Rheinische Friedrich-Wilhelms-Universität Bonn
Institut für Informatik I

Markus Rings

Das Problem der längsten Leiter

26. Oktober 2005

Diplomarbeit

Betreuer: Prof. Dr. Rolf Klein
Dr. Elmar Langetepe

Gutachter: Prof. Dr. Rolf Klein
Prof. Dr. Christel Baier

Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel 1

Einleitung

Jeder von uns hat sicher schon einmal die Erfahrung gemacht, wie schwierig es sein kann ein großes sperriges Möbelstück durch eine Wohnung oder noch schwieriger durch ein Treppenhaus zu transportieren. Nicht selten kommt es vor, dass zum Beispiel ein Sofa in einer Engstelle stecken bleibt. Für einige Momente lässt es sich weder vor noch zurück bewegen. In einem solchen Fall ist es sicher nützlich wenn man schon im Vorfeld feststellen kann, ob sich ein Gegenstand von Punkt A nach Punkt B transportieren lässt und, falls dies möglich ist, welches die entsprechenden Bewegungen sind. In der *Algorithmischen Geometrie* ist die Suche nach einer kollisionsfreien Bewegung eines Objektes in einem Raum unter dem Titel „Piano Mover's Problem“ bekannt.

Bereits Anfang der 1960er Jahre lenkte der Mathematiker Prof. John Horton Conway erste Aufmerksamkeit auf das „Sofa Problem“[19]. Dabei ist die größte Fläche S gesucht, die man um eine rechtwinklige Ecke eines Korridores gleichbleibender Breite bewegen kann.

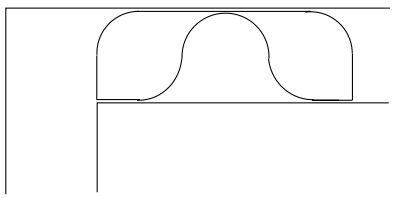


Abbildung 1.1: Das Sofa Problem

Das erste Paper dazu veröffentlichte Leo Moser im Jahre 1966, darin schlägt er die in Abb. 1.1 gezeigte Lösung vor. Diese hat bereits eine Fläche von ca. 2,044 Quadrateinheiten. Das größte bisher bekannte Objekt mit einer Fläche von ca. 2,2195 Quadrateinheiten wurde im Jahr 1992 von Joseph Gerver in *„On moving a sofa around a corner“* [4] veröffentlicht (s. Abb. 1.2). Darin beweist er „mit wenigen Lücken“ [5], dass seine Figur maximale Fläche unter allen Figuren hat, die sich um eine rechtwinklige Korridorecke mit einheitlicher Breite bewegen lassen.

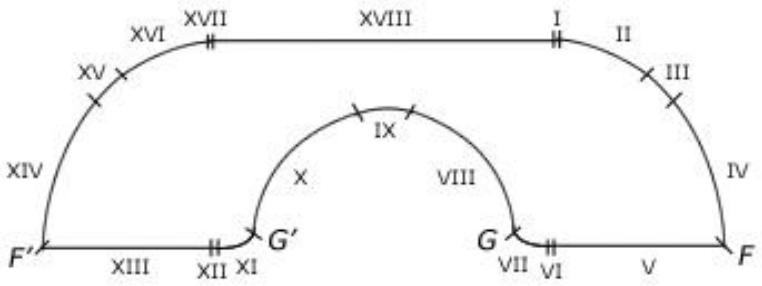


Abbildung 1.2: Gerver's maximales Sofa

Neben geometrischen Fragestellungen zu maximalen Objekten, die sich durch vorgegebene Engpässe bewegen lassen, gibt es noch weitere interessante Arbeiten zum „Piano Mover's Problem". Die Theoretische Informatik beschäftigt sich u.a. mit der Suche nach effizienten Berechnungsmethoden. J.T. Schwarz und M. Sharir haben auf diesem Gebiet eine ganze Serie von Paper [14, 15, 16, 18, 17] veröffentlicht. Im Ersten dieser Reihe [14] stellen sie einen Algorithmus vor, der in $O(n^5)$ (n = Anzahl der Ecken der Umgebung) eine kollisionsfreie Bewegung eines polygonalen Objektes in einer polygonalen 2-dimensionalen Umgebung berechnet, falls diese existiert. In ihrer Arbeit entwerfen Schwarz und Sharir zuerst ein Verfahren um die Bewegung einer Leiter (Liniensegment) zu berechnen. Dieses übertragen sie dann auf beliebige 2-d Objekte.

Im Jahr 1984 verbesserten O'Dunlaing, Sharir und Yap [13] die Laufzeit für die Berechnung einer Leiterbewegung auf $O(n^2 \log n \log^* n)$. Schon im folgenden Jahr veröffentlichten Leven und Sharir in [11] ein Verfahren aufbauend auf den Techniken aus [14] mit dem sie die Bewegung einer Leiter in einem 2-dimensionalen polygonalen Raum in $O(n^2 \log n)$ berechnen. Auch wenn Ke und O'Rourke [6] die Vermutung aufstellen, dass sich dieses Verfahren auf eine Laufzeit von $O(n^2)$ reduzieren lässt, so ist ein solcher Algorithmus jedoch noch nicht veröffentlicht worden.

Ziel dieser Diplomarbeit ist es die größte Leiter zu berechnen, die sich in einer 2-d Umgebung von A nach B transportieren lässt. Ich werde zeigen, dass dies in Zeit $O(n^2 \log n)$ berechenbar ist. Zuerst wird der Fall gelöst in dem nur eine Translation der Leiter erlaubt ist. Anschließend wird dieses Modell um die Möglichkeit der Rotation erweitert.

Anschließend wird gezeigt, dass sich die untere Schranke für Leitern fester Länge aus Kapitel 2.3 auch auf die Lösung des *Längste Leiter Problem* übertragen lässt. Abschließend wird erörtert wie sich die Laufzeit für mehrere Anfragen auf der gleichen Umgebung verbessern lässt.

Kapitel 2

Hintergrund

In diesem Kapitel werden zunächst grundlegende Begriffe und Definitionen eingeführt, die im Folgenden weiter verwendet werden. Anschließend werden Ideen des Algorithmus von Leven und Sharir [11] zur effektiven Berechnung einer Leiterbewegung in einer 2-dimensionalen polygonalen Umgebung kurz vorgestellt, da einige der Ideen auch bei dem Verfahren zur Berechnung der größten Leiter Verwendung finden. Zum Schluß wird die untere Schranke von J. O'Rourke [12, 6] für das obige Bewegungsplanungsproblem vorgestellt. Mit der Berechnung der größten Leiter ist auch die Frage gelöst, ob sich eine Leiter fester Länge von A nach B transportieren lässt, daher ist diese untere Schranke auch relevant für das *Längste Leiter Problem.*

2.1 Begriffe und Definitionen

Zur formalen Beschreibung der Problemstellung werden hier zunächst einige Definitionen eingeführt. Mit Hilfe dieser Darstellung lässt sich dann direkt eine Lösungsansatz erkennen.

Definition 2.1 (siehe auch [10])

(i) Eine **polygonale Umgebung** *oder einfach* **Umgebung** *ist gegeben durch ein umgrenzendes (bounding) Polygon, welches in seinem Inneren weitere Polygone enthalten kann. Diese inneren Polygone werden im folgenden* **Hindernis** *genannt. Hindernisse können weder einander noch das bounding Polygon schneiden.*

(ii) Eine **Plazierung** oder **Konfiguration** *ist eine Position der Leiter L in der Umgebung. Die Position ist dabei durch den Referenzpunkt $r = (x, y)$ eines Leiterendes und durch die Orientierung θ, in die die Leiter zeigt, eindeutig festgelegt. Die Plazierung lässt sich somit durch Angabe des Referenzpunktes und der Orientierung als $L(x, y, \theta) \in \mathbb{R}\times\mathbb{R}\times[0, 2\pi[$ (Kurz: $L(X, \theta)$) angeben (s. Abb. 2.2). Die Leiter hat somit drei Freiheitsgrade.*

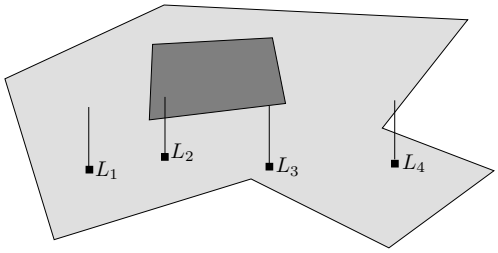


Abbildung 2.1: L_2, L_4: Verbotene Plazierungen, L_3: Semifree Position, L_1: Free Position

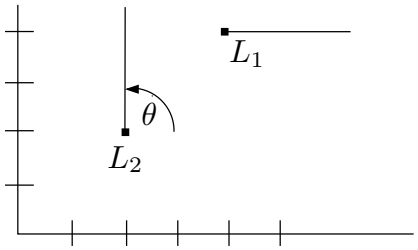


Abbildung 2.2: $L_1 = L(4, 4, 0)$, $L_2 = L(2, 2, \frac{\pi}{2})$

(iii) *Der* **Konfigurationsraum** *ist der Raum der möglichen Plazierungen der Leiter im Inneren des bounding Polygon.*

$$\mathcal{C} = ((\mathbb{R} \times \mathbb{R}) \cap P_{bound}) \times [0, 2\pi[$$

(iv) *Der* **Raum der verbotenen Konfigurationen** *umfasst alle Plazierungen im $\mathcal{C}$, bei denen die Leiter ein Hindernis schneidet oder das bounding Polygon verlässt (Abb. 2.1).* **Halbfreie Plazierungen**, *also das Berühren der Wand ist erlaubt. (Semifree Position)*

$$\mathcal{C}_{blocked} := \left\{ c \in \mathcal{C} \middle| L(c) \cap \bigcup \overset{\circ}{P}_i \cup cP_{bound} \neq \emptyset \right\}$$

(v) *Die* **freien Plazierungen** *(Free Position, FP) umfassen alle erlaubten Plazierungen im Konfigurationsraum.*

$$FP := \mathcal{C} \backslash \mathcal{C}_{blocked}$$

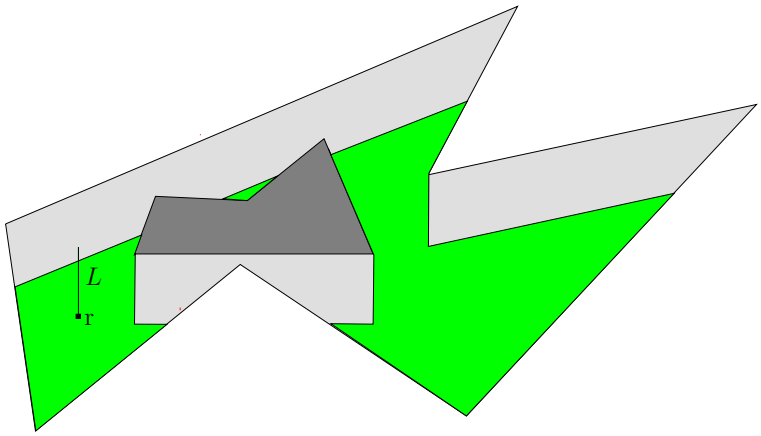


Abbildung 2.3: Grün dargestellt ist FP für die Leiter L mit festem Winkel $\theta = \frac{\pi}{2}$.

Die Frage, ob es eine kollisionsfreie Bewegung der Leiter von Punkt A nach B in der Umgebung gibt, lässt sich durch die oben beschriebenen Definitionen leicht formal beschreiben.

Lemma 2.2 *Eine Leiter L lässt sich in einer Umgebung U kollisionsfrei von Punkt A nach B bewegen, genau dann wenn* $L_A = L(X_A, \theta_A)$ *und* $L_B = L(X_B, \theta_B)$ *in der gleichen Zusammenhangskomponete von FP liegen.*

Beweis:
„⇒“ Angenommen es gibt eine kollisionsfreie Bewegung der Leiter von A nach B, dann gibt es auch einen kontinuierlichen Weg von L_A nach L_B der komplett in FP liegt. Daraus folgt, dass L_A und L_B in der gleichen Zusammenhangskomponente liegen.
„⇐“ Angenommen L_A und L_B liegen in der gleichen Zusammenhangskomponente, dann gibt es einen kontinuierlichen Weg innerhalb von FP der von L_A nach L_B führt. Daraus folgt, dass es eine kollisionsfreie Bewegung der Leiter von A nach B gibt. □

Gesucht ist nun eine effektive Berechnung des Raumes FP und dessen Zerlegung in die Zusammenhangskomponenten. Die folgenden Definitionen finden in Abschnitt 2.2 Anwendung und sind so oder in leicht veränderter Form auch für die Lösung des *Längste Leiter Problems* von Bedeutung.

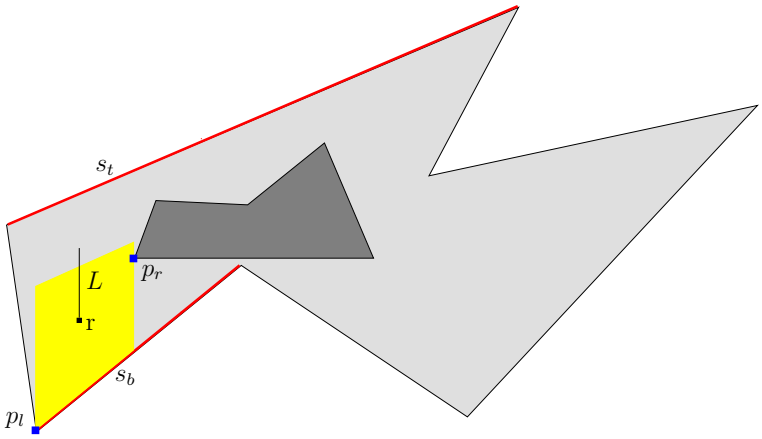


Abbildung 2.4: Gelb dargestellt ist die Region $\mathcal{R}_\pi$ in der sich die Leiter L mit dem Referenzpunkt r befindet. Das obere (untere) beschränkende Segment s_t (s_b) ist rot markiert. Die blauen Punkte p_r und p_l sind rechte bzw. linke Beschränkungen.

Definition 2.3 *Alle Plazierungen* $\mathcal{P}$ *im FP mit gleichen räumlichen Beschränkungen* Ψ *und festen* θ *bilden eine* **Region** $\mathcal{R}_\theta$ *(Abb. 2.4). Diese Beschränkungen sind folgendermaßen definiert:*

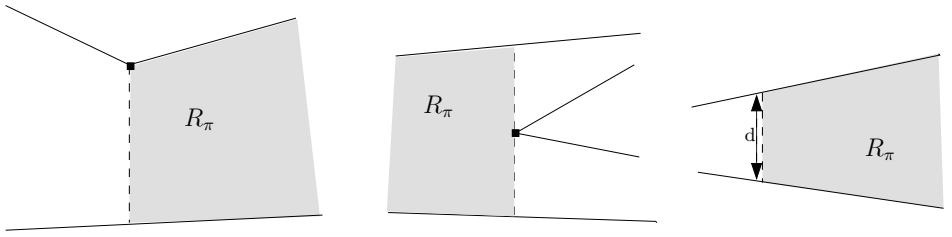


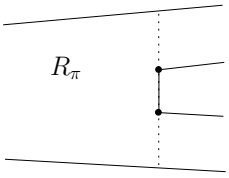
Abbildung 2.5: Drei Fälle der rechten und linken Begrenzung einer Zelle

- *Das erste Umgebungssegment in Leiterrichtung wird* **oberer Stop** *genannt.* $s_t(\mathcal{P})$
- *Dem entsprechend ist der* **untere Stop** *das erste Segment entgegen der Leiterrichtung.* $s_b(\mathcal{P})$
- **Rechter und linker Stop** *begrenzen die Region in Leiterrichtung rechts und links. An diesen Punkten* $p_r(\mathcal{P})$,$p_l(\mathcal{P})$ *kommt es bei festem* θ *zu einer Änderung der unteren oder oberen Stops oder FP endet dort. Diese Stops können in drei Fällen (Abb. 2.5) auftreten:*

  Fall 1: Der seitliche Stop ist ein Endpunkt des oberen oder unteren Segments

  Fall 2: Der seitliche Stop ist der Eckpunkt eines neuen Hindernises.

  Fall 3: Die Zelle hat eine Engstelle, durch die die Leiter ohne Drehung nicht bewegt werden kann.



In einigen Fällen ist dieser seitliche Stop nicht eindeutig, d.h. es gibt mehrere Punkte an denen sich gleichzeitig der untere und/oder obere Stop ändert. In diesem Fall befindet sich der Winkel θ *in einer* **kritischen Orientierung**.

Für jede Plazierung $\mathcal{P} = L(X, \theta)$ *ist somit ein* **Rahmen** Ψ *an Beschränkungen definiert durch:*

$$\Psi(\mathcal{P}) = \Big([s_t(\mathcal{P}), s_b(\mathcal{P})], [p_l(\mathcal{P}), p_r(\mathcal{P})]\Big)$$

Eine **Region** *lässt sich damit folgendermaßen beschreiben:*

$$\mathcal{R}_\theta = \{\mathcal{P}_1, \mathcal{P}_2 \in FP | \theta_1 = \theta_2, \Psi(\mathcal{P}_1) = \Psi(\mathcal{P}_2)\}$$

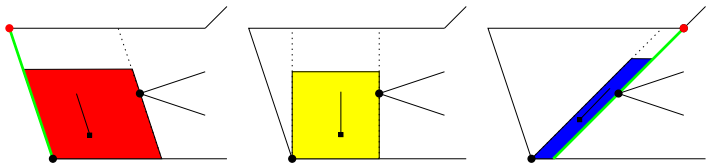


Abbildung 2.6: Rot: Region mit $\theta^+ = \frac{\pi}{2} + 0,2$, Gelb: Region mit $\theta = \frac{\pi}{2}$, Blau: Region mit $\theta^- = \frac{\pi}{4}$, Grün: Segment mit kritischer Orientierung

Bei der Definition der Region wurde stets ein fester Winkel angenommen, dies lässt sich allerdings leicht um die Möglichkeit der Rotation erweitern. Sei $\mathcal{P}_1 = L(X, \theta)$ eine Plazierung aus dem Inneren einer Region, so kann man ersehen, dass eine minimale Drehung δ im Punkt X den Rahmen Ψ von $\mathcal{P}_2 = L(X, \theta + \delta)$ nicht verändert. Das heißt $\Psi(\mathcal{P}_1) = \Psi(\mathcal{P}_2)$.

Nach Definition 2.3 gibt es Regionen mit kritischer Orientierung θ_{krit}, bei denen rechte oder linke Stops nicht eindeutig sind. Bei einer minimalen Veränderung des Winkels wird einer der beteiligten Punkte eindeutig zum entsprechenden Stop. Kritische Orientierungen sind somit die Winkel, in denen sich der Rahmen Ψ einer ganzen Region ändert. Dies führt zu folgender Definition.

Definition 2.4 *Eine* **Zelle** $\mathcal{Z}$ *ist der Raum aller Plazierungen aus FP mit gleichem Rahmen* Ψ *.*

$$\mathcal{Z} = \{\mathcal{P}_1, \mathcal{P}_2 \in FP | \Psi(\mathcal{P}_1) = \Psi(\mathcal{P}_2)\}$$

$$\Psi(\mathcal{Z}) = \Big([s_t(\mathcal{P}), s_b(\mathcal{P})], [p_l(\mathcal{P}), p_r(\mathcal{P})], [\theta^-, \theta^+]\Big) \text{ mit } \mathcal{P} \in \mathcal{Z}$$

Der Winkelbereich zwischen den kritischen Plazierungen $[\theta^-, \theta^+]$ *ist somit das maximale Intervall über dem eine Zelle definiert ist.*

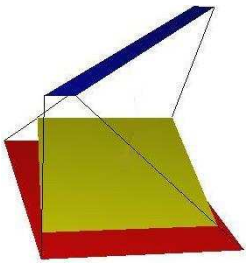


Abbildung 2.7: Seitenansicht der Zelle

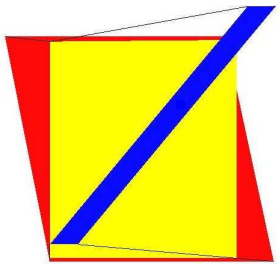


Abbildung 2.8: Zelle von oben

3D-Abbildungen der gleichen Zelle. Die Orientierung θ bildet die z-Achse. Die drei farbigen parallelen Ebenen entsprechen den Regionen aus Abbildung 2.6.

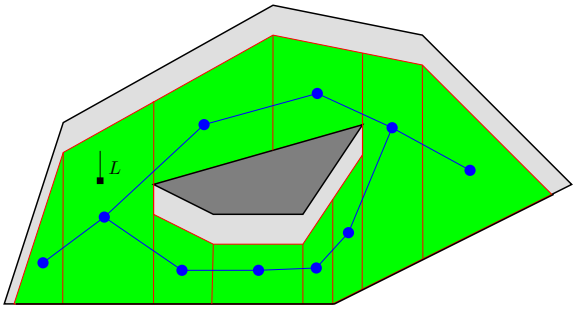


Abbildung 2.9: Roadmap: Der Raum FP_θ (grün) ist in Regionen zerlegt. Der Graph (blau) verbindet Knoten benachbarter Regionen.

2.2 Bewegungsplanung einer Leiter in $O(n^2 \log n)$

Im Folgenden wird nun ein Verfahren zur Bewegungsplanung einer Leiter in zwei Dimensionen in eigenen Worten kurz vorgestellt. Dieses wurde 1985 von Leven u. Sharir [11] veröffentlicht und berechnet in $O(n^2 \log n)$ (n = Anzahl der Ecken der Umgebung) ob sich eine Leiter mit Hilfe von Translations- und Rotationsbewegungen in einer 2-d Umgebung von einer Plazierung A zu einer Plazierung B bewegen lässt. Der Einfachheit halber wird das Verfahren zuerst für reine Translationen entwickelt und anschließend um die Möglichkeit der Rotation erweitert.

2.2.1 Reine Translationsbewegung

Bei reinen Translationsbewegungen ändert sich die Orientierung der Leiter nicht, somit reicht es aus, nur Regionen mit der Orientierung θ zu betrachten.

Das Verfahren beginnt somit mit der Partitionierung des Raum FP_θ in seine Regionen. Dabei wird außerdem ein Graph erstellt, in dem jede Region durch einen Knoten repräsentiert wird. Knoten, dessen Regionen einen gemeinsamen Rand haben, werden durch eine Kante verbunden. Es wird sozusagen eine Roadmap erstellt (Abb. 2.9).

Zur Lösung des Bewegungsplanungsproblems werden die Regionen bestimmt, in denen sich Start- und Zielpunkt (s, t) befinden. Anschließend wird durch eine Graphensuche nach einem Weg zwischen diesen beiden Knoten gesucht.

Wenn ein Weg im Graph gefunden wurde, lässt sich die Leiter von s nach t bewegen. Innerhalb einer Region kann eine Leiter auf direktem Weg von einer Plazierung zur Anderen bewegt werden (Korollar 3.4). Somit kann die Leiter vom Startpunkt aus zum gemeinsamen Rand mit der Nachbarregion bewegt werden, von dieser Region aus wieder zum Rand mit der nächsten

Region. Dies erfolgt so lange, bis die Zielregion erreicht ist. Dort kann die Leiter dann zur Zielposition bewegt werden.

Falls bei der Graphensuche kein Weg gefunden wurde, so lässt sich die Leiter nicht von s nach t bewegen, da die Knoten der Regionen in unterschiedlichen Zusammenhangskomponenten des Graphen und damit s und t in unterschiedlichen Zusammenhangskomponenten von FP_θ liegen.

Für die Zerlegung des FP in seine Regionen und die Berechnung der Roadmap wird o.b.d.A. angenommen, dass die Leiter eine unkritische Orientierung $\theta = \frac{\pi}{2}$ hat. In diesem Fall kann beides durch einen einfachen Sweep von links nach rechts in $O(n \log n)$ berechnet werden. Die Anzahl dieser Regionen liegt in $O(n)$.

2.2.2 Allgemeine Bewegungen

In Definition 2.4 wurde erklärt, dass sich das Konzept der Region durch Ergänzung um Drehungen auf Zellen erweitern lässt. Auch wenn eine Zelle nicht unbedingt konvex ist (Abb. 2.8), so ist dennoch eine kollisionsfreie Bewegung der Leiter zwischen allen Plazierungen der Zelle möglich ([11], S.201). Außerdem wurde festgestellt, dass sich bei kritischen Orientierungen der Rahmen der betroffenen Zellen ändert und somit neue Zellen entstehen. Kritische Orientierungen kommen in folgenden Fällen vor:

- Orientierungen der innerhalb des Bounding Polygon liegenden Sichtsegmente zwischen den Eckpunkten der Umgebung. # $O(n^2)$
- Orientierungen der innerhalb des Bounding Polygon liegenden Sichtsegmente zwischen einem Eckpunkt und einer Kante mit genau der Länge der Leiter (seitlicher Stop, Fall 3). # $O(n^2)$

Der Algorithmus:

1. Sei $\theta = 0$ eine unkritische Orientierung. Alle kritischen Orientierungen werden berechnet und von 0 aus aufsteigend sortiert.
2. Für die Orientierung $\theta = 0$ wird die Roadmap und der Graph wie im vorherigen Abschnitt berechnet. Zellen und Kanten erhalten die Startorientierung $\theta^- = 0$.
3. Für die nächste kritische Orientierung θ_{krit} werden die betroffenen Zellen bestimmt.
4. Die betroffen Zellen und die mit ihnen verbundenen Kanten werden mit der Endorientierung $\theta^+ = \theta_{krit}$ abgeschlossen.
5. Die neuen Zellen mit $\theta^- = \theta_{krit}$ werden berechnet. In der Roadmap werden neue Knoten erstellt und mit den jeweiligen Nachbarn und Vorgängern durch neue Kanten verbunden.

6. Falls es noch weitere kritische Orientierungen gib, fahre fort mit 3.

7. Am Ende des Verfahrens existieren Zellen mit den Orientierungsbereichen $[0, \theta_i]$ und $[\theta_j, —]$ aber sonst identischen Rahmen. Diese werden zu einer Zelle mit Orientierungsbereich $[\theta_j, \theta_i]$ zusammengefaßt und die entsprechenden Kanten angepaßt.

Da sich die Leiter innerhalb einer Zelle kollisionsfrei bewegen lässt, also auch von einem Randpunkt zu einem anderen, wurde somit das Bewegungsplanungsproblem auf die Wegesuche im Graphen (Roadmap) reduziert. Die Zellen von Start- und Zielplazierung müssen identifiziert werden und anschließend lässt sich zum Beispiel mit einer Tiefensuche ein Weg im Graphen suchen. Falls kein Weg gefunden wird, liegen beide Plazierungen in unterschiedlichen Zusammenhangskomponenten des Graphen und somit des Raumes FP, d.h. es gibt keine kollisionsfreie Bewegung zwischen den beiden Plazierungen.

2.2.3 Laufzeiten

Der oben beschriebene Algorithmus berechnet das Bewegungsplanungsproblem einer Leiter in einer 2-dimensionalen Umgebung mit Hindernissen in $O(n^2 \log n)$ Zeit. Die einzelnen Berechnungsschritte haben dabei folgende Laufzeiten:

- Die kritischen Orientierungen der Sichtbarkeitssegmente lassen sich u.a. mit dem Topologischen Sweep von Edelsbrunner und Guibas [3] berechnen. $\mathbf{O(n^2)}$

- Die $O(n^2)$ vielen kritischen Plazierungen müssen sortiert werden. $\mathbf{O(n^2 \log n)}$

- Berechnung der Roadmap für $\theta = 0$ mit einem Sweep. $\mathbf{O(n \log n)}$

- An jeder der $O(n^2)$ kritischen Orientierungen ändern sich nur konstant viele Zellen. Updates für jede neue Zelle können ebenfalls in konstanter Zeit durchgeführt werden. $\mathbf{O(n^2)}$

- Das Verbinden der Zellen aus Schritt 7 kann durch naives Vergleichen der jeweils $O(n)$ Zellen mit Winkelbereichen $[0, \theta_i]$ und $[\theta_j, —]$ erfolgen. $\mathbf{O(n^2)}$

- Lokalisieren der Zellen für Start- und Zielplazierung. $\mathbf{O(n^2)}$

- Wegesuche im Graph mit $O(n^2)$ Zell-Knoten (z.B. Tiefensuche). $\mathbf{O(n^2)}$

□

2.3 Untere Schranke $\Omega(n^2)$

Um die Güte des Algorithmus besser einordnen zu können, soll nun eine untere Schranke für das Leiter-Bewegungsplanungsproblem eingeführt werden. Diese beruht zum einen auf der Erkenntnis von O'Dunlaing, Sharir und Yap [13], dass FP in $\Omega(n^2)$ viele Zusammenhangskomponenten zerfallen kann. Zum anderen haben Ke und O'Rourke in [6, 12] gezeigt, dass selbst innerhalb einer Zusammenhangskomponente $\Omega(n^2)$ viele Bewegungsschritte notwendig sein können.

2.3.1 $\Omega(n^2)$ viele Zusammenhangskomponenten

In Abb. 2.10 wird gezeigt, dass FP in $\Omega(n^2)$ viele Zusammenhangskomponenten zerfallen kann.

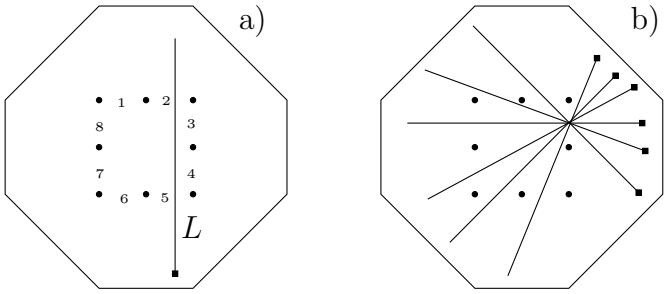


Abbildung 2.10: a) Umgebung in der FP der Leiter L in $\Omega(n^2)$ Zusammenhangskomponenten zerfällt. b) $\Omega(n)$ Leiterplazierungen in paarweise verschiedenen Zusammenhangskomponenten.

In Teil b) der Abbildung werden $\frac{3}{4}n$ Leiterplazierungen gezeigt, die in unterschiedlichen Zusammenhangskomponenten (ZK) von FP liegen, denn die Leiter ist jeweils zwischen zwei Pflock-Paaren positioniert und sie lässt sich mit keiner Kombination von Translations- und Rotationsbewegungen an einem Pflock vorbei bewegen. Die abgebildeten Leitern durchqueren alle Lücke 3 in Kombination mit einer der Lücken [1,2,5,6,7,8]. Für alle anderen n Lücken lassen ebenso $\frac{3}{4}n$ Leiterpositionen angeben, die wiederum in weiteren ZK's liegen. Daher zerfällt FP dieser Umgebung mit Leiter L in $\Omega(n^2)$ viele Zusammenhangskomponenten.

Ein naives Testen, in welcher Zusammenhangskomponente sich die Start- und Zielplazierung befinden, erfordert somit ebenfalls $\Omega(n^2)$ viele Berechnungsschritte.

2.3.2 $\Omega(n^2)$ viele Bewegungsschritte

Falls es nun möglich wäre, effizienter in der Menge der ZK's zu suchen, so würde diese Schranke nicht ausreichen. Für diesem Fall kommt die Erkennt-

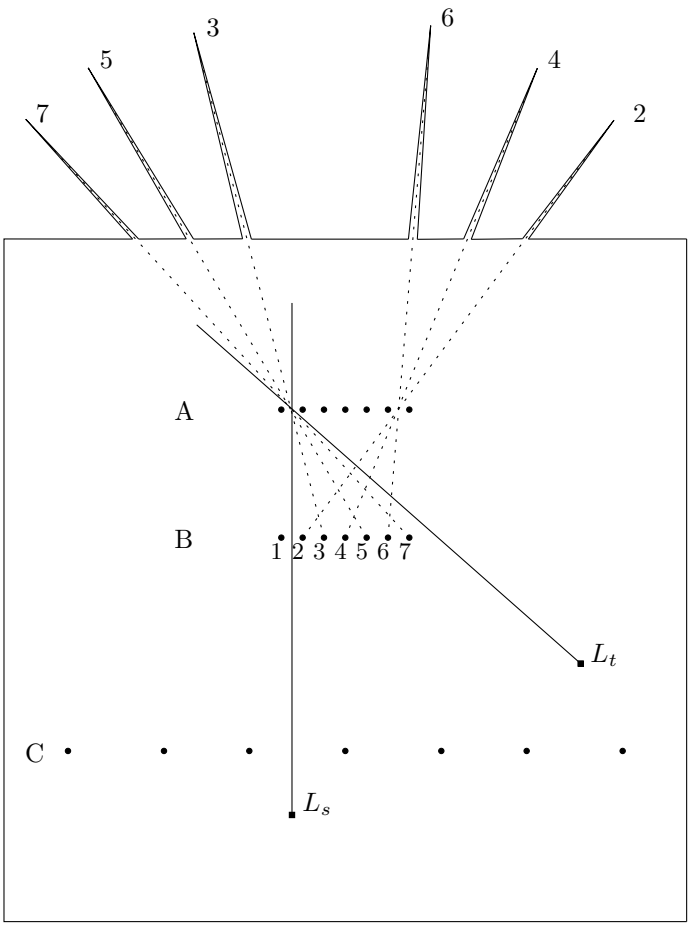


Abbildung 2.11: Die Bewegung der Leiter von Plazierung L_s zur Plazierung L_t erfordert $\Omega(n^2)$ viele Bewegungsschritte.

nis von Ke und O'Rourke [6, 12] hinzu, dass selbst innerhalb einer ZK $\Omega(n^2)$ viele Bewegungsschritte benötigt werden können. Abbildung 2.11 zeigt eine solche Umgebung.

In diesem Beispiel gibt es nur eine Möglichkeit, die Leiter an einem Pflock aus der Reihe B vorbei zu bewegen, die Leiter muss in die entsprechende mit gleicher Nummer beschriftete obere Einbuchtung geschoben werden.

Die oberen Einbuchtungen sind der Gestalt, dass sie nur durch die erste linke oder die letzte rechte Lücke in der Reihe A im vollen Umfang zugänglich sind. Da die Einbuchtungen für aufeinander folgende B-Pflöcke sich abwechselnd auf der linken oder rechten Seite befinden, muss die Leiter jedes mal an allen Pflöcken der Reihe A vorbei geführt werden.

Die Pflöcke aus der Reihe C verhindern wiederum, dass die Leiter durch eine Rotation an mehr als konstant vielen A-Pflöcken vorbei geführt werden

kann. Sie muss immer wieder durch Translation und Rotation an den C-Pflöcken vorbei bewegt werden.

Somit benötigt das Überschreiten eines B-Pflock $\Omega(n)$ viele Bewegungsschritte und für das Überschreiten aller $\Omega(n)$ B-Pflöcke werden $\Omega(n^2)$ viele Bewegungsschritte benötigt. □

Kapitel 3

Längste Leiter bei reinen Translationsbewegungen

Im ersten Schritt der Lösung des *Längste Leiter Problems* (LLP) sollen zunächst reine Translationsbewegungen betrachtet werden. Dazu wird ein Roadmap-Verfahren verwendet, das in den Grundzügen auf den von Leven und Sharir [11] verwendeten Ansätzen aufbaut. Dort wurde für das allgemeine Bewegungsplanungsproblem für eine Leiter fester Länge ein Weg im 3-dimensionalen Raum FP gesucht. D.h. die Leiter hat drei Freiheitsgrade (x- und y-Koordinate des Referenzpunktes und die Orientierung θ).

3.1 Definitionen und Vorüberlegungen

Wenn man dieses Modell auf das *Längste Leiter Problem* überträgt, so könnte man dort die Länge der Leiter als einen 4. Freiheitsgrad modellieren. In der folgenden Lösung geschieht dies allerdings nicht direkt als 4. Dimension des FP, sondern die Länge fließt indirekt über gewichtete Kanten in der Roadmap mit in die Lösung ein.

Da beim LLP die Länge der Leiter nicht fest vorgegeben, sondern gesuchte Größe ist, wird bei der Berechnung des FP von einer Leiter infinitesimaler Länge ausgegangen. Somit ist FP = Gesamter innerer Raum des Bounding Polygons, der nicht von Hindernissen überdeckt ist = Das Innere der Umgebung.

Für die Definition 2.3 der Region bedeutet dies, dass sich der 3. Fall der seitlichen Begrenzung (vgl. Abb. 2.5 und 3.1) ändert. Da die Leiter unendlich kurz ist, kann sie bis in die Ecken der Umgebung geschoben werden.

Definition 3.1 *Alle Plazierungen $\mathcal{P}$ im FP mit gleichen räumlichen Beschränkungen Ψ und festen θ bilden eine* **Region** $\mathcal{R}_\theta$.

Bis auf die seitlichen Beschränkungen bleiben alle anderen Beschränkungen

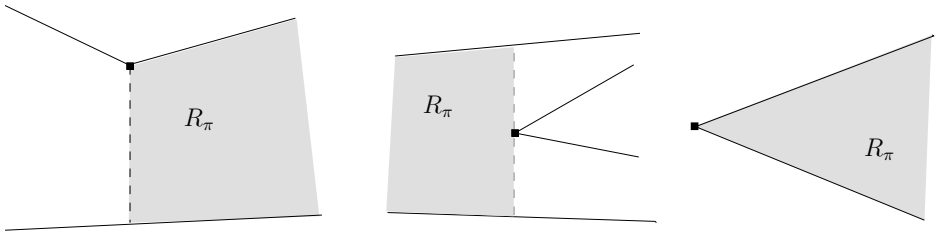


Abbildung 3.1: Drei Fälle der rechten und linken Begrenzung einer Zelle

und Feststellungen der Definition 2.3 unverändert. Für die seitlichen Stops gelten nun folgende Fälle:

Fall 1: Der seitliche Stop ist ein Endpunkt des oberen oder unteren Segmentes

Fall 2: Der seitliche Stop ist der Eckpunkt eines neuen Hindernises.

Fall 3: Oberes und unteres Segment haben einen gemeinsamen Endpunkt und bilden eine Ecke in der Umgebung.

Bevor nun mit dem Entwurf einer Lösung begonnen wird, sollen hier erst einmal einige Eigenschaften von Regionen festgehalten werden.

Lemma 3.2 *Der Rand einer Region besteht aus maximal vier Kanten und mindestens zwei dieser Kanten sind sind auch Ränder des FP.*

Beweis:
Betrachten wir zunächst Regionen, in denen eine seitliche Begrenzung des dritten Falls auftritt. Oberes und unteres Segment[1] starten hierbei aus einem gemeinsamen Punkt (o.b.d.A. *Linker Stop*[1]) und gehen in zwei Richtungen auseinander. Der Raum dazwischen bildet die Region $\mathcal{R}_\theta$, welche beim Auftreten einer der beiden Fälle 1 oder 2 endet (*Rechter Stop*[1]). Das Segment in Richtung θ, das oberes und unteres Segment verbindet und durch den rechten Stop läuft, beendet die Region, denn an dieser Stelle ändert sich der Rahmen Ψ. In diesem Fall hat die Region die Form eines Dreiecks.
In den anderen Fällen beginnt die Region links mit einem Segment in Richtung θ durch den linken Stop. Es verbindet oberes und unteres Segment, welche den Rand der Region fortsetzen. Das Segment durch den rechten Stop beendet wiederum die Region auf der rechten Seite. Die Region hat hierbei die Form eines Trapezes, da rechtes und linkes Segment parallel verlaufen (Orientierung θ). Abbildung 3.2 zeigt alle möglichen Kombinationen

[1]Nach Def. 2.3 werden oberer und untere Stop (Segment) sowie linker und rechter Stop immer relativ zur Orientierung θ angegeben.

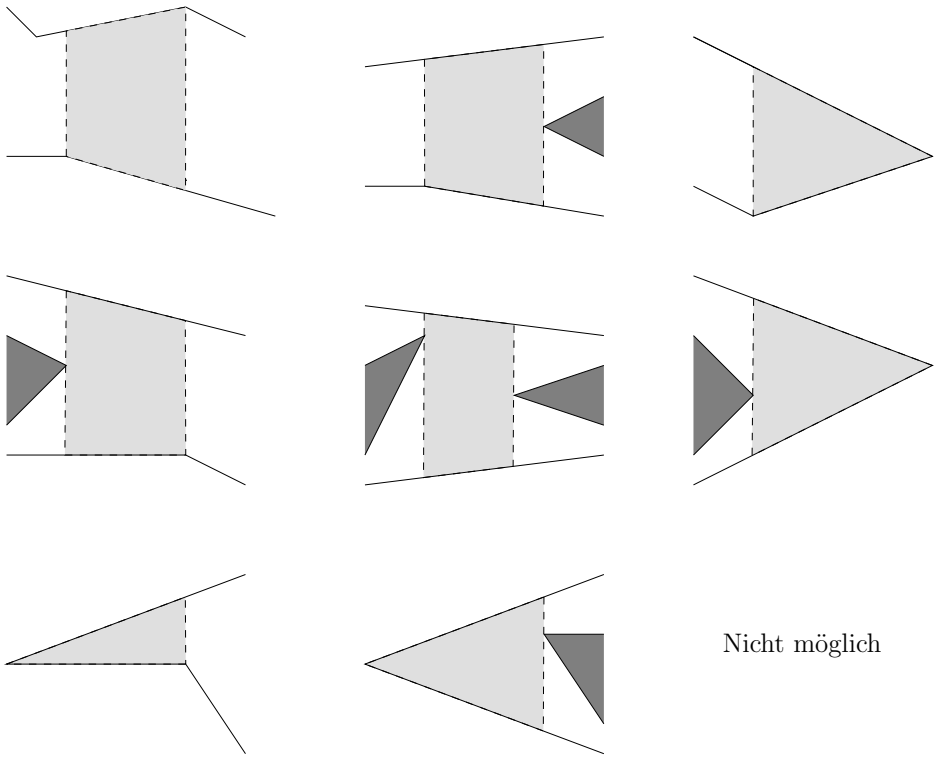


Abbildung 3.2: Regionen mit allen möglichen Kombinationen von Fällen rechter und linker Stops.

in denen die Fälle 1-3 vorkommen können.
Da die Teilstücke des oberen und unteren Segmentes, die im Bereich der Region liegen, immer zum Rand der Region beitragen und diese auch gleichzeitig Begrenzungen des FP sind, tragen mindestens zwei Kanten der Region auch zum Rand des FP bei. Bei kritischen Orientierungen θ_{krit} können in einigen wenigen Sonderfällen auch drei oder vier Kanten zum Rand von FP beitragen (siehe Abb. 3.3). □

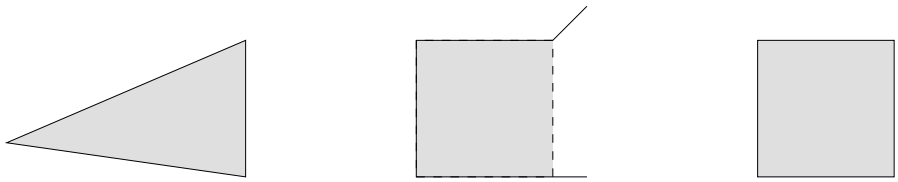


Abbildung 3.3: Sonderfälle, in denen mehr als zwei Kanten der Region zum Rand von FP beitragen.

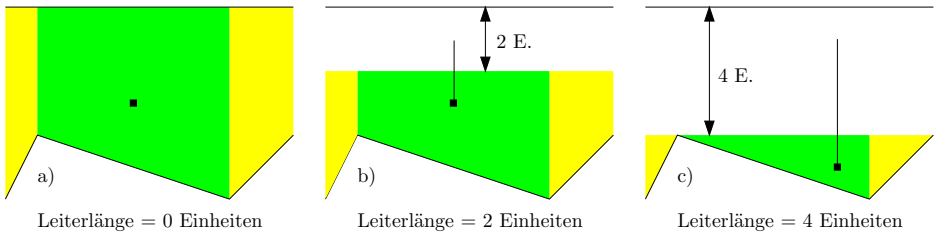


Abbildung 3.4: Veränderung einer Region bei unterschiedlichen Leiterlängen.

Korollar 3.3 *Eine Region kann bei einer nicht-kritischer Orientierung θ mit maximal vier anderen Regionen benachbart sein.*

Beweis:
Aus Lemma 3.2 folgt, dass maximal zwei Kanten nicht Teil des Randes von FP sind. Nur über diese Kanten ist ein Übergang zu einer anderen Region möglich. Im Fall von seitlichen Begrenzungen des Typs 2 aus Definition 3.1 können an einer Kante zwei benachbarte Regionen anliegen. Somit kann eine Region maximal vier Nachbarn haben. □

Korollar 3.4 *Regionen sind konvex und eine Leiter kann innerhalb einer Region auf direktem Weg (eine Translationsbewegung) von einem Punkt zum anderen bewegt werden.*[2]

Beweis:
In Lemma 3.2 wurde festgestellt, dass Regionen entweder dreieckig oder trapezförmig sind. Beide geometrischen Formen sind immer konvex. Aus der Konvexität folgt, dass das Liniensegment zwischen zwei Punkten aus der gleichen Region ebenfalls vollständig in der Region enthalten ist. Daraus folgt wiederum, dass sich die Leiter auf direktem Weg durch eine Translation zwischen den Punkten bewegen lässt. □

Für die Berechnung des Freespace und dementsprechend der Regionen wurde eine Leiter der Länge $l \to 0$ angenommen, bei der Problemstellung des LLP geht es aber darum, die größtmögliche Leiter zu finden, die sich durch die Umgebung bewegen lässt. Es stellt sich also die Frage, wie sich Regionen bei einer längeren Leiter verändern. Abbildung 3.4 zeigt, dass sich eine Region vom oberen Segment aus in Leiterrichtung um die Leiterlänge verkleinert. Dabei kann es wie in Teil c) der Abbildung vorkommen, dass der Kontakt zur Nachbarregion abreißt. Wegen der Konvexität der Region bleibt diese jedoch stets zusammenhängend und kann nicht in zwei ZK's zerfallen. Daraus folgt folgendes Lemma:

[2]Gilt auch für Leitern mit gegebener Länge. Die Region verkleinert sich lediglich duch eine Verschiebung des oberen Randes um die Leiterlänge in Leiterrichtung.

Lemma 3.5 *Die Engstellen einer Region $\mathcal{R}_\theta$ treten an den freien linken und rechten Rändern auf. Für den Übergang zwischen zwei benachbarten Regionen ist die Länge der gemeinsamen Randkante die kritische Leiterlänge.*

Beweis:
Freie Ränder sind Kanten der Region, an denen ein Übergang in eine benachbarte Region möglich ist und die nicht wie oberes und unteres Segment Teil des Randes von FP sind.
Wegen der Konvexität kann sich eine Region bei länger werdender Leiter nicht in zwei Teile spalten, in der Mitte einer Region kann somit keine Engstelle sein. Wie in Abbildung 3.4 gezeigt, kann an Übergängen zu benachbarten Regionen der Zusammenhang von FP unterbrochen werden, dort treten somit Engstellen auf. Da die freien linken und rechten Kanten die gleiche Orientierung haben wie die Leiter, ist die Länge der gemeinsamen Randkante zweier benachbarter Regionen die kritische Leiterlänge, an der der Zusammenhang zwischen den Regionen abreißt. □

Diese Feststellung hilft uns nun bei der Gewichtung der Kanten in der Roadmap. Da die Länge der gemeinsamen Randkante benachbarter Regionen der Länge der maximalen Leiter entspricht, die sich über diesen Rand von einer Region in die andere bewegen lässt, wird die entsprechende Kante der Roadmap mit dieser Länge gewichtet (siehe Abb. 3.5). Eine Region kann außerdem in der Mitte keine Engstellen haben, somit lässt sich eine Leiter, die durch zwei oder mehr Übergänge einer Region passt, auch durch das Innere der Region zwischen diesen Übergängen bewegen.

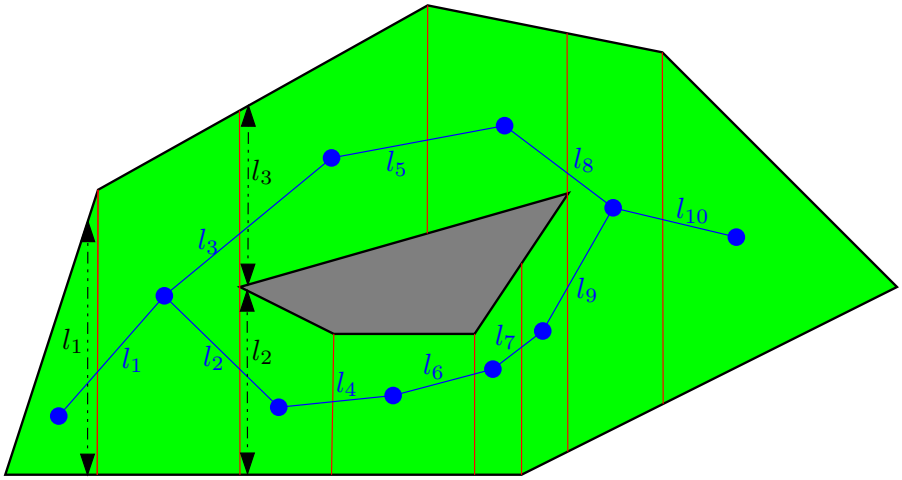


Abbildung 3.5: Roadmap mit gewichteten Kanten

3.2 Berechnung der Leiterbewegung

3.2.1 Idee

Auch wenn im vorherigen Abschnitt die Ideen für die Berechnung der längsten Leiter bereits angedeutet wurden, soll die Idee der Berechnung hier nocheinmal kompakt und zusammenhängend dargestellt werden.

In einem ersten Schritt wird das Innere der Umgebung in Regionen unterteilt. Innerhalb einer Region lässt sich die Leiter durch genau eine Translationsbewegung von einem Punkt zum anderen bewegen. Es wird eine Roadmap erstellt, in der die Regionen als Knoten repräsentiert werden. Benachbarte Regionen, welche einen gemeinsamen Rand haben, werden in der Roadmap durch Kanten verbunden. Da die Länge der gemeinsamen Randkante zweier Regionen der Länge der maximalen Leiter entspricht, die sich über diesen Rand bewegen lässt, wird die entsprechende Kante mit dieser Länge gewichtet.

Für die Berechnung der längsten Leiter, die sich zwischen zwei Punkten s und t bewegen lässt, werden zunächst die entsprechenden Regionen ermittelt. Es wird die längste Leiter bestimmt, die an s und t plaziert werden kann.

Durch eine modifizierte Tiefensuche in der Roadmap wird der Weg der maximalen Leiter von s nach t berechnet. Diese Suche funktioniert folgendermaßen:

Alle Kanten der Startregion werden in eine Priority-Queue eingefügt. Die Kante mit dem höchsten Gewicht wird entnommen und als begangen markiert. Von der Region am anderen Ende der begangenen Kante werden wiederum alle noch nicht begangenen Kanten in die Queue eingefügt. Es wird wiederum die Kante mit dem größten Gewicht entnommen und fortgefahren, bis die Zielregion erreicht ist. Das kleinste Gewicht einer dabei begangenen Kante entspricht der Länge der maximalen Leiter, die sich zwischen Start- und Zielregion bewegen lässt. Das Minimum aus dieser Länge und den maximalen Plazierungen an s und t ist die maximale Leiter, die sich von s nach t bewegen lässt.

Beispiel:
Abbildung 3.6 zeigt, wie eine solche Suche in der Roadmap funktionieren könnte. Vom Startknoten aus ist l_1 die einzige ausgehende Kante, sie wird begangen und ist somit die bis dahin längste Leiter. Jetzt kommen zwei neue Kanten hinzu l_2 und l_3, da l_2 die Längere (größeres Gewicht) von beiden ist, wird sie als nächstes begangen. Sie ist allerdings auch kürzer als l_1, und somit liegt der Engpass jetzt bei l_2. In der neuen Region kommt die Kante l_4 hinzu, unter den beiden noch nicht begangenen Kanten ist sie die Längere und wird als nächste exploriert. Da l_4 wiederum kürzer ist als die bisherige Engstelle, entspricht die bisher längste Leiter jetzt l_4. Die nun hinzukom-

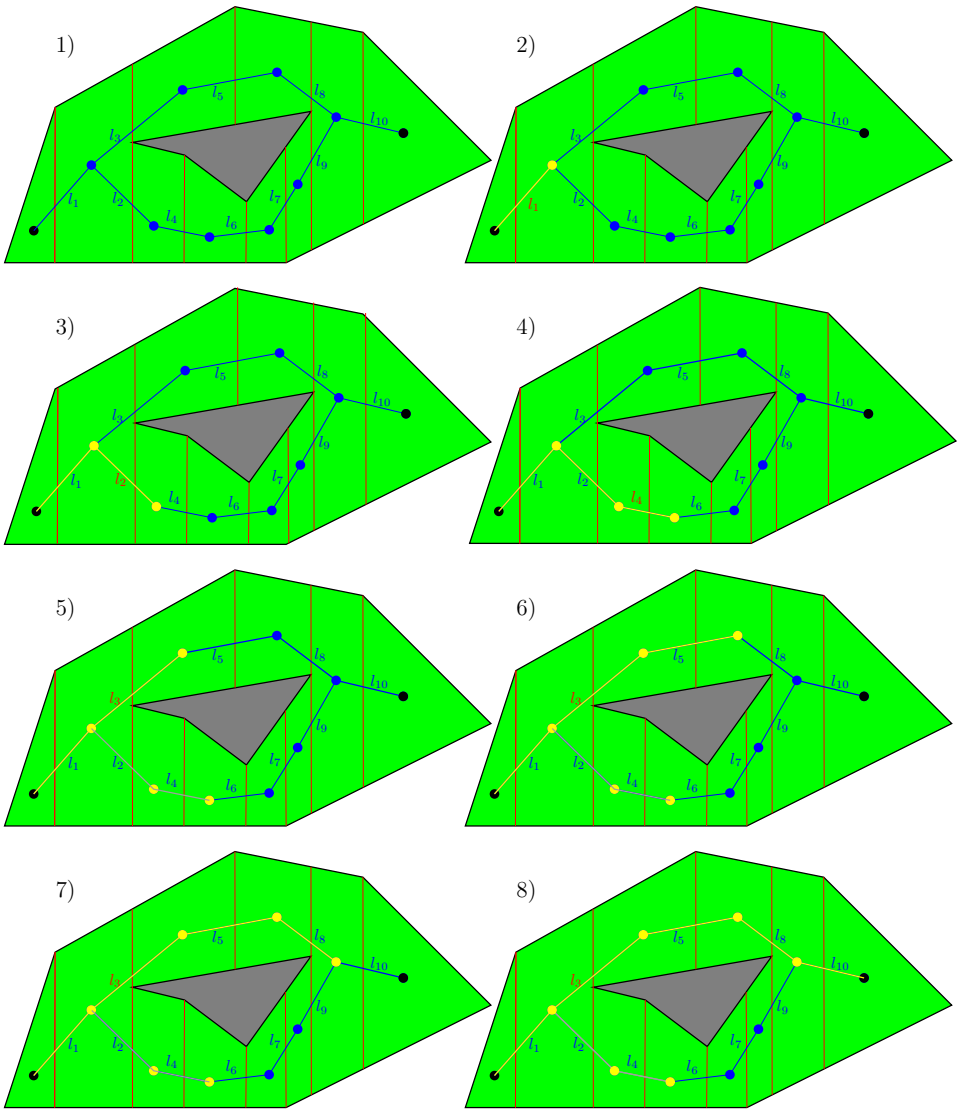


Abbildung 3.6: Die Abbildung zeigt die Schritte bei der Suche innerhalb der Roadmap. Start- und Endknoten sind schwarz markiert; Gelb dargestellt ist der aktuell verfolgte Weg; Bereits besuchte aber zur Zeit nicht relevante Kanten sind grau; Die Engstelle besitzt ein rotes Label.

mende Kante l_6 ist allerdings kürzer als l_3, die somit als nächstes begangen wird. l_3 ist gleichzeitig auch neue Engstelle. Als nächstes werden l_5 und l_8 exploriert, da sie länger sind als l_6. Nun kommen zusätzlich zu l_6 die Kanten l_9 und l_{10} hinzu. Da l_{10} die Längste ist, wird sie als nächstes begangen. Nun ist die Zielregion erreicht und die Engstelle l_3 entspricht der längsten Leiter, die sich zwischen den Regionen bewegen lässt.

3.2.2 Algorithmen

Bei den nun folgenden Algorithmen wird o.b.d.A. angenommen, dass die Leiter eine Orientierung von $\theta = \frac{\pi}{2}$ hat, somit lässt sich die Umgebung mit Hilfe eines horizontalen Sweeps in die Regionen unterteilen. Hat die Leiter eine andere Orientierung, so kann die ganze Umgebung um die Differenz zwischen $\frac{\pi}{2}$ und der tatsächlichen Orientierung der Leiter gedreht werden. Außerdem wird o.b.d.A. angenommen, dass sich die Umgebung in allgemeiner Lage befindet und somit keine Eckpunkte mit gleichen x-Koordinaten existieren. D.h. die Leiter befindet sich in keiner kritischen Orientierung. Eine Partitionierung der Umgebung in Regionen ist auch ohne diese Annahme möglich, sie vereinfacht lediglich die Beschreibung des Algorithmus.

Algorithmus 3.1 Zerlegung der Umgebung in Regionen und Erstellung der Roadmap

---

1. Die Eckpunkte der Umgebung werden nach x-Koordinaten aufsteigend sortiert und als Ereignisstruktur (ES) gespeichert.
2. Aus der ES wird der Punkt mit der kleinsten x-Koordinate entnommen.
3. Dieser Punkt muss ein Teil des Bounding Polygon sein, somit ist er linker Stop einer neuen Region. Die beiden angrenzenden Kanten bilden entsprechend ihrer Lage oberen und unteren Stop der Region.
4. Die offene Region wird in der Sweep-Status-Struktur (SSS) gespeichert.
5. Es wird ein entsprechender Knoten in der Roadmap erzeugt.
6. Der nächste Punkt wird aus der ES entnommen.
7. Dieser Punkt kann entweder Endpunkt eines oberen und/oder unteren Segmentes einer Region in SSS sein, oder es handelt sich um einen neuen freien Punkt. (Abb. 3.7)
   (a) Der Punkt ist Eckpunkt eines oberen **oder** unteren Segmentes **einer** Region aus SSS.
      - Es beginnt eine neue Region, da sich der Rahmen ändert. Die alte Region wird aus der SSS entnommen und als abgeschlossen gespeichert.
      - Es wird eine neue Region mit geändertem oberen/unteren Segment erzeugt und in der SSS gespeichert.
      - Außerdem wird ein neuer Knoten in der Roadmap erzeugt. Die beiden Knoten der alten und neuen Region werden durch

eine Kante verbunden, das Gewicht ergibt sich aus der Länge des gemeinsamen Randes.

(b) Der Punkt ist Eckpunkt des oberen **und** unteren Segmentes **einer** Region aus SSS.

- Oberes und unteres Segment treffen im Eckpunkt zusammen und beenden die Region. Die Region wird aus der SSS entnommen und als abgeschlossen gespeichert.

(c) Der Punkt ist Eckpunkt des oberen Segments einer Region und Eckpunkt des unteren Segments einer zweiten Region aus SSS.

- Die beiden betroffenen Segmente enden in diesem Eckpunkt, und die beiden betroffen Regionen verschmelzen dort zu einer neuen Region. Die zwei alten Regionen werden aus der SSS entnommen und als abgeschlossen gespeichert.
- Es wird eine neue Region mit den beiden verbleibenden Segmenten als oberer und unterer Stop erzeugt und in der SSS gespeichert.
- Außerdem wird ein neuer Knoten in der Roadmap erzeugt. Die Knoten der beiden alten Regionen werden mit dem Knoten der neuen Region durch eine Kante verbunden, das Gewicht ergibt sich aus der Länge des jeweiligen gemeinsamen Randes.

(d) Es handelt sich um einen neuen freien Punkt, der sich innerhalb einer offenen Region aus SSS befindet.

- Da der Punkt sich innerhalb einer Region befindet, handelt es sich um einen Eckpunkt eines Hindernisses oder des Bounding Polygons. Dort teilt sich die Region in zwei Regionen auf. Die alte Region wird aus der SSS entnommen und als abgeschlossen gespeichert.
- Es werden zwei neue Regionen erzeugt und in der SSS gespeichert. Das obere Segment der alten Region wird oberes Segment der oberen neuen Region, das untere Segment der alten Region wird unteres Segment der unteren neuen Region. Die beiden neuen Segmente, die am Eckpunkt anliegen, werden entsprechend oberes/unteres Segment der beiden Regionen.
- Außerdem werden zwei neue Knoten in der Roadmap erzeugt. Der Knoten der alten Region wird mit den Knoten der neuen Regionen durch eine Kante verbunden, das Gewicht ergibt sich aus der Länge des jeweiligen gemeinsamen Randes.

(e) Es handelt sich um einen neuen freien Punkt, der sich außerhalb der offenen Region der SSS befindet.

- Da der Punkt sich außerhalb einer Region befindet, ist dieser Eckpunkt linker Stop einer neuen Region.
- Es wird eine neue Region erzeugt und in der SSS gespeichert. Die beiden an den Eckpunkt angrenzenden Kanten bilden entsprechend ihrer Lage oberen und unteren Stop der Region.
- Es wird ein entsprechender Knoten in der Roadmap erzeugt.

8. Solange noch Eckpunkte in der ES gespeichert sind fahre fort mit Punkt 6.

9. Die Umgebung ist in Regionen aufgeteilt, der Algorithmus ist fertig.

Für die Berechnung der Regionen, in denen sich s und t befinden, kann man naiv alle Regionen testen. Die maximale Länge der Leiter an den Positionen von s und t ergibt sich aus der Entfernung in Leiterrichtung θ vom Punkt s oder t zum oberen Segment der Region.

Nun gilt es nur noch einen Weg zwischen Start- und Zielregion zu finden:

Algorithmus 3.2 Wegesuche in der Roadmap

---

1. **MaxLeiter** wird auf die Länge der maximalen Leiter am Startpunkt s gesetzt.

2. **AktKnoten** ist gleich dem Knoten der Region von s.

3. Füge alle Kanten von AktKnoten in die PriorityQueue **KantenQueue** ein. Kanten mit größtem Gewicht stehen an erster Stelle.

4. Solange **KantenQueue** nicht leer und **AktKnoten** $\neq$ **ZielKnoten** wiederhole:

   (a) Entnehme größte Kante aus der **KantenQueue** und markiere sie als gegangen.

   (b) Falls das Gewicht der Kante kleiner ist als **MaxLeiter** setze **MaxLeiter** auf dieses Gewicht.

   (c) **AktKnoten** wird auf den Zielknoten der Kante gesetzt.

   (d) Alle noch nicht gegangenen Kanten von **AktKnoten** werden in **KantenQueue** eingefügt. Kanten, die sich schon in der Queue befinden, werden nicht nochmals eingefügt, sondern sogar ganz aus der Queue entfernt, da beide Enden der Kante bereits besucht wurden.

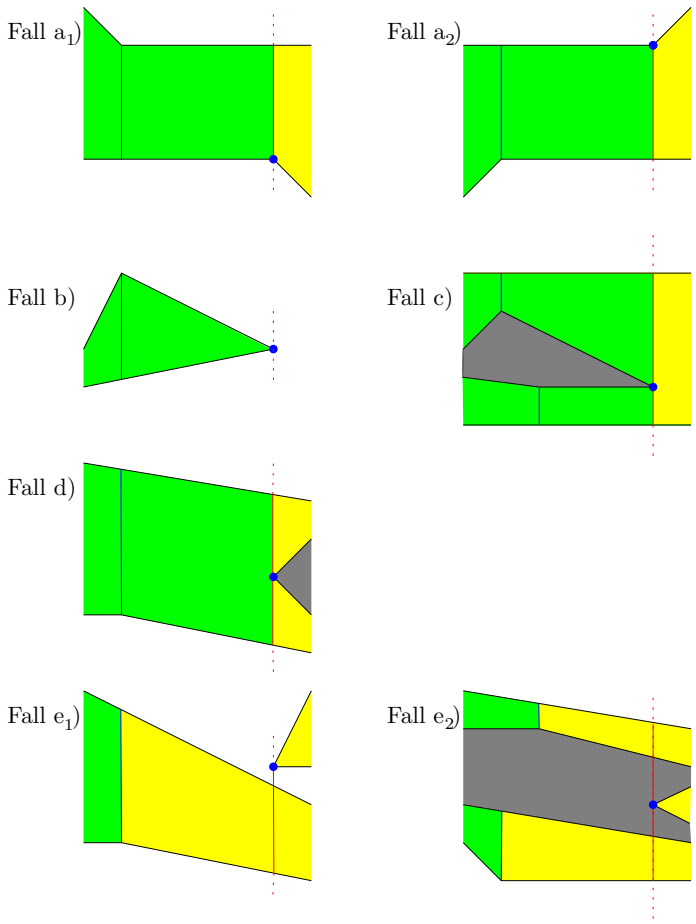


Abbildung 3.7: Fälle von Sweep-Ereignissen bei der Zerlegung der Umgebung in Regionen. Grüne Regionen sind bereits abgeschlossene und fertig bearbeitete Regionen. Offene Regionen aus der SSS sind gelb markiert. Die Sweepline ist rot dargestellt.

5. Falls **AktKnoten = ZielKnoten** ist, so gibt Min(**MaxLeiter**, maximale Leiter am Zielpunkt t) die gesuchte längste Leiter wieder. Andernfalls ist die Umgebung zwischen s und t nicht zusammenhängend und es gibt keinen Weg.

3.2.3 Verifikation

In diesem Abschnitt soll kein vollständiger formaler Beweis für die Korrektheit der Algorithmen gegeben werden. Vielmehr sollen einige Elemente der Algorithmen erklärt werden. Dabei wird gezeigt, dass sie das Richtige berechnen.

Zerlegung der Umgebung

Bei der Zerlegung der Umgebung in Regionen werden die Eckpunkte als Ereignisse des Sweeps benutzt, da sich nur an Eckpunkten eine Änderung des Rahmens Ψ und somit der Region ergeben kann.

Ist dieser Eckpunkt ein neuer freier Punkt, der keine bisherige Region berührt und sich nicht im Inneren einer Region befindet, so entsteht eine neue Region in der Form eines Dreiecks, wie in Abbildung 3.1 Fall 3 gezeigt. Abbildung 3.7 Fall e) zeigt auch, dass es für den Algorithmus unbedeutend ist, ob sich der Punkt außerhalb des bisherigen Bounding Polygon befindet (Fall e_1) oder eine konkave Ecke eines Hindernisses ist (Fall e_1). In beiden Fällen liegt der Punkt außerhalb der bisherigen Regionen und lässt somit eine neue Region entstehen.

Liegt ein freier Eckpunkt im Inneren einer Region (Fall d), so muss es sich um eine Hindernisecke handeln, die eine Region in zwei Teile teilt.

Genau das Entgegengesetzte tritt ein, wenn ein Eckpunkt Ende des unteren Segmentes einer Region und Ende des oberen Segmentes einer anderen Region ist (Fall c). Hierbei vereinen sich die beiden alten Regionen zu einer neuen Region.

Gegenpart zum ersten beschriebenen Fall ist die Konstellation, bei der ein Eckpunkt gleichzeitig Endpunkt des oberen und unteren Segments **einer** Region ist (Fall b). In diesem Fall handelt es ebenfalls um eine dreieckige Region, die dort endet.

Ist der Eckpunkt ein Endpunkt nur des oberen oder unteren Segmentes einer Region, so ändert sich nur das entsprechende Segment.

Durch diesen Sweep werden alle relevanten Ereignisse abgearbeitet, und die Umgebung ist nach der Ausführung komplett in ihre Regionen zerlegt.

Wegesuche

Bei der Wegesuche wird die KantenQueue mit den vom Startknoten ausgehenden Kanten gefüllt. **MaxLeiter** entspricht der Länge der Leiter an der Startposition.

Da die maximale Leiter gesucht ist, wird immer die Kante mit maximalem Gewicht aus der Queue entnommen und begangen. Ist dieses Gewicht kleiner als die bisherige **MaxLeiter**, so wird **MaxLeiter** auf diese Länge reduziert.

Bei Erreichen eines neuen Knotens werden dessen noch nicht begangene Kanten auch in die Queue eingefügt. Somit befinden sich alle zum jeweiligen Zeitpunkt begehbaren Kanten, die noch nicht begangen wurden, in der Queue.

Für den **AktKnoten**, der gerade durch Exploration einer Kante erreicht wurde, gilt somit folgende *Invariante*:

Lemma 3.6 *Für den im Algorithmus 3.2 gerade erreichten* **AktKnoten** *gibt* **MaxLeiter** *die Länge der längsten Leiter an, die sich vom Startpunkt aus zum Rand der* **AktKnoten** *entsprechenden Region bewegen lässt.*

Beweis: siehe nächste Seite

Wird nun der Knoten der Zielregion erreicht, ist auch eine maximale Leiter gefunden, die sich zum Rand der Zielregion bewegen lässt. Dieser Wert muss nur noch mit der maximalen Leiter am Zielpunkt verglichen werden. Das Minimum dieser beiden Werte beschreibt die Länge der längsten Leiter, die sich vom Start- zum Zielpunkt bewegen lässt (siehe Abbildung 3.8).

Wurden alle Kanten, die vom Startknoten aus erreichbar waren, begangen, bevor der Zielknoten erreicht ist, so muss dieser in einer anderen Zusammenhangskomponente der Umgebung liegen. Es gibt also keine Leiter, die sich zwischen Start- und Zielpunkt bewegen lässt.

In dem oben beschriebenen Algorithmus zur Wegesuche wurde bisher nur die Länge der längsten Leiter angegeben. Die Position der entsprechenden Engstelle lässt sich allerdings auf die gleiche Weise speichern, wie es im Algorithmus mit der Länge geschieht.

Ist als Ausgabe auch der Weg vom Start- zum Zielpunkt gefragt, so könnte man zum Beispiel zu jeder Kante in der Queue die zugehörige Kantenfolge bis zu dieser Kante speichern. Da der Weg der längsten Leiter nicht

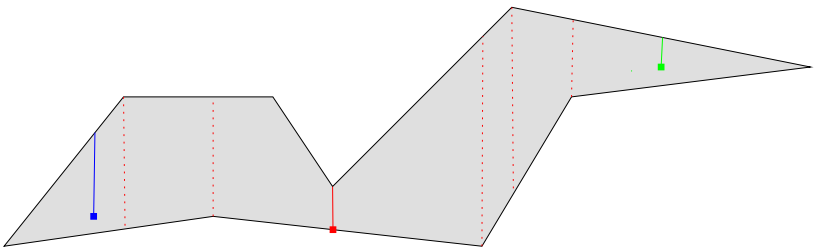


Abbildung 3.8: Engstellen bei der Bewegungsplanung. Blau: Längste Leiter an der Startposition. Grün: Längste Leiter an der Zielposition. Rot: Längste Leiter (Engstelle) bei der Bewegung zwischen den Rändern der Start- und Zielregion. Das Minimum dieser drei Werte ergibt die längste Leiter die sich vom Start- zum Zielpunkt bewegen lässt.

zwangsläufig eindeutig ist, gibt der Algorithmus nur einen möglichen Weg der Leiter wieder, der auch nicht dem kürzesten Weg entsprechen muss. Da aber immer die Kante mit dem größten Gewicht gewählt wird, gibt der Algorithmus den Weg wieder, der bei zentrierter Bewegung den oberen und unteren Abstand der Leiter zu den Rändern der Umgebung maximiert.

Beweis Lemma 3.6:

IA: MaxLeiter enthält die maximale Leiter, die sich am Startpunkt plazieren lässt, und AktKnoten enthält den Knoten der Startregion. Das Lemma ist in diesem Fall trivialerweise erfüllt. ✓

IV: Die Leiter der Länge MaxLeiter_{alt} ist die maximale Leiter, die sich vom Startpunkt aus zur Region von AktKnoten_{alt} bewegen lässt.

IS: In diesem Schritt ist nun zu zeigen, dass wenn Bedingung des Lemmas für den momentanen $AktKnoten_{alt}$ gilt, dann gilt sie auch für den nächsten $AktKnoten_{neu}$. Dabei muss zwischen zwei Fällen unterschieden werden:
Im ersten Fall sind die beiden Knoten durch die gerade begangene Kante verbunden, im zweiten Fall sind beide Knoten nicht durch die aktuelle Kante verbunden.
1.) Die beiden Knoten sind durch die gerade gegangene Kante verbunden. Nach IV ist die Leiter der Länge MaxLeiter_{alt} die maximale Leiter, die sich zur Region von AktKnoten_{alt} bewegen lässt. Wenn der gemeinsame Rand der beiden Regionen, also das Gewicht der Kante größer ist als die maximale Leiter, so lässt sich die Leiter auch in die neue Region bewegen. Im anderen Fall wird MaxLeiter auf die neue Länge angepasst und da immer die größte Kante aus der Queue begangen wird kann es (auch später) keinen maximaleren Weg mehr geben. In beiden Fällen ist die Bedingung des Lemma erfüllt. ✓
2.) Da immer die Kante mit dem größten Gewicht aus der Queue entnommen wird, müssen die beiden Knoten nicht durch die gerade begangene Kante verbunden sein. Da der Knoten K_v am anderen Ende der begangenen Kante zu AktKnoten_{neu} vor AktKnoten_{alt} exploriert wurde, muss dessen MaxLeiter_{K_V} größer/gleich MaxLeiter_{alt} sein. Für die Situation, dass die beiden gleich sind, lässt sich die Argumentation aus Fall 1.) auf diesen Fall übertragen. Für die andere Situation, muss das Gewicht der gerade begangenen Kante sogar kleiner sein als MaxLeiter_{alt}, da die Kante sonst früher aus der Queue entnommen worden wäre. MaxLeiter wird wieder auf die neue maximale Länge angepasst und die Bedingung des Lemma ist erfüllt. ✓

□

3.3 Laufzeit von $O(n \log n)$

Die in diesem Kapitel dargestellte Methode zur Berechnung der Längsten Leiter bei reinen Translationsbewegungen baut auf mehreren Schritten auf. Jeder Schritt hat eine maximale Laufzeit von $O(n \log n)$ und somit gilt auch für die Berechnung der längsten Leiter die Schranke von $O(n \log n)$.

3.3.1 Ausrichtung der Umgebung

Bei der Zerlegung der Umgebung in die Regionen und der Erstellung der Roadmap wird ein horizontaler Sweep von links nach rechts durchgeführt. Dies entspricht einer Orientierung der Leiter von $\theta = \frac{\pi}{2}$. Soll nun die längste Leiter für eine andere Orientierung θ gesucht werden, so muss die Umgebung um den Winkel $\alpha = \theta - \frac{\pi}{2}$ nach rechts gedreht werden. Dazu wird jeder Punkt mit der Matrix

$$\begin{pmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{pmatrix}$$

multipliziert, dadurch wird der Punkt um den Winkel α um den Ursprung gedreht.

Die Matrix-Multiplikation lässt sich in Zeit $O(1)$ durchführen. Für die Drehung der gesamten Umgebung muss jeder der n Eckpunkte einmal mit der Matrix multipliziert werden. Daraus ergibt sich für die Ausrichtung der Umgebung eine Laufzeit von $\mathbf{O(n)}$. □

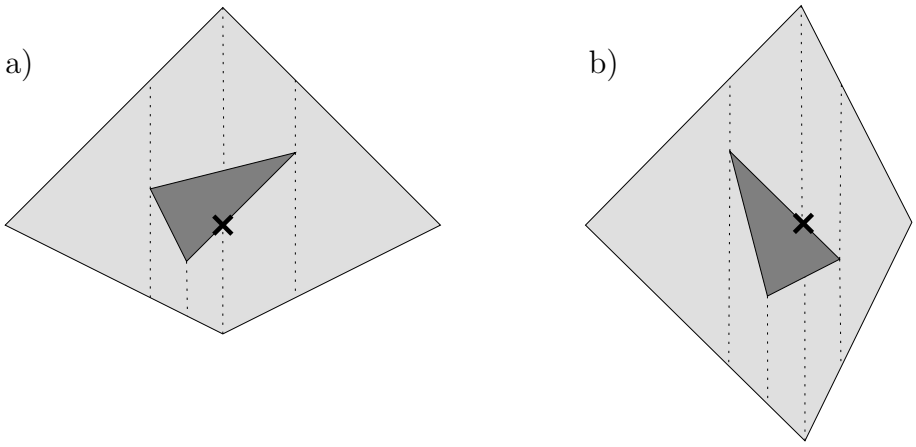


Abbildung 3.9: a) zeigt die ursprüngliche Umgebung und dessen Zerlegung in Regionen für eine Leiter der Orientierung $\theta = \frac{\pi}{2}$.
b) zeigt die um $\alpha = -\frac{\pi}{2}$ rechts gedrehte (= $\frac{\pi}{2}$ Linksdrehung) Umgebung. Dadurch konnte der Sweep die Regionen für eine Leiter der der Orientierung $\theta = 0$ berechnen.

3.3.2 Dekomposition in Regionen

Wie in den meisten Fällen bei der Verwendung eines Sweeps, benötigt der Sweep für die Zerlegung der Umgebung in die Regionen auch $O(n \log n)$ Arbeitsschritte.

- Sortieren der Eckpunkte der Umgebung nach x-Koordinaten und Speichern in der ES. $\mathbf{O(n \log n)}$
- Entnahme des Punktes mit kleinster x-Koordinate aus der ES. $\mathbf{O(1)}$
- In der SSS werden die offenen Regionen der Sweepline entlang vertikal sortiert gespeichert. Mit einer entsprechenden Datenstruktur (z.B. AVL-Baum) lassen sich die Aktionen *Einfügen*, *Entfernen* und *Anfragen* ob ein Punkt innerhalb einer Region liegt oder Eckpunkt einer Region ist in $O(\log n)$ Zeit bearbeiten. $\mathbf{O(\log n)}$
- Erzeugen einer neuen Region, eines neuen Roadmap-Knoten oder einer neuen Roadmap-Kante. $\mathbf{O(1)}$
- Die Umgebung hat n viele Eckpunkte und somit bearbeitet der Sweep n viel Ereignisse. Bei jedem Schleifendurchlauf wird ein Ereignis abgearbeitet. Innerhalb der Schleife werden nur die bereits oben analysierten Aktionen ausgeführt, diese kosten entweder $O(1)$ oder $O(\log n)$ viel Zeit. Somit ergibt sich für die gesamte Schleife: $\mathbf{O(n \log n)}$

□

3.3.3 Bestimmung der Regionen des Start- und Zielpunktes

Zur die Suche der Regionen, in denen sich der Start- und der Zielpunkt befindet, kann man naiv für jede Region testen, ob sich der jeweilige Punkt in dieser Region befindet. Da bei der Dekomposition der Umgebung an jedem neuen Eckpunkt nur maximal zwei neue Regionen entstehen, liegt die Anzahl aller Regionen in $O(n)$. Somit benötigt die naive Suche ebenfalls $\mathbf{O(n)}$ viele Schritte. Die maximale Leiter an diesem Punkt ergibt sich aus der Länge des Segmentes zwischen dem Punkt und dem oberen Segment der Region mit Orientierung θ. Dies lässt sich in konstanter Zeit berechnen.

□

Durch eine geschickte Speicherung der Regionen lässt sich diese Anfragezeit sicher noch auf $O(\log n)$ reduzieren. Darauf wird später noch eingegangen.

3.3.4 Wegesuche in der Roadmap

Wie soeben festgestellt, hat eine Umgebung $O(n)$ viele Regionen, die Roadmap besitzt die gleiche Anzahl an Knoten. Nach Lemma 3.3 haben Regionen bei nicht-kritischen Orientierungen maximal vier Nachbarn. Daraus folgt, dass die Anzahl aller Kanten in der Roadmap ebenfalls durch $O(n)$ beschränkt ist. Die Laufzeit der Wegesuche lässt sich somit folgendermaßen abschätzen:

Wird die PriorityQueue z.B. mit einem AVL-Baum implementiert, so benötigen die Aktionen insert, remove und getMax jeweils $O(\log n)$ Zeit. Jede Kante kann nur einmal in der Queue enthalten sein und wird auch nur einmal begangen. Die Schleife wird also maximal $O(n)$ mal ausgeführt. Da alle Aktionen innerhalb der Schleife maximal $O(\log n)$ Zeit benötigen, ergibt sich eine Laufzeit für die Schleife und damit für die gesamte Suche von $\mathbf{O(n \log n)}$. □

Kapitel 4

Längste Leiter bei allgemeinen Bewegungen

Nachdem sich nun die längste Leiter für reine Translationen berechnen lässt, stellt sich die Frage, wie dieses Verfahren auf allgemeine Bewegungen übertragen werden kann.

Wie auch bei der Lösung des Bewegungsplanungsproblems von Leven und Sharir [11], so wird auch hier das Konzept der Regionen um die Dimension der Orientierung auf Zellen erweitert. Dazu folgen nun einige weitere Definitionen und Vorüberlegungen.

4.1 Definitionen und Vorüberlegungen

Bereits in Kapitel 3 wurde erläutert, dass bei der Berechnung der Regionen für das Längste Leiter Problem eine Leiter mit infinitesimaler Länge verwendet wird. Eine Zelle nach Definition 2.4 lässt sich auch mit folgenden Worten beschreiben: *Eine Zelle ist die Vereinigung über alle Regionen mit gleichem Rahmen* Ψ *über dem Intervall* $[\theta^-, \theta^+]$. Da die Definition der Regionen bereits in Definition 3.1 auf unendlich kurze Leitern angepasst wurde, ändert sich somit für die Definition der Zelle nichts mehr.

Die Länge der maximalen Leiter wird wie bei der Lösung für reine Translationen über gewichtete Kanten bestimmt. Bei der Umsetzung dieser Idee kommen nun einige Fragen auf.

- Wie lassen sich gewichtete Kanten auf das Modell der Zellen übertragen?
- Wo befindet sich die maximale Leiter zur Gewichtung einer Kante zwischen benachbarten Zellen?
- Wie werden Drehungen bei der Gewichtung von Kanten berücksichtigt?

- Wie ändern sich Zellen bei größer werdender Leiter? Engstellen von Regionen können nur an den freien Rändern auftreten, nicht aber im Inneren. Gilt dies auch auf für Zellen?

All diese Fragen sollen im Folgenden beantwortet werden, beginnen möchte ich dabei mit der letzten Frage.

4.1.1 Engstellen einer Zelle

Im letzten Kapitel wurde in Lemma 3.5 gezeigt, dass bei Regionen Engstellen nur an den freien Rändern auftreten. Wie die Abbildungen 4.1 und 4.2 verdeutlichen, kann eine Zelle jedoch auch eine Engstelle im Inneren haben.

Es wurde auch festgestellt, dass sich Regionen bei größer werdender Leiter um die Leiterlänge verkleinern. Der obere Rand bewegt sich dabei in Leiterrichtung um die Leiterlänge auf den unteren Rand zu. In den unten gezeigten Abbildungen kommt es somit bei einer Leiterlänge von sechs Einheiten zu einem Verschwinden der mittleren gelben Region. Die Zelle zerfällt dort somit bei Leitern der Länge > 6 E. in zwei nicht zusammenhängende Komponenten.

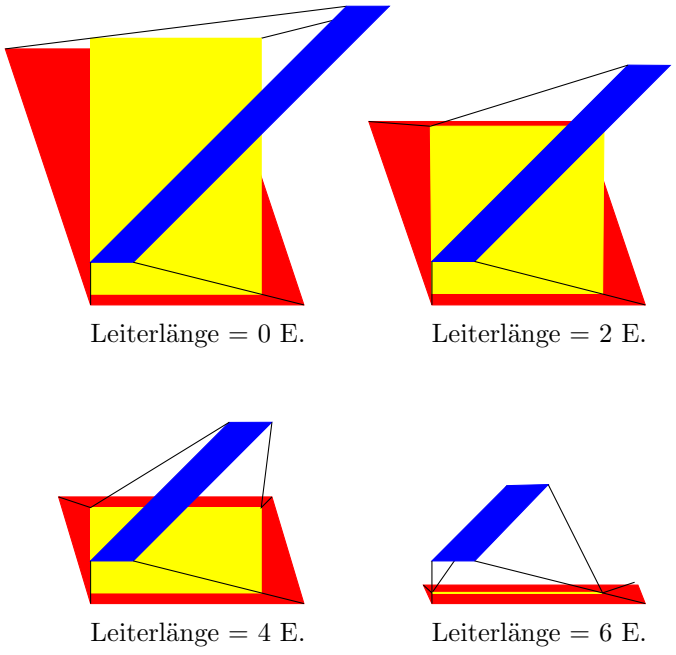


Abbildung 4.1: Veränderung der Zelle bei unterschiedlichen Leiterlängen. Rot: Region mit $\theta^+ = \frac{\pi}{2} + 0,2$; Gelb: Region mit $\theta = \frac{\pi}{2}$; Blau: Region mit $\theta^- = \frac{\pi}{4}$; Grün: Segment mit kritischer Orientierung

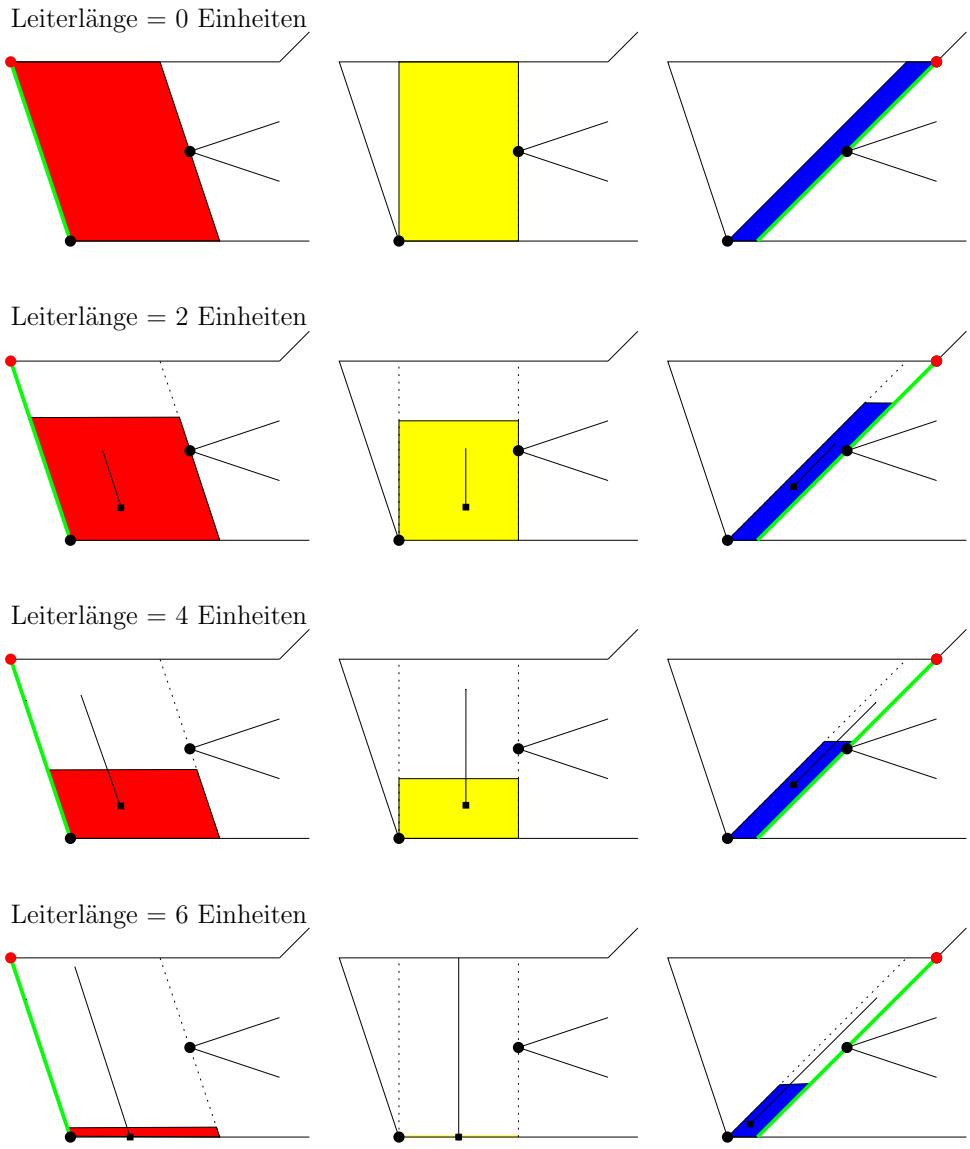


Abbildung 4.2: Veränderung der Regionen in einer Zelle bei unterschiedlichen Leiterlängen. Rot: Region mit $\theta^+ = \frac{\pi}{2} + 0,2$; Gelb: Region mit $\theta = \frac{\pi}{2}$; Blau: Region mit $\theta^- = \frac{\pi}{4}$; Grün: Segment mit kritischer Orientierung

Falls eine Zelle eine solche Engstelle bei einer Orientierung θ_{eng} und einer Leiterlänge l in ihrem Inneren besitzt, so wird die entsprechende Zelle in zwei Zellen mit den Orientierungsbereichen $[\theta^-, \theta_{eng}]$ und $[\theta_{eng}, \theta^+]$ aufgeteilt. In der Roadmap werden die Knoten der beiden Zellen durch eine Kante mit dem Gewicht l verbunden.

Es stellt sich nun die Frage, wann und wie eine Engstelle im Inneren einer Zelle entstehen kann. Da nach Lemma 3.5 eine Region sich nicht in zwei Teile teilen kann, kann eine Zelle bei wachsender Leiter nur den Zusammenhang verlieren, wenn eine ganze Region aus dem Inneren der Zelle „zusammengeschrumpft“ ist. D.h. diese Region $\mathcal{R}_{\theta_{eng}}$ kann keine Leitern ab einer bestimmten Länge enthalten. Wenn es darüber oder darunter Regionen gibt, die größere Leitern enthalten können, so gibt es aber keinen Weg in der Zelle zwischen diesen Regionen, da diese Leitern nicht die Orientierung θ_{eng} überschreiten können.

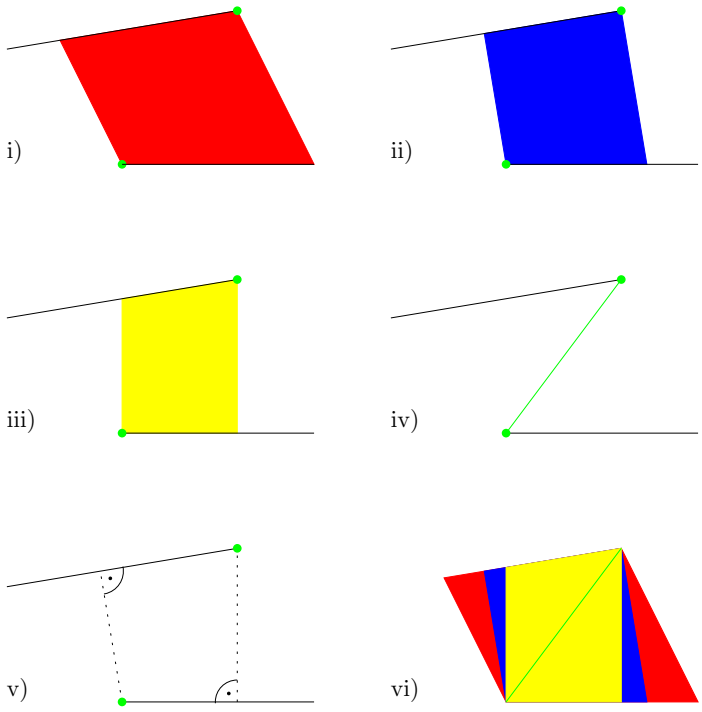


Abbildung 4.3: Engpass in einer Zelle bei seitlichen Stops des Falls 1. Darstellung von verschiedenen Regionen einer Zelle, die seitlichen Stops sind grün markiert. i) Region mit kritischer Orientierung θ^+. ii) Region mit Orientierung senkrecht auf dem oberen Segment. iii) Region mit Orientierung senkrecht auf dem unteren Segment. iv) Region mit kritischer Orientierung θ^-. v) Kürzeste linke und rechte Kante. vi) Topansicht der Zelle.

Daraus folgt, dass es im Inneren einer Zelle eine Region geben muss, die sich schneller verkleinert als Regionen, die darüber oder darunter liegen. Abbildung 4.2 zeigt noch einmal, wie sich eine Zelle bei wachsender Leiter verkleinert. Da die beiden seitlichen freien Kanten der Region beide in Leiterrichtung ausgerichtet sind und der obere Rand sich bei wachsender Leiter in die gleiche Richtung nach unten bewegt, entspricht die längere der beiden seitlichen Kanten der Länge der größten Leiter, die noch in der Region plaziert werden kann. Nun gilt es zu analysieren, bei welcher Orientierung θ_{eng} die entsprechende Region die kürzeste maximale Leiter hat.

Abbildung 4.3 zeigt den Fall, in dem linker und rechter Stop Eckpunkt des oberen oder unteren Segmentes sind (Fall 1, Def. 3.1). Da beide Stops nach Definition des Rahmens Ψ auch immer Eckpunkte in allen Regionen der Zelle sind, kann man sehr einfach die Orientierung des kürzesten linken bzw. rechten Randes der Regionen bestimmen. Aus der Algorithmischen Geometrie ist bekannt, dass der kürzeste Abstand zwischen einem Punkt und einer Gerade durch das Lot vom Punkt auf die Gerade gegeben ist. Der kürzeste linke/rechte Rand unter allen Regionen entspricht somit dem Lot des linken/rechten Eckpunktes auf das gegenüberliegende Segment. Dies gilt nur, solange die entsprechende Orientierung im Orientierungsbereich $[\theta^-, \theta^+]$ der Zelle liegt. Ansonsten hat der entsprechende kürzeste Rand die Orientierung θ^- oder θ^+ (siehe Abbildung 4.4).

Wenn ein seitlicher Stop eine neue Hindernisecke ist (Fall 2, Def. 3.1), so wird die Bestimmung der entsprechenden kürzesten Randkante etwas komplizierter. Der Punkt dieses seitlichen Stops ist ebenfalls in allen Regionen der Zelle enthalten, allerdings ist er kein Eckpunkt der Region sondern er liegt im Inneren des seitlichen Randes. Abbildung 4.5 i) zeigt: Weder das

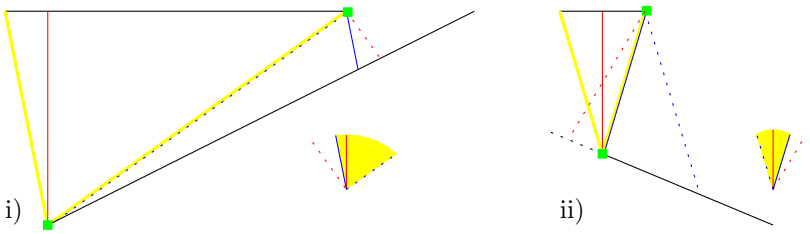


Abbildung 4.4: In beiden Fällen liegt die Senkrechte des rechten Stops auf das untere Segment nicht im Orientierungsbereich $[\theta^-, \theta^+]$. Die seitlichen Stops sind grün markiert, die kritischen Orientierungen θ^-, θ^+ sind gelb, die Senkrechten sind rot und die Alternativen für den rechten Stop sind blau dargestellt. i) Zelle hat eine Engstelle, da die kürzeste linke Randkante länger ist und dessen Region im Inneren der Zelle liegt. ii) Zelle hat keine Engstelle, da die kürzeste rechte Randkante länger ist und dessen Region eine Außenfläche der Zelle ist.

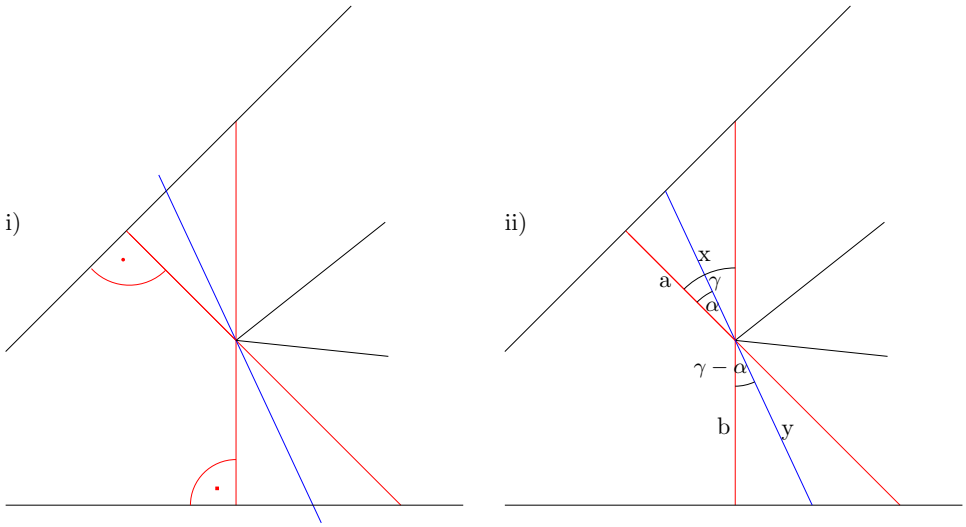


Abbildung 4.5: Engpass in einer Zelle bei seitlichem Stop des Falls 2

Lot auf das untere noch das Lot auf das obere Segment entsprechen der kürzesten Randkante der Regionen. Die Orientierung der kürzesten Randkante liegt zwischen den Orientierungen der beiden Senkrechten. Mit den Beschriftungen aus Abbildung 4.5 ii) lässt sich die Orientierung der kürzesten Randkante folgendermaßen berechnen.

$$\cos(\alpha) = \frac{a}{x}\ ,\ \cos(\gamma - \alpha) = \frac{b}{y}$$

$$x + y = \frac{a}{\cos(\alpha)} + \frac{b}{\cos(\gamma - \alpha)}$$

$$(x + y)' = \frac{a * \sin(\alpha)}{\cos^2(\alpha)} + \frac{b * \sin(\gamma - \alpha)}{\cos^2(\gamma - \alpha)}$$

Die Nullstelle dieser Ableitung im Intervall von $[0, \gamma]$ abgezogen von der Orientierung des Segments a ergibt die Orientierung des kürzesten Randes einer Region an diesem seitlichen Stop. Dies gilt wiederum nur, solange die entsprechende Orientierung im Orientierungsbereich $[\theta^-, \theta^+]$ der Zelle liegt. Ansonsten hat der entsprechende kürzeste Rand wieder die Orientierung θ^- oder θ^+ (siehe Abbildung 4.6).

Bei einem seitlichen Stop wie in Fall 3 (Def 3.1) hat die Region an dieser Seite keine Kante und deren Länge ist folglich gleich null.

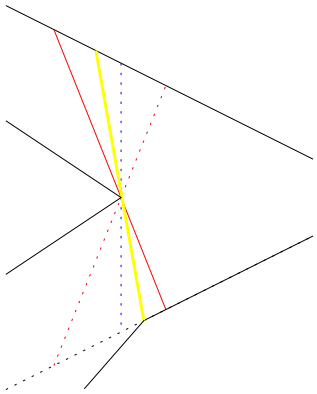


Abbildung 4.6: Engpass in einer Zelle bei seitlichem Stop des Falls 2 - Sonderfall. Das kürzeste Segment (blau) kann nicht im Orientierungsbereich liegen. Das Segment mit kritischer Orientierung (gelb) ist somit die kürzeste linke Randkante.

Da die längere der beiden seitlichen Kanten einer Region der Länge der größten Leiter entspricht, die noch in der Region plaziert werden kann, ist die größere der kürzesten linken und rechten Randkante die längste Leiter, die in der Engpassregion platziert werden kann. Die Orientierung dieser Kante entspricht der Orientierung der Engpassregion. Der Zusammenhang einer Zelle wird allerdings nur dann unterbrochen, wenn diese Orientierung im Inneren des Orientierungsbereichs der Zelle liegt (vgl. Abbildung 4.4 ii)).

Abschließend kann man noch feststellen, dass es wegen Lemma 4.1 nur eine Engstelle in der Zelle geben kann. Denn bei einer Rechts- oder Linksdrehung von der Orientierung θ_{eng} weg kann die maximale Leiterlänge in den entsprechenden Regionen nur wachsen.

Lemma 4.1 *Die Länge der seitlichen Kanten einer Region wächst streng monoton bei einer Rechts- oder Linksdrehung, die sich von der Orientierung der kürzesten Kante entfernt.*

Beweis:
Zunächst einmal kann man feststellen, dass die Orientierungen der oberen und unteren Kanten ebenfalls kritische Orientierungen sind, an denen sich der Rahmen Ψ ändert.
Die Analyse der Funktion für seitliche Kanten des Typs 2 (Abb. 4.7)

$$x + y = \frac{a}{\cos(\alpha)} + \frac{b}{\cos(\gamma - \alpha)}$$

zeigt genau das im Lemma beschriebene Verhalten für α aus dem Schnitt der Intervalle $]-\frac{\pi}{2}, \frac{\pi}{2}[\cap](-\frac{\pi}{2}+\gamma), (\frac{\pi}{2}+\gamma)[$. Die Intervallgrenzen entsprechen

den beiden relativen Orientierungen der unteren und oberen Segmente. Für seitliche Kanten des Typs 1 gilt (Abb. 4.8):

$$\text{Kantenlänge} = \frac{\text{Länge der Kürzesten}}{\cos(\beta)}$$

Die Drehung um β kann dabei maximal im Intervall $]-\frac{\pi}{2}, \frac{\pi}{2}[$ liegen. Der COS hat über diesem Intervall bei $\beta = 0$ sein Maximum und fällt zu beiden seiten streng monoton ab. Die Funktion der Kantenlänge hat somit ebenfalls die im Lemma beschriebene Eigenschaft. □

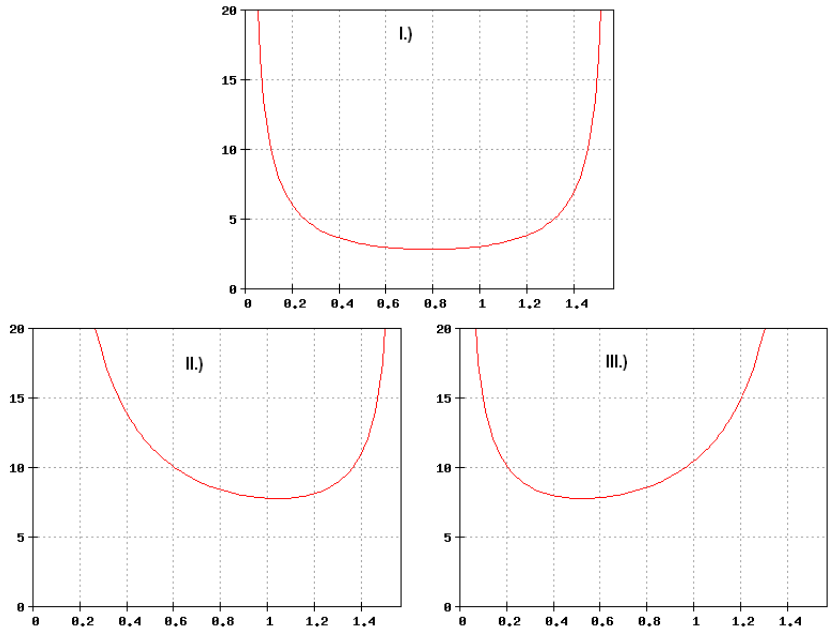


Abbildung 4.7: Beispiele der Funktion $x + y = \frac{a}{\cos(\alpha)} + \frac{b}{\cos(\gamma-\alpha)}$ mit $\gamma = \frac{\pi}{2}$: I.) a = 1, b = 1. II.) a = 1, b = 5. III.) a = 5, b = 1.

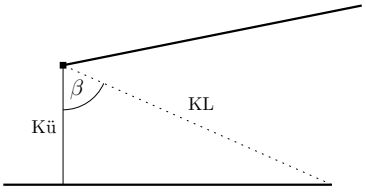


Abbildung 4.8: Kantenlänge am Eckpunkt des Typs 1: KL = $\frac{\text{Kü}}{\cos(\beta]}$

4.1.2 Gewichtete Kanten bei Zellen

Nun ist zu klären, wie sich gewichtete Kanten auf das Modell der Zellen übertragen lassen.

Die erste Art von Kanten wurde im vorherigen Kapitel eingeführt. Wenn eine Zelle in ihrem Inneren eine Engstelle hat, so wird sie in der Roadmap durch zwei Knoten repräsentiert. Beide Knoten sind durch eine Kante verbunden, die mit der Länge der maximalen Leiter an dieser Engstelle gewichtet ist. Es handelt sich dabei um eine Kante, die eine Drehung im Inneren einer Zelle beschreibt.

Weiter gibt es noch eine zweite Art von Drehungskanten. Wenn bei einer Drehung der Leiter die kritische Orientierung der Zelle überschritten wird, so ändert sich nach Definition der Rahmen Ψ und die Leiter befindet sich in einer neuen Zelle. Abbildung 4.9 zeigt ein Beispiel eines solchen Übergangs.

Somit werden in der Roadmap die Knoten der beiden zugehörigen Zellen durch eine Kante verbunden. Gewichtet wird die Kante mit der Länge der maximalen Leiter der kritischen Region. Diese entspricht wie im vorherigen Kapitel 4.1.1 der längeren der beiden seitlichen Kanten der Region.

Als drittes gibt es Kanten, die den Übergang zwischen zwei benachbarten Zellen mittels Translation beschreiben. Da dabei keine Drehung stattfindet, entspricht dies dem Übergang zwischen Regionen aus den benachbarten Zellen, die beide die gleiche Orientierung haben. Da es Aufgabe des Algorithmus ist, die maximale Leiter zu finden, wird die Roadmapkante zwischen den Zellknoten mit der Länge der maximalen gemeinsamen Randkante der Regionen der beiden Zellen gewichtet.

Die Orientierung der Regionen mit maximalem gemeinsamen Rand muss im Bereich des Schnitts der Orientierungsbereiche der beiden Zellen liegen,

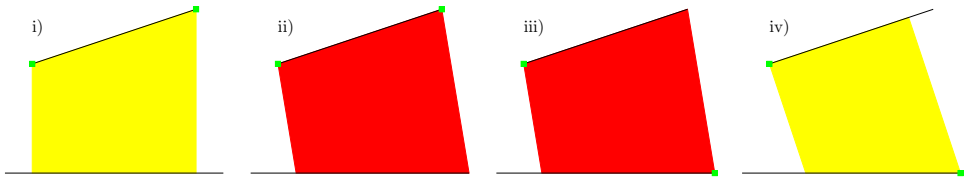


Abbildung 4.9: Drehung über die kritische Orientierung. i) Nicht kritische Region mit den seitlichen Stops oben links und rechts. ii) Region mit kritischer Orientierung und den seitlichen Stops oben links und rechts. iii) Region mit kritischer Orientierung und den seitlichen Stops links oben und rechts unten. iv) Nicht kritische Region mit den seitlichen Stops links oben und rechts unten.

da die benachbarten Regionen gleiche Orientierung haben müssen.

$$[\theta^-_{Schnitt}, \theta^+_{Schnitt}] = [\theta^-_1, \theta^+_1] \cap [\theta^-_2, \theta^+_2]$$

Außerdem folgt aus Lemma 4.1, dass die maximalen seitlichen Randkanten der Regionen nicht im Inneren des Orientierungsbereiches liegen kann. Daher ergibt sich die maximale Leiterlänge, mit der die Translationskante gewichtet wird, aus dem Maximum der gemeinsamen Randkante der beiden Regionen mit Orientierung $\theta^-_{Schnitt}$ und $\theta^+_{Schnitt}$.

4.1.3 Bewegung der Leiter innerhalb der Zelle

Nachdem nun die Fragen zu Engstellen innerhalb der Zellen und maximalen Übergängen zwischen den Zellen geklärt ist, bleibt die Frage offen, wie eine Leiter innerhalb einer Zelle bewegt werden kann. Wenn in diesem Kapitel die Rede von der Zelle ist, so ist immer die bereits an der Engstelle geteilte Zelle gemeint.

Lemma 4.2 *Innerhalb einer Zelle ohne Engstelle lässt sich eine Leiter zwischen zwei Plazierungen bewegen, falls beide Plazierungen im FP liegen. Die Bewegung ist mit der Kombination* Translation, Rotation, Translation *möglich.*

Beweis:
Zunächst wird die Leiter durch Translation von der Position der Startplazierung zu dem längeren seitlichen Rand bewegt. Dies ist nach Korollar 3.4 innerhalb von Regionen problemlos möglich. Da die Zelle keine Engstelle hat, lässt sich die Leiter an dieser Stelle in die Orientierung der Zielplazierung drehen. Anschließend lässt sich die Leiter durch Translation an die Position der Zielplazierung bewegen. □

Für die Suche nach der Längsten Leiter bedeutet dies, dass sich die Leiter innerhalb der Zelle zwischen den Plazierungen von zwei Übergängen (Kanten) bewegen lässt. Denn die Längste Leiter entspricht der kleinsten begangenen Kante im Graphen.

4.1.4 Veränderungen an kritischen Orientierungen

Zum besseren Verständnis der späteren Berechnungen soll nun erläutert werden, wie sich Regionen und damit auch die Zellen an kritischen Orientierungen verändern.

Bei kritischen Orientierungen auf Grund einer Engstelle von Zellen wird der Knoten mit den zugehörigen Kanten lediglich dupliziert, alles andere bleibt unverändert. Von größerem Interesse sind die kritischen Orientierungen, an denen sich der Rahmen der Regionen bzw. Zellen ändert.

Bereits in Kapitel 2 wurde erwähnt, dass die Orientierungen der Sichtbarkeitskanten zwischen Eckpunkten der Umgebung genau den Orientierungen entsprechen, an denen sich der Rahmen einer Region ändert. Denn für den Fall dieser kritischen Orientierung wären beide Eckpunkte des Sichtbarkeitssegmentes gleichzeitig seitlicher Stop, bei einer minimalen Änderung der Orientierung aber nur einer von beiden (siehe Abbildung 4.9 i) - iv)).

Somit kann man aber auch feststellen, dass von einer kritischen Orientierung eines Sichtbarkeitssegmentes nur solche Regionen/Zellen betroffen sein können, die mindestens einen Endpunkt des Segmentes als seitlichen Stop haben. Jeder Eckpunkt kann für eine feste Orientierung maximal in drei Regionen/Zellen seitlicher Stop sein. Da eine Region beide Eckpunkte des Sichtbarkeitssegmentes enthält, können von einer kritischen Orientierung maximal fünf Regionen/Zellen betroffen sein.

Im folgenden werden einige Beispiele mit unterschiedlichen Kombinationen von Eckpunkten gezeigt und erläutert.

Fall 1:

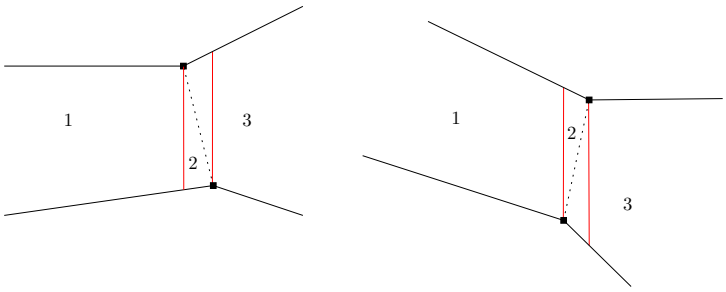


Abbildung 4.10: Änderung der Regionen bei Überschreitung der Kritischen Orientierung zwischen zwei Eckpunkten des Typs 1

Bei der linken Region (1) bleiben oberes und unteres Segment unverändert, es ändert sich nur der Eckpunkt für den rechten Stop. In der mittleren Region (2) tauschen die beiden Eckpunkte ihre Position als rechter und linker

Stop, außerdem ändern sich oberes und unteres Segment. Die rechte Region (3) hat nur einen geänderten seitlichen linken Stop.

Fall 2:

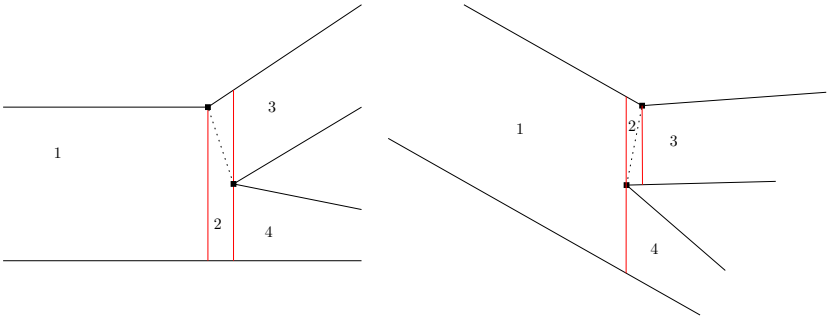


Abbildung 4.11: Änderung der Regionen bei Überschreitung der Kritischen Orientierung zwischen einem Eckpunkt des Typs 1 und einem des Typs 2

Bei der linken Region (1) bleiben oberes und unteres Segment wieder unverändert, es ändert sich nur der Eckpunkt für den rechten Stop. In der mittleren Region (2) tauschen die beiden Eckpunkte ihre Position als rechter und linker Stop, außerdem ändern sich oberes und unteres Segment. Die rechte Region (3) hat nur einen geänderten seitlichen linken Stop, der Rahmen der Region (4) bleibt unverändert.

Fall 3:

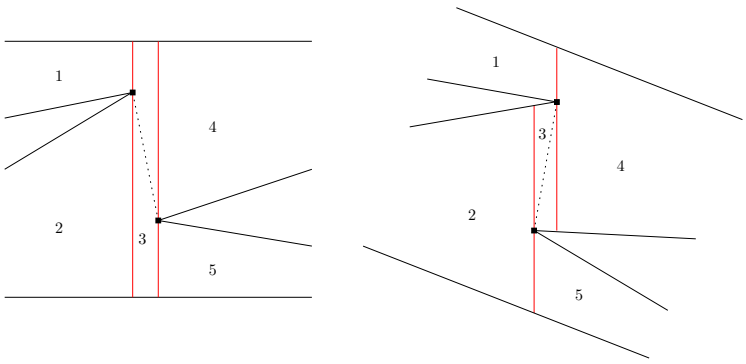


Abbildung 4.12: Änderung der Regionen bei Überschreitung der Kritischen Orientierung zwischen zwei Eckpunkten des Typs 2

Region (1) Änderung des rechten Stops, Region (3) Änderung des gesamten Rahmens, Region (4) Änderung des linken Stops, Regionen (2)u.(5) keine Änderung.

Fall 4:

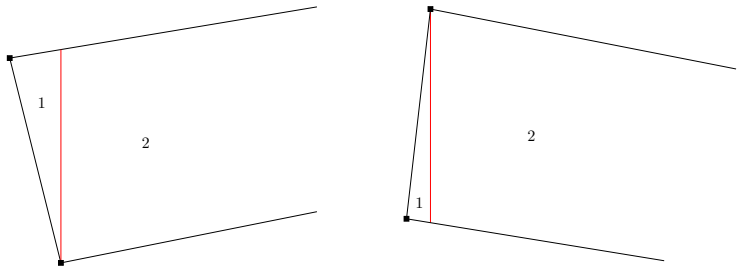


Abbildung 4.13: Änderung der Regionen bei Überschreitung der Kritischen Orientierung zwischen einem Eckpunkt des Typs 1 und einem des Typs 3

Region (1) Änderung des gesamten Rahmens, Region (2) Änderung des linken Stops.

Fall 5:

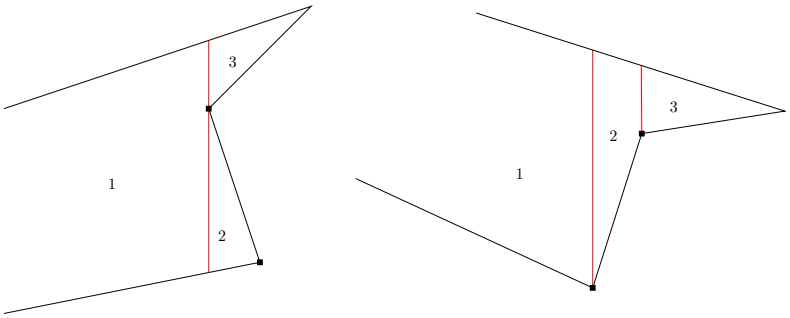


Abbildung 4.14: Änderung der Regionen bei Überschreitung der Kritischen Orientierung zwischen einem Eckpunkt des Typs 2 und einem des Typs 3

Region (1) Änderung des rechten Stops, Region (2) Änderung des gesamten Rahmens, Region (3) keine Änderung.

Damit sind alle möglichen Kombinationen von Eckpunkten abgedeckt. Aus diesen Beispielen kann man erkennen, dass sich nur bei Regionen, die beide Eckpunkte des kritischen Sichtsegmentes enthalten, der gesamte Rahmen ändert. Bei den Anderen kann sich nur der betroffene Eckpunkt ändern.

Aus dieser Erkenntnis lässt sich somit eine Idee zur Berechnung dieses neuen Teilbereichs ableiten. Die linken Regionen (nur der rechte Stop ist Eckpunkt des kritischen Sichtsegmentes) kommen zum Start als offene Regionen in die SSS und die beiden Eckpunkte des Sichtsegmentes sind Ereignisse. Die neuen Regionen können somit für die neue Orientierung mit einem Sweep

berechnet werden. Bei den offen Regionen, die am Ende des Sweeps noch in der SSS sind, handelt es sich um die rechten Regionen (nur der linke Stop ist Eckpunkt des kritischen Sichtsegmentes), deren rechter Stop unverändert bleibt und von dessen alten Regionen übernommen werden kann.

Da sich wie z.B. im Fall 3 nicht der Rahmen aller Regionen ändert, befinden sich nur die neu berechneten Regionen auch in neuen Zellen, deren Rahmen sich geändert hat. Für diese neuen Zellen werden entsprechende Roadmapknoten und -kanten erzeugt.

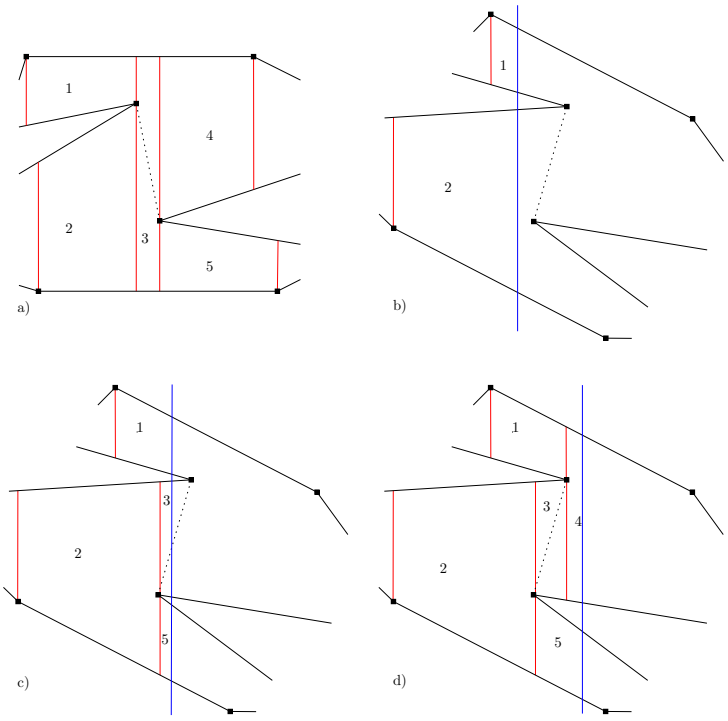


Abbildung 4.15: Beispiel für die Neuberechnung der Regionen mittels Sweep bei einer Beispielumgebung mit kritischer Orientierung des Falls 3

Die Abbildung 4.15 - a) zeigt die Umgebung vor Überschreiten der kritischen Orientierung. In b) wurden die betroffenen Regionen gedreht. Die linken Regionen 1;2 wurden als offene Regionen in der SSS gespeichert, denn deren linker Stop sowie deren oberes und unteres Segment bleiben unverändert. In c) überschreitet die Sweepline den ersten Eckpunkt. Die Region 2 wird abgeschlossen und zwei neue offene Regionen 3;5 werden in der SSS gespeichert. Nach Überschreiten des zweiten Eckpunktes d) werden die Regionen 1;3 abgeschlossen und die neue offene Region 4 wird in der SSS gespeichert. Da kein weiteres Ereignis folgt, können die Regionen 4;5 ihren rechen Stop von den Vorgängern aus a) übernehmen.

4.2 Berechnung der Leiterbewegung

Nachdem nun weitere Grundlagen gelegt wurden, soll im folgenden Abschnitt die Idee zur Berechnung der längsten Leiter für Translation und Rotation kompakt beschrieben werden. Anschließend werden wie im Kapitel 3 für Translationen die verwendeten Algorithmen beschrieben, und abschließend wird die Plausibilität des Verfahrens verifiziert.

4.2.1 Idee

Wie bei der Lösung des *Längste Leiter Problems* für reine Translationen, so beginnt die Lösung des allgemeinen Falls mit der Zerlegung der Umgebung in Regionen.

Im allgemeinen Fall (Translations- und Rotationsbewegungen) wird die Umgebung in Regionen der Orientierung $\theta = 0$ zerlegt, diese Orientierung ist o.b.d.A. eine nicht-kritische Orientierung. Für die Zerlegung in Regionen führt man anstatt eines horizontalen Sweeps einen vertikalen Sweep von oben nach unten durch. Die Eckpunkte werden in diesem Fall nach y-Koordinaten absteigend sortiert. Alternativ kann die Umgebung auch in linearer Zeit um $\frac{\pi}{2}$ nach links gedreht und dann ein horizontaler Sweep ausgeführt werden.

Da $\theta = 0$ eine nicht-kritische Orientierung ist, liegt jede Region in genau einer Zelle. Der Rahmen der Zelle entspricht dem Rahmen der Region und als Start-Orientierung wird $\theta^- = 0$ gewählt. θ^+ bleibt vorerst offen, so dass die Zelle einen vorläufigen Orientierungsbereich von $[0, —]$ hat. Wie in Kapitel 3 wird für jede Zelle/Region ein Roadmapknoten erzeugt, welcher jeweils mit den Knoten der Nachbarregionen/-zellen durch Kanten verbunden ist. Die Kanten erhalten ebenfalls einen vorläufigen Orientierungsbereich von $[0, —]$ und vorerst kein Gewicht.

Bereits in Kapitel 2 wurde festgestellt, dass sich bei einer minimalen Drehung der Orientierung die Struktur der entsprechenden Roadmap der Regionen nicht ändert. Erst an einer kritischen Orientierung ändert sich der Rahmen einer Zelle und es entsteht eine neue Zelle. Kritische Orientierungen sind alle Orientierungen der Kanten des Sichtbarkeitsgraphen der Umgebung. Da es sich bei diesem Verfahren zur Berechnung der Längsten Leiter um eine 360°-Lösung handelt, sind beide Orientierungen einer Sichtbarkeitskante auch relevante kritische Orientierungen. In Kapitel 4.1.1 wurde beschrieben, dass Zellen an ihren Engstellen aufgeteilt werden, somit sind die Orientierungen der Engstellen ebenfalls kritisch.

Der Sichtbarkeitsgraph der Umgebung lässt sich z.B. mit dem Topologischen Sweep von Edelsbrunner und Guibas [3] berechnen. Dabei können $\Theta(n^2)$ viele Sichtbarkeitskanten entstehen. Für die $O(n)$ vielen bereits berechneten Zellen gibt es ebenfalls schon die Orientierung der Engstelle. Die Orientierung der Engstellen von Zellen, die im Laufe des Verfahrens erst entstehen, werden später hinzugefügt.

Alle kritischen Orientierungen werden von 0 an aufsteigend sortiert. Die kleinste Orientierung dieser Liste ist die erste Orientierung, bei der sich Zellen ändern. Die betroffenen Zellen und deren Kanten werden mit dieser Orientierung als θ^+ abgeschlossen, außerdem lassen sich nun die Gewichte der abgeschlossen Kanten bestimmen. Es werden neue Zellen berechnet und entsprechende Roadmapknoten und -kanten erzeugt. Die Orientierungen der Engstellen der neuen Zellen werden in die sortierte Liste der kritischen Orientierungen eingefügt. Dies wiederholt sich, so lange es noch kritische Orientierungen in der Liste gibt.

Nachdem alle kritischen Orientierungen bearbeitet wurden gibt es Zellen mit den Orientierungsbereichen $[0, \theta_i]$ und $[\theta_j, —]$. Da aber 0 keine kritische Orientierung ist, gibt es darunter immer zwei Zellen mit gleichem Rahmen Ψ, die zu einer Zelle mit dem Orientierungsbereich $[\theta_j, \theta_i]$ zusammengefasst werden können.

Somit ist eine komplette Roadmap für Rotations- und Translationsbewegungen berechnet worden. Für beliebige Start- und Zielpositionen kann jetzt wieder die längste Leiter bestimmt werden, die sich zwischen beiden Positionen bewegen lässt. Dazu werden die Start- und Zielzelle ermittelt und mit dem Suchalgorithmus 3.2 die maximale Leiter berechnet.

4.2.2 Algorithmen

Nachdem die Idee zur Berechnung der längsten Leiter für allgemeine Bewegungen kompakt zusammengefasst wurde, sollen in diesem Kapitel Details zur Berechnung behandelt werden. Dabei können die Algorithmen aus Kapitel 3.2.2 direkt oder leicht modifiziert übernommen werden.

Da die Umgebung in Regionen mit der Orientierung $\theta = 0$ zerlegt und der Algorithmus 3.1 verwendet werden soll, wird die Umgebung zuerst um $-\frac{\pi}{2}$ nach rechts gedreht. Dies geschieht durch Multiplikation jedes Eckpunktes mit der Matrix $\begin{pmatrix} \cos -\frac{\pi}{2} & \sin -\frac{\pi}{2} \\ -\sin -\frac{\pi}{2} & \cos -\frac{\pi}{2} \end{pmatrix} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$. Nun kann die Umgebung mit Algorithmus 3.1 in die Regionen zerlegt werden. Der Algorithmus wird lediglich dahingehend modifiziert, dass Kanten nicht gewichtet werden und zusätzlich der Orientierungsbereich $[0, —]$ gespeichert wird. Außerdem werden Zeiger von den Eckpunkten auf Regionen/Zellen, deren seitliche Stops sie sind, gespeichert. Diese Zeiger helfen später, die bei einer kritischen Orientierung betroffenen Zellen schnell zu finden.

Für jede berechnete Zelle wird nun die Orientierung der Engstelle bestimmt, falls eine solche existiert. Dazu werden zuerst die minimalen Leitern an den beiden seitlichen Stops berechnet. Wenn die Orientierung der Längeren der beiden Leitern im Inneren des Orientierungsbereichs der Zelle liegt, so befindet sich dort eine Engstelle. Abschnitt 4.1.1 zeigt, dass bei seitli-

chen Stops des Typs 1 die minimale Leiter an dieser Position dem Lot vom Stop auf das gegenüberliegende Segment entspricht. Bei Stops des Typs 2 ergibt die Nullstelle von $(x+y)' = \frac{a*\sin(\alpha)}{\cos^2(\alpha)} + \frac{b*\sin(\gamma-\alpha)}{\cos^2(\gamma-\alpha)}$ die Orientierung der minimalen Leiter an diesem Stop.

Mit dem Topologischen Sweep von Edelsbrunner und Guibas [3] werden alle Sichtbarkeitssegmente innerhalb der Umgebung berechnet.

Alle bis jetzt bekannten kritischen Orientierungen werden in einer sortierten Datenstruktur gespeichert. Die kleinste Orientierung entspricht der Orientierung, an der sich die ersten Zellen ändern. Somit gilt zu Beginn jedes Schleifendurchlaufs des folgenden Algorithmus die Invariante: Die Roadmap enthält alle Knoten und Kanten für die Suche der Größten Leiter über dem Intervall $[0, \theta_i]$. (θ_i = nächste kritische Orientierung)

Algorithmus 4.1 Update der Roadmap an kritischen Orientierungen

---

1. Entnehme kleinste kritische Orientierung θ_i aus der Datenstruktur.
2. Ermittle die betroffenen Zellen.
3. Berechne neue Zellen mit dem in Kapitel 4.1.4 beschriebenen Sweep. Handelt es sich um eine Orientierung auf Grund einer Engstelle einer Zelle, so wird diese Zelle nur geteilt und es ist kein Sweep erforderlich.
4. Knoten, deren Zellen sich geändert haben, und deren Kanten werden abgeschlossen.
   (a) Die Knoten erhalten die End-Orientierung $\theta^+ = \theta_i$.
   (b) Alle Translationskanten der Knoten erhalten die End-Orientierung $\theta^+ = \theta_i$.
   (c) Kanten, die eine Translation beschreiben, können wie in Kapitel 4.1.2 beschrieben gewichtet werden, da sich der gemeinsame Orientierungsbereich mit Nachbarzellen nicht mehr ändern kann.
5. Für jede neue Zelle wird ein Knoten in der Roadmap mit den zugehörigen Kanten erzeugt.
   (a) Die neuen Knoten erhalten die Start-Orientierung $\theta^- = \theta_i$.
   (b) Zwischen dem neuen Knoten und seinem Vorgänger werden Drehungskanten in die Roadmap eingefügt, die wie in Kapitel 4.1.2 beschrieben gewichtet werden.
   (c) Die neuen Translationskanten zwischen benachbarten Zellen werden erzeugt, sie erhalten die Start-Orientierung $\theta^- = \theta_i$ und vorläufig kein Gewicht.

6. Die Engstellen der neuen Zellen werden berechnet und, falls sie existieren, werden deren kritische Orientierungen in der sortierten Datenstruktur gespeichert.

7. Die Referenzen von den Eckpunkten auf die Zellen, deren seitlicher Stop sie sind, werden aktualisiert.

8. Falls es weitere kritische Orientierungen im Bereich $\theta < 2\pi$ gibt, so fahre fort mit 1.

Da $\theta = 2\pi$ gleich $\theta = 0$ entspricht, sind ab dieser Stelle alle relevanten kritischen Orientierungen berücksichtigt. Falls in der sortierten Datenstruktur noch kritische Orientierungen sind, so handelt es sich um Engstellen von Zellen mit dem bisherigen Orientierungsbereich $[\theta_j, —]$. Da $0 = 2\pi$ aber keine kritische Orientierung ist, gibt es dazu eine Zelle mit gleichem Rahmen und einem Orientierungsbereich $[0, \theta_i]$. Somit wurden die betroffenen Engstellen bereits ganz am Anfang bearbeitet.

Nun gilt es diese Zellen zu vereinen. Da es von Beiden $O(n)$ viele gibt, kann durch naives vergleichen in $O(n^2)$ Zeit festgestellt werden, welche Zellen den gleichen Rahmen haben. Beide werden zu einer Zelle mit $[\theta_j, \theta_i]$ vereinigt und die Gewichte der Translationskanten werden wegen des neuen Orientierungsbereichs neu berechnet. Die Rotationskanten können ohne Änderung übernommen werden, da diese für ein festes θ gelten.

Die Berechnung der Roadmap ist somit abgeschlossen, und mit dem Algorithmus 3.2 kann darin wieder nach einem Weg der maximalen Leiter gesucht werden. Dazu müssen zunächst einmal die Zellen bestimmt werden in denen die Start- und Zielplazierung liegen. Hierfür wird an dieser Stelle auch erstmal ein naives Verfahren verwendet, bei der jede Zelle überprüft wird, ob sie die entsprechende Plazierung enthält. Dabei wird für jede Zelle bestimmt, ob die Orientierung θ_s/θ_t der Plazierung s/t im Orientierungsbereich der Zelle liegt, außerdem muss der Strahl vom Start- oder Zielpunkt in Richtung θ_s/θ_t das obere Segment der Zelle schneiden. Wenn beide Bedingungen erfüllt sind, liegt die Plazierung innerhalb der Zelle. Die Distanz vom Start-/Zielpunkt zum Schnittpunkt ergibt die Maximale Leiter an dieser Position.

Für die Suche in der Roadmap kann dann der Algorithmus 3.2 unverändert übernommen werden.

4.2.3 Verifikation

Die oben beschriebenen Algorithmen sind größtenteils mit leichten Modifikation aus der Lösung für reine Translationen übernommen worden. Hier soll nur nochmals geklärt werden, wie und warum dies möglich ist.

Nach den Definitionen aus den Kapiteln 2 - 4 sind Regionen Plazierungen fester Orientierung mit gleichem Rahmen Ψ. Der Rahmen ist ein Set von Beschränkungen, bestehend aus oberem und unterem Segment und linkem und rechtem Stop. Eine Zelle ist die Vereinigung über alle Regionen unterschiedlicher Orientierung mit gleichem Rahmen Ψ. An den kritischen Orientierungen sind jedoch die seitlichen Stops nicht eindeutig, da mehrere Eckpunkte auf einer Linie liegen. Bei einer minimalen Drehung ändert sich der Rahmen, und ein Eckpunkt wird eindeutiger seitlicher Stop. Daher hat der Rahmen einer Zelle den Orientierungsbereich $[\theta^-, \theta^+]$ als zusätzliche Beschränkung.

Regionen sind somit Ebenen einer Zelle mit fester Orientierung und für eine nicht-kritische Orientierung können sie ihre Zellen repräsentieren. Erst bei einer kritischen Orientierung ändern sich wenige betroffene Regionen und somit deren Zellen. Zwischen zwei kritischen Orientierungen bleibt also die Struktur der Roadmap unverändert. Diese Eigenschaft macht sich das vorgestellte Berechnungsverfahren zu Nutze. Für eine nicht-kritische Startorientierung werden die Regionen berechnet, welche ihre Zellen repräsentieren. Anschließend werden die kritischen Orientierungen sukzessive z.B. Counter-Clockwise abgearbeitet und die dort neu entstehenden Zellen berechnet. Kritische Orientierungen sind die Orientierungen aller Sichtsegmente, denn in diesem Fall liegen beide Eckpunkte auf einer Linie. Zusätzlich sind die Engstellen der Zellen auch kritische Orientierung, da an dieser Stelle die Zelle geteilt wird.

Da die kritischen Orientierungen in sortierter Reihenfolge abgearbeitet werden, gelten vor jedem Schleifendurchlauf des Algorithmus 4.1 folgende Invarianten:

1. Die Roadmap enthält alle Knoten und Kanten für die Suche der größten Leiter über dem Intervall $[0, \theta_i]$. (θ_i = nächste kritische Orientierung)

2. Alle abgeschlossen Kanten sind gewichtet.

3. Alle offen Knoten und Kanten entsprechen der Struktur der Roadmap der Regionen mit den Orientierungen aus dem Intervall $[\theta_{i-1}, \theta_i]$.

Diese drei Invarianten sind zu Beginn des Algorithmus 4.1 alle erfüllt. Wie oben beschrieben ändert sich bis zum ersten kritischen Stop die Struktur der Roadmap nicht (3), und damit sind schon alle Knoten und Kanten für die Suche über $[0, \theta_1]$ vorhanden (1). Es wurden auch noch keine Knoten und Kanten abgeschlossen (2).

Bei der Ausführung des Algorithmus geschieht nun Folgendes: Die von der kritischen Orientierung betroffenen Knoten und deren Kanten werden abgeschlossen und gewichtet (2). Es werden an dieser Stelle neue Knoten erstellt, die untereinander, mit den Nachbarn und den Vorgängern durch Kanten verbunden sind. Drehungskanten werden direkt abgeschlossen und gewichtet gespeichert (2). Die von θ_i nicht betroffen Zellen und somit deren Knoten und Kanten sind auch bis mindesten θ_{i+1} gültig und werden nicht verändert. Die neuen Knoten und Kanten starten mit θ_i und nehmen für den Bereich $[\theta_i, —]$ den Platz der abgeschlossenen Vorgänger-Zellen ein. Nach dem Inkrementieren des Zählers i bilden somit die offenen Knoten die Struktur der Roadmap der Regionen mit den Orientierungen aus dem Intervall $[\theta_{i-1}, \theta_i]$ (3), und für alle Knoten und Kanten ist (1) erfüllt.

Nachdem der Algorithmus 4.1 alle relevanten kritischen Orientierungen abgearbeitet hat, enthält die Roadmap alle Knoten und Kanten für die Suche über dem Intervall $[0, 2\pi]$. Ein Übergang zwischen den beiden Orientierungen 0 und 2π ist jedoch noch nicht möglich. Die offenen Knoten und Kanten sind genau die Knoten und Kanten mit $[\theta_j, —]$, die mit ihrem Gegenstück mit dem Intervall $[0, \theta_i]$ vereinigt werden. Dabei werden die Knoten und Kanten abgeschlossen und die Kanten gewichtet.

Damit ist eine vollständige Roadmap für die Suche nach der längsten Leiter berechnet.

Da die grundlegende Struktur der Roadmap für den allgemeinen Fall mit der Roadmap für reine Translationen übereinstimmt, lässt sich Lemma 3.6 auf Zellen und damit der Algorithmus 3.2 zur Wegesuche eins zu eins auf den allgemeinen Fall übertragen.

4.3 Laufzeit von $O(n^2 \log n)$

Da für die Berechnung der Längsten Leiter bei Translation und Rotation Algorithmen des vorherigen Kapitels übernommen wurden, kann auch auf deren Laufzeitbetrachtungen aus Kapitel 3.3 zurückgegriffen werden. Dies trifft im Besonderen auf die Berechnung der Regionen für $\theta = 0$ und die Wegesuche in der Roadmap zu. Die Berechnung der gesamten Roadmap mit Algorithmus 4.1 wird dagegen etwas genauer untersucht. Alle drei Stufen haben eine maximale Laufzeit von $O(n^2 \log n)$ und damit lässt sich auch die längste Leiter mit dieser Laufzeit berechnen.

4.3.1 Berechnung der Regionen für $\theta = 0$

Für die Wiederverwendung des Algorithmus 3.1 muss die Umgebung zuerst um $\frac{\pi}{2}$ nach links ($-\frac{\pi}{2}$ nach rechts) gedreht werden. Dies geschieht durch Multiplikation der Matrix $\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$ mit jedem Eckpunkt der Umgebung. Da pro Eckpunkt konstant viel Zeit benötigt wird, lässt sich die Umgebung in linearer Zeit drehen.

$\mathbf{O(n)}$

Der Algorithmus 3.1 wird auf dieser Umgebung in leicht modifizierter Form ausgeführt. Roadmapknoten und -kanten wurden dort in konstanter Zeit erstellt. In der modifizierten Version werden Kanten nicht gewichtet und das zusätzliche Speichern des festen Orientierungsbereichs [0,—] pro Knoten erfolgt ebenfalls in O(1).

Jeder Eckpunkt kann maximal dreimal seitlicher Stop einer Region/Zelle sein. Wenn die Eckpunkte und deren Referenzen auf die Roadmapknoten, deren seitlicher Stop sie sind, zum Beispiel in einer Hash-Tabelle gespeichert werden, so ist der Zugriff ebenfalls in konstanter Zeit möglich. Somit ändert sich durch die Modifikationen die Laufzeit des Algorithmus nicht, und die Berechnung der Regionen für $\theta = 0$ erfolgt in Zeit:

$\mathbf{O(n \log n)}$

4.3.2 Berechnung der Roadmap mit Algorithmus 4.1

Bei der weiteren Berechnung der Roadmap sind die kritischen Orientierungen die Ereignisse, an denen neue Zellen entstehen und somit neue Knoten und Kanten in die Roadmap erzeugt werden müssen. Die kritischen Orientierungen entsprechen den Orientierungen der Sichtbarkeitssegmente innerhalb der Umgebung. Mit dem Topologischen Sweep von Edelsbrunner und Guibas [3] lassen sich diese in $O(n^2)$ berechnen. Da es $O(n^2)$ viele Sichtbarkeitskanten geben kann, und sich an jedem kritischen Sichtbarkeitssegment

nur drei Zellen ändern, können maximal $O(n^2)$ viele Zellen entstehen. Jede Zelle kann höchstens eine Engstelle habe, so dass die Anzahl dieser kritischen Orientierungen ebenfalls in $O(n^2)$ liegt.

Die kritischen Orientierungen werden z.B. alle in einem Heap gespeichert, dafür sind $O(n^2 \log n)$ Schritte nötig. Die Orientierungen der erst später berechneten Engstellen lassen sich in logarithmischer Zeit darin einfügen. Somit benötigt die Unterhaltung dieser Datenstruktur:

$$\mathbf{O(n^2 \log n)}$$

Algorithmus 4.1

Die Berechnungsschritte des Algorithmus haben alle eine maximale Laufzeit von $O(\log n)$, und da es $O(n^2)$ viele kritische Orientierungen gibt kann der Algorithmus nur quadratisch oft durchlaufen werden. Daraus ergibt sich eine Laufzeit von:

$$\mathbf{O(n^2 \log n)}$$

- Entnahme der kleinsten kritischen Orientierung aus dem Heap. $O(\log n)$
- Ermittlung der betroffenen Zellen über die Referenzen von den Eckpunkten auf die Zellen. $O(1)$
- Berechnung der neuen Zellen. Es sind maximal fünf Zellen betroffen, daher kann dieser Ausschnitt in konstanter Zeit berechnet werden. $O(1)$
- Knoten und Kanten, die sich geändert haben (max. 3) werden abgeschlossen und gewichtet. Der Zugriff ist über die Referenzen möglich, das Schreiben der End-Orientierung und die Berechnung der Kantengewichte ist in konstanter Zeit möglich. $O(1)$
- Für jede neue Zelle (max. 3) wird ein neuer Knoten mit zugehörigen Kanten in die Roadmap eingefügt. $O(1)$
- Die Engstellen der neuen Zellen werden in konstanter Zeit bestimmt und anschließend in den Heap eingefügt. $O(\log n)$
- Update der Referenzen von den Eckpunkten auf die Zelle dessen seitlicher Stop sie sind. $O(1)$

□

Die Vereinigung der Zellen mit gleichem Rahmen an der Nahtstelle $\theta = 0 = 2\pi$ lässt sich durch einfaches Vergleichen der Zellen durchführen. Da es jeweils $O(n)$ viele Zellen mit den Bereichen $[0, \theta_i]$ und $[\theta_j, —]$ gibt, würde dieses Vergleichen eine quadratische Laufzeit haben.

Mit Hilfe der Referenzen von den Eckpunkten auf die Zellen ließe sich die Vereinigung auch auf eine lineare Laufzeit reduzieren.

$\mathbf{O(n)}$

4.3.3 Bestimmung der Zellen der Start- und Zielplazierung

Bei der naiven Suche wird jede Zelle überprüft, ob sie die Start- oder Zielplazierung enthält. Dazu muss zunächst die Orientierung der Plazierung im Orientierungsbereich der Zelle liegen. Falls dies erfüllt ist kann durch einen Schnitttest vom oberen Segment mit dem Strahl der Plazierung festgestellt werden, ob die Plazierung in der Zelle liegt. Pro Zelle kann dieser Test in konstanter Zeit durchgeführt werden, daher ergibt sich eine Gesamtlaufzeit von:

$\mathbf{O(n^2)}$

4.3.4 Wegesuche in der Roadmap

Bereits in Kapitel 3.3.4 wurde festgestellt, dass der Algorithmus 3.2 eine Laufzeit von $O(n \log n)$ (n = Anzahl der Kanten in der Roadmap) hat.

Die anfängliche Roadmap bei der Start-Orientierung $\theta = 0$ hat $O(n)$ viele Kanten. Bei der Berechnung mit dem Algorithmus 4.1 entstehen bei jedem Durchlauf nur konstant viele neue Kanten. Es existieren somit maximal $O(n^2)$ viele Kanten. Daraus ergibt sich eine Laufzeit für die Wegesuche von:

$\mathbf{O(n^2 \log n)}$

Kapitel 5

Untere Schranke für die Berechnung der längsten Leiter

In Kapitel 4 wurde gezeigt, dass die längste Leiter, die sich in einer Umgebung von A nach B transportieren lässt, mit einer Laufzeit von $O(n^2 \log n)$ berechnet werden kann. Damit ist auch automatisch die Frage gelöst, ob sich eine Leiter fester Länge in einer Umgebung von A nach B bewegen lässt. Der Algorithmus von Leven und Sharir [11] (Kapitel 2.2) benötigt für diese Berechnung ebenfalls $O(n^2 \log n)$ viel Zeit, zusätzlich hat der Algorithmus aber auch den Weg vom Start zum Ziel als Ausgabe.

Für die Lösung des Bewegungsplanungsproblems für eine Leiter fester Länge, welches u.a. durch den in Kapitel 2.2 beschriebenen Algorithmus gelöst wird, konnten O'Dunlaing, Sharir und Yap [13] sowie Ke und O'Rourke [6, 12] eine untere Schranke von $\Omega(n^2)$ zeigen. Da beim *Längsten Leiter Problem* eine Ausgabe des Weges nicht erforderlich ist, lässt sich die untere Schranke nicht direkt auf dieses Problem übertragen. Im Folgenden soll jedoch gezeigt werden, dass auch für das *Längste Leiter Problem* eine untere Schranke von $\Omega(n^2)$ gilt.

Die Abbildung 5.1 zeigt eine Beispielumgebung in der $\Omega(n^2)$ viele unterschiedliche kritische Leiterlängen möglich sind. Bei jeder Leiterlänge verbinden sich zwei Zusammenhangskomponenten, die vorher nicht zusammenhängend waren. Das Beispiel zeigt $\Omega(n)$ viele Leiterlängen an einer Lücke. Auch wenn durch Spiegelung diese Leiterlängen an drei weiteren Positionen vorkommen, so gibt es dennoch $\Omega(n)$ viele Lücken, die wiederum jeweils $\Omega(n)$ viele zusätzliche Leiterlängen haben. Somit ergibt sich eine Gesamtzahl von $\Omega(n^2)$ vielen unterschiedlichen Leiterlängen.

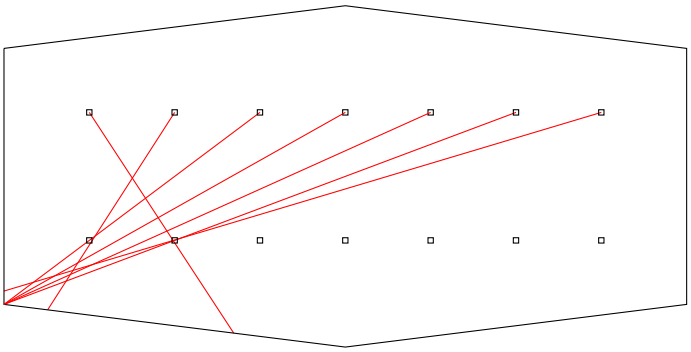


Abbildung 5.1: $\Omega(n^2)$ viele kritische Leiterlängen sind möglich

Falls eine dieser Engstellen auf dem Weg der längsten Leiter liegt, so ist diese kritische Leiterlänge auch relevant für die Bestimmung der längsten Leiter. Die Abbildung 2.11 von Ke und O'Rourke aus Kapitel 2 zeigt, dass ein Weg $\Omega(n^2)$ viele Bewegungsschritte haben kann, und dabei ebensoviele Engstellen überschritten werden. Damit können quadratisch viele kritische Leiterlängen für die Bestimmung der längsten Leiter relevant sein.

Somit gilt auch für die Berechnung der längsten Leiter eine untere Schranke von $\Omega(n^2)$.

Kapitel 6

Verbesserung der Laufzeit

Die vorherigen Kapitel haben gezeigt, dass für die Berechnung der längsten Leiter eine untere Schranke von $\Omega(n^2)$ gilt. Der in dieser Arbeit vorgestellte Algorithmus benötigt für die Berechnung $O(n^2 \log n)$ viel Zeit. Es stellt sich nun die Frage, ob sich diese kleine Lücke auch noch schließen lässt. Hier soll gezeigt werden, dass sich die Laufzeit für mehrere Anfragen auf der gleichen Umgebung verbessern lässt. Anschließend werden weitere Ideen zur Verbesserung der Laufzeit aufgezeigt.

6.1 Verbesserung der Laufzeit für mehrere Anfragen auf der gleichen Umgebung

Das in dieser Diplomarbeit entwickelte Verfahren zur Berechnung der längsten Leiter hat eine Laufzeit von $O(n^2 \log n)$. Grob betrachtet besteht es aus zwei Phasen, welche jeweils beide diese Laufzeit haben. In der ersten Phase wird schrittweise eine Roadmap berechnet, welche den kontinuierlichen Freespace durch eine Roadmap diskretisiert. In der zweiten Phase werden die Roadmapknoten der Start- und Zielplazierung bestimmt und ein maximaler Weg in der Roadmap gesucht.

Bei mehreren Anfragen auf der gleichen Umgebung ändert sich an der Berechnung der ersten Phase nichts, es wird immer die gleiche Roadmap berechnet. Es würde also ausreichen diesen Schritt nur einmal auszuführen. Phase zwei, die Suche innerhalb der Roadmap, muss allerdings jedesmal wieder durchgeführt werden. Eine Verbesserung dieser Laufzeit würde auch die Laufzeit für mehrere Anfragen verbessern.

Die in der ersten Phase berechnete Roadmap kann quadratisch viele Knoten und Kanten enthalten. Da bei der Suche nach der längsten Leiter immer nur die größten begehbaren Kanten als nächstes bearbeitet werden, muss während der Suche immer eine sortierte Datenstruktur unterhalten werden. Allein diese erzwingt bereits eine Laufzeit von $O(n^2 \log n)$. Eine verbesserte Lösung muss also diese sortierte Datenstruktur vermeiden.

Die Lösung dafür liegt in einem Maximum-Spanning-Tree. Mit Hilfe eines modifizierten Algorithmus von Kruskal lässt sich am Ende der ersten Phase in $O(n^2 \log n)$ aus dem Roadmap-Graphen ein Roadmap-Baum erstellen. Dieser maximale Spannbaum enthält genau die Kanten, die einen Zusammenhang aller Roadmapknoten mittels maximaler Kanten gewährleisten. Alle Kanten, die nicht im Roadmap-Baum enthalten sind, können auch nicht eine längste Leiter repräsentieren, da es längere Leitern gibt, die sich über einen anderen Wege zwischen den beiden Knoten bewegen lassen.

Der so ermittelte Roadmap-Baum kann allerdings weiterhin quadratisch viele Kanten enthalten, da er auch quadratisch viele Knoten enthalten kann. Die Wegesuche in dem Roadmap-Baum kann jetzt aber mit einer einfachen Tiefensuche durchgeführt werden. Diese lässt sich linear in der Anzahl der Kanten durchführen.

Für die Bestimmung des Start- und Zielknotens wurde in der bisherigen Lösung bereits eine Laufzeit von $O(n^2)$ benötigt. Da die Tiefensuche die gleiche Laufzeit hat, ergibt sich für jede Anfrage auf einer bereits berechneten Roadmap (Roadmap-Baum) eine Laufzeit von:

$$\mathbf{O(n^2)}$$

In einem weiteren Schritt könnte man sich überlegen, ob es mit Hilfe von geeigneten Datenstrukturen möglich ist, die Zellen der Start- und Zielplazierung schneller zu lokalisieren. Da der maximale Roadmap-Spannbaum allerdings nicht notwendigerweise ausgeglichen ist, ist es eher unwahrscheinlich, dass sich für jeden Fall die Suche in der Roadmap beschleunigen lässt. Außerdem können wegen der unteren Schranke quadratisch viele Engstellen für die Bestimmung der längsten Leiter relevant sein. Somit lässt sich die Laufzeit für eine Anfrage nicht weiter reduzieren.

6.2 Weitere Verbesserungsmöglichkeiten

Für Anfragen auf einer Roadmap lässt sich die Laufzeit des Algorithmus also auf $O(n^2)$ pro Anfrage reduzieren. Ist eine solche Verbesserung auch für die Berechnung der Roadmap möglich?

In ihrem Paper zu unteren Schranken [6] stellen Ke und O'Rourke die Vermutung auf, dass sich die Laufzeit des Algorithmus von Leven und Sharir zur Bewegungsplanung einer Leiter fester Länge von $O(n^2 \log n)$ auf $O(n^2)$ reduzieren lässt. Da der Algorithmus zur Berechnung der Roadmap für das *Längste Leiter Problem* auf eine ähnliche Art und Weise aufgebaut ist, kann man solche Verbesserungen möglicherweise auch hierauf übertragen.

Auch wenn ich keine Veröffentlichung gefunden habe, die zeigt, wie eine solche Verbesserung der Laufzeit möglich ist, so möchte ich hier aber trotzdem eine Idee vorschlagen, mit deren Hilfe die Laufzeitverbesserung möglich wäre.

Zur Berechnung der Roadmap werden an den kritischen Orientierungen die betroffenen Roadmap-Knoten abgeschlossen und neue Knoten und Kanten hinzugefügt. Damit dies in der richtigen Reihenfolge geschieht, müssen die kritischen Orientierungen aufsteigend sortiert sein. Da es quadratisch viele kritische Orientierungen geben kann, beträgt die Laufzeit für das Sortieren $\Theta(n^2 \log n)$.

Bei der Berechnung aller kritischen Orientierungen mittels Topologischen Sweep von Edelsbrunner und Guibas [3] werden diese in $O(n^2)$ berechnet. Die Ausgabe der Sichtbarkeitssegmente erfolgt hierbei zwar nicht streng nach Orientierung sortiert, aber die Verwendung der oberen und unteren Horizontbäume sorgt dafür, dass die Sichtbarkeitssegmente in einer gewissen Reihenfolge ausgegeben werden. Man könnte also jetzt untersuchen, ob diese Ordnung bereits für die Updates der Roadmap ausreichend ist. Wäre dies der Fall, so könnte die Sortierung der kritischen Orientierungen der Sichtbarkeitssegmente entfallen.

Übrig bleiben dann lediglich die kritischen Orientierungen der Engstellen der Zellen. Da an diesen Stellen jedoch keine große Neuberechnung erforderlich ist, könnte man die Zellen am Ende der Berechnung out-of-order teilen. Falls die Ordnung des Topologischen Sweeps ausreicht, könnte somit die Roadmap im $O(n^2)$ berechnet werden.

Um aber die schnellere Suche in der Roadmap durchführen zu können, müßte wiederum der Algorithmus von Kruskal ausgeführt werden. Dieser hat jedoch eine Laufzeit von $O(n^2 \log n)$.

Wenn sich der Maximum-Spanning-Tree auf anderem Wege in quadratischer Laufzeit berechnen ließe, dann hätte das Verfahren die optimale Laufzeit von $\Theta(n^2)$ erreicht.

Kapitel 7

Ausblick

In dieser Diplomarbeit wurde ein Algorithmus entworfen, der die maximale Leiter, die sich in einer polygonalen 2-dimensionalen Umgebung zwischen zwei Punkten bewegen lässt, in $O(n^2 \log n)$ berechnet. Mit diesem Verfahren könnte man z.B. in idealisierter Form den längsten Schwertransport berechnen, der sich von Fabrik A zu Fabrik B bewegen lässt.

Bereits in dieser Arbeit wurden Ansätze aufgezeigt, wie man die Laufzeit des Algorithmus hin zum Optimum verbessern kann. In weiteren Arbeiten könnte man neben Laufzeitverbesserungen auch die Problemstellung um weitere Aspekte erweitern:

- Man könnte die längste Leiter in einer 3-d Umgebung berechnen. Für eine Leiter fester Länge haben Ke und O'Rourke [6] bereits $\Omega(n^4)$ und $O(n^6 \log n)$ als Schranken ermittelt.

- Man könnte untersuchen, welches die größte skalierte Kopie eines Roboterpolygon ist, die sich durch eine 2-d Umgebung bewegen lässt.

- Als Fernziel steht dann die Berechnung von maximalen 3-d Objekten, die in einer 3-d Umgebung bewegt werden sollen. Mit diesem Verfahren könnte man z.B. schon in der Planungsphase berechnen, wo sich in einem geplanten Industriegebäude Engstellen für das zukünftige Wachstum von Containern befinden.

Literaturverzeichnis

[1] N. Blum. Theoretische Informatik - Eine anwendungsorientierte Einführung, 2. Auflage. *Oldenbourg Verlag*, München, 2001.

[2] H. T. Croft, K. J. Falconer and R. K. Guy. Unsolved Problems of Geometry. *Springer Verlag*, Bonn, 1991.

[3] H. Edelsbrunner and L. J. Guibas. Topologically sweeping an arrangement. *Proc. of 18th ACM Sympos. Theory Comput.*, p. 389–403, 1986.

[4] J. L. Gerver. On moving a sofa around a corner. *Geometriae Dedicata*, 42:267–283, 1992.

[5] A. Kaufman and R. Rempel. The Piano Mover's Problem: Variation on a Theme. *http://www.bethelks.edu/academics/math/piano/amykaufman2003.html*, 2003.

[6] Y. Ke und J. O'Rourke. Moving a Ladder in Three Dimensions: Upper and Lower Bounds. *Symposium on Computational Geometry*, 3:136–146, 1987.

[7] K. Kedem und M. Sharir. An Efficient Motion-Planning Algorithm for a Convex Polygonal Object in Two-Dimensional Space. *Discrete & Comuptational Geometry*, 5:43-75, 1990.

[8] K. Kedem, M. Sharir and S.Toledo. On Critical Orientations in the Kedem-Sharir Motion Planning Algorithm. *Discrete & Comuptational Geometry*, 17:227-239, 1997.

[9] R. Klein. Algorithmische Geometrie. *Addison-Wesley*, Bonn, 1997.

[10] R. Klein, T. Kamphans, E. Langetepe. Bewegungsplanung für Roboter. *Skript zur Vorlesung*, Universität Bonn, 2004.

[11] D. Leven und M. Sharir. An Efficient and Simple Motion Planing Algorithm for a Ladder Amidst Polygonal Barriers. *Journal of Algorithms*, 8:192-215, 1987.

[12] J. O'Rourke. A lower bound on moving a ladder. *Technical Report 85/20*, Johns Hopkins University, Baltimore, 1985.

[13] C. O'Dunlaing, M. Sharir und C. Yap Generalized Voronoi Diagrams for a Ladder: II. Efficient Construction of the Diagram. *Algorithmica*, 2:27-59, 1987.

[14] J. T. Schwarz und M. Sharir. On the Piano Movers' Problem: I. The case of a tow-dimensional rigid polygonal body moving admidst polygonal barriers. *Comm. Pure Appl. Math.*, 36:345–398, 1983.

[15] J. T. Schwarz und M. Sharir. On the Piano Movers' Problem: II. General techniques for calculating topological properties of real algebraic manifolds. *Advances Appl. Math.*, 4:298–351, 1983.

[16] J. T. Schwarz und M. Sharir. On the Piano Movers' problem: III. Coordinating the motion of several independent bodies: The special case of circular bodies moving amidst polygonal barriers. *Robotics Research*, 2:46–75, 1983.

[17] J. T. Schwarz und M. Sharir. On the Piano Movers' problem: V. The case of a rod moving in 3-D space amidst polyhedral obstacles. *Comm. Pure Appl. Math.*, 37:815–848, 1984.

[18] M. Sharir und E. Ariel-Sheffi. On the Piano Movers' problem: IV. Various decomposable two-dimensional motion planning problems. *Comm. Pure Appl. Math.*, 37:479–493, 1984.

[19] G. P. Vennebush. Move that Sofa! *Mathematics Teacher*, 95:92-97, 2002.

Erklärung

Hiermit erkläre ich, gemäß § 19 Abs. 7 der DPO vom 15. August 1998, dass ich meine Diplomarbeit selbständig durchgeführt und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe.

Leubsdorf, den 26. Oktober 2005 Markus Rings