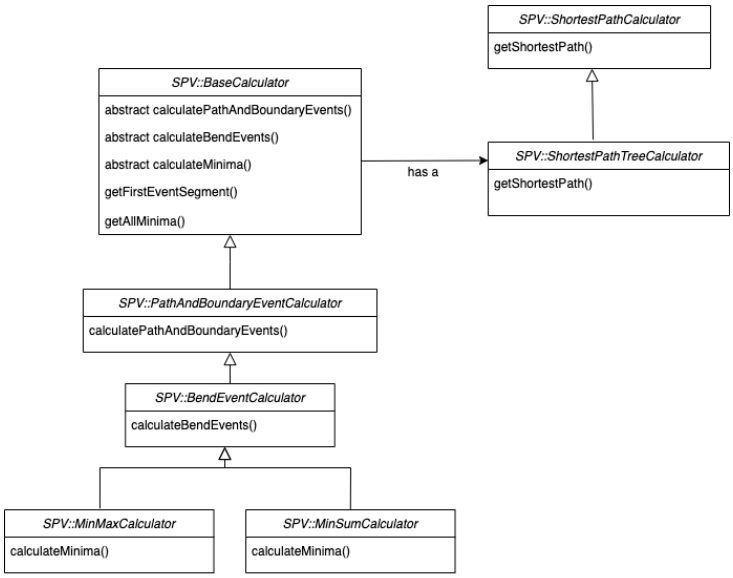

Abbildung 3.1: Die wichtigsten Klassen der Library

Die Bibliothek hat den namespace SPV, kurz für **S**hortest **P**ath to **V**isibility.

Ihre wichtigsten Klassen sind die in Abbildung 3.1 dargestellten Calculator-Klassen. Sie haben die folgenden Aufgaben:

1. Die Klasse *SPV::ShortestPathCalculator* berechnet anhand des Algorithmus, der in [15, 16][1] beschrieben wird, den kürzesten Pfad vom Start- zum Endpunkt. Die dazu notwendige Triangulation des Polygons wird von CGAL berechnet. Wenn man die Funktion *getShortestPath* auf einer Instanz von *SPV::ShortestPathCalculator* aufruft, enthalten die Knoten des kürzesten Pfads keine Informationen über die von der Sichtbarkeitsgeraden überfegten Segmente auf dem Polygonrand.

2. Die Klasse *SPV::ShortestPathTreeCalculator* ist von *SPV::ShortestPathCalculator* abgeleitet und berechnet für jeden Knoten des kürzesten Pfads die Segmente auf dem Polygonrand, die von einer Sichtbarkeitsgeraden, die an dem Knoten gedreht wird, überstreift werden. Für den Aufbau der Shortest Path Maps werden der in [13, 21][2] beschriebene Algorithmus und die CGAL Triangulierung genutzt. Die überstreiften Segmente werden bei den einzelnen Knoten des kürzesten Pfads gespeichert. Da die relevanten Sichtbarkeitsgeraden lediglich diese Segmente überstreifen, ist es ausreichend, nur sie bei der Berechnung der Ereignisse zu betrachten. Wenn man die (überschriebene) Funktion *getShortestPath* auf einer Instanz von *SPV::ShortestPathTreeCalculator* aufruft, enthalten die Knoten des kürzesten Pfads Informationen über die von der Sichtbarkeitsgeraden überfegten Segmente.

3. Die Klasse *SPV::BaseCalculator* ist die abstrakte Basisklasse für alle Klassen, die die Ereignisse und die Minima berechnen. Sie enthält Code, der von allen abgeleiteten Klassen benötigt wird.

4. Die Klasse *SPV::PathAndBoundaryEventCalculator* ist von *SPV::BaseCalculator* abgeleitet und implementiert die Funktion *calculatePathAndBoundaryEvents*, die die Pfad- und Randereignisse berechnet, wie in

[1] [16] habe ich unter dem Titel „Teaching notes“ (http://page.mi.fu-berlin.de/mulzer/#teaching) auf der Webseite von Prof. W. Mulzer gefunden. Die Notizen sind keine offizielle Veröffentlichung, waren aber zum besseren Verständnis des Algorithmus sehr hilfreich, so dass ich sie erwähnen möchte.

[2] [21] ist die Mitschrift einer Vorlesung von L. Guibas und daher mit Vorsicht zu genießen. Die darin enthaltenen Beispiele waren aber zum besseren Verständnis des Algorithmus sehr hilfreich, so dass ich diese Quelle nicht unzitiert lassen will.

Kapitel 2.3.1 dieser Arbeit beschrieben. Das Ergebnis dieser Berechnungen ist eine doppelt verkettete Liste von *SPV::EventSegment* Instanzen (siehe VisibilityProblemLibrary/Models/eventsegment.h in [18]). Jede Instanz dieser Klasse beschreibt ein Segment, das von je zwei aufeinanderfolgenden Ereignissen begrenzt wird, und hat eine Referenz auf seinen Vorgänger und seinen Nachfolger. Lediglich die erste Instanz hat keinen Vorgänger und die letzte keinen Nachfolger. Nach Aufruf von *calculatePathAndBoundaryEvents* besteht die Liste aus aufeinanderfolgenden Pfad- und Randereignissen.

5. Die Klasse *SPV::BendEventCalculator* ist von *SPV::PathAndBoundaryEventCalculator* abgeleitet und implementiert die Funktion *calculateBendEvents*. Diese Funktion berechnet alle Biegungsereignisse und fügt sie in die Liste der *SPV::EventSegment* Instanzen ein, so wie es in Kapitel 2.3.2 beschrieben wird.

6. Die Klasse *SPV::MinMaxCalculator* ist von *SPV::BendEventCalculator* abgeleitet und implementiert die Funktion *calculateMinima* für die minmax Variante des Problems. Die Lage der Sichtbarkeitsgerade, bei der die Distanzen gleich sind, wird approximativ durch rekursive Aufrufe der Funktion *findInnerMinimum* (definiert als private Funktion in VisibilityProblemLibrary/Minima/minmaxcalculator.h in [18]) berechnet. Diese Funktion teilt ein Segment in der Mitte und sucht in einer der Hälften weiter, bis ein Minimum gefunden ist.

7. Die Klasse *SPV::MinSumCalculator* ist von *SPV::BendEventCalculator* abgeleitet und implementiert die Funktion *calculateMinima* für die minsum Variante des Problems. Falls die Lotfußpunkte auf der Sichtbarkeitsgeraden von Polygonkanten versperrt sind, wird das Minimum approximativ berechnet. Der dazu benutzte Algorithmus ist recht krude: zwischen die Sichtbarkeitsgeraden, die das relevante Segment begrenzen, werden in bestimmten Abständen weitere Sichtbarkeitsgeraden gelegt, und die Distanz zu jeder dieser Geraden wird berechnet. Alle lokalen Minima werden während dieser Berechnungen gespeichert. Da meine Erfahrung bei der Berechnung der lokalen Minima darauf hinweist, dass ein lokales Minimum bei versperrter Sicht nah an einem Ereignis liegt, ist die Anzahl der erzeugten Sichtbarkeitsgeraden in der Nähe der Ereignisse größer als in weiter entfernten Bereichen. Dadurch wird ein Kompromiss zwischen Genauigkeit und Schnelligkeit erzielt.

Um Minima berechnen zu lassen, muss eine Instanz von *SPV::MinSumCalculator* oder *SPV::MinMaxCalculator* erzeugt werden. Als Argumente für den Konstruktor werden ein Polygon, ein Start- und ein Endpunkt erwartet. Anschließend muß die Funktion *calculateMinima* aufgerufen werden. Die Minima kann man dann per Aufruf der Funktion *getAllMinima* erhalten.

3.3 Visualisierung

Das Unterprojekt *VisibilityProblemApp* dient dazu, das Sichtbarkeitsproblem und seine Lösung zu visualisieren. Dazu werden einige der vom Qt-Framework bereitgestellten Funktionen genutzt. Der Aufbau von *VisibilityProblemApp* ist sehr einfach: die Klasse *MainWindow* baut ein Fenster auf, das ein Menü und eine graphische Fläche enthält. Auf der graphischen Fläche kan man mit der Maus ein Polygon, sowie einen Start- und Endpunkt eingeben. Die Klasse *VisibilityProblemScene*, die die graphische Fläche kontrolliert, kommuniziert über die Klasse *CGALGeometryConnector* mit der oben beschriebenen Bibliothek. Aufgabe von *CGALGeometryConnector* ist es, zwischen Qt und der Bibliothek, die auf CGAL basiert, zu übersetzen.

Über das Menü kann entschieden werden, welche Minima bzw. welche Ereignisse gezeigt werden sollen. Außerdem kann man einige Testdaten laden, die im Testteil der Applikation verwendet wurden. Die Testdaten können in verschiedenen Vergrößerungen dargestellt werden. Es ist auch möglich, die Koordinaten der Punkte darzustellen, sowie ihre Genauigkeit einzustellen. Wenn man *Animate* klickt, werden die Drehung der Sichtbarkeitsgeraden um die Knoten des kürzesten Pfads sowie die Änderungen der Pfade zur Sichtbarkeitsgeraden animiert.

Der Menüeintrag *Min Max Circle*, der ein Unterpunkt von *Minimum Type* ist, hat folgende Funktion: wenn die Lösung eines min-max Problems auf einer Seite nicht eindeutig bestimmt ist und der Menüeintrag aktiviert ist, wird ein Kreis eingezeichnet, in dem die Lösungen liegen können. Da dies keine wirklich schöne Darstellung des gegebenen Sachverhalts ist, kann der Kreis durch Klicken des Menüeintrags gezeigt und versteckt werden.

3.4 Tests

Das Unterprojekt *VisibilityProblemTest* enthält Code, durch den die Berechnungen der Bibliothek auf ihre Korrektheit getestet werden. Der Code basiert dabei auf dem QtTest Framework. Um Testdaten zu generieren habe ich anhand von GeoGebra ([12]) Polygone erzeugt und manuell die Ereignisse sowie die Minima berechnet. Die von mir berechneten Lösungen werden in den Tests mit den Lösungen, die die Bibliothek berechnet, verglichen. Bei der Auswahl der Testpolygone habe ich versucht, möglichst viele Grenzfälle zu beachten, z.B.:

1. Polygone, bei denen der direkte Weg zu Sichtbarkeitsgeraden durch Polygonkanten versperrt ist.

2. Polygone, bei denen sich die Drehrichtung der Sichtbarkeitsgeraden ändert.

3. Polygone, bei denen Sichtbarkeitsgeraden über Einbuchtungen im Polygon verlaufen.

4. Polygone, bei denen mehrere Knoten aufgrund von Randereignissen zum kürzesten Pfad zur Sichtbarkeit hinzugefügt werden und dann wieder verschwinden.

5. Polygone, die mehr als ein min-sum Minimum haben.

6. Polygone mit min-sum Minima, die auf einer Sichtbarkeitsgeraden liegen, die nicht mit einem Pfad-, Rand- oder Biegungsereignis zusammenfällt.

7. Polygone, bei denen es eine Menge von Lösungen für das min-max Problem gibt.

8. Polygone, bei denen die Lösung des min-max Problems bei einem Ereignis eintritt.

9. Polygone, bei denen der Start- und Endpunkt auf einer Kante der Polygontriangulierung liegen.

Alle Tests werden zweifach durchgeführt: einmal mit Punkt s als Startpunkt und t als Endpunkt, ein zweites Mal mit t als Startpunkt und s als Endpunkt. Das Ergebnis muss in beiden Fällen natürlich dasselbe sein.

Ich habe das Programm auf Ubuntu außerdem mit Valgrind [23] auf Speicherlecks getestet. Im derzeitigen Zustand konnten damit weder Speicherlecks noch andere Fehler gefunden werden.

Um aussagekräftige Daten zur Schnelligkeit der Bibliothek zu erhalten, müsste man Polygone mit vielen Knoten (hundert und mehr) erzeugen. Das ist mit einem erheblichen Arbeitsaufwand verbunden, den ich als Teil dieser Masterarbeit nicht zu investieren bereit war. Allerdings ist es auch ohne Schnelligkeitstests möglich, den ineffizientesten Teil der Implementierung zu identifizieren: es ist die approximative Bestimmung der lokalen min-sum Minima bei versperrter Sicht auf die Lotfußpunkte.[3] Dabei werden hunderte Sichtbarkeitsgeraden und Distanzen berechnet und miteinander verglichen. Um diesen Approximationsalgorithmus etwas effizienter zu gestalten, werden in der näheren Umgebung der Ereignisse mehr Geraden erzeugt als in weiter entfernten Regionen, aber besonders gut ist diese Implementierung nicht.

3.5 Verbesserungsmöglichkeiten

Die offensichtlichste Verbesserungsmöglichkeit besteht darin, dass sich jemand die Mühe macht, den Code einer eingehenden Code-Review zu unterziehen. Ich habe jede Zeile selbst geschrieben und bin mir sicher, dass ich viele Dinge umständlich und unverständlich programmiert habe. Auch ist keinesfalls auszuschließen, dass der Code Fehler enthält. Die Tests deuten zwar darauf hin, dass der Code recht robust ist, aber das ist natürlich keine Garantie dafür, dass die Berechnungen immer korrekt verlaufen.

Eine weitere Verbesserungsmöglichkeit besteht darin, den Algorithmus zum Auffinden von lokalen min-sum Minima bei versperrter Sicht auf die Lotfußpunkte zu verbessern. Derzeit ist der Algorithmus (wie oben beschrieben) eher krude und kann bei eingehender Beschäftigung mit den Distanzfunktionen möglicherweise stark verbessert werden.

Wenn die Lösung in der min-max Version auf einer Seite nicht eindeutig bestimmt ist, kann das Programm den Kreis anzeigen, in dem die Lösungen

[3] Die Implementierung ist in den Funktion *handleLocalMinimaWithBothPathsBlocked* und *handleLocalMinimaWithOnePathBlocked* der Klasse *SPV::MinSumCalculator* zu finden.

liegen müssen. Allerdings werden dabei weder Ränder von Sichtbarkeitszellen noch der Rand des Polygons berücksichtigt, so dass hier eine weitere Verbesserungsmöglichkeit besteht.

Kapitel 4

Zusammenfassung und Ausblick

Ich habe mich in dieser Masterarbeit intensiv mit dem paarweisen Sichtbarkeitsproblem zweier Punkte in einem einfachen Polygon beschäftigt. Dabei habe ich zwei nennenswerte Erkenntnisse finden können:

1. Neben Pfad-, Rand und Biegungsereignissen müssen auch degenerierte Biegungsereignisse berechnet werden.

2. Wenn die Sicht auf die Lotfußpunkte zwischen zwei Ereignissen nicht von Polygonkanten versperrt ist, liegt das lokale min-sum Minimum bei einem dieser Ereignisse und nicht zwischen ihnen.

Darüber hinaus habe ich eine praktische Implementierung des Algorithmus in C++ geschrieben.

Für Studenten oder Wissenschaftler, die sich weiter mit dem paarweisen Sichtbarkeitsproblem beschäftigen wollen, fallen mir folgende interessante Fragestellungen ein:

1. Es wäre interessant, die Distanzfunktionen genauer zu untersuchen, die gelten, wenn die Sicht auf die Lotfußpunkte auf einer oder auf beiden Seiten durch Polygonkanten versperrt ist. Ist in diesem Fall das lokale min-sum Minimum eindeutig bestimmt, oder gibt es mehr als ein lokales Minimum? Kann man abschätzen, um wieviel sich ein solches Minimum vom optimalen Minimum auf den Halbkreisen unterscheidet? Unter welchen Voraussetzungen lassen sich die Gleichungen für das min-max Problem exakt lösen?

2. Eine weitere interessante Frage ist, wie das Sichtbarkeitsproblem zu lösen ist, wenn mehr als zwei Punkte gegenseitig sichtbar sein sollen.

3. Wer dann immer noch nicht genug hat, kann versuchen, das paarweise Sichtbarkeitsproblem im n-dimensionalen Raum mit $n > 2$ zu lösen bzw. seine Komplexität zu bestimmen.

Neben diesen Fragestellungen ist es natürlich möglich, den Quellcode zu verbessern. Zum Beispiel kann man den kruden Algorithmus zum Finden von lokalen min-sum Minima durch einen effizienteren ersetzen. Eine schöne Weiterentwicklung würde auch darin bestehen, die Eingabe des Polygons und die Wiedergabe der Lösung per Webbrowser zu ermöglichen. Die von mir geschriebene Bibliothek könnte dabei auf einem Webserver laufen. Dies würde es einer größeren Zahl von Interessierten (falls es Interessierte gibt) ermöglichen, die Visualisierung des Problems zu nutzen, ohne extra Software installieren zu müssen. Desweiteren könnte man Polygone mit hunderten von Knoten erzeugen, das Programm damit testen und den Quellcode anhand der dabei gewonnenen Erkenntnisse verbessern.

Anhang A

Distanzfunktionen bei freier Sicht

Die Sichtbarkeitsgerade l durch v mit Steigung α wird durch die folgende Gleichung beschrieben:

$$y = \alpha \cdot x + s \tag{A.1}$$

Da jede Sichtbarkeitsgerade durch v geht, gilt:

$$v.y = \alpha \cdot v.x + s \tag{A.2}$$

$$s = v.y - \alpha \cdot v.x \tag{A.3}$$

Wenn s in A.1 ersetzt wird, erhält man:

$$\alpha \cdot x - y + (v.y - \alpha \cdot v.x) = 0 \tag{A.4}$$

Wenn eine Gerade durch die Gleichung $a' \cdot x + b' \cdot y + c' = 0$ beschrieben wird, ist der Abstand eines Punktes (x_0, y_0) zu dieser Geraden:

$$d(a' \cdot x + b' \cdot y + c' = 0, (x_0, y_0)) = \frac{|a' \cdot x_0 + b' \cdot y_0 + c'|}{\sqrt{a'^2 + b'^2}} \tag{A.5}$$

Mit $a' = \alpha$, $b' = -1$ und $c' = v.y - \alpha \cdot v.x$ kann die gemeinsame Distanz von u und u' zu l in Abhängigkeit der Steigung α daher folgendermassen geschrieben werden:

$$d(\alpha) = \frac{|(u.x - v.x) \cdot \alpha + v.y - u.y| + |(u'.x - v.x) \cdot \alpha + v.y - u'.y|}{\sqrt{\alpha^2 + 1}} \tag{A.6}$$

Anhang B

Beweis der Gleichheit der Winkel

In Lemma 13 wird angenommen, daß die Winkel β und γ (siehe Abbildung B.1) gleich sind. Diese Annahme wird hier bewiesen. Dazu benötigt man die folgenden Tatsachen:

- die gegenüberliegenden Winkel von zwei sich schneidenden Geraden sind gleich. In Abbildung B.1 bedeutet das, dass die blauen Winkel bei v beide gleich ρ sind, und die roten beide gleich σ
- der Innenwinkel eines Dreiecks beträgt 180°
- die Basiswinkel eines gleichschenkligen Dreicks sind gleich

In Abbildung B.1 ist das Dreick $\Delta c_2 u' f_{u'}^{l'}$ gleichschenkelig mit Basis $\overline{u' f_{u'}^{l'}}$ und es gilt:

$$\angle f_{u'}^{l'} u' v = 90° - \rho \tag{B.1}$$

Somit gilt für den Winkel γ:

$$\gamma = 180° - 2 \cdot (90° - \rho) = 2 \cdot \rho \tag{B.2}$$

Mit einer entsprechenden Argumentation kann man für die andere Seite beweisen:

$$\angle f_{u}^{l'} c_1 u = 2 \cdot \sigma \tag{B.3}$$

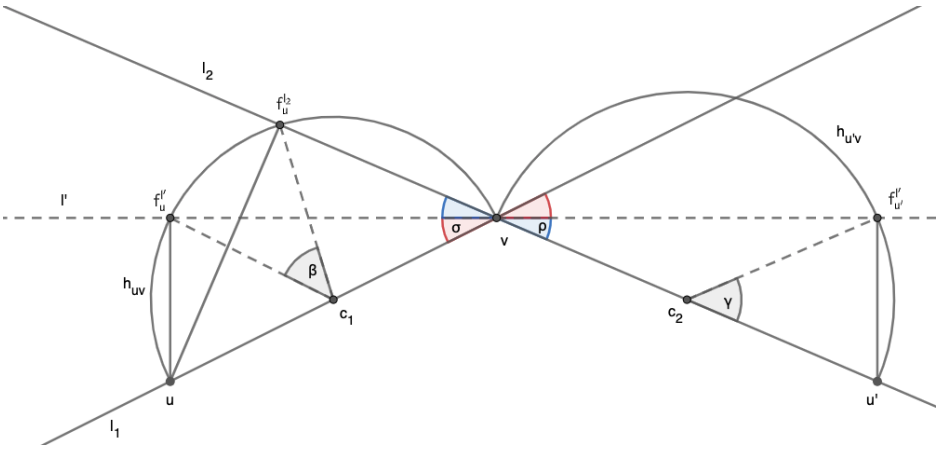

Abbildung B.1: Die Winkel β und γ sind gleich.

$$\angle f_u^{l_2} c_1 v = 180° - 2 \cdot (\rho + \sigma) \tag{B.4}$$

Unter Berücksichtigung von:

$$180° = \angle f_u^{l_2} c_1 v + \beta + \angle f_u^{l'} c_1 u \tag{B.5}$$

gilt:

$$\beta = 180° - (180° - 2 \cdot (\rho + \sigma) + 2 \cdot \sigma) = 2 \cdot \rho = \gamma \tag{B.6}$$

Anhang C

Werte des Beispiels bei versperrter Sicht

Bei dem in Kapitel 2.4.3 verwendeten Beispiel (siehe Abbildung 2.28) sind die Gerade, die die Polygonkante auf der linken Seite enthält, die Punkte u, v und u' sowie die Distanzfunktionen folgendermaßen definiert:

$$y_e = -\frac{1}{4} \cdot x + 2 \tag{C.1}$$

$$u = (1, -2), v = (6, 0), u' = (11.1, -2.04) \tag{C.2}$$

$$d_l(\alpha) = \sqrt{(1 - \frac{6 \cdot \alpha + 2}{\frac{1}{4} + \alpha})^2 + (-2 - \frac{\frac{1}{2} \cdot \alpha}{\frac{1}{4} + \alpha})^2} + \frac{|5.1 \cdot \alpha + 2.04|}{\sqrt{\alpha^2 + 1}}$$
$$\text{für } -0.4 \leq \alpha \leq -0.386 \tag{C.3}$$

$$d_l(\alpha) = \frac{|-5 \cdot \alpha + 2|}{\sqrt{\alpha^2 + 1}} + \frac{|5.1 \cdot \alpha + 2.04|}{\sqrt{\alpha^2 + 1}} \text{ für } -0.386 < \alpha \leq 0.4 \tag{C.4}$$

Diese Werte kann man z.B. in [10] eingeben und damit das Minimum, das bei einer Steigung von etwa -0.397 liegt, berechnen lassen.

Anhang D

Anleitung zur Installation

Um den Quellcode ändern und kompilieren zu können, müssen CGAL und Qt sowie ein paar andere C++ Bibliotheken installiert werden. Ich habe diese Bibliotheken sowohl auf macOS Catalina als auch auf Ubuntu 18.04 installiert und den Quellcode in beiden Betriebssystemen kompiliert und getestet. Ich empfehle eine Installation auf Ubuntu, da es sehr einfach ist, die notwendigen externen Bibliotheken einzubinden.

Falls Ubuntu nicht das Betriebssystem des verwendeten Computers ist, kann es anhand einer virtuellen Maschine genutzt werden. Dazu kann z.B. VirtualBox (https://www.virtualbox.org/) und das entsprechende Ubuntu Image (http://releases.ubuntu.com/18.04/) installiert werden. Im Internet gibt es viele Tutorials, die bei der Installation helfen.

D.1 Installation auf Ubuntu

In Ubuntu öffnet man einen Terminal und installiert folgende Bibliotheken:

```
sudo apt-get install build-essential
sudo apt-get install qtcreator
sudo apt-get install qt5-default
sudo apt-get install libcgal-dev
```

Dann muss der Quellcode noch von github heruntergeladen werden:[1]

```
git clone https://github.com/klauste/ShortestPathToVisibility.git
```

Anschließend kann das Projekt im qtCreator geöffnet und kompiliert werden.

D.2 Installation auf macOS Catalina

Die freie Qt Version kann von https://www.qt.io/download-qt-installer heruntergeladen werden.

Darüber hinaus werden folgende Bibliotheken benötigt, die man im Internet herunterladen kann:

CGAL

CGAL_Core

boost_thread-mt

boost_system-mt

gmp

mpfr

Dann muß der Quellcode von github heruntergeladen werden:

```
git clone https://github.com/klauste/ShortestPathToVisibility.git
```

In den .pro Dateien muss möglicherweise noch der Pfad zu den externen Bibliotheken angepaßt werden, z.B. in der Datei VisibilityProblemLibrary.pro. Anschließend kann das Projekt im qtCreator geöffnet und kompiliert werden.

[1] git ist eigentlich Bestandteil von Ubuntu, wenn es nicht installiert ist, kann dies leicht nachgeholt werden.

Um eine dmg Datei zu erstellen, muss man das Projekt 'VisibilityProblemApp' im 'Release' Modus kompilieren. Dann führt man die folgenden Schritte aus:

```
cd <pathToYourBuildFolder>/VisibilityProblemApp
    /VisibilityProblemApp.app/Contents

mkdir Frameworks

cd <pathToYourBuildFolder>/VisibilityProblemApp/

<pathToQt>/bin/macdeployqt VisibilityProblemApp.app -always-overwrite
    -appstore-compliant -dmg
```

Literaturverzeichnis

[1] H. Ahn, E. Oh, L. Schlipf, F. Stehn und Darren Strash. „On Romeo and Juliet Problems: Minimizing Distance-to-Sight“. In: *arXiv e-prints*, arXiv:1906.01114 (Juni 2019), arXiv:1906.01114. arXiv: 1906.01114 [cs.CG].

[2] N. Amato, M. Goodrich und E. A. Ramos. „A Randomized Algorithm for Triangulating a Simple Polygon in Linear Time“. In: *Discrete & Computational Geometry* 26 (2001), 245–265. DOI: https://doi.org/10.1007/s00454-001-0027-x.

[3] E. M. Arkin, A. Efrat, C. Knauer, J. S. B. Mitchell, V. Polishchuk, G. Rote, L. Schlipf und T. Talvitie. „Shortest Path to a Segment and Quickest Visibility Queries“. In: *Journal of Computational Geometry* 7.2 (2016), S. 77–100. DOI: https://doi.org/10.20382/jocg.v7i2a5.

[4] S. Carlsson, H. Jonsson und B. Nilsson. „Finding the Shortest Watchman Route in a Simple Polygon“. In: *Discrete & Computational Geometry* 22 (1999), 377–402. DOI: https://doi.org/10.1007/PL00009467.

[5] *CGAL*. URL: https://www.cgal.org/ (besucht am 27. 02. 2020).

[6] B. Chazelle. „Triangulating a simple polygon in linear time“. In: *Discrete & Computational Geometry* 6 (1991), 485–524. DOI: https://doi.org/10.1007/BF02574703.

[7] WP. Chin und S. Ntafos. „Shortest watchman routes in simple polygons“. In: *Discrete & Computational Geometry* 6.1 (1991), S. 9–31. DOI: https://doi.org/10.1007/BF02574671.

[8] V. Chvátal. „A combinatorial theorem in plane geometry“. In: *Journal of Combinatorial Theory, Series B* 18.1 (1975), S. 39–41. DOI: https://doi.org/10.1016/0095-8956(75)90061-1.

[9] M. de Berg, O. Cheong, M. van Kreveld und M. Overmars. *Computational Geometry - Algorithms and Applications*. 3. Aufl. Springer-Verlag GmbH Germany, 2008, 21 ff. ISBN: 978-3-540-77973-5. DOI: https://doi.org/10.1007/978-3-540-77974-2.

[10] *Desmos*. URL: https://www.desmos.com/calculator (besucht am 17. 03. 2020).

[11] *Draw*. URL: https://www.draw.io/ (besucht am 05. 03. 2020).

[12] *GeoGebra*. URL: https://www.geogebra.org/geometry (besucht am 12. 04. 2020).

[13] L. Guibas, J. Hershberger, D. Leven, M. Sharir und R. E. Tarjan. „Linear time algorithms for visibility and shortest path problems inside simple polygons". In: *Algorithmica* 2.1-4 (1987), S. 209–233. DOI: https://doi.org/10.1007/BF01840360.

[14] R. Khosravi und M. Ghodsi. „The fastest way to view a query point in simple polygons." In: *Proceedings of the 21st European Workshop on Computational Geometry* (Jan. 2005), S. 187–190.

[15] D. T. Lee und F. P. Preparata. „Euclidean Shortest Paths in the Presence of Rectilinear Barriers". In: *Networks* 14 (1984), S. 393–410. DOI: https://doi.org/10.1002/net.3230140304.

[16] W. Mulzer. *Shortest Paths in Polygons - Teaching notes*. URL: https://page.mi.fu-berlin.de/mulzer/notes/alggeo/polySP.pdf (besucht am 04. 12. 2019).

[17] *Overleaf*. URL: https://www.overleaf.com/ (besucht am 15. 04. 2020).

[18] *Pairwise Visibility Implementation*. URL: https://github.com/klauste/ShortestPathToVisibility (besucht am 27. 02. 2020).

[19] *Qt*. URL: https://www.qt.io/ (besucht am 27. 02. 2020).

[20] *SageMath*. URL: http://http://www.sagemath.org/de/ (besucht am 29. 08. 2019).

[21] J. Stewart. *Handout for Lecture 8, Mitschrift einer Vorlesung von L. Guibas*. 1992. URL: https://graphics.stanford.edu/courses/cs268-09-winter/notes/handout7.pdf (besucht am 20. 08. 2019).

[22] X. Tan. „Fast computation of shortest watchman routes in simple polygons". In: *Information Processing Letters* 77 (1 2001), S. 27–33. DOI: https://doi.org/10.1016/S0020-0190(00)00146-0.

[23] *Valgrind*. URL: https://valgrind.org/ (besucht am 27. 02. 2020).

[24] H. Wang. „Quickest Visibility Queries in Polygonal Domains“. In: *Discrete & Computational Geometry* 62 (2019), S. 374–432.

[25] Wikipedia. *Sweep (Informatik) — Wikipedia, Die freie Enzyklopädie*. 2018. URL: https://de.wikipedia.org/w/index.php?title=Sweep_(Informatik)&oldid=174576584 (besucht am 28. 03. 2020).

[26] E. Wynters und J. Mitchell. „Shortest Paths for a Two-robot Rendezvous.“ In: *Proceedings of the 5th Canadian Conference on Computational Geometry* (Jan. 1993), S. 216–221.

Abbildungsverzeichnis

Alle verwendeten Abbildungen außer 3.1 wurden von mir mit GeoGebra [12] erstellt und per Screenshot abgespeichert. Zur Erstellung von 3.1 habe ich draw.io [11] verwendet.

Die Arbeit wurde im Online Editor overleaf [17] geschrieben.