Fernuniversität 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: 1.5.2020

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

Zusammenfassung

Der Artikel „On Romeo and Juliet Problems: Minimizing Distance-to-Sight“ ([1]) stellt folgendes Problem der Algorithmischen Geometrie vor: gegeben seien 2 Punkte s und t innerhalb eines einfachen Polygons P mit n Eckpunkten. Was ist dann die minimale Distanz, die sie zurücklegen müssen, um einander sehen zu können? Dieses Problem kann anhand eines im Artikel 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 [2] und des Qt-Frameworks [11].

Inhaltsverzeichnis

Kapitel 1

Einleitung

Der Artikel „On Romeo and Juliet Problems: Minimizing Distance-to-Sight“ [1] stellt eine Variante des sogenannten „Watchman Route Problem“ [3] vor, das paarweise Sichtbarkeitsproblem:

Paarweises Sichtbarkeitsproblem: *Gegeben seien zwei Punkte s und t in einem einfachen Polygon P. Berechne die minimale Distanz, die s und t zurücklegen müssen, um sich in P sehen zu können.*

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 entwickelt der Artikel einen Algorithmus mit linearer Komplexität.

Kurz zusammengefaßt besteht der Algorithmus darin, diejenigen Ereignisse zu berechnen, bei denen sich die Struktur der Pfade von Punkt s und t zur gegenseitigen Sichtbarkeit ändert. Diese Ereignisse werden im Artikel als *Pfadereignisse (path events)*, *Randereignisse (boundary events)* und *Biegungsereignisse (bend-events)* bezeichnet. Nachdem diese Ereignisse berechnet sind, müssen noch die lokalen Minima zwischen ihnen gefunden werden. Durch einen Vergleich der lokalen Minima lassen sich dann das bzw. die globalen Minima finden.

Diese Masterarbeit hat zum Ziel, den vorgestellten Algorithmus wiederzugeben und praktisch zu implementieren. Als Programmiersprache wurde C++ gewählt. Diese Wahl 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 einzu-

bindende Bibliotheken und Frameworks gibt, die für die Implementierung notwendig sind: zum einen die CGAL Bibliothek [2], die viele wichtige geometrische Algorithmen bereitstellt (etwa zur Triangulierung eines Polygons), und zum anderen das Qt-Framework [11], das eine Visualisierung der Ergebnisse und Tests ermöglicht.

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.

Ich bin bei der Implementierung auf zwei solcher Probleme bzw. Ungenauigkeiten gestoßen. Das erste Problem ist bei der Berechnung der *Biegungsereignisse* aufgetreten. Der Artikel erwähnt drei Möglichkeiten bei den *Biegungsereignissen*:

1. Der Pfad zur Sichtbarkeitsgeraden verläuft am Polygonrand und stößt dann auf eine Polygonecke.

2. Der Pfad zur Sichtbarkeitsgeraden stößt auf einen Punkt des kürzesten Pfads zwischen Punkten s und t.

3. Zwei Kanten auf dem Pfad zur Sichtbarkeitsgeraden werden parallel.

Diese Fallunterscheidung berücksichtigt nicht, daß es für die Berechnung eines lokalen Minimums von Bedeutung ist, ob der direkte Pfad zu einer Sichtbarkeitsgeraden (also der Pfad zum Lotfußpunkt auf der Sichtbarkeitsgeraden) frei ist, oder von einer Polygonkante versperrt wird. Falls eine Polygonkante bei Drehung der Sichtbarkeitsgerade beginnt bzw. aufhört, den direkten Pfad zum Lotfußpunkt zu versperren, wird dem kürzesten Pfad zur Sichtbarkeit zwar kein Punkt hinzugefügt oder weggenommen; dennoch sind derartige Ereignisse für die Berechnung der Minima von Bedeutung, da sich bei ihnen die zu verwendende Distanzfunktion ändert. Sie werden in dieser Arbeit als *degenerierte Biegungsereignisse (degenerate bend-events)* bezeichnet (eine Namensgebung, die auf meine Betreuerin Frau Schlipf zurückgeht) und werden bei der Implementierung des Algorithmus berücksichtigt.

Ein zweites Problem bzw. eine zweite Ungenauigkeit ist mir bei der Berechnung des lokalen min-sum Minimums zwischen zwei Ereignissen aufgefallen. Der Artikel [1, Seite 8] sagt dazu:

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

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

1. Falls der direkte Pfad zur Sichtbarkeitsgeraden von keiner Polygonkante versperrt wird, ist es nicht nötig, Ableitungen der Distanzfunktion zu berechen. Das lokale Minimum liegt in diesem Fall immer auf einer der Sichtbarkeitsgeraden, die mit einem Pfad-, Rand- oder Biegungsereignis zusammenfällt.

2. Falls der direkte Pfad zur Sichtbarkeitsgerade von einer oder mehreren Polygonkanten versperrt wird, liegt das Minimum nicht notwendigerweise auf einer Sichtbarkeitsgeraden, die mit einem Pfad-, Rand- oder Biegungsereignis zusammenfällt. 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 minmax Minimums, die im Artikel [1] nur knapp erwähnt wird, so ausführlich, daß eine praktische Implementierung möglich ist.

Die Masterarbeit gliedert sich entsprechend der Aufgabenstellung in zwei Teile: im ersten Teil (Kapitel 2) wird der Algorithmus zur Berechnung des minimalen Pfads zur gemeinsamen 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.
- ∂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 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, die 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 Knotenpunkt 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.

- $\langle v_0, v_1, ..., v_k - 1, v_k \rangle$ bezeichnet die Knotenpunkte 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 Knotenpunkt sein oder zwischen zwei Knotenpunkten 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 2.1.

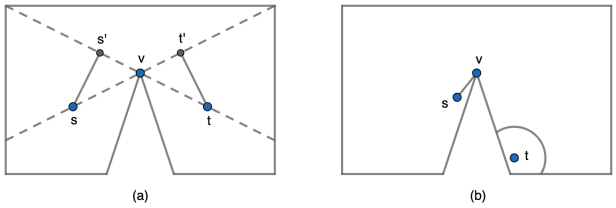

Abbildung 2.1: In Abbildung (a) sind sowohl das Punktepaar (s', t) als auch (s, t') Lösungen des min-sum Problems. In Abbildung (b) sind alle Punktepaare (v, t') mit $|\overline{tt'}| \leq |\overline{sv}|$ und $t' \in P$ Lösungen des min-max Problems.

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 [6],[7]. SPT_s und SPM_s werden bei der Erläuterung des Algorithmus immer wieder benötigt und sind folgendermaßen definiert:

Definition 1. *s sei ein Punkt in P und p_i seien die Knotenpunkte 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 Knotenpunkt 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 Knotenpunkten von P verläuft. Der Vorgänger von q auf $\pi(s,q)$ ist ein Knotenpunkt p' von P. Die Zelle, in der sich die Punkte q befinden, wird im folgenden als Sichtbarkeitszelle von p' bezeichnet und als $S_{p'}$ abgekürzt. p' wird als Scheitelpunkt dieser Zelle bezeichnet. Ein Knotenpunkt $p' \in P$ kann der Scheitelpunkt mehrerer Sichtbarkeitszellen sein.

SPT_s kann aus einer Triangulation von P in $\mathcal{O}(n)$ Zeit berechnet werden. SPM_t kann in Zeit $\mathcal{O}(n)$ aus SPT_s berechnet werden, indem jede Kante von SPT_s erweitert wird, bis sie ∂P erreicht. Für ein Beispiel von SPT_s und SPM_s siehe Abbildung 2.2.

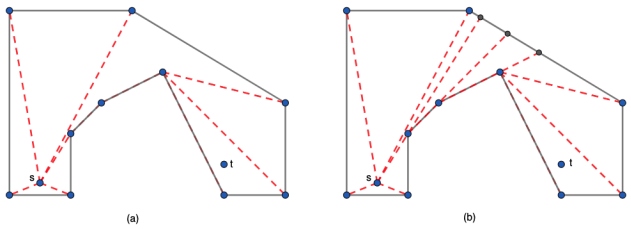

Abbildung 2.2: Abbildung (a) zeigt den Shortest Path Tree von s, Abbildung (b) die Shortest Path Map von s.

2.2 Allgemeine Eigenschaften einer Lösung des Sichtbarkeitsproblems

Im folgenden werden einige allgemeine Eigenschaften einer Lösung des Sichtbarkeitsproblems erläutert und bewiesen. Das Punktepaar $(s^*\ t^*)$ sei dabei zunächst eine optimale Lösung.

Für die inneren Knotenpunkte von $\pi(s,s^*)$ und $\pi(t,t^*)$ gilt folgendes Lemma:

Lemma 1. *Alle inneren Knotenpunkte von $\pi(s,s^*)$ und $\pi(t,t^*)$ sind Knotenpunkte von P.*

Beweis. Der Endpunkt von $\pi(s,s^*)$ befindet sich in einer Zelle der Shortest Path Map SPM_s. Da alle inneren Knotenpunkte von SPM_s Knotenpunkte

von P sind, gilt dies auch für die inneren Punkte von $\pi(s, s^*)$. Für $\pi(t, t^*)$ gilt eine entsprechende Argumentation, wenn SPM_t betrachtet wird. □

Lemma 2. *Für jede optimale Lösung des Sichtbarkeitsproblem (s^*, t^*) gilt: $\overline{s^*t^*}$ tangiert $\pi(s, t)$ an mindestens einem Knotenpunkt $v \in \pi(s, t)$.*

Beweis. Falls $s = s^*$ oder $t = t^*$, gilt das Lemma auf jeden Fall, da s und t Endknoten von $\pi(s, t)$ sind. 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^*. $l = \overline{xx'}$ sei die maximale in P enthaltene Strecke, die $\overline{s^*t^*}$ enthält. s' sei der Vorgängerpunkt von s^* auf $\pi(s, s^*)$, und t' der Vorgängerpunkt von t^* auf $\pi(t, t^*)$. Dann liegen s' und t' entweder auf derselben Seite oder auf unterschiedlichen Seiten von l.

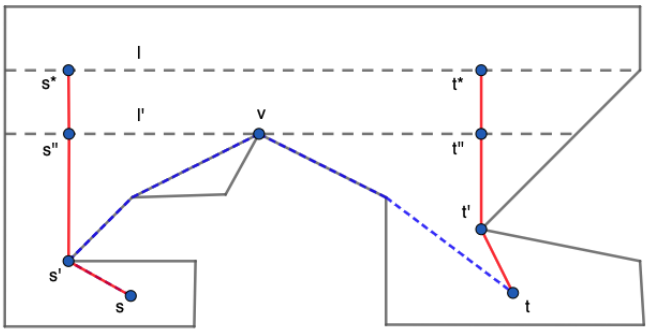

Abbildung 2.3: Wenn s' und t' unter l liegen, muß die Strecke zwischen einem optimalen Punktepaar v tangieren.

Zunächst betrachte ich den Fall, daß s' und t' unterhalb von l liegen. Angenommen l tangiere keinen Knotenpunkt von P. Dann können die Distanzen $\overline{s's^*}$ sowie $\overline{t't^*}$ durch eine Parallelverschiebung von l nach unten verkürzt werden (siehe Abbildung 2.3). Das ist aber ein Widerspruch zur Optimalität von (s^*, t^*), so daß l einen Knotenpunkt v von P tangieren muss. Da s unterhalb von $\overline{xv}$ liegt und t unterhalb von $\overline{vx'}$, muss der Pfad $\pi(s, t)$ die Strecke l schneiden. Angenommen der Schnittpunk mit $\overline{xv}$ sei y, und der mit $\overline{vx'}$ 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.

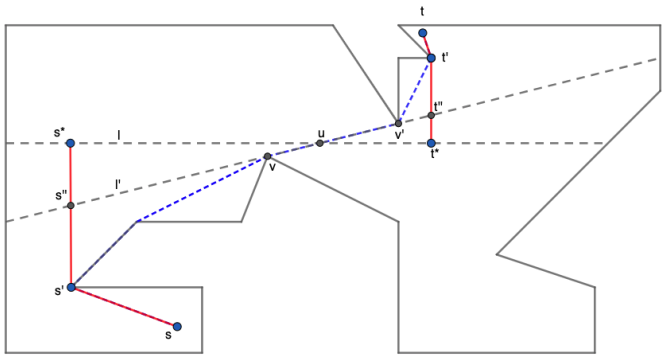

Abbildung 2.4: s' und t' liegen auf unterschiedlichen Seiten von l

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 liegen. Wenn $\overline{s^*t^*}$ keinen Knotenpunkt von P tangiert, kann l an einem Punkt $\in \overline{s^*t^*}$ in Richtung s' und t' gedreht werden, bis $\overline{s^*t^*}$ einen Knotenpunkt v von P tangiert. An diesem Knotenpunkt kann l weiter gedreht werden, bis $\overline{s^*t^*}$ einen weiteren Knotenpunkt v' von P tangiert. Die Schnittpunkte von l mit $\overline{s's^*}$ bzw. $\overline{t't^*}$ während dieser Drehungen seien s'' und t''. Für diese gilt: $\overline{s''t''} \subset P$ und $|\pi(s',s'')| + |\pi(t',t'')| < |\pi(s',s^*)| + |\pi(t',s^*)|$, was der Annahme der Optimalität von s^* und t^* widerspricht (siehe Abbildung 2.4). Also muss das Segment $\overline{s^*t^*}$ zwei Knotenpunkte von P tangieren.

Nun sei angenommen, daß $\overline{s^*t^*}$ zwei Knotenpunkte 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 $y \in \overline{xv}$, bei einem Punkt $u \in \overline{vv'}$ und bei einem Punkt $y' \in \overline{v'x'}$. 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.

□

Die maximalen Strecken in P, die eine Knotenpunkt von $\pi(s,t)$ tangieren,

sind für das Sichtbarkeitsproblem so wichtig, daß sie eine eigene Definition erhalten:

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 einem Knotenpunkt $v \in \pi(s,t)$ tangiert.*

Jede Sichtbarkeitsgerade l, die einen Punkt 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 2.5).

Auch die Lotfußpunkte von einem Punkt auf die Gerade, die eine Sichtbarkeitsgerade enthält, sind wichtig genug für eine eigene Definition:

Definition 3. *g sei die Gerade, die eine Sichtbarkeitsgerade l enthält, q sei ein beliebger Punkt $\notin g$ und 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 l_q^l geschrieben.*

Da eine Sichtbarkeitsgerade als Strecke innerhalb von P definiert wurde, ist zu beachten, daß l_q^l auch außerhalb des Polygons liegen kann.

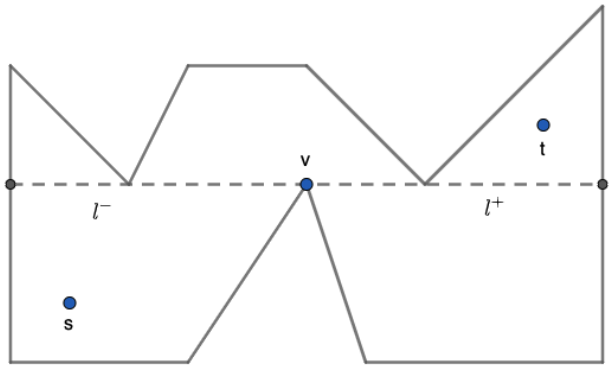

Abbildung 2.5: Einteilung von l in l^- und l^+

Lemma 2 besagt, daß jede Lösung des Sichtbarkeitsproblems auf einer Sichtbarkeitsgeraden l liegt. Das folgende Lemma spezifiert dies noch weiter:

Lemma 3. *Der Endpunkt von $\pi(s,l)$ auf l befindet sich auf l^-, der Endpunkt von $\pi(t,l)$ befindet sich auf l^+.*

Beweis. v sei der von l tangierte Knotenpunkt 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^- 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^- liegt. Durch eine entsprechende Argumentation kann bewiesen werden, daß $\pi(t,l)$ bei einem Punkt auf l^+ endet. □

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

Lemma 4. *g sei eine Gerade, q ein beliebiger Punkt $\notin g$ und l_q^g der Lotfußpunkt von q auf g. Der kürzeste Pfad von q nach g ist die Strecke $\overline{ql_q^g}$. Je weiter ein Punkt $p \in g$ von l_q^g entfernt ist, umso größer wird sein Abstand zu q.*

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

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

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

Man beobachte, daß die Hälfte l^- einer fixen Sichtbarkeitsgeraden mehrere Zellen von SPM_s durchqueren kann. Das folgende Lemma bestimmt sowohl die Sichtbarkeitszelle von SPM_s, in der $\pi(s,l^-)$ endet, als auch die Lage des Endpunkts in der Zelle. s^* ist dabei nicht Teil der Lösung des Sichtbarkeitsproblems, sondern der Endpunkt auf einer gegebenen Sichtbarkeitslinie l, der $\pi(s,l^-)$ minimiert.

Lemma 5. *Der Endpunkt s^* von $\pi(s,l^-)$ liegt in einer Sichtbarkeitszelle von SPM_s mit Scheitelpunkt p^*, so daß eine der folgenden Bedingungen erfüllt ist:*

(i) $p^* = s^*$
(ii) $\overline{p^*s^*}$ ist orthogonal zur Geraden g, die l^- enthält, oder
(iii) $s^* \in \partial P$ und kein Punkt $\in S_{p^*}$ ist näher an $l_{p^*}^l$ als s^*.

Beweis. Für den Beweis betrachtet man alle von l^- durchquerten Sichtbarkeitszellen. Man beginnt bei der Sichtbarkeitszelle bzw. bei einer der Sichtbarkeitszellen S_p, deren Scheitelpunkt p am nächsten an s liegt.

Wenn $p \in l^-$, dann ist $\pi(s,p)$ der kürzeste Pfad von s zu l^- und das Lemma ist bewiesen.

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

Wenn $l_p^{l^-} \notin S_p$, dann schneidet l^- den Rand von S_p in zwei Punkten. i^* sei derjenige Schnittpunkt, der näher an $l_p^{l^-}$ liegt. Dann ist der minimale Pfad von p zu l^- in S_p aufgrund von Lemma 4 die Strecke $\overline{pi^*}$. Wenn $i^* \in \partial P$ und keine weitere Sichbarkeitszelle i^* enthält, dann kann die Distanz $|\pi(s,l^-)|$ nicht weiter verkürzt werden. Somit besteht $\pi(s,l^-)$ aus $\pi(s,p)$ und $\overline{pi^*}$ und das Lemma ist bewiesen. (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 3 gilt weiterhin.)

Andernfalls liegt i^* auf dem Rand einer weiteren Sichtbarkeitszelle von SPM_s mit Scheitelpunkt p'. In diesem Fall betrachtet man den Lotfußpunkt $l_{p'}^{l^-}$ und wendet die bisherige Argumentation auf die Sichtbarkeitszelle $S_{p'}$ und $l_{p'}^{l^-}$ an. Dies kann solange wiederholt werden, bis der Lotfußpunkt in der 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 $\pi(s,p^*)$, oder besteht aus $\pi(s,p^*)$ und der Strecke $\overline{p^*l_{p^*}^l}$, oder aus $\pi(s,p^*)$ und der Strecke von p^* zum Schittpunkt von l^- mit ∂P, der am nächsten zu $l_{p^*}^l$ liegt. Der Beweis ist in Abbildungen 2.6 und 2.7 visualisert.

□

Durch Vertauschen von s und t gilt Lemma 5 auch für die optimalen Punkte t^*.

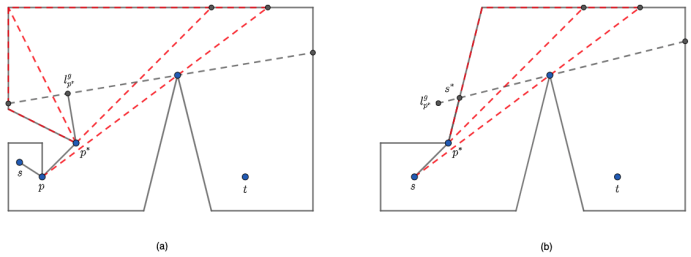

Abbildung 2.6: In Abbildung (a) fallen s^* und $l^l_{p^*}$ zusammen, in Abbildung (b) ist das nicht der Fall.

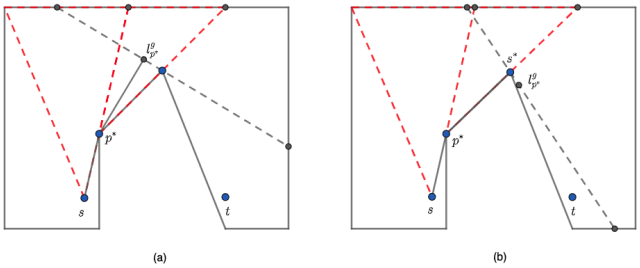

Abbildung 2.7: In Abbildung (a) fallen s^* und $l^l_{p^*}$ zusammen, in Abbildung (b) ist das nicht der Fall.

2.3 Die Ereignisse eines Sweep-Line Ansatzes

Der Algorithmus zur Lösung des Sichtbarkeitsproblems ähnelt einem klassischen Sweep-Line Algorithmus, mit dem Unterschied, daß die Sweep-Line nicht über die Szene hinwegfegt, sondern daß eine Sichtbarkeitsgerade an allen inneren Punkten $\langle v_1, ..., v_k - 1\rangle$ von $\pi(s,t)$ gedreht wird. Der Algorithmus wird mit einer Sichtbarkeitsgeraden, die s und v_1 enthält, initialisiert. Diese Gerade wird an v_1 gedreht (während sie v_1 tangiert), bis sie auf v_2 trifft. Dann wird die Gerade an v_2 gedreht usw., solange bis die Gerade, die an $v_k - 1$ gedreht wird, auf t trifft. Während dieser Drehungen werden aufgrund von Lemma 2 alle Sichtbarkeitsgeraden angetroffen, die für eine Lösung des Sichtbarkeitsproblems in Frage kommen.

Wie bei einem klassischen Sweep-Line Algorithmus müssen die Ereignisse berechnet werden, an denen sich die Struktur der Lösung ändert. s^* und t^*

seien jetzt und im folgenden die Endknoten der Pfade zu einer gegebenen Sichtbarkeitsgeraden. Dann ändert sich die Struktur der Lösung wenn sich der Drehpunkt ändert, wenn sich die Polygonkante ändert, die von der sich drehenden Sichtbarkeitsgeraden überstreift wird, oder wenn sich die Pfade $\pi(s, s^*)$ oder $\pi(t, t^*)$ strukturell ändern.

Die folgenden Ereignisse können unterschieden werden:

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

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

3. **Biegungsereignisse (bend events)** sind Endpunkte von Sichtbarkeitsgeraden, bei denen dem kürzesten Pfad $\pi(s, l)$ oder $\pi(t, l)$ ein Eckpunkt $p \in P$ hinzugefügt oder weggenommen wird, oder wo sich die zu verwendende Distanzfunktion ändert.

Diese Ereignisse werden als Punkte auf dem Polygonrand ∂P dargestellt, wobei für jedes Ereignis beide Endpunkte der Sichtbarkeitsgerade gespeichert werden. Wenn alle Ereignisse berechnet 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.

2.3.1 Berechnung der Pfad- und Randereignisse

Die Berechnung der Pfad- und Randereignisse kann so aufgeteilt werden, daß zunächst l^+ betrachtet wird. Die Berechnung der Ereignisse, die durch l^- ausgelöst werden, kann in einem zweiten Durchgang durch Vertauschen von s und t erfolgen.

Bei der Drehung von l^+ (bzw. l^-) an einem Knotenpunkt v_i ist folgendes zu beachten: l^+ dreht sich an v_i von der Position, wo l den Punkt v_{i-1} enthält, bis zur Position, wo l den Punkt v_{i+1} enthält. Die Region, die l^+ (l^-) während dieser Drehung überstreift, wird als A_i^+ (A_i^-) bezeichnet. Sie besteht nur aus

Sichtbarkeitszellen bzw. bei s und t aus Teilmengen von Sichtbarkeitszellen von SPM_s (SPM_t), die v_i als Scheitelpunkt haben (siehe Abbildung 2.8).

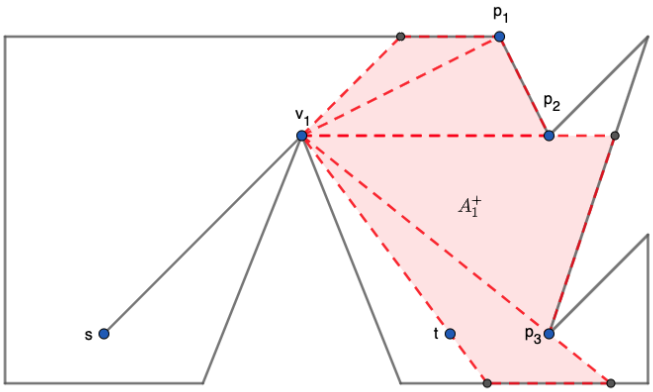

Abbildung 2.8: A_1^+ setzt sich aus Zellen von SPM_s zusammen, die v_1 als Knotenpunkt haben.

Das Innere der Sichtbarkeitszellen einer Shortest Path Map ist paarweise disjunkt. Für alle Punkte q innerhalb einer der überstreiften Sichtbarkeitszellen ist der Drehpunkt außerdem der Vorgänger von q auf $\pi(s, q)$. Daher gelten bezüglich der Regionen A_i^+ und A_i^- folgende Eigenschaften. Da diese Eigenschaften im weiteren Verlauf noch öfters benötigt werden, erhalten sie eigene Namen:

- **$\mathbf{E_1}$**: das Innere der Regionen A_i^+ ist für $0 < i < k$ paarweise disjunkt
- **$\mathbf{E_2}$**: das Innere der Regionen A_i^- ist für $0 < i < k$ paarweise disjunkt
- **$\mathbf{E_3}$**: für alle Punkte $q \in A_i^+$ mit $0 < i < k$ ist v_i der Vorgänger von q auf dem kürzesten Pfad $\pi(s, q)$
- **$\mathbf{E_4}$**: für alle Punkte $q \in A_i^-$ mit $0 < i < k$ ist v_i der Vorgänger von q auf dem kürzesten Pfad $\pi(t, q)$

Nach diesen vorbereitenden Überlegungen kann das folgende Lemma für Pfadereignisse bewiesen werden:

Lemma 6. *Für ein einfaches Polygon P mit n Knotenpunkten und Punkten $s, t \in P$ können alle Pfadereignisse in der Reihenfolge, in der sie von der Sweep-Line angetroffen werden, in Zeit $\mathcal{O}(n)$ berechnet werden.*

Beweis. Für die Berechnung der Pfadereignisse muss zunächst die Shortest Path Map SPM_s von s berechnet werden. Ein Pfadereignis tritt ein, wenn zwei aufeinanderfolgende Punkte von $\pi(s,t)$ auf einer Sichtbarkeitsgeraden l liegen. Für jedes Pfadereignis ist l^+ somit eine Kante von SPM_s. Man kann einen Endpunkt eines Pfadereignisses also finden, indem man für jeden Punkt v_i mit $0 < i < k$ alle Kanten von SPM_s betrachtet, die zu v_i inzident sind. Die Kante, die $\overline{v_{i-1}v_i}$ enthält, ist das gesuchte Pfadereignis. Ihr Endpunkt auf ∂P kann als der eine Endpunkt des Pfadereignisses gespeichert werden. Um die gegenüberliegenden Endpunkte der Pfadereignisse zu finden, muss man bei t starten und die Kanten von SPM_t finden, die Kanten in $\pi(t,s)$ enthalten. Da es $\mathcal{O}(n)$ Kanten in SPM_s und SPM_t gibt und jede Kante maximal einmal betrachtet werden muss, können alle Pfadereignisse in Zeit $\mathcal{O}(n)$ berechnet werden.

Das Pfadereignis, das von Punkt v_{k-1} und t bzw. von v_1 und s ausgelöst wird, ist eine Ausnahme. Allerdings kann es in Zeit $\mathcal{O}(1)$ berechnet werden, wenn man die dreieckige Zelle betrachtet, in der sich t bzw. s befindet.

□

Für Randereignisse gilt das folgende Lemma:

Lemma 7. *Für ein einfaches Polygon P mit n Knotenpunkten und Punkten $s,t \in P$ können alle Randereignisse in der Reihenfolge, in der sie von der Sweep-Line angetroffen werden, in Zeit $\mathcal{O}(n)$ berechnet werden.*

Beweis. Ein Randereignis tritt genau dann ein, wenn die Sichtbarkeitsgerade, die den Punkt v_i tangiert, einen Eckpunkt p von $P \setminus \pi(s,t)$ (möglicherweise als Endpunkt) enthält. Aus Eigenschaft E_3 folgt, daß $\overline{v_i p}$ eine Kante des Shortest-Path-Tree SPT_s ist. Das bedeutet, daß alle Eckpunkte von P, deren Elternknoten in SPT_s Punkte von $\pi(s,t)$ sind, potentielle Randereignisse sind. Um nun alle Randereignisse zu berechnen, wandert man zwischen zwei aufeinanderfolgenden Pfadereignissen auf ∂P entlang und kontrolliert für jeden Eckpunkt p von P, ob sein Elternknoten in SPT_s der Drehpunkt v_i ist. Falls dies so ist und p der Endpunkt von l^+ ist, kann p als einer der beiden Endpunkte des Randereignisses gespeichert werden. Dies ist z.B. bei p_1 in Abbildung 2.8 der Fall. Wenn p nicht der Endpunkt von l^+ ist, kann dieser anhand von SPM_s gefunden werden. Dies ist z.B. bei p_2 in Abbildung 2.8 der Fall. Die Randereignisse für l^- können analog berechnet werden.

Nun muss noch für jedes gefundene Randereignis der gegenüberliegende Endpunkt $\tilde{p}$ der Sichtbarkeitsgerade durch p und v_i auf ∂P berechnet wer-

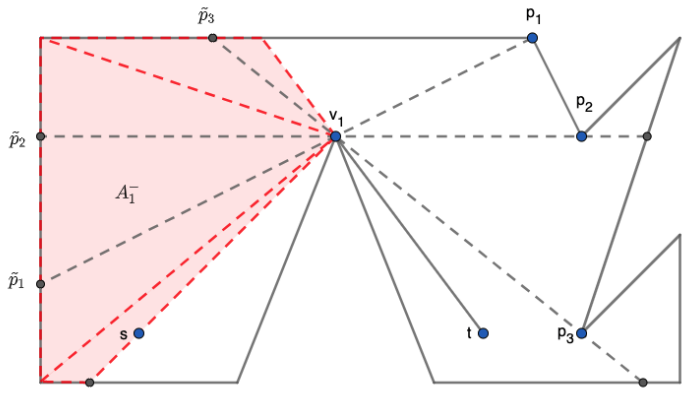

Abbildung 2.9: Die gegenüberliegenden Endpunkte der Randereignisse können anhand der Zellen von SPM_t inzident zum Drehpunkt gefunden werden.

den. Dazu kann man die dreieckigen Zellen der Shortest Path Map SPM_t, die zu v_i inzident sind und zwischen den gerade betrachteten Pfaderegnissen liegen, benutzen. Man findet die Zelle, die den gegenüberliegenden Endpunkt für das erste Grenzereignis enthält, und berechnet den Schnittpunkt auf ∂P. In Abbildung 2.9 ist das $\tilde{p_1}$. Dort wird ein Randereignis gespeichert. Dann sucht man in derselben Zelle den gegenüberliegenden Punkt für das nächste Randereignis. Wenn ein gegenüberliegender Punkt in der gerade betrachteten Zelle nicht gefunden werden kann, geht man zur nächsten Zelle und sucht dort. So findet man etwa $\tilde{p_3}$. Das wird wiederholt, bis alle gegenüberliegenden Endpunkte der Randereignisse gefunden sind. Da jede Zelle von SPM_t maximal einmal von l^- überstreift wird, muss man in $\mathcal{O}(n)$ Zellen nach Schnittpunkten suchen. Da die Kante der Zelle von SPM_t, die auf ∂P liegt, bekannt ist, kann der Schnittpunkt mit einer Geraden durch p und v_i in konstanter Zeit berechnet werden. Insgesamt muss man nach $\mathcal{O}(n)$ Schnittpunkten suchen, so daß sich eine Zeitkomplexität von $\mathcal{O}(n)$ ergibt.

Die Randereignisse, auf die l^- trifft, können analog berechnet werden. D.h. man wandert zwischen zwei Pfadereignissen auf ∂P in der entgegengesetzten Richung entlang und überprüft für jeden Knotenpunkt von P SPT_t. Die gegenüberliegenden Punkte können anhand von SPM_s gefunden werden.

Da alle Berechnungen in $\mathcal{O}(n)$ zu bewältigen sind, ergibt sich für die Berechnung der Pfad- und Randereignisse eine Gesamtkomplexität von $\mathcal{O}(n)$.

□

2.3.2 Berechnung der Biegungsereignisse

Vorbemerkungen

Nachdem die Pfad- und Randereignisse berechnet sind, können die Biegungsereignisse behandelt werden. Da der kürzeste Pfad von $\pi(s, l)$ nach Lemma 3 auf l^- endet und $\pi(t, l)$ auf l^+, wird im folgenden nur der Pfad $\pi(s, l^-)$ behandelt. Für $\pi(t, l^+)$ gelten alle Ergebnisse durch eine entsprechende Anpassung.

Wie weiter oben schon erwähnt, können die folgenden Biegungsereignisse unterschieden werden können:

- Ein Eckpunkt $p \in P$ 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 Knotenpunkt 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 $l^-_{p^*}$ 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 definieren somit die Regionen, die von l^- überstreift werden und die zur Berechnung der Biegungsereignisse betrachtet werden müssen.

Um die Biegungsereignisse zu berechnen, beginnt man bei der Sichtbarkeitsgerade l, die durch s und v_1 definiert ist, und betrachtet alle Pfad- und

Randereignisse in der Reihenfolge, in der sie auf ∂P auftreten. u sei dabei der letzte Knotenpunkt auf $\pi(s, l^-)$, bevor der Pfad l^- erreicht. Zu Beginn der Berechnungen wird der kürzeste Pfad zur Sichtbarkeitsgerade $\pi(s, l^-)$ mit dem Punkt s als Startpunkt initialisiert. Bei jedem Biegungsereignis, bei dem entweder Punkte hinzukommen oder verlorengehen, wird er aktualisiert. Degenerierte Biegungereignisse werden auch auf dem Polygonrand eingetragen, damit die Berechnung der Distanzen 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 l_s^l zusammen.

Nach diesen Vorüberlegungen untersuche ich, welche Biegungsereignisse mit einem Pfad- oder Randereignis zusammenfallen können. Dann werden die Biegungsereignisse besprochen, die zwischen aufeinanderfolgenden Pfad- bzw. Randereignissen liegen. Als letztes werden Degenerierte Biegungsereignisse untersucht. Es folgt ein Lemma, das beweist, daß alle Biegungsereignisse in Zeitkomplexität $\mathcal{O}(n)$ berechnet werden können.

Biegungsereignisse bei Pfad- und Randereignissen

Lemma 8. *Bei einem Pfadereignis ändern sich die Knotenpunkte von $\pi(s, l^-)$ nur, wenn sich die Drehrichtung von l ändert. In diesem Fall ist der neue kürzeste Pfad zum Knotenpunkt vor l^- der kürzeste Pfad von s zum bisherigen Drehpunkt v.*

Beweis. v' sei der nächste Drehpunkt auf $\pi(s, t)$ und $\overline{p^- p^+}$ sei die maximale Strecke $\in P$, die $\overline{vv'}$ enthält. Falls l an v' inifinitesimal in die bisherige Richtung weitergedreht wird, dann ändert sich der bisherige Weg zu l^- nur infinitesimal. Daher bleibt u der letzte Punkt vor l^- (siehe Abbildung 2.10 (a)).

Wenn l hingegen in die andere Richtung gedreht wird, betrachte man die Kante zwischen den Knoten, die das Pfadereignis auslösen, also z.B. $\overline{vv'}$ in Abbildung 2.10 (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 aufgrund von Eigenschaft E_2 alle in P^t. Daraus folgt, daß für jeden kürzesten Pfad $\pi(s, l^-)$ mit $l^- \in P^t$ 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 $l^- \in P^t$: $\pi(s, v) \subset \pi(s, l^-)$. □

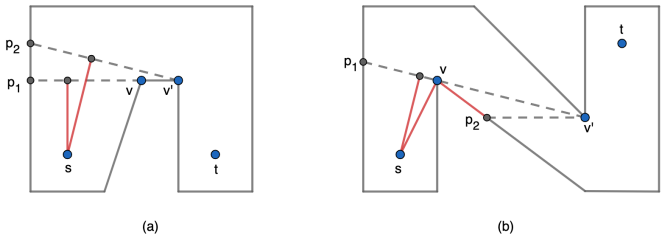

Abbildung 2.10: In Abbildung (a) bleibt s nach dem Pfadereignis der letzte Punkt vor l^-. In Abbildung (b) wird beim Pfadereignis v der letzte Punkt vor l^-.

Für Randereignisse gilt das folgende Lemma:

Lemma 9. *Bei einem Randereignis, das durch einen Knotenpunkt $p \in P \setminus \pi(s,t)$ ausgelöst wird, ändern sich die Knotenpunkte 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 Knotenpunkt u hinzugefügt.*

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 Punkte von $\pi(s,l^-)$ nicht.

Im folgenden wird daher angenommen, daß p der Scheitelpunkt von $S_p \in SPM_s$ ist. Nach Lemma 5 liegt der optimale Punkt s^* bei weiterer Drehung von l nur dann in S_p, wenn $l_p^{l^-} \in S_p$ oder wenn $l_p^{l^-}$ 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 Punkte von $\pi(s,l^-)$ nicht, siehe auch Abbildung 2.11.

□

Der Vollständigkeit halber sei darauf hingewiesen, daß bei der Entscheidung, ob der Knotenpunkt 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 i^* vor dem Randereignis auf der Polygonkante, deren Anfangspunkt u ist. Diese Situation ist in Abbildung 2.11 dargestellt.

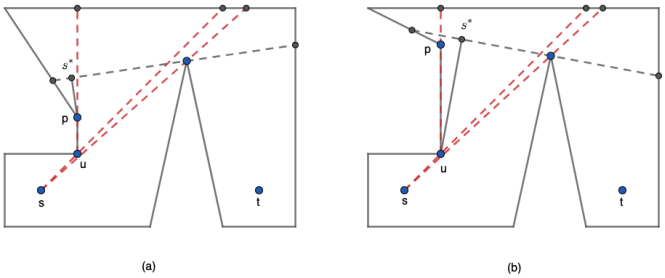

Abbildung 2.11: In Abbildung (a) liegt s^* nach dem Randereignis bei p in S_p und p kommt zu $\pi(s, l^-)$ hinzu. In Abbildung (b) bleibt s^* nach dem Randereignis bei p in S_u und die inneren Punkte von $\pi(s, l^-)$ ändern sich nicht.

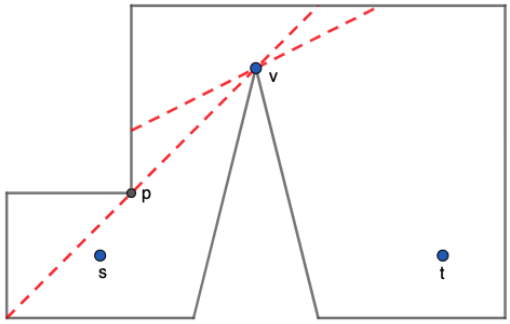

Abbildung 2.12: Vor dem Randereignis bei p verläuft $\pi(s, l^-)$ im Polygoninneren von s zu $l_s^{l^-}$. Der Knotenpunkt p ragt so in das Polygoninnere hinein, daß nach dem Randereignis der Weg von s nach $l_s^{l^-}$ versperrt ist, so daß p zu $\pi(s, l^-)$ hinzugefügt werden muss.

2. Im zweiten Fall verläuft der Pfad von u zu i^* vor dem Randereignis im Polygoninneren. p wird in diesem Fall zu $\pi(s, l^-)$ nur dann hinzugefügt, wenn der Knotenpunkt p so in das Polygon hineinragt, daß er den direkten Weg zu $l_u^{l^-}$ versperrt, siehe 2.12.

Biegungsereignisse zwischen Pfad- und Randereignissen

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

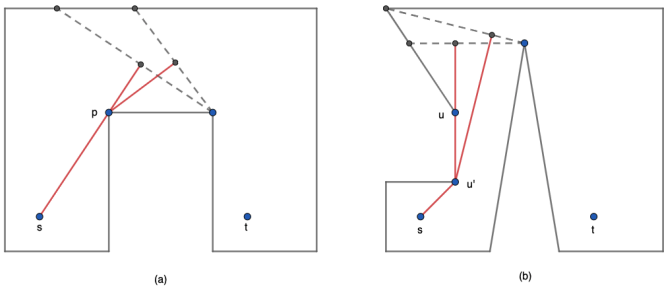

Abbildung 2.13: In Abbildung (a) trifft $\overline{us^*}$ auf Punkt p, der zu $\pi(s, l^-)$ hinzugefügt wird. In Abbildung (b) werden $\overline{us^*}$ und $\overline{u'u}$ parallel und u wird von $\pi(s, l^-)$ entfernt.

Wie schon erwähnt, ändern sich die inneren Punkte 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 $\pi(s, l^-)$ auf einen Punkt $p \in P$ und i^* liegt im folgenden in S_p, siehe Abbildung 2.13 (a). 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 2.13 (b).

Für den ersten Fall, gilt folgendes Lemma:

Lemma 10. *Wenn $\pi(u, l^-)$ zwischen zwei aufeinanderfolgenden Pfad- oder Grenzereignissen auf einen Knotenpunkt $p \in P$ trifft, dann muss dieser Knotenpunkt 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 [8]). 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 Streckenendpunkte (siehe Abbildung 2.14). Der erste Knotenpunkt, den e_l 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.

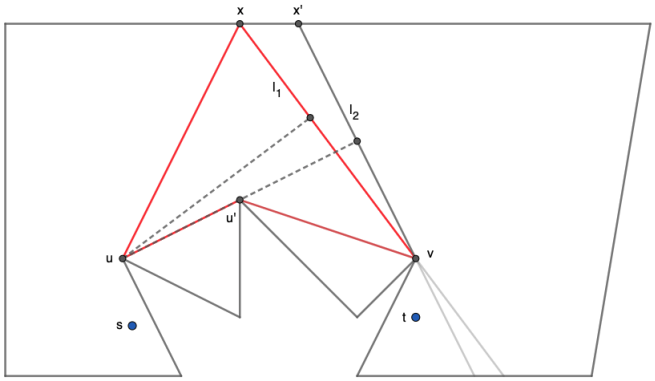

Abbildung 2.14: Der Trichter von u zu l_1 is rot eingezeichnet.

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 Knotenpunkt $p \in P$ trifft, ist folgendes zu beachten:

1. wenn der Endpunkt von $\pi(u, l^-)$ auf einen Knotenpunkt $p \notin \pi(s, v)$ trifft, definiert p ein Randereignis, so daß Lemma 9 gilt.

2. wenn der Endpunkt von $\pi(u, l^-)$ auf einen Knotenpunkt $p \in \pi(s, v)$ trifft, dann muß dieser Knotenpunkt nach Lemma 10 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 is ja Teil der sich drehenden Sichtbarkeitsgeraden (siehe Abbildung 2.15).

Der zweite Fall, bei dem sich die inneren Punkte 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:

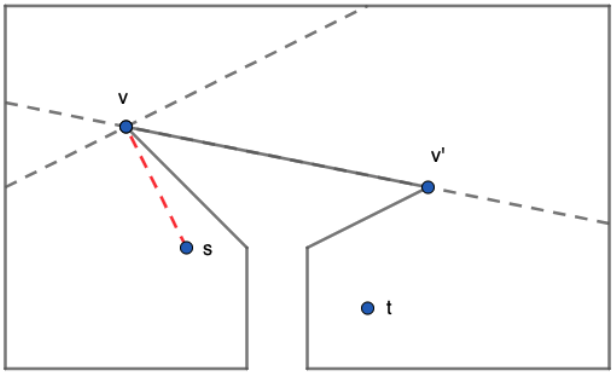

Abbildung 2.15: Der Drehpunkt v ist zwischen den eingezeichneten Sichbarkeitsgeraden der Endpunkt von $\pi(s, l^-)$.

Lemma 11. *Ein Punkt $u \in \pi(s,t)$, kann vom Pfad $\pi(s, l^-)$ nicht wieder verschwinden, wenn er einmal hinzugefügt worden ist.*

Beweis. Ein Punkt $u \in \pi(s,t)$ kann nach Lemmata 8 und 10 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 Knotenpunkt $u \in \pi(s,t)$.

Im ersten Fall besagt Lemma 8, daß der hinzugefügte Punkt nicht mehr verschwinden kann. Im zweiten Fall gilt nach Lemma 10: wenn der hinzugefügte Punkt u wieder vom kürzesten Pfade verschwindet, muss es einen Trichter zwischen seinem Vorgänger u' auf $\pi(s, l^-)$ und einer 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 Punkt nicht wieder vom Pfad $\pi(s, l^-)$ verschwinden. □

Lemma 12. *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 11 können Punkte $p \in \pi(s, t)$ nicht mehr von $\pi(s, l^-)$ verschwinden. Also müssen nur die Punkte betrachtet werden, die nicht zu $\pi(s, t)$ gehören und zu $\pi(s, l^-)$ hinzugefügt werden.

Nach Lemma 9 werden Knotenpunkte $p \notin \pi(s, t)$ bei Randereignissen zu $\pi(s, l^-)$ hinzugefügt, wenn für s^* bei weiterer Drehung von l gilt: $s^* \in S_p$. Wegen Beobachtung E_1 und E_2, d.h. weil die von l überstreiften Regionen paarweise disjunkt sind, können diese Randereignisse bei Drehung der Sichtbarkeitsgeraden nur einmal angetroffen werden. Das heißt, daß ein Punkt $p \notin \pi(s, t)$ nur einmal zu $\pi(s, l^-)$ hinzugefügt 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 Biegungseregnisse*. Bei diesen werden dem kürzesten Pfad zur Sichtbarkeitsgeraden keine Knotenpunkte hinzugefügt, bzw. weggenommen, aber die Polygonkante beginnt bzw. hört auf, den direkten Weg von u zum Lotfußpunkt $l_u^{l^-}$ zu versperren (siehe Abbildung 2.16 als Beispiel). Degenerierte Biegungsereignisse lassen sich folgendermaßen berechnen: die Lotfußpunkte $l_u^{l^-}$ bilden nach dem Thalessatz einen Halbkreis h_{uv} über u und den 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 $l_u^{l^-}$ zu versperren. Diese Entscheidung kann in konstanter Zeit durchgeführt werden, so daß sich für die Berechnung der degenerierten Biegungsereignisse eine Gesamtkomplexität von $\mathcal{O}(n)$ ergibt.

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

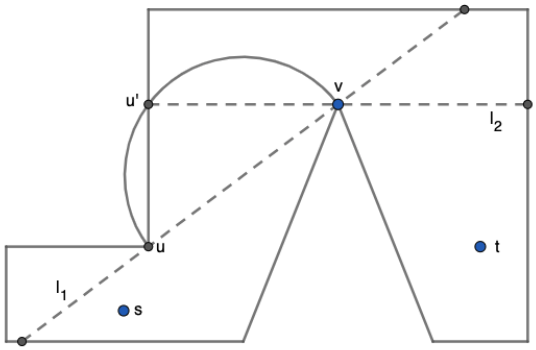

Abbildung 2.16: Ab dem Randereignis, das bei u eintritt, verläuft der kürzeste Pfad von u nach l^- zunächst auf der Polygonkante. Bei u′ tritt ein degeneriertes Biegungsereignis ein, ab hier ist der Weg von u nach l^- die Strecke $\overline{ul_u^{l^-}}$, die im Polygoninneren verläuft.

Komplexität der Berechnung aller Biegungsereignisse

Lemma 13. *Alle degenerierten und nicht degenerierten Biegungsereignisse können in der Reihenfolge, in der sie auf ∂P auftreten, in $\mathcal{O}(n)$ Zeit berechnet werden.*

Beweis. Ein Biegungsereignis fällt mit einem Pfadereignis zusammen, wenn sich die Drehrichtung beim Pfadereignis ändert. Hier sei v' der Nachfolger von v und v'' der Nachfolger von v' auf $\pi(s,t)$. Um die Drehrichtung an v' zu bestimmen, muss berechnet werden, in welcher der beiden Halbebenen, die durch die Gerade durch v und v' aufgespannt werden, v'' liegt. Dies ist in konstanter Zeit möglich.

Ein Biegungsereignis fällt mit einem von $p \in P$ ausgelösten Randereignis zusammen, wenn s^* bei weiterer Drehung von l in S_p liegt. Dies kann in konstanter Zeit entschieden werden.

Da es $\mathcal{O}(n)$ Pfad- und Randereignisse gibt, kann in $\mathcal{O}(n)$ Zeit entschieden werden, ob die Pfad- und Randereignisse mit einem Biegungsereignis zusammenfallen.

Somit bleiben noch die Biegungsereignisse die zwischen zwei aufeinanderfolgenden Pfad- bzw. Randereignissen auftreten. Für die Reihenfolge der potentiellen Biegungsereignisse gilt folgendes: nach Lemma 10 kann ein Kno-

tenpunkt $p \in \pi(s,t)$ nur zu $\pi(s,l^-)$ hinzukommen, wenn $u \in \pi(s,t)$. Wenn für den letzten Knotenpunkt 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 Punkte verlieren muss, bevor ein zu $\pi(s,t)$ gehörender Punkt hinzugefügt werden kann.

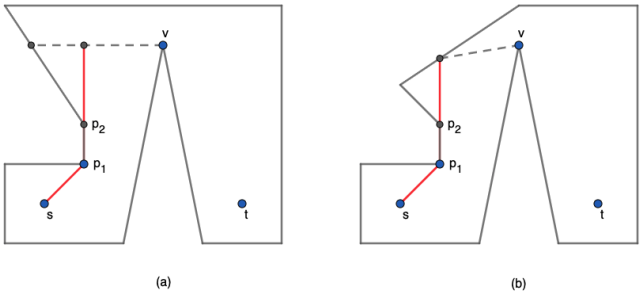

Abbildung 2.17: In Abbildung (a) steht die Gerade durch p_1 und p_2 senkrecht auf einer Sichtbarkeitsgeraden, die die gerade überfegte Polygonkante schneidet. In Abbildung (b) schneidet die Gerade durch p_1 und p_2 die gerade überfegte Polygonkante. In beiden Fällen geht p_2 verloren.

Um zu überprüfen, ob zwischen einem Start- und Endereignis ein Knotenpunkt verloren geht, kann man folgendermassen vorgehen: wenn u nicht zu $\pi(s,t)$ gehört und einen Vorgänger auf $\pi(s,l^-)$ hat, betrachtet man die Gerade durch u und diesen Vorgänger. Wenn diese Gerade entweder auf einer Sichtbarkeitsgerade, die die gerade überfegte Polygonkante schneidet, senkrecht steht, oder aber die gerade überfegte Polygonkante schneidet, verliert der Pfad den Punkt u (siehe Abbildung 2.17). Diese Überprüfung kann in konstanter Zeit erfolgen. Ein Punkt p kann nach Lemma 12 nur einmal von $\pi(s,l^-)$ entfernt werden. Ausserdem gibt es $\mathcal{O}(n)$ Randereignisse, bei denen ein Punkt $p \notin \pi(s,t)$ zu $\pi(s,l^-)$ hinzukommen kann. Insgesamt gibt es daher $\mathcal{O}(n)$ Biegungsereignisse, bei denen ein Punkt verloren geht, und ihre Berechnung hat insgesamt die Zeitkomplexität $\mathcal{O}(n)$.

Auch bei der Überprüfung, ob der Nachfolger u' von u auf $\pi(s,t)$ zu $\pi(s,l^-)$ hinzukommt, muss eine Fallunterscheidung gemacht werden: hier betrachtet man die Gerade durch u und u'. Wenn diese Gerade entweder auf einer Sichtbarkeitsgeraden, die die gerade überfegte Polygonkante schneidet,

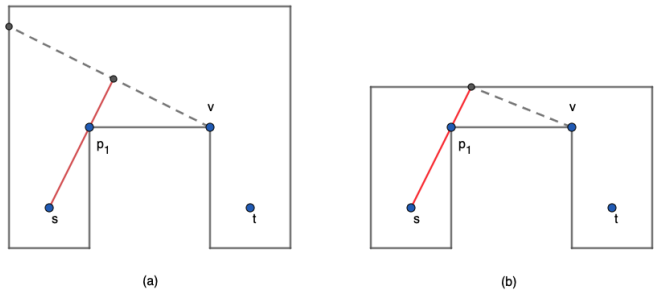

Abbildung 2.18: In Abbildung (a) steht die Gerade durch s und p_1 senkrecht auf einer Sichtbarkeitsgeraden, die die gerade überfegte Polygonkante schneidet. In Abbildung (b) schneidet die Gerade durch s und p_1 die gerade überfegte Polygonkante. In beiden Fällen kommt p_1 zum Pfad dazu.

senkrecht steht, oder aber die gerade überfegte Polygonkante schneidet, gewinnt der Pfad den Knotenpunkt u' hinzu (siehe Abbildung 2.18). Diese Überprüfung kann in konstanter Zeit erfolgen. Nach Lemma 11 kann ein Knotenpunkt $p \in \pi(s,t)$ nur einmal zu $\pi(s,l^-)$ hinzukommen. Da es $\mathcal{O}(n)$ Knotenpunkte von $\pi(s,t)$ gibt, ist die Gesamtkomplexität der Berechnung von Biegungsereignissen, bei denen ein Knotenpunkt $p \in \pi(s,t)$ zu $\pi(s,l^-)$ hinzukommt, $\mathcal{O}(n)$.

Auch die Berechnung der degenerierten Biegungsereignisse kann in $\mathcal{O}(n)$ Zeit erfolgen. Hier 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. Da es $\mathcal{O}(n)$ solcher Ereignisse gibt, liegt die Komplexität der Berechnung aller degenerierten Biegungsereignisse in $\mathcal{O}(n)$.

Für die Berechnung der Biegungsereignisse auf l^+ ist ein zweiter Durchlauf nötig, der bei t beginnt. Auch bei diesem Durchlauf liegt die Komplexität der Berechnungen bei $\mathcal{O}(n)$, so daß alle Biegungsereignisse in Zeitkomplexität $\mathcal{O}(n)$ berechnet werdn können. □

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 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 der Lotfußpunkt l_u^l von u aus sichtbar ist 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 l_u^l 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 Eregnisse definieren, 90° ist. Denn spätestens, wenn der Drehwinkel 90° erreicht, tritt ein Biegungsereignis ein, bei dem der aktuelle Drehpunkt Teil des kürzesten Pfads zur Sichtbarkeitslinie wird. Im folgenden wird daher - wenn nichts anderes erwähnt ist - $\rho \leq 90°$ angenommen.

2.4.1 Vorbemerkungen

In den folgenden Abschnitten bezeichnen u und u' die vorletzten Knoten auf den Pfaden $\pi(s, l^-)$ bzw. $\pi(t, l^+)$, also die Knoten der Pfade, bevor die gemeinsame Sichtbarkeitsgerade l erreicht wird. v ist der Knotenpunkt, an dem die Sichtbarkeitsgerade gedreht wird. l_1 sei die Sichtbarkeitsgerade durch u und v, ihre Steigung sei α_1, l_2 sei die Sichtbarkeitsgerade durch u' und v, ihre Steigung sei 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. Wenn man eine fixe Sichtbarkeitsgerade l' durch v betrachtet, dann ist der kürzeste Weg von u nach l' die Strecke $\overline{ul'_u}$. Ebenso ist der kürzeste Weg von u' nach l' die Strecke $\overline{u'l'_{u'}}$. 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 (siehe Abbildung 2.19).

Wenn man in einem Polygon beliebige Knoten u, u' und v betrachtet, dann können diese in den unterschiedlichsten Positionen liegen. Allerdings ist es immer möglich das Polygon so zu verschieben und zu drehen, daß u in Quadrant 3 liegt, 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. Dadurch kann bei der Darstellung der Beweise auf unnötige Fallunterscheidungen verzichtet werden.

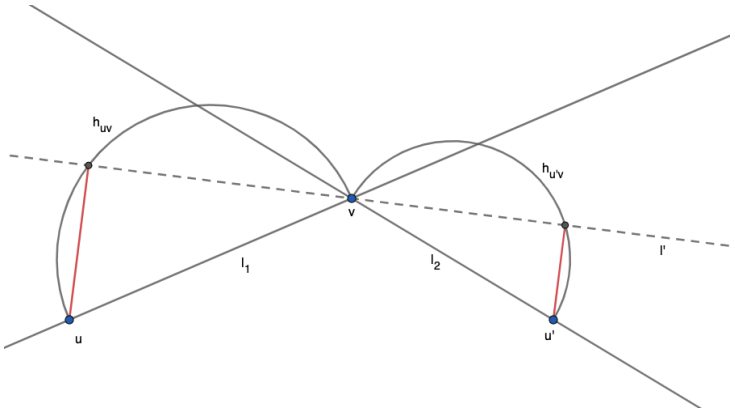

Abbildung 2.19: Die Endpunkte der kürzesten Wege von u (u') zu den Sichtbarkeitsgeraden durch v liegen auf den Halbkreisen h_{uv} ($h_{u'v}$).

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):

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

Für die Steigungen α_1 und α_2 gilt folgendes:

$$\alpha_1 = \frac{v.y - u.y}{v.x - u.x} \tag{2.3}$$

$$\alpha_2 = \frac{v.y - u'.y}{v.x - u'.x} \tag{2.4}$$

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 Variablen ersetzt, also $a = u.x - v.x$, $b = v.y - u.y$, $c = u'.x - v.xc$ 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:

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

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. Da die zweite Ableitung von $d(\alpha) = 0$, kann sie nicht zur Beantwortung der Frage benutzt werden. Statt dessen hilft das folgende Lemma, das besagt, daß der gemeinsame Pfad zu jeder Sichtbarkeitsgeraden zwischen l_1 und l_2 länger ist als der gemeinsame Pfad zu l_1 oder l_2:

Lemma 14. *Ohne Beschränkung der Allgemeinheit sei $\overline{uv} \leq \overline{u'v}$. Dann gilt für jede Sichtbarkeitsgerade l zwischen l_1 und l_2, d.h. für jedes l mit Steigung $\alpha : \alpha_2 < \alpha < \alpha_1$:*

$$\textit{(i)}\ |\pi(s, l_2)| + |\pi(t, u')| < |\pi(s, l)| + |\pi(t, l)| \tag{2.12}$$

Wenn $\overline{uv} = \overline{u'v}$ dann gilt außerdem:

$$\textit{(ii)}\ |\pi(s, l_2)| + |\pi(t, u')| = |\pi(s, u)| + |\pi(t, l_1)| \tag{2.13}$$

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.

Der zweite Teil des Lemmas (ii) ist einfach zu sehen. p_1 sei der Schnittpunkt von l_2 mit h_{uv} und p'_1 sei der Schnittpunkt von l_1 mit $h_{u'v}$. Dann sind die Dreiecke Δuvp_1 und $\Delta u'vp'_1$ kongruent, wenn $|\overline{uv}| = |\overline{u'v}|$. Damit gilt $|\overline{up_1}| = |\overline{u'p'_1}|$ und folglich $|\pi(s, l_2)| + |\pi(t, u')| = |\pi(s, u)| + |\pi(t, l_1)|$.

Für den Beweis von (i) wird zunächst angenommen, daß die Strecken $\overline{uv}$ und $\overline{u'v}$ gleich lang sind. Außerdem gelten folgende Definitionen (siehe auch Abbildung 2.20):

- c_1 ist der Mittelpunkt des Halbkreises h_{uv}
- c_2 ist der Mittelpunkt des Halbkreises $h_{u'v}$
- β ist der Winkel zwischen $l_u^2 l$, c_1 und l_u^l

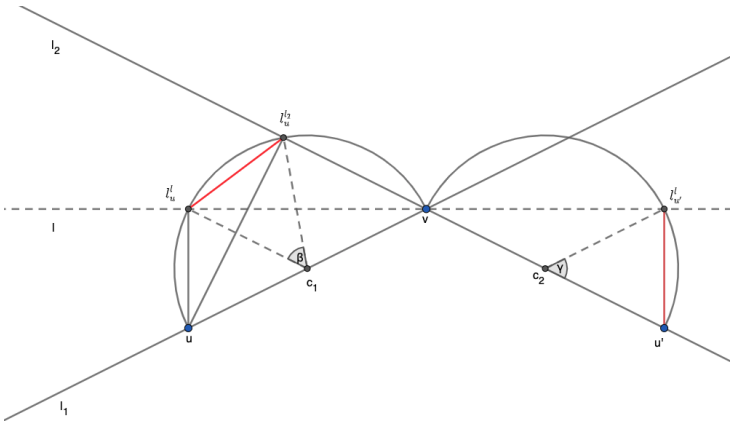

Abbildung 2.20: Die gemeinsame Distanz zu l_2, d.h. $|\overline{ul_u^{l_2}}|$ ist minimal.

- γ ist der Winkel zwischn u', c_2 und $l^l_{u'}$
- die Länge der Strecke zwischen l^l_u und $l^{l_2}_u$ ist d_1
- die Länge der Strecke zwischen u' und $l^l_{u'}$ 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 $|\overline{ul^l_u}| + d_2$. Die gemeinsame Distanz zu l_2 hingegen ist $|\overline{ul^{l_2}_u}|$. Beachte, daß auf jeden Fall gilt: $|\overline{ul^l_u}| + d_1 > |\overline{ul^{l_2}_u}|$. 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: $|\overline{ul^l_u}| + d_1 = |\overline{ul^l_u}| + d_2 > |\overline{ul^{l_2}_u}|$. 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 auch d_2 länger als d_1, und die Distanz $|\overline{ul^{l_2}_u}|$ ist wiederum minimal. □

Das bedeutet, daß der Wert der Distanzfunktion beim Nullpunkt von mindestens einer Seite von unten erreicht wird. Also ist er entweder ein Maximum oder ein Wendepunkt, aber auf keinen Fall ein Minimum. Mithilfe dieser Erkenntnis, kann auch die Annahme fallengelassen werden, daß u auf l_1 und u' auf l_2 liegt, und es gilt folgendes Lemma:

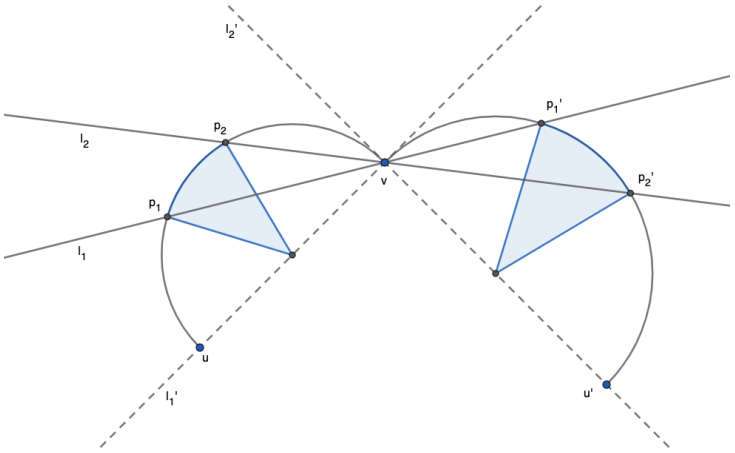

Abbildung 2.21: Die blau markierten Teile der Halbkreise sind von u bzw. u' aus sichtbar.

Lemma 15. *Der Halbkreis h_{uv} sei von u aus im Bereich zwischen l_1 und l_2 sichtbar. Der Schnittpunkt von l_1 mit h_{uv} sei p_1, der Schnittpunkt von l_2 mit h_{uv} sei p_2. Analog sei der Halbkreis $h_{u'v}$ von u' aus im Bereich zwischen l_1 und l_2 sichtbar. Der Schnittpunkt von l_1 mit $h_{u'v}$ sei p'_1, der Schnittpunkt von l_2 mit $h_{u'v}$ sei p'_2. Dann ist das lokale Minimum der gemeinsamen Distanzfunktion $|\pi(s,l)| + |\pi(t,l)|$ im Bereich zwischen l_1 und l_2 entweder $|\pi(s,p_1)| + |\pi(t,p'_1)|$ oder $|\pi(s,p_2)| + |\pi(t,p'_2)|$.*

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 2.21). 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 oder wenn der Nullpunkt ein Wendepunkt ist, dann muss die gemeinsame Distanz gemäß Lemma 14 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 $|\pi(s,p_2)| + |\pi(t,p'_2)|$ 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 konstant ab oder konstant zu. Das lokale Minimum ist also $|\pi(s, p_1)| + |\pi(t, p'_1)|$ oder $|\pi(s, p_2)| + |\pi(t, p'_2)|$. 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 konstant ab, wenn l vom Maximum weggedreht wird. Daher muss auch in diesem Fall das lokale Minimum entweder $|\pi(s, p_1)| + |\pi(t, p'_1)|$ oder $|\pi(s, p_2)| + |\pi(t, p'_2)|$ sein. □

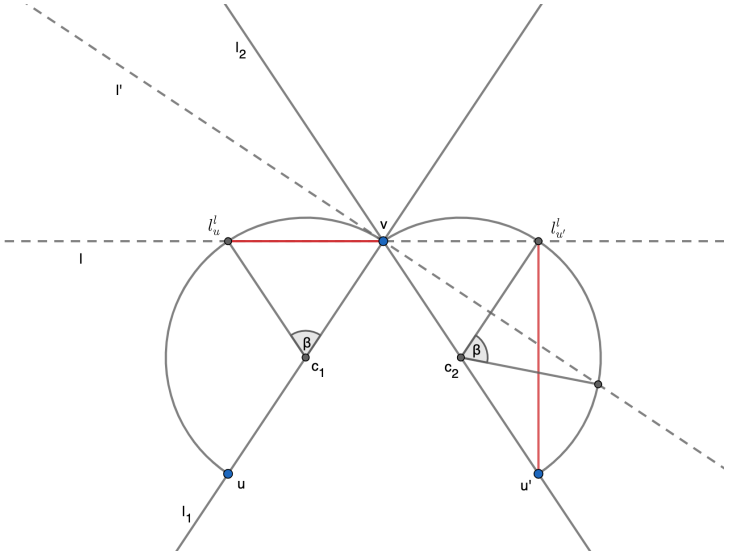

Abbildung 2.22: Bei l' tritt ein Biegungsereignis ein und v wird zu $\pi(s, l^-)$ hinzugefügt. Zwischen l_1 und l' ist $\overline{uv}$ minimal, weil für jedes l zwischen l_1 und l' die Strecke zwischen l^l_u und v kleiner ist als die Strecke zwischen $l^l_{u'}$ und u'.

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 Biegungesereignis gilt Lemma 14 (siehe Abbildung 2.22). 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 gleich sind, gibt es 2 lokale Minima, ansonsten nur eines. Damit gilt folgendes Lemma:

Lemma 16. *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

Im 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.

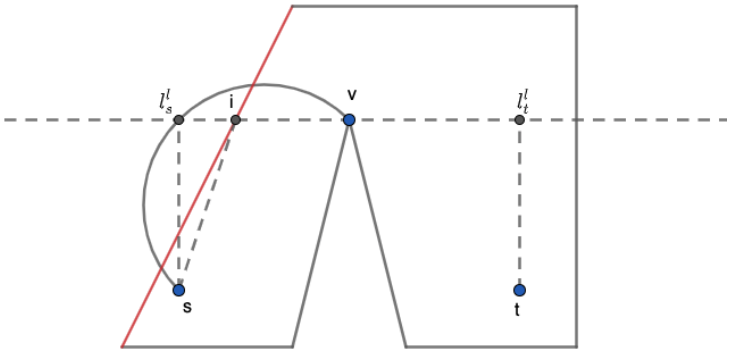

Abbildung 2.23: Die Pfade zu Schnittpunkten auf der Kante sind immer länger als die Pfade zu Schnittpunkten auf dem Halbkreis.

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 als die Distanz zum Schnittpunkt derselben Sichtbarkeitsgeraden mit einer im Halbkreis verlaufenden Polygonkante. In Abbildung 2.23 etwa versperrt die rot eingezeichnete Kante auf einer Teilstrecke den direkten Weg von s zu den Lotfußpunken l_s^l auf h_{sv}. i sei der Schnittpunkt von l mit der Polygonkante in diesem Bereich. Dann gilt: $|\overline{sl_s^l}| \leq |\overline{si}|$, wobei die Gleichheit nurin den zwei Schnittpunkten des Halbkreises mit der Polygonkante gegeben ist.

Nach Lemma 15 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 $l_u^{l_2}$ und von u' auf $l_{u'}^{l_1}$ frei ist, müssen daher keine Sichtbarkeitslinien 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 $l_u^{l_2}$ und/oder $l_{u'}^{l_1}$ verdeckt wie z.B. in Abbildung 2.24.

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:

$$y = \beta \cdot x + b \tag{2.14}$$

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

$$y = x \cdot \alpha + v.y - v.x \cdot \alpha \tag{2.15}$$

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

$$i.x(\alpha) = \frac{-v.x \cdot \alpha + v.y - b}{\beta - \alpha} \tag{2.16}$$

$$i.y(\alpha) = \frac{(-b - \beta \cdot v.x) \cdot \alpha + \beta \cdot v.y}{\beta - \alpha} \tag{2.17}$$

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

$$d_{ui}(\alpha) = \sqrt{(u.x - i.x)^2 + (u.y - i.y)^2} \tag{2.18}$$

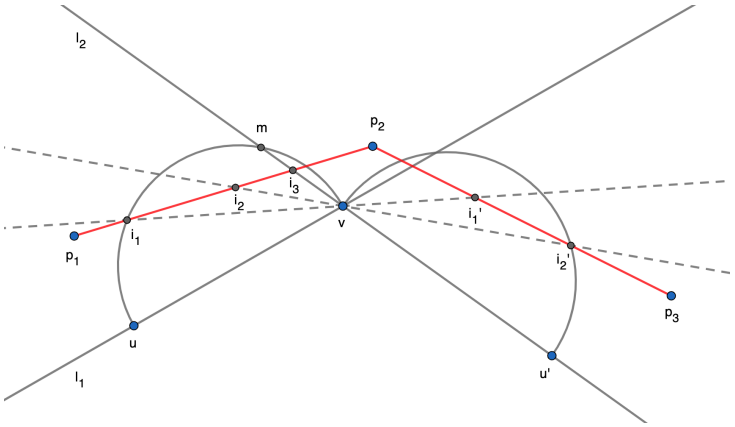

Abbildung 2.24: Die Polygonkanten zwischen p_1, p_2 und p_3 verdecken die Sichtbarkeit der Halbkreise.

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 2.24 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:

$$d(\alpha) = \sqrt{(u.x - i.x)^2 + (u.y - i.y)^2} + \frac{|(u'.x - v.x) \cdot \alpha + (v.y - u'.y)|}{\sqrt{(\alpha^2 + 1)}} \tag{2.19}$$

Zwischen der Geraden, die i_2 und i'_2 enthält, und der, 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 ist i' der Schnittpunkt von l^+ mit der Polygonkante $\overline{p_2 p_3}$. Die gemeinsame Distanz zu l ist:

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

Bei einer weiteren Drehung von l ist nur die Sicht von u' auf $h_{u'v}$ versperrt und Gleichung 2.19 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 den Halbkreis zu verdecken (z.B. bei i_1 oder i_2' in Abbildung 2.24), oder beim Schnittpunkt von l_1 bzw. l_2 mit der Polygonkante (z.B. bei i_3 in 2.24).

Die Antwort ist, daß dies nicht immer der Fall ist. Abbildung 2.25 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' zu $h_{u'v}$ 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 -0.379. Der andere Kandidat für das lokale Minimum ist die Sichtbarkeitsgerade durch i_3. Hier ist die gemeinsame Distanz aber etwas höher.

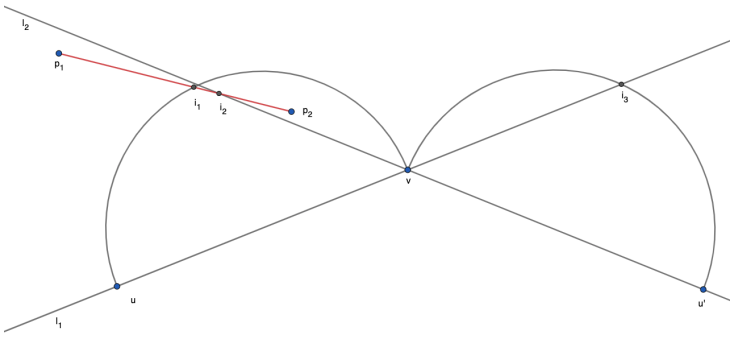

Abbildung 2.25: Das lokale Minimum liegt in dieser Konstellation bei einer Sichtbarkeitsgeraden durch einen Punkt zwischen i_1 und i_2.

Bei meinen Berechnungen von lokalen Minima in verschiedenen Konstellationen ist mir aufgefallen, daß ein Fall wie in Abbildung 2.25 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 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_1. Ich habe versucht, genauere Abschätzungen dafür zu finden, wie die Polygonkante verlaufen muss, damit das Minimum zwischen zwei Schnittpunkten

liegen kann wie in Abbilung 2.25, bzw. um wieviel das Minimum im Inneren der Polygonkante kleiner sein kann als die Distanz bei einem der Schnittpunkte. 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 2.19 bzw. 2.20 nach α ableiten. Potentielle Kandidaten für lokale Minima wären dann die Nullstellen dieser Ableitungen. Allerdings sind die ersten Ableitungen dieser Distanzfunktionen und die Ausdrücke für ihre Nullstellen so kompliziert, daß sie sich für eine praktische Implementierung des Algorithmus kaum eignen (siehe auch den Anhang). Daher benutze ich in meiner Implementierung einen approximativen Ansatz, um die lokalen Minima bei versperrenden Polygonkanten zu bestimmen. Wenn die Anzahl der Iterationen beim gewählten approximativen Ansatz durch eine konstante Zahl begrenzt ist, gilt sowohl für eine genaue als auch eine approximative Bestimmung der lokalen Minima das folgende Lemma:

Lemma 17. *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 untersucht, wie entschieden werden kann, ob die Lösung der min-max Version des Sichtbarkeitsproblems zwischen zwei aufeinanderfolgenden Ereignissen liegt und, wenn ja, bei welcher Sichtbarkeitsgeraden. Dabei ist v wieder der aktuelle Drehpunkt, u der letzte Knotenpunkt von $\pi(s, l)$ vor l und u' der letzte Knotenpunkt von $\pi(t, l)$ vor l. Das Hauptergebnis dieses Abschnitts ist, daß die Entscheidung und Berechnung der Sichtbarkeitsgeraden zwischen zwei Ereignissen in konstanter Zeit durchgeführt werden kann.

Bei der Berechnung der Minima helfen die folgenden beiden Lemmata:

Lemma 18. *l_1 und l_2 seien 2 aufeinanderfolgende Ereignisse, die auf dem kürzesten Pfad zwischen s und l auftreten. wobei l_1 von s aus gesehen früher*

eintritt. Dann wächst die Distanz $|\pi(s,l)|$ während der Drehung von l zwischen l_1 zu 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 beobachte man, daß ein Biegungsereignis eintritt, wenn der Drehwinkel ρ zwischen l und l_1 $90°$ erreicht. Ab diesem Biegungsereignis ist der kürzeste Weg von u nach l $\overline{uv}$, der sich auch bei weiterer Drehung von l nicht verändert. Der maximale Distanzzuwachs ist also $|\overline{uv}|$

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 l_u^l:

$$|\overline{ul_u^l}| = sin(\rho) \cdot |\overline{uv}| \tag{2.21}$$

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. In diesem Fall sei e die Polygonkante, die die Sicht auf h_{uv} versperrt, und g_e sei die Gerade, die die Polygonkante e enthält. Der von l zuerst erreichte Schnittpunkt von g_e und h_{uv} sei i. Dann muß e steiler sein als die Kante $\overline{iv}$. Denn wenn das nicht der Fall ist, muß auf e eine weitere, steilere Kante folgen, damit das Polygon einfach bleibt. Das aber bedeutet, daß e von v aus nicht sichtbar ist und für die Berechnung des Minimums keine Bedeutung hat, siehe Abbildung 2.26. Deshalb müssen für den Beweis nur solche Polygonkanten e betrachtet werden, die bei Eintritt in h_{uv} steiler sind als $\overline{iv}$. Wenn dies der Fall ist, liegt der Lotfußpunkt $l_u^{g_e}$ außerhalb von h_{uv} und der Endpunkt des kürzesten Wegs von u nach e bewegt sich während der Drehung von l von $l_u^{g_e}$ weg. Nach Lemma 4 steigt daher seine Länge kontinuierlich an.

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

□

Lemma 18 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:

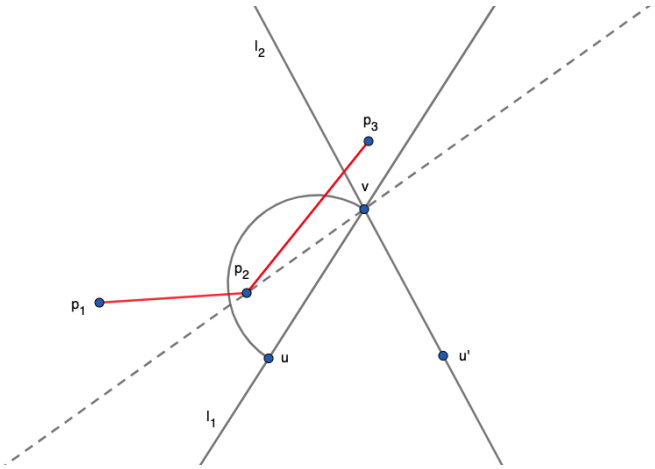

Abbildung 2.26: Die Polygonkante $\overline{p_1p_2}$ ist bei Eintritt in den Halbkreis nicht steil genug und ist daher von v aus nicht sichtbar. Für die Berechnung des Minimums spielt nur $\overline{p_2p_3}$ eine Rolle.

Lemma 19. *Die Länge des Pfads $\pi(s, l)$ wächst auf dem Weg von s nach t kontinuierlich. Ebenso wächst die Länge des Pfads $\pi(t, l)$ auf dem Weg von t nach s kontinuierlich.*

Beweis. Der Beweis ist für beide Richtungen gleich, daher wird das Lemma nur für die Richtung von s nach t bewiesen.

Wegen Lemma 18 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 Knotenpunkte von $\pi(s, l)$ ändern: Wenn kein neuer Punkt zu $\pi(s, l)$ hinzu- oder wegkommt, ändert sich die Länge von $\pi(s, l)$ bei dem Ereignis nicht. Wenn ein neuer Punkt zu $\pi(s, l)$ hinzukommt, kann die Länge von $\pi(s, l)$ nur größer werden. Wenn ein Punkt u von $\pi(s, l)$ entfernt wird und sein Vorgänger u' der letzte Punkt 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 $\pi(s, l)$ nich ändert. □

Da $|\pi(s, l)|$ in Richtung von s nach t kontinuierlich wächst, und $|\pi(t, l)|$ in Richtung von t nach s, liegt die 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:

$$|\pi(s,u)| \leq |\pi(t,u')| + |\pi(u',l_1)| \tag{2.22}$$

$$|\pi(t,u')| \leq |\pi(s,u)| + |\pi(u,l_2)| \tag{2.23}$$

Um genau zu entscheiden, wo die Länge der Pfade gleich wird, müssen die korrekten Distanzfuntionen für die Strecken $\overline{ul}$ und $\overline{u'l}$ gewählt werden, je nachdem ob h_{uv} und $h_{u'v}$ von u bzw. u' sichtbar sind oder nicht. Diese Distanzfunktionen können in Anhang D gefunden werden. Sie spielen bei der praktischen Implementierung des Algorithmus keine Rolle, da sie zu kompliziert für den praktischen Gebrauch sind.

Die Lösung des min-max Problems weist folgende Besonderheit auf: wenn die Gleichheit der Distanzen zwischen zwei Ereignissen eintritt, wo einer der Pfade $\pi(s,l)$ oder $\pi(t,l)$ auf dem Drehpunkt v endet, dann ist die Lösung des min-max Problems auf der anderen Seite nicht eindeutig bestimmt. Wenn etwa $\pi(s,l)$ bei v endet, kann der Pfad, der bei t beginnt, bei jedem Punkt t', für den gilt: $|\pi(t,v')| \leq |\pi(s,v)|$, siehe 2.27.

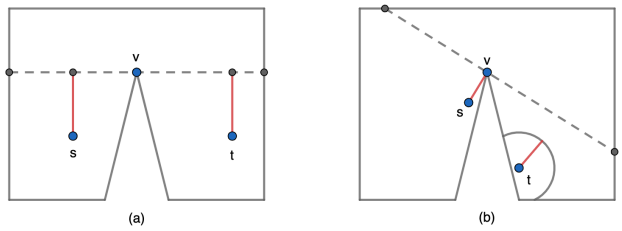

Abbildung 2.27: In Abbildung (a) befindet sich die Lösung des min-max Problems auf der eingezeichneten Sichtbarkeitsgerade, in Abbildung (b) kann der Pfad, der bei t beginnt, bei jedem Punkt innerhalb des eingezeichneten Kreisausschnitts enden.

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

Lemma 20. *Die Entscheidung, ob eine Lösung des min-max Problems zwischen zwei Ereignissen l_1 und l_2 liegt, kann in konstanter Zeit getroffen werden. Wenn die Entscheidung positiv ausfällt, kann die Sichtbarkeitsgerade l, bei der die Lösung liegt, in konstanter Zeit gefunden werden.*

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 Sichtbarkeit auf die Lotfußpunkte $l_{u'}^{l_1}$ und $l_u^{l_2}$ frei ist oder nicht. In beiden Fällen, ist die Berechnung der Länge in konstanter Zeit möglich.

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: $|\pi(s,u)| + |\pi(u,l)| = |\pi(t,u')| + |\pi(u'l)|$. Statt einer genauen Lösung, kann auch ein approximativer Ansatz verwendet werden. Solange die Anzahl der Iterationen dabei durch eine konstante Zahl begrenzt ist, kann die Lösung in beiden Fällen in konstanter Zeit gefunden werden. □

2.6 Komplexität des paarweisen Sichtbarkeitsproblems

Für die Berechnung der lokalen Minima zwischen zwei aufeinanderfolgenden Ereignissen gilt folgendes Lemma, das die Lemmata 16, 17 und 20 zusammenfaßt:

Lemma 21. *Die Sichtbarkeitsgeraden l, die zwischen zwei aufeinanderfolgenden Ereignissen bzw. auf einem der Ereignisse verlaufen und eine Lösung des min-sum bzw. min-max Problems sind, können in konstanter Zeit gefunden werden.*

Beweis. Degenerierte Biegungsereignisse treten ein, wenn eine Polygonkante anfängt bzw. aufhört, die Sicht von u bzw. u' zu h_{uv} bzw. $h_{u'v}$ zu versperren. Bei der Berechnung einer Lösung für das min-sum Problem weiß man daher bei jedem Ereignis, welche der in Kapitel 2.4 beschriebenen Distanzfunktionen zu verwenden sind. Wenn weder die Sicht von u noch von u' auf die Halbkreise versperrt ist, ist es ausreichend, die gemeinsamen Distanzen bei den zwei aufeinanderfolgenden Ereignissen zu überprüfen. Das lokale Minimum, bzw. die lokalen Minima können nur dort liegen. Dies ist in konstanter Zeit möglich. Wenn die Sicht von u, u' oder von beiden versperrt ist, muss man entweder die Distanzfunktion 2.19 oder 2.20 benutzen, um das Minimum zu bestimmen. Theoretisch kann man dafür elementare Analysis anwenden, was in konstanter Zeit möglich ist. Bei einer praktischen Implementierung

des Algorithmus ist dies aufgrund der Kompliziertheit der Ableitungen nur bedingt möglich und eine approximative Bestimmung der lokalen Minima ist notwendig. Solange die Anzahl der Iterationen bei dem gewählten approximativen Ansatz beschränkt ist, ist auch die approximative Bestimmung der lokalen Minima in konstanter Zeit möglich.

Für das min-max Problem ist es nach Lemma 20 in konstanter Zeit möglich, zu entscheiden, ob die Lösung zwischen zwei aufeinanderfolgenden Ereignissen liegt. Falls diese Entscheidung positiv ausfällt, kann die Steigung der Sichtbarkeitsgeraden, bei der die Lösung liegt, ebenfalls in konstanter Zeit berechnet werden. □

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

Theorem 1. *Gegeben sei ein einfaches Polygon P mit n Knotenpunkten und keinen Löchern. 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, in $\mathcal{O}(n)$ Zeit berechnet werden.

Beweis. Für die Berechnung der Ereignisse werden der kürzeste Pfad zwischen s und t, die Shortest Path Maps und die Shortest Path Trees 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 werden können. Nach der Berechnung der Ereignisse muß zwischen je zwei aufeinanderfolgenden Ereignissen eine Lösung gesucht werden. 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 [11] unter Zuhilfenahme von CGAL [2] 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 Progroamm mit macOS Catalina und mit Ubuntu 18.04 getestet. Instruktionen zum Kompilieren des Projekts sind im Anhang E und auf github zu finden. Ich empfehle, das Programm auf Ubuntu zu kompilieren, da die Installation von CGAL und Qt 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 Programm 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

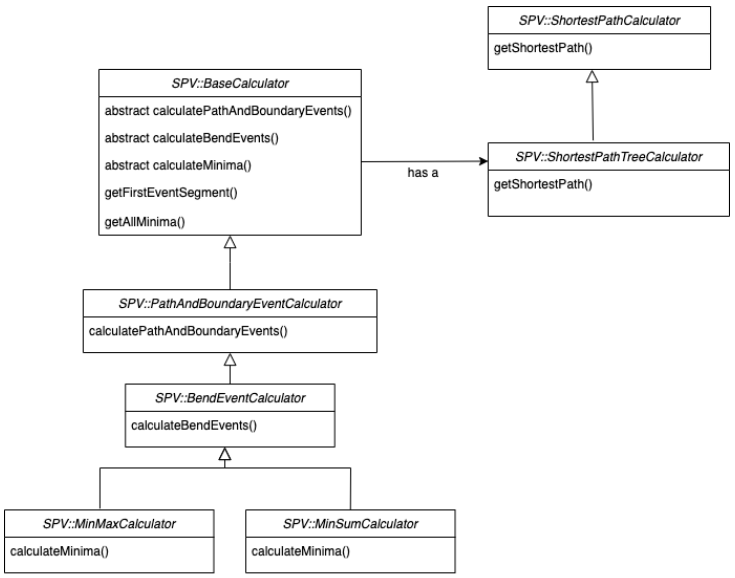

Abbildung 3.1: Die wichtigsten Klassen der Library

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 [8] beschrieben wird, den kürzesten Pfad vom Start- zum Endpunkt. Die dazu notwendige Triangulation des Polygons wird von CGAL berechnet.

2. Die Klasse *SPV::ShortestPathTreeCalculator* berechnet für jeden Punkt des kürzesten Pfads die Segmente auf dem Polygonrand, die von einer Sichtbarkeitsgeraden, die an dem Punkt gedreht wird, überstreift werden. Dazu wird der in [7] beschriebene Algorithmus benutzt (mit einigen zusätzlichen Fallunterscheidungen, die in dem Paper vernachlässigt wurden). Auch hier wird die CGAL Triangulation genutzt. Die überstreiften Segmente werden bei den einzelnen Punkten des kürzesten Pfads gespeichert. Da die relevanten Sichtbarkeitsgeraden lediglich diese Segmente überstreifen, ist es ausreichend, nur sie bei der Berechnung der Ereignisse zu betrachten.

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

4. Die Klasse *SPV::PathAndBoundaryEventCalculator* ist von *SPV::BaseCalculator* abgeleitet und implementiert die Funktion *calculatePathAndBoundaryEvents*, die die Pfad- und Randereignisse wie in dieser Arbeit beschrieben berechnet. Das Ergebnis dieser Berechnung ist eine doppelt verkettete Liste von *SPV::EventSegment* Instanzen (siehe VisibilityProblemLibrary/Models/eventsegment.h in [10]). 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 Function *calculateBendEvents*. Diese Funktion berechnet degenerierte und nicht-degenerierte Biegungsereignisse und fügt sie in die Liste der *SPV::EventSegment* Instanzen ein.

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 [10]) 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 der direkte Pfad zu einer Sichtbarkeitsgeraden von einer Polygonkante versperrt wird, wird das Minimum approximativ berechnet. Der dazu benutzte Algorithmus ist recht krude: das relevante Segment wird in 1000 Sichtbarkeitsgeraden aufgeteilt und die Distanz zu jeder dieser Geraden wird betrachtet. Die Anzahl der betrachteten Sichtbarkeitsgeraden wird durch die Variable *stepPrecision* (definiert als private Variable in VisibilityProblemLibrary/basecalculator.h in [10]) bestimmt, die im derzeitigen Code den willkürlichen Wert 0,001 hat. Wenn ein Polygon betrachtet wird, in dem sehr viele Sichtbarkeitsgeraden von Polygonkanten versperrt werden, kann diese Berechnung zu einer merklichen Verlangsamung des Programms führen. Durch Vergrössern des Wertes von *stepPrecision* kann diese Verlangsamung verringert werden, allerdings um den Preis, daß die Berechnungen nicht mehr so genau sind.

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.

Bei der Berechnung der Segmente, die von Sichtbarkeitsgeraden überstreift werden, und bei der Berechnung der Pfad- und Grenzereignisse war folgende Überlegung von Bedeutung: beim Aufbau der Shortest Path Map werden die Endpunkte der von Sichtbarkeitsgeraden überstreiften Segmente auf dem Polygonrand in Vektoren gespeichert, und zwar in der Reihenfolge, in der sie angetroffen werden. Zuerst werden anhand von SPM_s die Segmente mit dem Startpunkt als Wurzel berechnet, dann anhand von SPM_t die Segmente mit dem Endpunkt als Wurzel. Jeder der so erhaltenen Punkte auf dem Polygonrand definiert die eine Hälfte eines Rand- bzw. Pfadereignisses. Um die Pfad- und Grenzereignisse in der richtigen Reihenfolge zu berechnen, geht man den Vektor V_s, der anhand von SPM_s angefüllt wurde, von vorne nach

hinten durch, und den Vektor V_t, der anhand von SPM_t angefüllt wurde, von hinten nach vorne. Der erste Eintrag in V_s definiert mit dem letzten Eintrag von V_t das (von s aus gesehen) erste Pfadereignis. Der letzte Eintrag in V_s definiert mit dem ersten Eintrag von V_t das zweite Pfadereignis. Die Randereignisse müssen dann nur noch in der richtigen Reihenfolge eingefädelt werden. Zum besseren Verständis dient Abbildung 3.2.

Beim ersten und letzten Punkt des kürzesten Pfades muß berücksichtigt werden, daß die Shortest Path Map nicht beim Start- und Endpunkt endet. Es bedarf also bei diesen Punkten einer speziellen Funktionalität, die in der Funktion *SPV::PathAndBoundaryEventCalculator::getSegmentsForFinalPoint* implementiert ist.

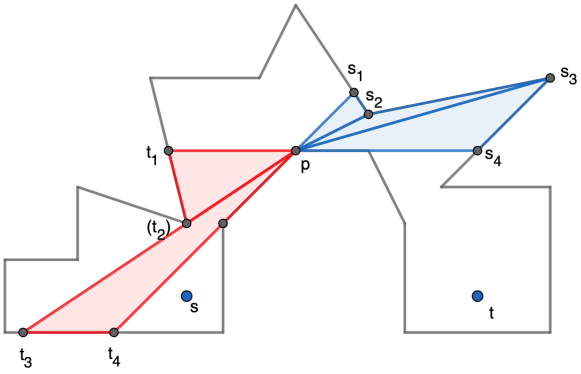

Abbildung 3.2: Die Punkte s_1 und t_4 definieren das von s aus gesehen erste Pfadereignis, die Punkte s_4 und t_1 das zweite Pfadereignis bei Drehung um p. Die restlichen Punkt sind Randereignisse, die in der richtigen Reihenfolge eingefädelt werden müssen. Der Punkt (t_2) braucht besondere Aufmerksamkeit, weil er ein Randereignis definiert, bei dem (t_2) nur tangiert wird.

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 kann mit der Maus ein Polygon, sowie ein Start- und Endpunkt eingegeben werden. 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 übersetzten.

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.

3.4 Tests

Das Unterprojekt *VisibilityProblemTest* enthält Code, durch den die Berechnungen der Bibliothek getestet werden. Der Code basiert dabei auf dem Qt-Test Framework. Um Testdaten zu generieren habe ich anhand von GeoGebra ([5]) 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 Edge-Cases 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 Punkte 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 das min-max Minimum nicht eindeutig ist (sondern innerhalb einer Scheibe liegt).

8. Polygone, bei denen der Start- und Endpunkt auf einer Kante der Polygontriangulation 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 Ergebins muß in beiden Fällen natürlich daßelbe sein.

Ich habe das Programm auf Ubuntu außerdem mit Valgrind [12] auf memory leaks getestet. Im derzeitigen Zustand konnten keine memory leaks oder andere Fehler gefunden werden.

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 eines lokalen min-sum Minimums bei versperrtem Pfad zu den Sichtbarkeitslinien zu verbessern. Derzeit ist der Algorithmus (wie oben beschrieben) eher krude und kann bei eingehender Beschäftigung mit der Distanzfunktion möglicherweise stark verbessert werden.

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 der direkte Pfad zu den Sichtbarkeitsgeraden zwischen zwei Ereignissen nicht versperrt ist, liegt das lokale min-sum Minimum entweder auf der Sichtbarkeitsgeraden, die mit dem Anfangsereignis zusammenfällt, oder auf der, die mit dem Endereignis zusammenfällt.

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

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

1. Für den Fall, daß der direkte Pfad zur Sichtbarkeitslinie auf einer oder auf beiden Seiten durch Polygonkanten versperrt ist, wäre es interessant, genauer abzuschätzen, unter welchen Umständen ein lokales min-sum Minimum auf einer Sichtbarkeitsgeraden zwischen den Ereignissen liegt. Meine Erfahrung bei der Berechnung der Minima deutet darauf hin, daß dies nur in sehr seltenen Fällen möglich ist. Allerdings ist es mir nicht gelungen, dies mathematisch genauer zu fassen.

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

Neben diesen Fragestellungen ist es natürlich möglich, den Quellcode zu verbessern. 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 Visualiserung des Problems zu nutzen, ohne extra Software installieren zu müssen.

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 14 wird angenommen, daß die Winkel β und γ (siehe Abbildung B.1) gleich sind. Diese Annahme wird hier bewiesen. Dazu benötigt man die Tatsachen, daß der Innenwinkel eines Dreiecks 180° beträgt und die Basiswinkel eines gleichschenkligen Dreicks gleich sind.

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

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

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:

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

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

Unter Berücksichtigung von:

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

gilt:

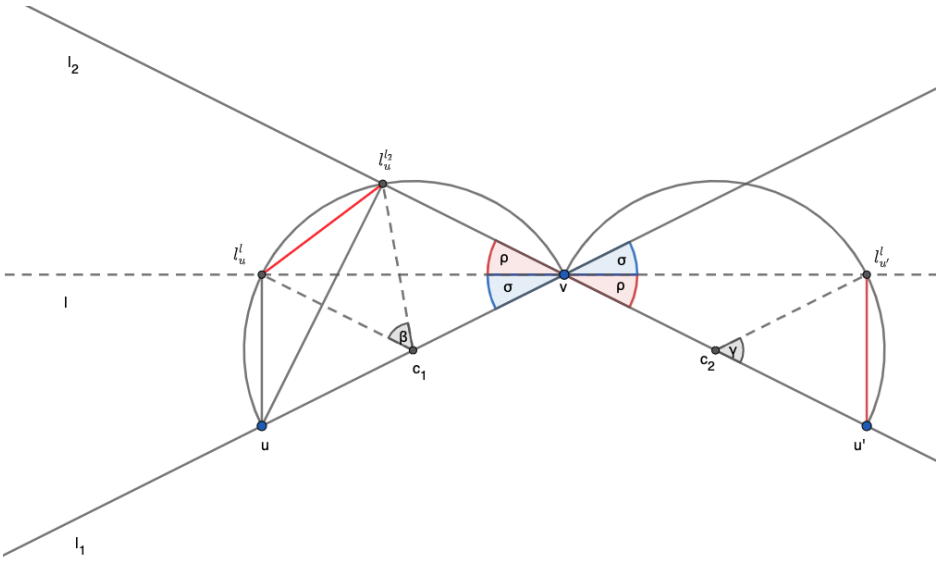

Abbildung B.1: Die Winkel β und γ sind gleich.

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

Anhang C

Werte des Beispiels bei versperrter Sicht

Bei dem in Kapitel 2.4.3 verwendeten Beispiel (siehe Abbildung 2.25) sind die Gerade, die die Polygonkante auf der linken Seite enthält, die Punkte u, v und u' sowie die Distanzfunktionen folgendermaßen definiert:

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

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

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

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

Anhang D

Distanzfunktionen für die min-max Version des Algorithmus

In der min-max Version des Algorithms liegt eine Lösung bei der Sichtbarkeitsgeraden l, für die gilt: $|\pi(s,u)| + |\pi(u,l)| = |\pi(t,u')| + |\pi(u',l)|$. Die Länge von $\pi(s,u)$ bzw. $\pi(t,u')$ ist einfach die Summe der Kantenlängen. Für die Distanzen $|\pi(u,l)|$ und $|\pi(t,l)|$ 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 $|\pi(u,l)|$ und $|\pi(u',l)|$ anhand der Erläuterungen in Appendix A leicht hergeleitet werden. In diesem Fall ist folgende Gleichung für α zu lösen:

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

Falls die Sicht auf h_{uv} oder $h_{u'v}$ versperrt ist, muß mit Gleichung 2.18 eine der folgenden Gleichungen für α gelöst werden:

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

$$|\pi(s,u)| + \sqrt{(u.x - i.x)^2 + (u.y - i.y)^2} = \\ |\pi(t,u')| + \frac{|(u'.x - v.x) \cdot \alpha + v.y - u'.y|}{\sqrt{\alpha^2 + 1}} \quad \text{(D.3)}$$

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

$$|\pi(s,u)| + \sqrt{(u.x - i.x)^2 + (u.y - i.y)^2} = \\ |\pi(t,u')| + \sqrt{(u'.x - i'.x)^2 + (u'.y - i'.y)^2} \quad \text{(D.4)}$$

Anhang E

Anleitung zur Installation der Software

Das Programm kann sowohl auf macOS Catalina als auch auf Ubuntu 18.04 installiert werden. 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.

E.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 das Programm noch von github heruntergeladen werden (git ist eigentlich Bestandteil von Ubuntu, wenn es nicht installiert ist, kann dies leicht nachgeholt werden):

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

Anschließend kann das Projekt im qtCreator geöffnet und kompiliert werden.

E.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:

CGAL

CGAL_Core

CGAL_ImageIO

boost_thread-mt

boost_system-mt

gmp

mpfr

Diese können im Internet gefunden werden.

Dann muß das Programm 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 werden und anhand von clang kompiliert werden.

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

[3] 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.

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

[5] *GeoGebra*. URL: https://www.geogebra.org/geometry (besucht am 29. 08. 2019).

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

[7] Leonidas Guibas. *Handout for Lecture 8*. 1992. URL: https://graphics.stanford.edu/courses/cs268-09-winter/notes/handout7.pdf (besucht am 20. 08. 2019).

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

[9] *Overleaf*. URL: https://www.overleaf.com/.

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

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

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

Abbildungsverzeichnis

Alle verwendeten Abbildungen außer 3.1 wurden von mir mit GeoGebra [5] erstellt und per Screenshot abgespeichert. Zur Erstellung von 3.1 habe ich draw.io [4] verwendet.

Die Arbeit wurde im Online Editor overleaf [9] geschrieben.