



---

Elmar Langetepe

# Methoden der Offline Bewegungsplanung

BA-INF 124

---

---

Wintersemester 2016/2017  
Dozent: Elmar Langetepe  
Skript: Tom Kamphans/Elmar Langetepe



## Einleitung

Die Vorstellung von autonomen Maschinen, die den Menschen schwierige und gefährliche Arbeiten abnehmen, hat immer wieder die Phantasie vieler Science-Fiction Autoren<sup>1</sup> beflügelt und viele Wissenschaftler unterschiedlicher Disziplinen beschäftigt. Das Feld “Robotics” involviert daher verschiedenste Bereiche, darunter:

- Elektrotechnik, Elektronik, Mechanik
- Kontroll- und Regelungstechnik
- Systemtheorie
- Softwareengineering
- Künstliche Intelligenz (Neuronale Netze, Fuzzy Logic)
- Algorithmik

Im Rahmen dieser Vorlesung werden wir uns auf den letzten Punkt konzentrieren und Probleme aus der Algorithmischen Geometrie behandeln, die im Zusammenhang mit der Steuerung von Robotern interessant sind.

Wir benötigen zur Behandlung dieser Fragestellungen einige Begriffe und Verfahren der Algorithmischen Geometrie, die hier nicht in aller Ausführlichkeit vorgestellt werden können, sondern nur kurz angerissen oder gar nur als “black-box” verwendet werden. Für ein tieferes Verständnis dieser Themen sei auf die Vorlesung Algorithmische Geometrie oder auf die einschlägige Literatur in diesem Bereich verwiesen.

Die diskutierten Themen sind Varianten einer Grundaufgabe: Gegeben ist ein Robotersystem mit  $k$  Freiheitsgraden, das sich in einer Umgebung  $U$  mit oder ohne (stationären) Hindernissen bewegt, ein Startpunkt  $s$  und ein Zielpunkt  $t$ . Stelle fest, ob es eine kollisionsfreie und kostengünstige Bewegung von  $s$  nach  $t$  gibt und falls ja, bestimme eine solche. Wichtige Kriterien für die vorgestellten Lösungen sind dabei:

- Korrektheit, d.h. es soll genau dann eine kollisionsfreie Bewegung gefunden werden, wenn eine solche existiert.
- Effizienz der Berechnung, denn natürlich soll die Bewegung schnell berechnet werden, der Roboter soll nicht vor lauter Rechenaufwand auf der Stelle stehen bleiben.
- Qualität der Lösung, d.h. die gefundene Bahn ist auch günstig bzgl. der festgelegten Kostenmaße.

Zunächst wollen wir die Betrachtung von Kollisionsproblemen zurückstellen, uns auf punktförmige Roboter beschränken und die Länge der vom Roboter zurückgelegten Bahnen als Qualitätskriterium betrachten.

## Literatur

Begleitend zu diesem Skript sei folgende Literatur empfohlen:

- Mark de Berg, Marc van Kreveld, Mark Overmars und Otfried Schwarzkopf:  
*Computational Geometry: Algorithms and Applications.*  
 Springer-Verlag, Berlin 1997, [dBvKOS97]

---

<sup>1</sup>Der Begriff “Roboter” wurde von dem tschechischen Autor Karel Čapek (1890-1938) geprägt. [ČČ61]

- Rolf Klein:  
*Algorithmische Geometrie.*  
Addison–Welsey, Bonn 1997, [Kle97]
- Micha Sharir und Pankaj K. Agarwal:  
*Davenport-Schinzel Sequences and Their Geometric Applications.*  
Cambridge University Press, New York 1995, [SA95]
- Micha Sharir:  
*Algorithmic Motion Planning.*  
Kapitel 40 in: Jacob E. Goodman und Joseph O'Rourke: *Handbook of Discrete and Computational Geometry.*  
CRC Press LLC, Boca Raton, FL, 1997, [Sha97]
- Joseph S. B. Mitchell:  
*Geometric Shortest Paths and Network Optimization.*  
Kapitel 15 in: Jörg-Rüdiger Sack and Jorge Urrutia: *Handbook of Computational Geometry.*  
Elsevier Science Publishers B.V. North-Holland, 2000, [Mit00]
- Jean-Claude Latombe:  
*Robot Motion Planning.*  
Kluwer Academic Publishers, Boston, 1991, [Lat91]
- Jakob T. Schwartz und Chee-Keen Yap (Editor):  
*Advances in Robotics.*  
Lawrence Earlbaum, 1987, [Yap87]
- Sowie viele in Journalen und Konferenzbänden erschienene Arbeiten und auch Dissertationen wie:  
A. Frank van der Stappen:  
*Motion Planning amidst Fat Obstacles.*  
Dissertation, Dept. Comput. Sci., Utrecht Univ., NL, 1994, [vdS94]

An dieser Stelle sei auch auf das Geometrie–Labor

<http://www.geometrylab.de/>

mit Java–Applets aus dem Bereich Algorithmische Geometrie und Bewegungsplanung für Roboter hingewiesen.

## Danksagungen

Sehr herzlich danken wir Annette Ebberts–Baumann, Dorte Lübbert, Andrea Eubeler, Stefan Kamphans und allen Teilnehmern der Vorlesung für viele Korrekturhinweise und Verbesserungsvorschläge.

## Korrekturhinweise

Trotz aller Sorgfalt ist dieses Skript bestimmt nicht fehlerfrei. Bitte teilt uns Fehler aller Art oder unverständlich aufgeschriebenes per Mail an [Elmar.Langetepe@cs.uni-bonn.de](mailto:Elmar.Langetepe@cs.uni-bonn.de) mit.

# Kapitel 1

## Kürzeste Pfade

Die Untersuchung kürzester Wege beschäftigt die Wissenschaft bereits seit dem Altertum. In der Mathematik befaßt man sich in der Theorie der metrischen Räume und in der Differentialgeometrie damit. So zeigten bereits Joh. Bernoulli (1698) und L. Euler (1728), daß für Kurven  $C$  von  $a$  nach  $b$  auf einer glatten Fläche  $S$  gilt:  $C$  ist Kürzeste  $\implies C$  ist Geodätische, d. h.  $\forall p \in C : \text{Schmiegeebene}(C, p) \perp \text{Tangentialebene}(S, p)$ .

Dabei ist die Schmiegeebene die Ebene, an die  $C$  sich lokal anschmiegt,<sup>1</sup> und die Tangentialebene die in  $p$  tangential zu  $S$  liegende Ebene. Geodätische Kurven sind nur lokal nicht verkürzbar, im Gegensatz zu kürzesten Wegen, die auch global unverkürzbar sind. Anschaulich kann man sich geodätische Kurven als straff gespannte Gummibänder vorstellen. Die Umkehrung obigen Satzes gilt nicht: In Abbildung 1.1 ist die Helix von  $a'$  nach  $b'$  zwar eine Geodätische, aber keine Kürzeste.

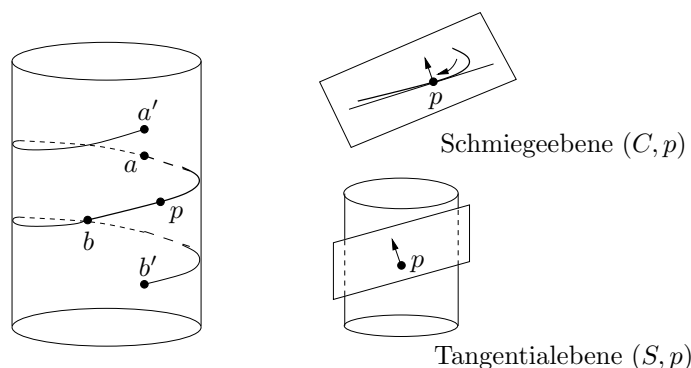


Abbildung 1.1: Kürzeste von  $a$  nach  $b$ , Geodätische von  $a'$  nach  $b'$ .

Wir interessieren uns hier für diskrete Umgebungen.

### 1.1 Kürzeste Pfade in der Ebene mit polygonalen Hindernissen

Die Umgebung, in der sich der Roboter bewegt, sei durch  $h$  Hindernisse in Form von Polygonen  $P_i$  mit insgesamt  $n$  Ecken gegeben. Dabei sind auch Polygone mit  $\overset{\circ}{P}_i = \emptyset$  erlaubt,<sup>2</sup> also zu einem Linienzug degenerierte Polygone. Weiterhin seien ein Startpunkt  $s$  und Zielpunkt  $t$  gegeben, die nicht im Inneren eines Polygons liegen dürfen; es muß gelten  $s, t \notin \bigcup \overset{\circ}{P}_i$ . Die Aufgabe ist nun, einen Pfad  $\pi$  minimaler Länge von  $s$  nach  $t$  zu finden, der keine Hinderniskante kreuzt.

<sup>1</sup>Genauer: die Grenzlage einer durch drei benachbarte Kurvenpunkte  $p'$ ,  $p$ ,  $p''$  gegebene Ebene für die Grenzwerte  $p' \rightarrow p$  und  $p'' \rightarrow p$  [Stö99].

<sup>2</sup> $\overset{\circ}{P}$  bezeichnet das Innere eines Polygons, also alle Punkte aus  $P$ , deren  $\epsilon$ -Umgebung vollständig in  $P$  liegt.

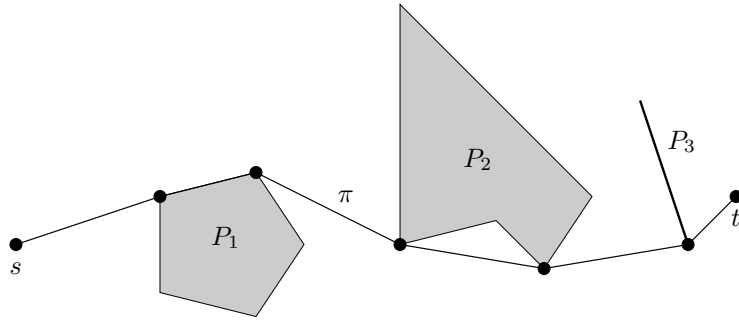


Abbildung 1.2: Umgebung mit polygonalen Hindernissen.

Der kürzeste Pfad ist nicht immer eindeutig, je nach Umgebung kann es exponentiell in  $h$  viele kürzeste Wege geben, Abbildung 1.3 zeigt eine Umgebung mit  $h = 4m + 1$  Hindernissen (Liniensegmente) und  $2^{m+1}$  kürzesten Wegen (Akman, 1987 [Akm87b]).

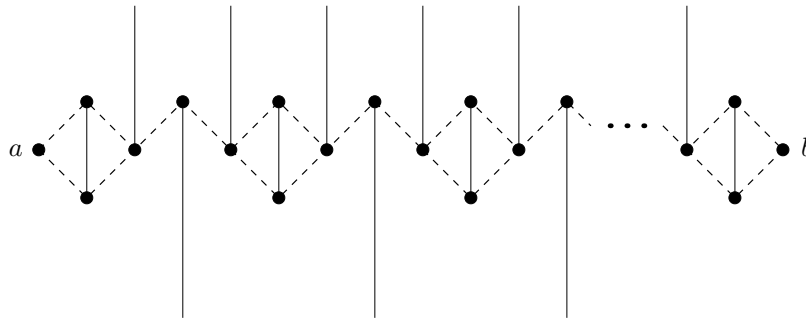
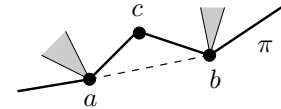


Abbildung 1.3: Es kann exponentiell viele kürzeste Wege geben.

Kürzeste Pfade haben die Gestalt einer polygonalen Kette, die nur an konvexen Polygonecken<sup>3</sup> abknicken kann, da sich die Kette andernfalls verkürzen ließe. Dies wird klar, wenn man sich den kürzesten Pfad als ein Gummiband vorstellt, das von  $s$  nach  $t$  gespannt ist: die Knicke im Gummiband müssen sich an einer Polygonecke abstützen, ohne eine Polygonecke würde sich das Gummiband strammziehen. In nebenstehender Abbildung ist  $\pi$  also kein kürzester Pfad, da sich der Knick in Punkt  $c$  nicht auf eine Polygonecke abstützt. Der kürzeste Pfad zwischen  $a$  und  $b$  wäre die gestrichelte Linie.



Genauer bestehen kürzeste Pfade aus einer Startkante von  $s$  aus, einer Zielkante zu  $t$  hin und ansonsten nur aus Kanten, die *gegenseitig sichtbare*, konvexe Hindernisecken verbinden. Diese Beobachtung führt zu einem ersten Ansatz zur Lösung des Problems, bei der wir auf eine aus der Algorithmischen Geometrie bekannte Struktur zurückgreifen können:

**Definition 1.1** Sei  $\mathcal{L}$  eine Menge sich nicht kreuzender Liniensegmente in der Ebene. Dann ist der **Sichtbarkeitsgraph** von  $\mathcal{L}$  ein Graph  $\text{VisG}(\mathcal{L}) = (V, E)$  mit

$$V = \{ \text{alle Endpunkte der Liniensegmente in } \mathcal{L} \}$$

$$E = \{ (p, q) \mid p, q \in V, \overline{pq} \text{ kreuzt kein Liniensegment aus } \mathcal{L} \}.$$

Der Sichtbarkeitsgraph einer Menge von Polygonen  $P_i$  entsteht aus  $\text{VisG}(\mathcal{L})$  mit  $\mathcal{L} = \{ \text{Kanten der } P_i \}$  durch Entfernen aller im Inneren eines Polygons liegenden Sichtbarkeitskanten.

Der Sichtbarkeitsgraph von  $n$  Liniensegmenten hat die Komplexität  $O(n^2)$ . Abbildung 1.4(i) zeigt ein Beispiel eines Arrangements von  $n$  Liniensegmenten, dessen Sichtbarkeitsgraph  $\Omega(n^2)$  viele Kanten enthält. Jedoch kann der Sichtbarkeitsgraph auch deutlich weniger Kanten enthalten,

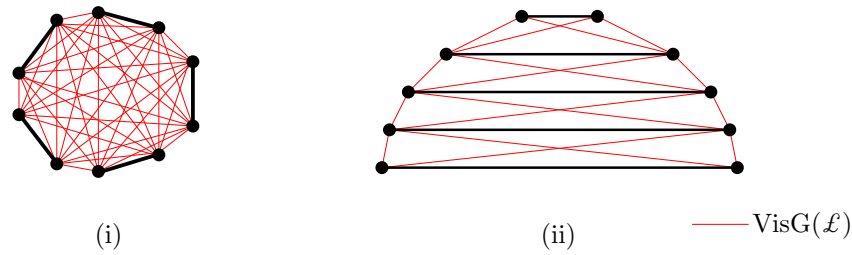


Abbildung 1.4: Der Sichtbarkeitsgraph kann (i)  $\Omega(n^2)$  viele Kanten enthalten, aber auch (ii) nur  $O(n)$  viele.

wie das Beispiel in Abbildung 1.4(ii) zeigt.

---

**Algorithmus 1.1** Berechnung des kürzesten Weges mit Hilfe des Sichtbarkeitsgraphen

---

1. Berechne den Sichtbarkeitsgraph  $\text{VisG}(P)$ .
  2. Füge alle Kanten  $(s, v)$  mit  $v \in V$  und  $s$  sieht  $v$  ein:  
 $\forall v \in V$  : Teste, ob das Segment  $\overline{sv}$  von einer Randkante eines der Hindernisse geschnitten wird. Ist dies nicht der Fall, dann füge das Segment  $\overline{sv}$  zum Sichtbarkeitsgraphen hinzu.
  3. Füge alle Kanten  $(v, t)$  mit  $v \in V$  und  $t$  sieht  $v$  ein (analog).
  4. Berechne mit dem Algo. von Dijkstra in dem modifizierten Graph mit Distanzen als Kantengewichten alle kürzesten Wege von  $s$ .
- 

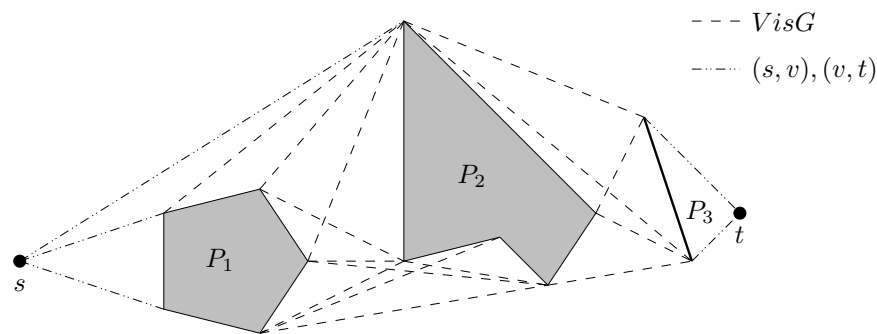
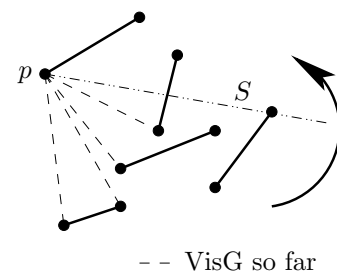


Abbildung 1.5: VisG und Kanten von  $s$  und zu  $t$ .

Damit lässt sich ein kürzester Weg wie in Algorithmus 1.1 beschrieben berechnen. Abbildung 1.5 zeigt das Ergebnis der Schritte 1 bis 3 auf der Umgebung aus Abbildung 1.2. Die zusätzlichen Kanten lassen sich offensichtlich in Zeit  $O(n^2)$  einfügen, die Komplexität der anderen beiden Schritte bedarf einer genaueren Analyse. Beginnen wir mit der Berechnung des Sichtbarkeitsgraphen: Dieser lässt sich naiv in Zeit  $O(n^3)$  berechnen, indem für jedes Paar  $(p, q)$  von Ecken das Segment  $\overline{pq}$  auf Kreuzung mit den  $n$  Polygonrandsegmenten getestet wird. Ein besseres Laufzeitverhalten hat der in Algorithmus 1.2 skizzierte radiale Sweep, der für ein festes  $p$  Zeit  $O(n \log n)$  benötigt, insgesamt also  $O(n^2 \log n)$ .

Dies kann noch verbessert werden, indem nicht jeder Punkt für sich einen radialen Sweep durchführt, sondern die Bearbeitung so synchronisiert wird, daß Sichtinformationen weitergereicht




---

<sup>3</sup>Polygonecke, deren Außenwinkel  $> \pi$  ist.

---

**Algorithmus 1.2** Sweep zur Berechnung des Sichtbarkeitsgraphen
 

---

Für jeden Endpunkt  $p$ :

- Sortiere alle Endpunkte rechts von  $p$  nach Winkeln von  $p$ .
  - Drehe einen Strahl  $S$  von  $p$  gegen den Uhrzeigersinn von Süd nach Nord:
    - Führe in balanciertem Baum sortiert nach Abstand von  $p$  Buch darüber, welche Segmente von  $S$  gerade geschnitten werden.
    - Sei  $q$  der Schnittpunkt von  $S$  mit dem zu  $p$  nächsten Liniensegment auf  $S$ . Falls  $q \in V$ , berichte Sichtbarkeitskante  $\overline{pq}$ .
- 

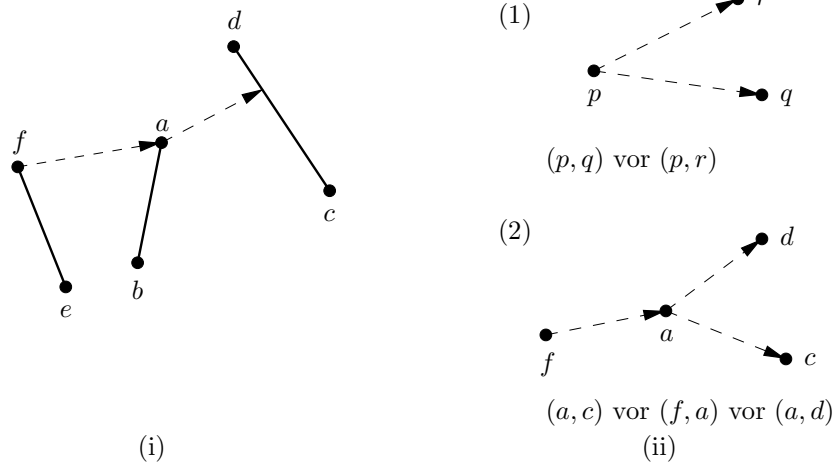


Abbildung 1.6: (i) Sichtbarkeitsinformationen können von  $a$  an  $f$  weitergereicht werden, (ii) Typen von Bedingungen an die Bearbeitungsreihenfolge.

werden können. Wenn z. B. in Abbildung 1.6(i) der Punkt  $f$  beim Sweep den Punkt  $a$  entdeckt, kann  $a$  die Information " $\overline{cd}$  ist sichtbar" an  $f$  weiterreichen.

Die Wahl der Bearbeitungsreihenfolge  $R$  hängt dabei von der Steigung  $\text{slope}(p, q)$  des Segmentes zwischen  $p$  und  $q$  ab. Wir müssen dabei zwei Typen von Bedingungen erfüllen, siehe Abbildung 1.6(ii):

- (1)  $\text{slope}(p, q) < \text{slope}(p, r) \Rightarrow (p, q)$  vor  $(p, r)$  bearbeiten
- (2)  $\text{slope}(a, c) < \text{slope}(f, a) < \text{slope}(a, d) \Rightarrow (a, c)$  vor  $(f, a)$  vor  $(a, d)$

Man beachte, dass in Bedingung (2) weder die erste noch die zweite Ungleichung aus Bedingung (1) folgt.

Die Wahl der Bearbeitungsreihenfolge werden wir später betrachten. Nehmen wir zunächst an, wir hätten eine solche Bearbeitungsreihenfolge  $R$  bereits festgelegt. Nun wird zur Initialisierung zu jedem Punkt  $p$  ein Zeiger  $\wp$  auf das erste unterhalb von  $p$  liegende Segment berechnet; also das Liniensegment, das ein Strahl von  $p$  in negativer  $Y$ -Richtung als erstes treffen würde. Liegt unter  $p$  kein Liniensegment, wird der Zeiger auf NIL gesetzt.

Bei der Bearbeitung der Paare  $(p_i, q_i)$  aus der Liste  $R$  soll folgende Invariante erhalten bleiben:

Sei  $q$  der zuletzt bearbeitete Partner von  $p$ . Dann gibt es einen Zeiger  $\wp$  von  $p$  auf das in Sweeprichtung kurz hinter  $q$  sichtbare Segment  $\sigma$ .

Nach der Initialisierung ist die Invariante erfüllt,  $\wp$  zeigt auf das von  $p$  in Richtung Süden liegende Segment. Vor der Bearbeitung eines Paares  $(p, r)$  sei  $q$  der zuletzt bearbeitete Partner



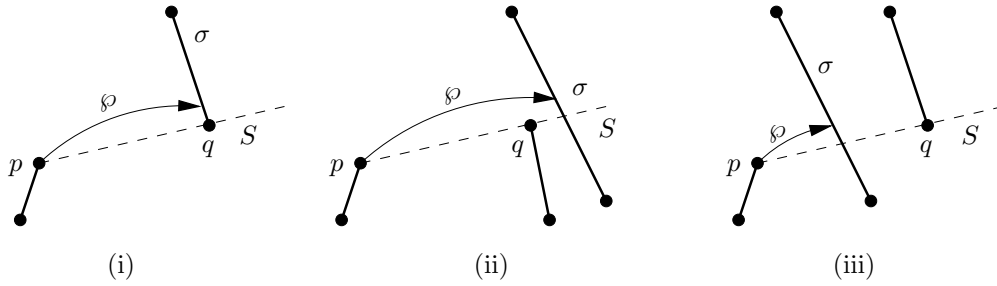


Abbildung 1.7: Es gibt drei mögliche Ausgangssituationen eines Punktes  $p$  mit zuletzt bearbeitetem Partner  $q$  und sichtbarem Segment  $\sigma$ .

von  $p$ , und der Zeiger von  $p$  zeige auf  $\sigma$ . Es gibt drei mögliche Ausgangssituationen von  $p$ ,  $q$  und  $\sigma$ , siehe Abbildung 1.7:

- (i)  $q$  ist von  $p$  aus sichtbar und Anfangspunkt eines Segments  $\sigma$ .
- (ii)  $q$  ist von  $p$  aus sichtbar und Endpunkt eines Segments.  $\sigma$  ist das auf dem Strahl  $S$  von  $p$  durch  $q$  nächste Segment hinter  $q$ .
- (iii)  $q$  ist von  $p$  aus nicht sichtbar.  $\sigma$  ist das zu  $p$  nächste Segment auf dem Strahl  $S$  von  $p$  durch  $q$ .

Bei der Bearbeitung des Paares  $(p, r)$  können nun folgende Fälle eintreten, siehe Abbildung 1.8:

1.  $r$  liegt hinter  $\sigma$ . In allen drei Ausgangssituationen zeigt  $\wp$  nach wie vor auf  $\sigma$ .
2.  $r$  liegt vor  $\sigma$ . Dann ist  $r$  Anfangspunkt eines Segments  $\tau$  und  $\wp$  zeigt nun auf  $\tau$ .
3.  $r$  ist Endpunkt von  $\sigma$ . Hier muss die Sichtbarkeitsinformation für  $r$  benutzt werden. Je nach Ausgangssituation gibt es in diesem Fall folgende drei Möglichkeiten, die prinzipiell analog behandelt werden können:
  - (i)  $\wp$  wird auf das erste Segment auf dem Strahl zwischen  $p$  und  $r$  hinter  $r$  gesetzt. Dies entspricht der Ausgangssituation (ii).
  - (ii) In diesem Fall kann der Winkelbereich  $W$  zwischen den Strahlen von  $p$  durch  $q$  und  $r$  keinen Punkt enthalten, ansonsten wäre  $r$  nicht der nächste Partner von  $p$ . Außerdem muß  $r$  von  $p$  aus sichtbar sein. Andernfalls müßte sich ein Endpunkt in  $W$  befinden, oder  $\sigma$  wäre nicht das nach  $q$  sichtbare Segment.  
Falls ein Segment  $\tau$  existiert auf das  $r$  zeigt, ist es durch die Bearbeitung des letzten Paares  $(r, v)$  bezüglich  $r$  entstanden, dass vor  $(p, r)$  dran war. Der Punkt  $v$  liegt unterhalb von  $W$  wegen Teil 2) der Bearbeitungsreihenfolge. Das Segment  $\tau = \overline{uw}$  muss vollständig den obigen Winkelbereich  $W$  durchqueren und ist somit sichtbar von  $p$  aus. Der untere Endpunkt  $u$  kann nicht oberhalb vom Strahl durch  $r$  und  $v$  liegen ( $u$  könnte allerdings mit  $v$  identisch sein), da  $\tau$  ja die Sichtbarkeitsbedingung erfüllen muss. Der obere Endpunkt  $w$  kann nicht zwischen dem Strahl durch  $r$  und  $v$  und dem Bereich  $W$  liegen, da dann  $(r, w)$  noch nach  $(r, v)$  und vor  $(p, r)$  betrachtet worden wäre. Der Endpunkt  $w$  kann wie oben bereits erwähnt nicht im Bereich  $W$  liegen. Folglich sieht  $p$  einen Punkt  $r'$  auf  $\tau$  und  $\wp$  wird also auf  $\tau$  gesetzt.
  - (iii) Analog zu (ii).

Damit können wir den Sichtbarkeitsgraphen nach der Berechnung der Bearbeitungsreihenfolge  $R$  in Zeit  $O(n^2)$  bestimmen. Um  $R$  festzulegen, müssen wir die Liste der Punktpaare  $(p_i, q_i)$  so sortieren, daß die Steigungen der Liniensegmente  $\overline{p_i q_i}$  die geforderten Eigenschaften (1) und (2) haben. Dabei können wir eine aus der Algorithmischen Geometrie bekannte Eigenschaft

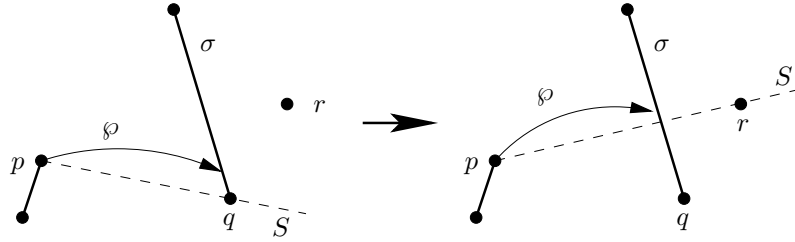
ausnutzen, die Dualität zwischen Punkten und Geraden, die gegeben ist durch die Abbildung<sup>4</sup>

$$\text{Punkt } p = (p_x, p_y) \mapsto \text{Gerade } p^* = \{ Y = p_x X - p_y \}.$$

Seien  $p = (p_x, p_y)$  und  $q = (q_x, q_y)$  o. E. mit  $p_x < q_x$  gegeben. Dann gilt für die Steigung des Segments

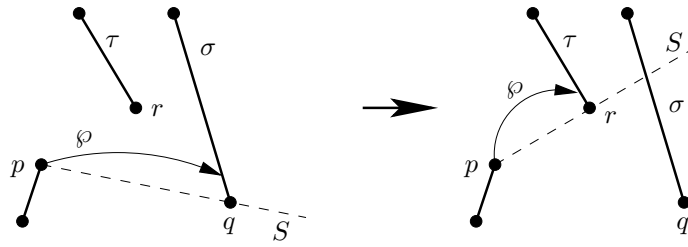
$$\text{slope}(p, q) = \frac{q_y - p_y}{q_x - p_x}.$$

Fall 1:  $r$  liegt hinter  $\sigma$  (Situation (i))



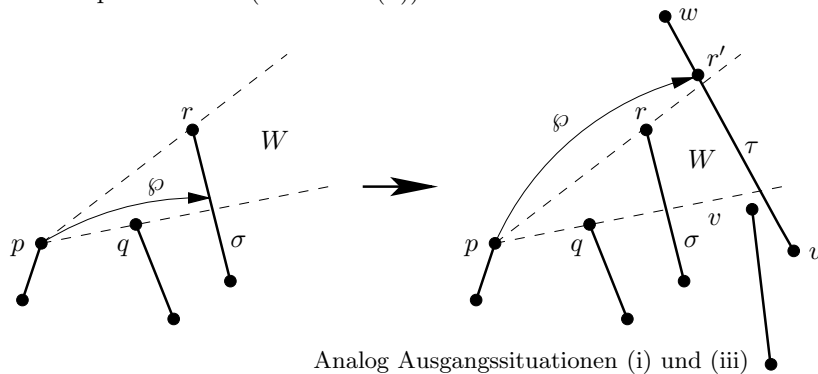
Analog Ausgangssituationen (ii) und (iii)

Fall 2:  $r$  liegt vor  $\sigma$  (Situation (i))



Analog Ausgangssituationen (ii) und (iii)

Fall 3:  $r$  ist Endpunkt von  $\sigma$  (Situation (ii))



Analog Ausgangssituationen (i) und (iii)

Abbildung 1.8: Drei Fälle bei der Bearbeitung eines Paares  $(p, r)$ .

<sup>4</sup>Diese Dualisierung unterscheidet sich nur im Vorzeichen von der in [Kle05] betrachteten Dualisierung  $p = (p_x, p_y) \mapsto p^* = \{ Y = p_x X + p_y \}$ .

Dem entspricht die X-Koordinate des Schnittpunktes  $(x, y)$  der zu  $p$  und  $q$  dualen Geraden  $p^*$  und  $q^*$ :

$$\begin{aligned} Y &= p_x X - p_y = q_x X - q_y \\ \Rightarrow X &= \frac{q_y - p_y}{q_x - p_x} = \text{slope}(p, q) . \end{aligned}$$

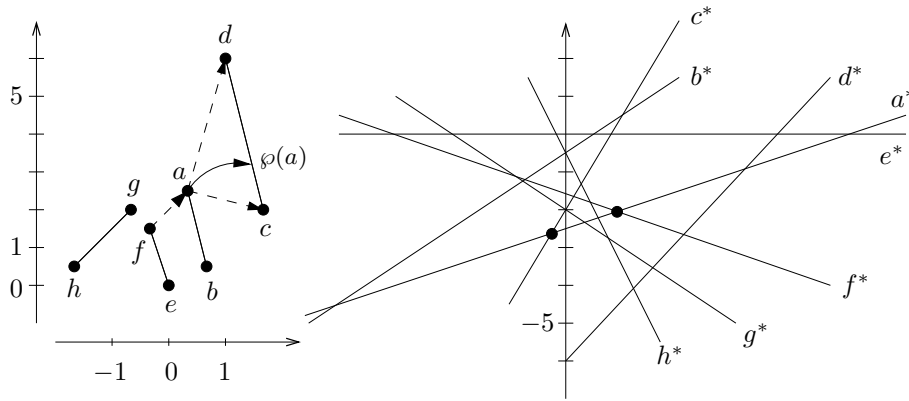


Abbildung 1.9: Dualisierung  $p = (p_x, p_y) \mapsto p^* = \{ Y = p_x X - p_y \}$ .

Wir können also die Bedingungen aus Abbildung 1.6(ii) so übersetzen:

- (1)  $\text{slope}(p, q) < \text{slope}(p, r) \Leftrightarrow p^* \cap q^*$  links von  $p^* \cap r^*$  auf Gerade  $p^*$
- (2a)  $\text{slope}(a, c) < \text{slope}(f, a) \Leftrightarrow a^* \cap c^*$  links von  $f^* \cap a^*$  auf Gerade  $a^*$
- (2b)  $\text{slope}(f, a) < \text{slope}(a, d) \Leftrightarrow f^* \cap a^*$  links von  $a^* \cap d^*$  auf Gerade  $a^*$

Also müssen wir die Schnittpunkte im Arrangement der Geraden  $p_i^*$  so bearbeiten, daß längs jeder Geraden die richtige Reihenfolge eingehalten wird. Abbildung 1.9 zeigt ein Beispiel eines Arrangements mit den dazu dualen Geraden. Da  $\text{slope}(a, c) < \text{slope}(f, a)$  gilt, muß der Schnittpunkt von  $a^*$  mit  $c^*$  links des Schnittpunktes von  $f^*$  mit  $a^*$  auf  $a^*$  liegen.

Zu lösen ist nun folgendes Problem: Gegeben seien  $n$  Geraden im  $\mathbb{R}^2$  in allgemeiner Lage, die eine Zerlegung der Ebene bewirken. Gesucht ist eine Reihenfolge der Schnittpunkte, die mit der X-Ordnung längs jeder Geraden verträglich ist. Dazu sind folgende Verfahren bekannt:

		Rechenzeit	Speicherplatz
Inkrementelle Konstruktion des Arrangements der Geraden (Edelsbrunner, O'Rourke, Seidel, 1986, [EOS86])		$O(n^2)$	$O(n^2)$
Sweep mit einer Geraden		$O(n^2 \log n)$	$O(n)$
Sweep mit einer Kurve (topologischer Sweep, Edelsbrunner, Guibas, 1986, [EG86, EG89])		$O(n^2)$	$O(n)$

Im folgenden konzentrieren wir uns auf den topologischen Sweep. Die wesentliche Idee dabei ist, den Sweep in der Ebene nicht mit einer Geraden durchzuführen, sondern mit einer Kurve, die topologisch einer Geraden entspricht:

**Definition 1.2** In einem Arrangement  $\mathcal{A}$  von Geraden heißt eine Kurve  $S$  **Pseudogerade**, wenn sie jede Gerade aus  $\mathcal{A}$  in höchstens einem Punkt schneidet.

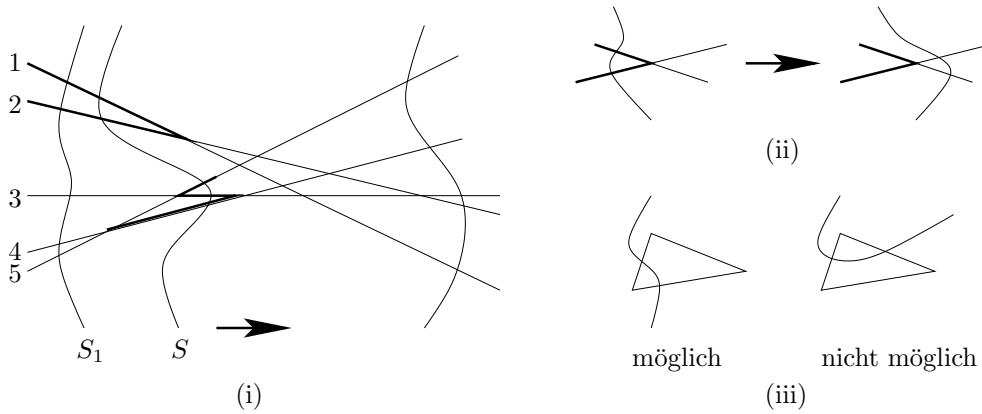


Abbildung 1.10: (i) Topologischer Sweep, (ii) Elementarschritt, (iii) Die Sweepkurve kann eine Zelle nur von oben betreten und nach unten verlassen.

Beim Sweep wird also eine Sweepkurve  $S$  (Pseudogerade) über das Arrangement  $\mathcal{A}$  bewegt (Abbildung 1.10(i)). Die von  $S$  geschnittenen Geraden werden längs  $S$  sortiert in einem Verzeichnis mitgeführt. In einem Elementarschritt des Sweeps wird die Sweepkurve über genau einen Schnittpunkt hinweg geführt (Abbildung 1.10(ii)). Das Problem dabei ist, zwei konsekutive Geraden längs  $S$  mit Schnittpunkt rechts von  $S$  für das nächste Elementarereignis zu finden, wenn im Mittel dafür Zeit  $O(1)$  zur Verfügung steht.

Die Existenz solcher zwei Geraden ist jedenfalls sichergestellt: Seien  $G_1$  und  $G_2$  die Geraden, deren Schnittpunkt rechts von  $S$  am weitesten links liegt, dann müssen  $G_1$  und  $G_2$  entlang  $S$  benachbart sein. Ein Elementarschritt ist also stets durchführbar, wenn es auch im allgemeinen mehrere Kandidaten dafür gibt.

Ein Ansatz zur Lösung dieses Problems ist, in einer Liste  $N$  entlang  $S$  sortiert für jede Gerade  $G$  festzuhalten, von welcher anderen Geraden  $G'$  das von  $S$  durchkreuzte Segment von  $G$  rechts begrenzt wird. Für das Arrangement in Abbildung 1.10(i) ergibt sich also entlang der ersten eingezeichneten Sweepkurve  $S_1$  die Liste

$$N = \{ (1, 2), (2, 1), (3, 5), (4, 5), (5, 4) \}.$$

Aus dieser Liste läßt sich ablesen, daß die Geraden 1 und 2 sowie die Geraden 4 und 5 benachbart und somit Kandidaten für den nächsten Elementarschritt sind. Für die zweite Sweepkurve  $S$  ergibt sich die Liste

$$N = \{ (1, 2), (2, 1), (5, 1), (3, 4), (4, 3) \},$$

und Kandidaten für den nächsten Elementarschritt sind die Paare  $(1, 2)$  und  $(3, 4)$ .

Nach einem Elementarschritt ist die Liste  $N$  zu aktualisieren. Dabei helfen uns obere und untere Horizontbäume, die wie folgt definiert sind, vgl. Abbildung 1.11:

Oberer Horizontbaum  $T^O$ :

- Markiere jede von  $S$  geschnittene Kante.
- Setze Markierung längs der Geraden nach rechts fort.
- Wo sich zwei markierte Kanten treffen, setze diejenige mit größerer Steigung nach rechts fort.

Unterer Horizontbaum  $T_U$ :

- Markiere jede von  $S$  geschnittene Kante.
- Setze Markierung längs der Geraden nach rechts fort.
- Wo sich zwei markierte Kanten treffen, setze diejenige mit kleinerer Steigung nach rechts fort.

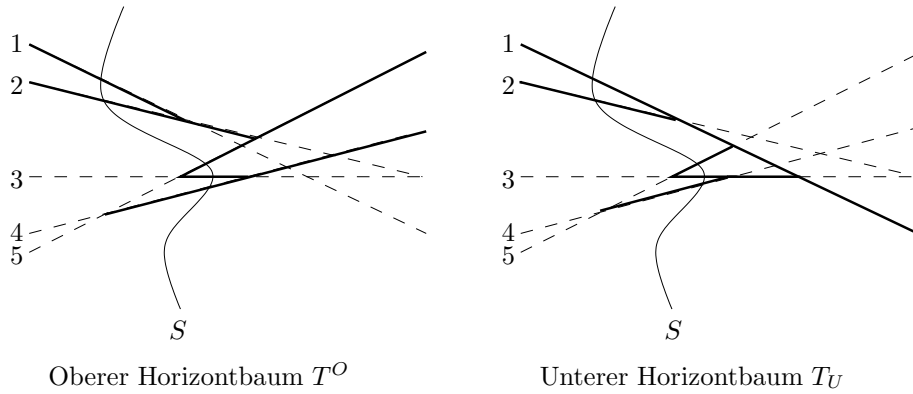


Abbildung 1.11: Oberer und unterer Horizontbaum.

Eine Eigenschaft der Horizontbäume ist folgende: Sei  $e$  eine von  $S$  geschnittene Kante von  $\mathcal{A}$ , und seien  $e^O$  und  $e_U$  ihre Verlängerungen in  $T^O, T_U$ . Dann ist  $e = \min(e^O, e_U)$ , d. h. aus  $T^O$  und  $T_U$  kann schnell bestimmt werden, von welchen Geraden die Kante  $e$  begrenzt wird; die kürzere Kante liefert den Schnittpunkt mit der begrenzenden Geraden.

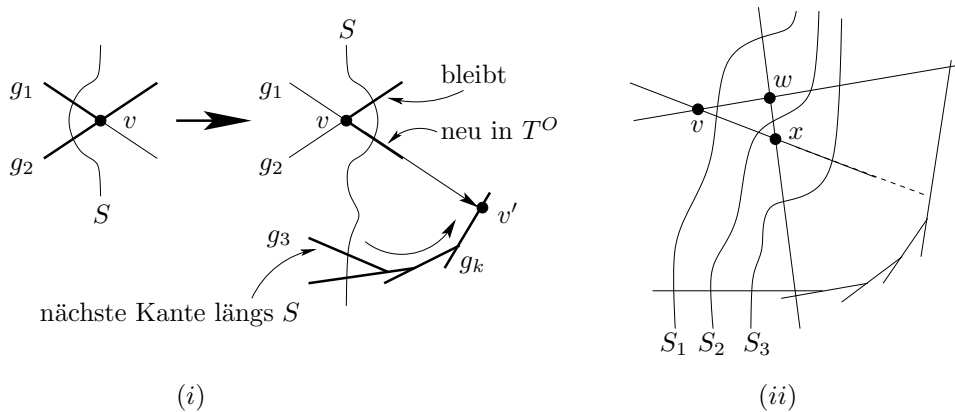
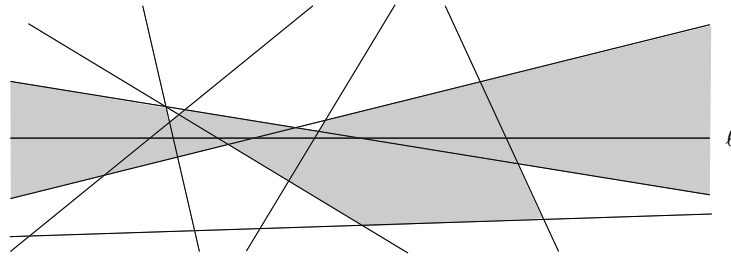
Abbildung 1.12: Aktualisierung der Horizontbäume am Beispiel von  $T^O$ .

Abbildung 1.12(i) zeigt die Aktualisierung der Horizontbäume am Beispiel von  $T^O$ : Nachdem die Sweepkurve  $S$  den Punkt  $v$  überschritten hat, wird die Gerade  $g_1$ , die bisher nur bis  $v$  markiert war, hinter  $v$  von  $S$  geschnitten. Somit wird das Segment  $vv'$  der Geraden rechts von  $v$  Teil des Horizontbaumes.  $v'$  ist dabei der zu  $v$  nächstgelegene Schnittpunkt von  $g_1$  mit einer markierten Geraden  $g_k$  mit größerer Steigung. Um diese zu finden, suchen wir längs  $S$  die nächste Schnittkante  $g_3$  und folgen der konvexen Kantenfolge bis zum Schnittpunkt mit  $g_1$ . Die Suche nach einem Schnittpunkt kann dabei  $O(n)$  viele Schritte erfordern. Es kann auch vorkommen, daß dieselben Kanten mehrfach abgesucht werden müssen; Abbildung 1.12(ii) zeigt ein Arrangement, in dem bei den Knoten  $v$  und  $x$  dieselben Kanten durchsucht werden.

Um eine Übersicht über die Kosten zu bekommen, wollen wir die Kosten des Besuchs einer Kante  $e$  dem verursachenden Knoten  $v$  in Rechnung stellen. Ideal wäre, wenn Bilanzen nur zwischen Knoten und Kanten derselben Zelle im Arrangement aufgestellt werden müßten. Abbildung 1.12(ii) zeigt allerdings, daß dies nicht ohne weiteres möglich ist: die Kanten, die bei der Bearbeitung von  $v$  besucht werden, liegen nicht alle in derselben Zelle wie  $v$ . Dies wird erst bei der Bearbeitung von  $x$  festgestellt, und  $x$  liegt in derselben Zelle wie die Kanten, also transferieren wir die Kosten dieser Kanten auf  $x$ . Zur Berechnung dieser Kosten brauchen wir Folgendes:

Abbildung 1.13: Zone einer Geraden  $\ell$  in einem Arrangement  $\mathcal{A}$ .

**Definition 1.3** Sei  $\mathcal{A}$  ein Arrangement von  $n$  nicht senkrechten Geraden, und sei  $\ell$  eine beliebige Gerade. Die **Zone** von  $\ell$  in  $\mathcal{A}$  besteht aus allen Zellen, deren Abschluß von  $\ell$  geschnitten wird:

$$\text{Zone}(\ell) := \{ \text{Zellen } Z \text{ in } \mathcal{A} \mid \ell \cap \overline{Z} \neq \emptyset \}$$

**Theorem 1.4** (Zonentheorem)

Sei  $\mathcal{A}$  ein Arrangement von  $n$  nicht senkrechten Geraden und sei  $\ell$  eine beliebige Gerade. Die Zone von  $\ell$  hat die Komplexität  $O(n)$ .

**Beweis.**

In jeder Zelle gibt es einen rechten, obersten Punkt und einen linken, untersten Punkt. Diese Punkte teilen den Rand einer Zelle eindeutig in einen linken und einen rechten Rand.

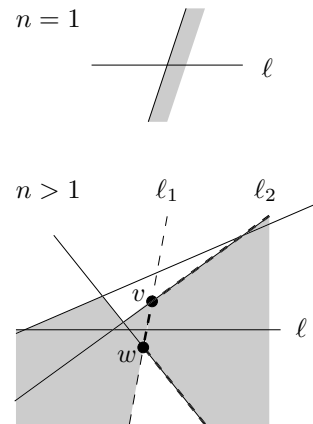
Wir zeigen durch Induktion über  $n$ , daß es höchstens  $5n$  linke Kanten auf den Rändern der Zonenzellen geben kann. Aus Symmetriegründen gibt es auch höchstens  $5n$  rechte Kanten.

Für  $n = 1$  haben wir eine linke Kante.

Für  $n > 1$  sei  $\ell_1$  die Gerade im Arrangement, deren Schnitt mit  $\ell$  am weitesten rechts liegt (wir nehmen an, diese sei eindeutig).

Sei  $v$  der niedrigste Schnitt von  $\ell_1$  mit einer Geraden aus  $\mathcal{A}$  oberhalb von  $\ell$  und  $w$  der höchste Schnitt von  $\ell_1$  mit einer Geraden aus  $\mathcal{A}$  unterhalb von  $\ell$ .

Sei  $\mathcal{A}'$  das Arrangement ohne  $\ell_1$ . Lt. Induktionsvoraussetzung hat die Zone von  $\ell$  in  $\mathcal{A}'$  höchstens  $5(n-1)$  linke Kanten.



Wieviele linke Kanten werden nun beim Einfügen von  $\ell_1$  höchstens hinzugefügt?

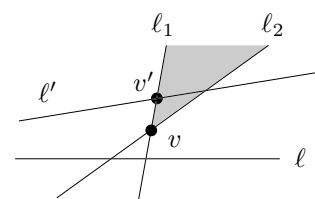
- (i) Die Kante  $\overline{vw}$ .
- (ii) Sei  $\ell_2$  die Gerade, die  $\ell_1$  in  $v$  schneidet.  $\ell_2$  hat vor dem Einfügen von  $\ell_1$  möglicherweise mehrere linke Kanten gestellt. Höchstens eine davon kann durch Einfügen von  $\ell_1$  in  $v$  geteilt werden.
- (iii) Entsprechend für  $w$ .

Es kommen also höchstens drei linke Kanten hinzu, also hat  $\text{Zone}(\ell)$  in  $\mathcal{A}$  höchstens  $5(n-1)+3 \leq 5n$  linke Kanten.

Zu zeigen ist noch, warum beim Einfügen von  $\ell_1$  keine weiteren linken Kanten geteilt werden können. Nehmen wir dazu an, die Gerade  $\ell'$  würde  $\ell_1$  oberhalb von  $\ell$  schneiden

$\Rightarrow \exists$  Schnittpunkt  $v'$  oberhalb von  $v$ .

$\Rightarrow$  kurz rechts von  $\ell_1$  kann kein Stück von  $\ell'$  zu  $\text{Zone}(\ell)$  gehören, weil  $\ell'$  dort in einer Region liegt, die nach unten



hin von  $\ell_1$  und  $\ell_2$  begrenzt wird.

$\Rightarrow$  es kann keine Teilung stattgefunden haben.

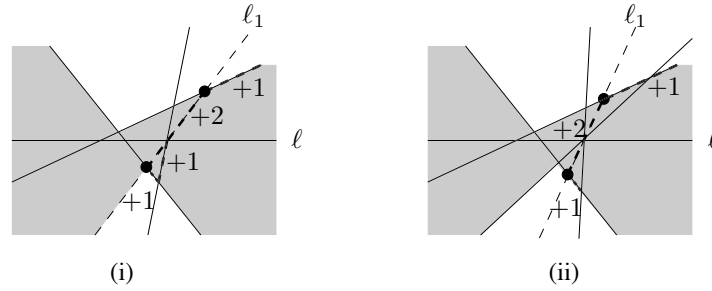


Abbildung 1.14: Im rechtesten Schnittpunkt schneiden sich zwei oder mehrere Geraden. Die gestrichelten Kanten werden neu eingefügt.

Ist die Gerade  $\ell_1$  nicht eindeutig, d.h. in dem weitesten rechts liegenden Schnittpunkt mit  $\ell$  schneiden sich mehrere Geraden, so wählen wir eine dieser Geraden als  $\ell_1$ . Mit der gleichen Zählweise wie oben stellen wir fest, daß wir für zwei schneidende Geraden 5 neue linke Kanten bekommen können, siehe Abbildung 1.14(i), für drei oder mehrere Geraden 4 linke Kanten (Abbildung 1.14(ii)).  $\square$

Wir können die Kosten für die Aktualisierung der Horizontbäume wie folgt abschätzen:

$$\begin{aligned}
 & \text{Kosten der Aktualisierung von } T^O, T_U \\
 & \approx \text{Summe aller Kosteneinheiten Kante-Knoten aus je einer Zelle} \\
 & \leq \sum_{\text{Zellen } Z} (\# \text{Kanten von } Z) \cdot (\# \text{Knoten von } Z) \\
 & = \sum_{\text{Zellen } Z} (\# \text{Kanten von } Z)^2 \\
 & \leq \sum_{\text{Geraden } G} \underbrace{\sum_{\text{Zellen } Z \text{ mit Kante auf } G} (\# \text{Kanten von } Z)}_{O(n) \text{ nach Theorem 1.4}} \\
 & \in O(n^2)
 \end{aligned}$$

Bemerkung: Die Kosten der Aktualisierung der Horizontbäume, nachdem die Sweep Pseudogerade über einen Schnittpunkt hinweggehoben wurde, entstehen durch entlang laufen des Randes der aktuellen Zelle  $z$ . Der Rand einer Zelle  $z$  kann also insgesamt maximale Anzahl Knoten von  $z$  mal durchlaufen werden. Die Gesamtkosten ergeben sich durch Summation über alle Zellen  $z$ .

Damit haben wir folgende Laufzeiten gezeigt:

Aktualisierung der Horizontbäume	$O(n^2)$
$\Rightarrow$ Topologischer Sweep	$O(n^2)$
$\Rightarrow$ Wahl der Bearbeitungsreihenfolge	$O(n^2)$
$\Rightarrow$ Konstruktion des Sichtbarkeitsgraphen	$O(n^2)$

Also gilt:

**Theorem 1.5** (Welzl, 1985)

Der Sichtbarkeitsgraph von  $n$  Liniensegmenten kann in Zeit  $O(n^2)$  und in Speicherplatz  $O(n)$  konstruiert werden. [Wel85]

Dabei werden sichtbare Paare von Endpunkten nur berichtet, nicht gespeichert. Außerdem setzen wir voraus, daß sich die Punkte in allgemeiner Lage befinden, insbesondere keine drei Punkte kollinear sind.<sup>5</sup>

Den Sichtbarkeitsgraphen in Algorithmus 1.1 können wir also in Zeit  $O(n^2)$  konstruieren, auch das Hinzufügen der Kanten zu Start- und Zielpunkt ist in  $O(n^2)$  möglich. Damit bleibt aus Algorithmus 1.1 das Problem der Suche nach einem kürzesten Weg in ungerichteten Graphen mit nicht-negativen Kantengewichten; dies ist ein diskretes Problem.

Dieses Problem läßt sich mit dem Algorithmus von Dijkstra (Algorithmus 1.3, [Dij59]) lösen. Dieser verwaltet eine Welle  $W$  mit einem Ausläufer  $A$  um  $a$  und aktualisiert in jedem Schritt die Knotenmarkierung

$$d(p) = \begin{cases} \text{min. Länge eines Pfads von } a \text{ nach } p & \text{für } p \in W \\ \text{min. Länge eines Pfads von } a \text{ nach } p, \text{ der bis auf } p \text{ in } W \text{ verläuft} & \text{für } p \in A \\ \infty & \text{sonst} \end{cases}$$

---

**Algorithmus 1.3** Kürzeste Pfade in gewichtetem Graph (Dijkstra, 1959)

---

**Gegeben:** Graph  $G = (V, E)$  zusammenhängend,  $|V| = n$ ,  $|E| = m$   
 Kantengewichtung  $g : E \rightarrow \mathbb{R}^{\geq 0}$ , Knoten  $a, b \in V$ .

**Gesucht:** Kantenfolge von  $a$  nach  $b$  mit minimalem Gesamtgewicht.

Solange  $A \neq \emptyset$ :

- Entnehme  $A$  ein  $p$  mit minimalem  $d(p)$ .
  - $W := W \cup \{p\}$ .
  - Für alle direkten Nachbarn  $q$  von  $p$  in  $W^C$ :
    - $d(q) := \min \{ d(q), d(p) + g(p, q) \}$
    - Wenn  $q \notin A$  dann  $A := A \cup \{q\}$ .
- 

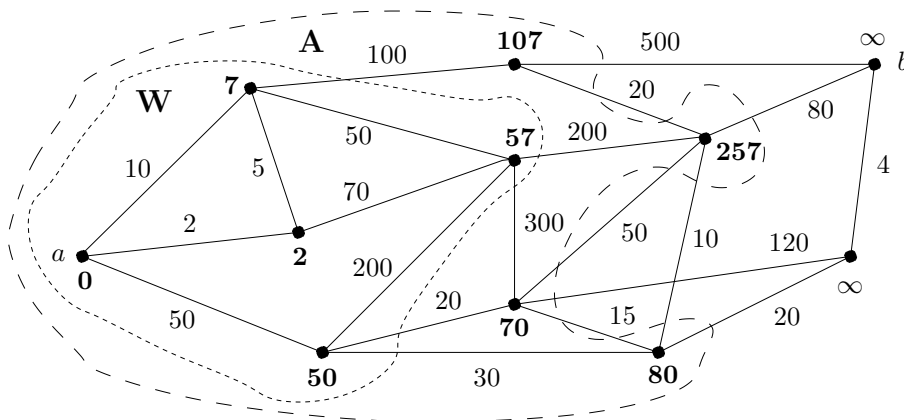


Abbildung 1.15: Welle, Ausläufer und Knotenmarkierungen während des Algorithmus von Dijkstra.

<sup>5</sup>Ansonsten wären noch einige Spezialfälle zu beachten, die aber an Laufzeit und Speicherplatz nichts ändern würden.



Abbildung 1.15 zeigt einen Zwischenstand bei der Berechnung des kürzesten Weges mit der aktuellen Welle  $W$ , dem aktuellen Ausläufer  $A$  und den aktuellen Knotenmarkierungen (fett).

Zur Implementierung des Algorithmus von Dijkstra wird eine Prioritätswarteschlange für den Ausläufer  $A$  benötigt. Diese soll eine Menge von Paaren  $(p, d(p))$  verwalten. Die Operationen Einfügen eines Elementes, Löschen des Minimums werden insgesamt  $O(n)$  mal durchgeführt. Vermindern des Wertes  $d(p)$  kann insgesamt  $O(m)$  mal durchgeführt werden. Die Elemente befinden sich bei diesen Operationen in bekannter Position, d. h. es kann ohne vorherige Suche darauf zugegriffen werden. Zur Realisierung dieser Warteschlange gibt es folgende Möglichkeiten:

Führen wir  $i$  mal einfügen,  $j$  mal vermindern und  $k$  mal löschen des Minimums in beliebiger Reihenfolge aus.

- Lineare Liste, unsortiert: Einfügen, Vermindern in  $O(1)$ , Löschen–Min in  $O(i)$ . Gesamtlaufzeit in  $O(i^2 + j)$
- Fibonacci Heap (Fredman, Tarjan, 1987; beschrieben u.a. im Buch von Ottmann und Widmayer, [OW93]): Einfügen, Vermindern in  $O(1)$  und Löschen–Min  $O(\log i)$ . Gesamtlaufzeit in  $O(k \log i + i + j)$ . [FT87]
- Relaxed Heap (Driscoll, Gabow, Shrairaman, Tarjan, 1988): Einfügen, Vermindern in  $O(1)$ , Löschen–Min in  $O(\log i)$ . Gesamtlaufzeit in  $O(k \log i + j)$  [DGST88]

Wird der Algorithmus von Dijkstra mit Fibonacci–Heaps implementiert, kann man in einem Graph  $G$  mit  $n$  Knoten und  $m$  Kanten die kürzesten Wege von einem Knoten  $a$  zu allen übrigen Knoten in Zeit  $O(n \log n + m)$  bestimmen.

Damit ist folgendes gezeigt:

### Theorem 1.6

*Gegeben seien  $h$  Polygone mit  $n$  Ecken, ein Startpunkt  $s$  und ein Zielpunkt  $t$ . Ein kürzester Weg von  $s$  nach  $t$  kann in Zeit  $O(n^2)$  berechnet werden, die kürzesten Wege von  $s$  zu allen Ecken ebenfalls in Zeit  $O(n^2)$ .*

Der Engpaß bei der bisherigen Berechnung kürzester Wege ist die Konstruktion des Sichtbarkeitsgraphen. Wie bereits erwähnt, kann dieser  $\Omega(n^2)$  viele Kanten enthalten, jedoch auch viel weniger, vgl. Abbildung 1.4. Beim obigen Algorithmus brauchen wir stets eine Laufzeit von  $\Omega(n^2)$  wegen der Konstruktion des Arrangements der dualen Geraden. Die Frage ist nun, ob sich der Sichtbarkeitsgraph auch output–sensitiv bzgl. seiner Größe berechnen läßt. Tatsächlich ist dies möglich: Zu einem Arrangement aus  $n$  Liniensegmenten, dessen Sichtbarkeitsgraph aus  $m$  Kanten besteht, sind folgende output–sensitiven Algorithmen bekannt:

Overmars, Welzl, 1988: Zeit  $O(m \log n)$  und Platz  $O(n)$  [OW88],  
 Ghosh, Mount, 1987: Zeit  $O(n \log n + m)$  und Platz  $O(m)$  [GM87, GM91],  
 Pocchiola, Vegter, 1995: Zeit  $O(n \log n + m)$  und Platz  $O(n)$  [PV95].

Damit gilt:

**Theorem 1.7** *Mit der Fibonacci–Heap Implementierung des Algorithmus von Dijkstra und der output–sensitiven Berechnung des Sichtbarkeitsgraphen lassen sich kürzeste Wege in Zeit  $O(n \log n + m)$  berechnen.*

Bei der Laufzeit ist jedoch der Anteil von  $O(n \log n)$  unvermeidlich:

**Theorem 1.8** (untere Schranke) *In einer Umgebung von Polygonen mit insgesamt  $n$  Ecken braucht die Berechnung eines kürzesten Pfades Zeit  $\Omega(n \log n)$ .*

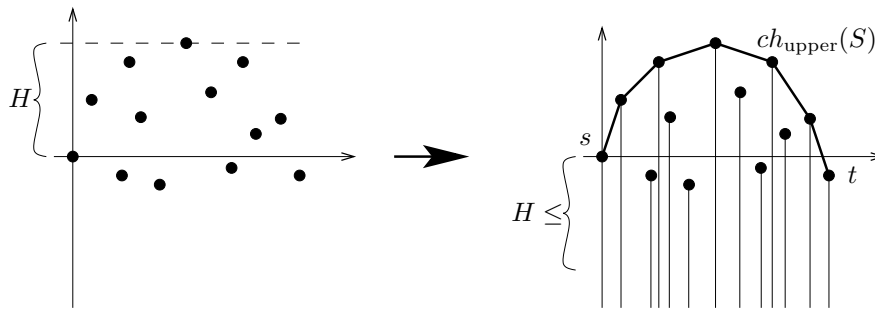


Abbildung 1.16: Reduktion: Konstruktion der konvexen Hülle auf Bestimmung des kürzesten Pfades.

### Beweis.

Wir zeigen, daß sich die Konstruktion der konvexen Hülle auf die Berechnung kürzester Pfade reduzieren läßt. Da sich Sortieren auf die Konstruktion der konvexen Hülle reduzieren läßt, erhalten wir eine Laufzeit von  $\Omega(n \log n)$ .

Sei eine Menge von Punkten gegeben. Sei  $s$  der linkeste Punkt,  $t$  der rechteste Punkt und  $H$  die Höhe der vertikale Abstand zwischen niedrigsten und höchsten Punkt. Lege an jeden Punkt ein senkrecht nach unten zeigendes Liniensegment einer Länge  $> 2H$ . Dann entspricht der kürzeste Weg von  $s$  nach  $t$  der oberen konvexen Hülle, siehe Abbildung 1.16.  $\square$

Die nächste Frage ist, ob man wirklich den Sichtbarkeitsgraphen konstruieren muß. In der Tat ist dies nicht nötig, folgender Ansatz ist als Locus Approach bekannt und sei hier nur kurz umrissen:

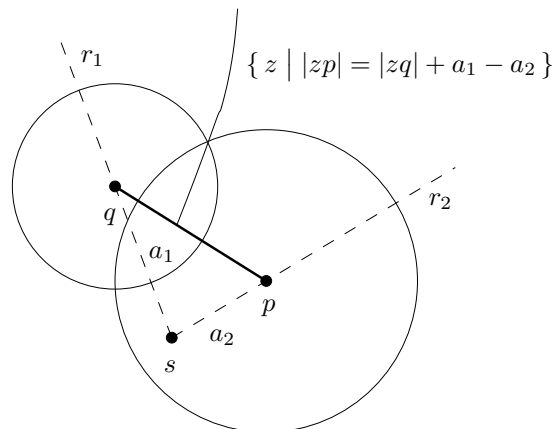


Abbildung 1.17: Berechnung der Shortest Path Map.

Wir zerlegen den freien Teil der Ebene in Regionen, so daß alle Punkte einer Region kombinatorisch gleiche<sup>6</sup> Kürzeste zum Startpunkt  $s$  haben (Shortest Path Map). In Abbildung 1.17 sind genau für die Punkte auf der Hyperbel  $H = \{ z \mid |zp| = |zq| + a_1 - a_2 \}$  die Wege von  $s$  über  $p$  oder über  $q$  gleich lang. Dies entspricht einem Voronoi-Diagramm mit additiven Gewichten, siehe Abschnitt A.3.4. Die Shortest Path Map läßt sich mit der Continuous Dijkstra Methode berechnen: Wir lassen um  $s$  einen Kreis wachsen. Wenn dieser eine Hindernisecke berührt, entsteht ein neuer Kreis, der mit der gleichen Geschwindigkeit wie die bisherigen Kreise um die Hindernisecke wächst (Wellenfront). Es wird festgehalten, von welchem Kreis ein Punkt der Ebene zuerst erreicht wird. Auf die Continuous Dijkstra Methode werden wir in Abschnitt 1.4 noch zurückkommen, wir halten hier nur fest:

<sup>6</sup>D. h. sie berühren dieselben Ecken in derselben Reihenfolge.

**Theorem 1.9** (Hershberger, Suri, 1997)

Gegeben seien polygonale Hindernisse mit insgesamt  $n$  Ecken und ein fester Startpunkt  $s$ . Dann läßt sich in Zeit  $O(n \log n)$  und Platz  $O(n \log n)$  eine Karte der kürzesten Wege berechnen. Diese braucht  $O(n)$  Platz und ermöglicht für einen beliebigen Zielpunkt  $t$

- (i) in Zeit  $O(\log n)$  die Entfernung von  $s$  zu bestimmen und
- (ii) in Zeit  $O(\log n + k)$  den kürzesten Weg von  $s$  nach  $t$  zu berichten, wobei  $k$  die Anzahl der Kanten auf dem Weg ist.

[HS99]

## 1.2 Kürzeste Pfade im Inneren eines einfachen Polygons

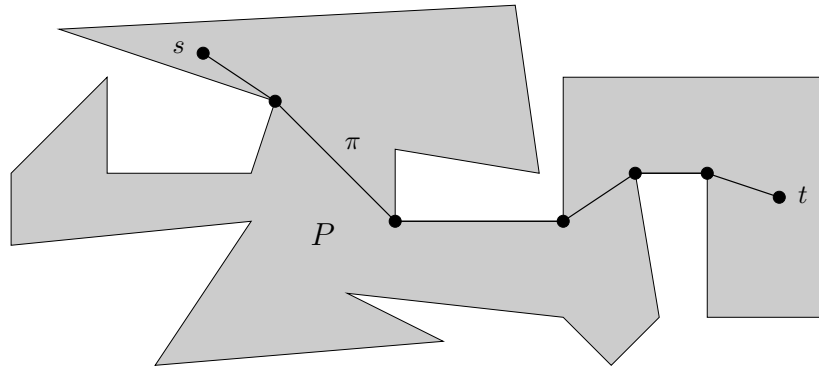


Abbildung 1.18: Kürzester Pfad im Inneren eines einfachen Polygons.

Betrachten wir nun das Problem, zu einem gegebenem einfachen Polygon mit  $n$  Ecken und zwei beliebigen Punkten  $s$  und  $t$  im Inneren des Polygons den kürzesten Pfad  $\pi$  zu bestimmen, der im Inneren des Polygons verläuft. Im Gegensatz zu dem Problem in Abschnitt 1.1 ist der kürzeste Pfad hier eindeutig bestimmt. Man kann sich den Pfad auch in diesem Fall als straff gespanntes Gummiband zwischen  $s$  und  $t$  vorstellen, das wie in Abschnitt 1.1 nur an den Ecken des Polygons abknicken kann.

### 1.2.1 Der Algorithmus von Lee und Preparata

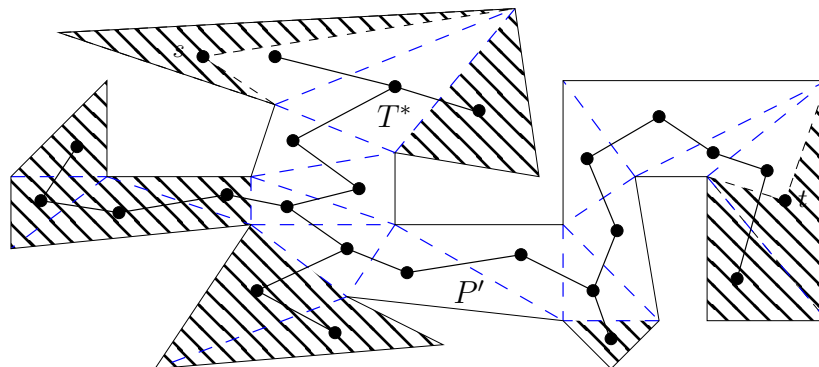


Abbildung 1.19: Triangulation  $T$  und dualer Graph  $T^*$ .

Wie in Abschnitt 1.1 ist unser Ansatz auch hier, das gegebene Problem in ein diskretes Problem zu verwandeln; dabei hilft uns die aus der Algorithmischen Geometrie bekannte Triangulation des Polygons:

**Definition 1.10** Zu einem einfachen Polygon  $P$  ist die **Triangulation**  $T$  eine maximale Menge von kreuzungsfreien Liniensegmenten im Innern von  $P$ , deren Endpunkte auf den Ecken von  $P$  liegen. Der zur Triangulation  $T$  duale Graph  $T^*$  enthält die Dreiecke der Triangulation als Knoten und Kanten zwischen den Knoten von Dreiecken mit einer gemeinsamen Seite.

Aus der Algorithmischen Geometrie [Kle97] kennen wir ein randomisiertes Verfahren zur Berechnung der Triangulation in Zeit  $O(n \log n)$ , ein lineares Verfahren wurde 1991 von Chazelle vorgestellt [Cha91]; Abbildung 1.19 zeigt die Triangulation und den dualen Graphen zu dem Beispiel aus Abbildung 1.18.

Der duale Graph  $T^*$  ist ein Baum mit Knotengrad  $\leq 3$ , also entspricht dem kürzesten Pfad  $\pi$  von  $s$  nach  $t$  ein eindeutiger Pfad  $\pi_{T^*}$  im Baum  $T^*$  von dem Dreieck, das  $s$  enthält, zu dem

Dreieck, das  $t$  enthält. Nach Lokalisation der Dreiecke, die  $s$  bzw.  $t$  enthalten, läßt sich  $\pi_{T^*}$  durch Tiefensuche in  $T^*$  in linearer Zeit bestimmen.

Betrachten wir das Teilpolygon  $P'$ , das die Dreiecke längs  $\pi_{T^*}$  sowie zwei angesetzte Dreiecke mit Ecken  $s$  bzw.  $t$  enthält und sich in Zeit  $O(n)$  bestimmen läßt, so muß der kürzeste Pfad  $\pi$  in  $P'$  verlaufen. Der nicht zu  $P'$  gehörende Teil ist in Abbildung 1.19 schraffiert dargestellt.

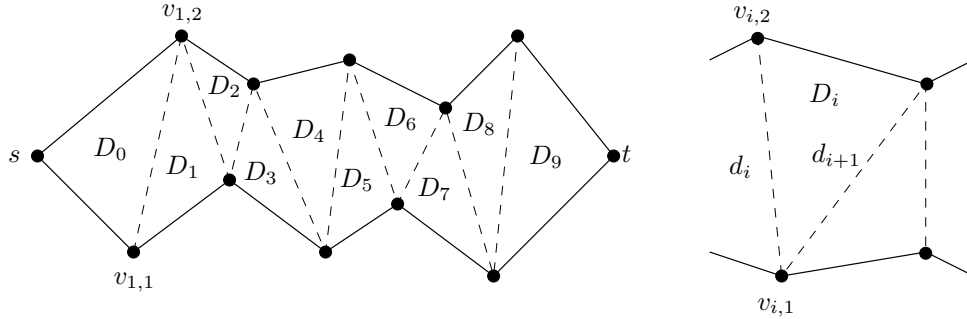


Abbildung 1.20: Kette von Dreiecken.

Damit reduziert sich das anfängliche Problem auf die Bestimmung des kürzesten Pfades von  $s$  nach  $t$  in  $P'$ . Dieses Polygon hat jedoch eine spezielle Eigenschaft: es besteht aus einer Kette von Dreiecken, bei der zwei konsecutive Diagonalen  $d_i, d_{i+1}$  immer eine gemeinsame Ecke haben, vgl. Abbildung 1.20. Das erlaubt uns, induktiv die Kürzesten  $\pi(s, v_{i,1})$  und  $\pi(s, v_{i,2})$ ,  $1 \leq i \leq n$  von  $s$  zu den Endpunkten  $v_{i,1}$  und  $v_{i,2}$  der Diagonalen  $d_i$  zu bestimmen: der Fall  $i = 1$  ist trivial, der kürzeste Weg besteht aus den beiden Kanten des Polygons mit Ecke  $s$ . Nehmen wir also per Induktion an, der kürzeste Pfad von  $s$  zu  $v_{i,1}$  und  $v_{i,2}$  sei bekannt. Die Gestalt der Pfade  $\pi(s, v_{i,1})$  und  $\pi(s, v_{i,2})$  ist in Abbildung 1.21 dargestellt: nach einem eventuellen gemeinsamen Teil vom Startpunkt  $s$  bilden sie einen Trichter mit der Diagonalen  $d_i$  als Oberseite.

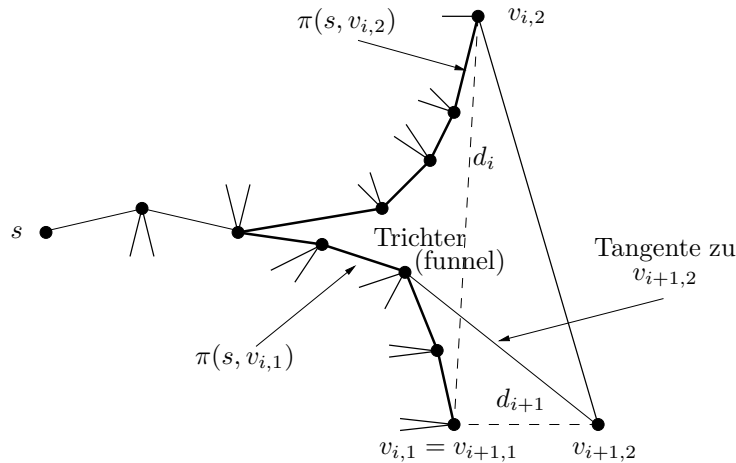


Abbildung 1.21: Trichter.

Sei o. B. d. A.  $v_{i+1,1} = v_{i,1}$ . Zu berechnen ist also der kürzeste Weg von  $s$  nach  $v_{i+1,2}$ . Dazu starte in  $v_{i,2}$  und suche die Wand des alten Trichters gegen den Uhrzeigersinn nach einem Tangentenpunkt von  $v_{i+1,2}$  ab. Diese Tangente gehört zum kürzesten Weg von  $s$  nach  $v_{i+1,2}$ . Dabei durchlaufene Kanten fallen entweder weg oder gehören zu einem gemeinsamen Stück des neuen Trichters und werden nie wieder besucht, so daß sich eine Laufzeit von  $O(n)$  für die Bestimmung des kürzesten Weges von  $s$  nach  $t$  in  $P'$  ergibt. Damit haben wir folgendes Theorem gezeigt, vgl. Algorithmus 1.4:

**Theorem 1.11** (Lee, Preparata, 1984)

Der kürzeste Pfad zwischen zwei Punkten in einem einfachen Polygon kann in Zeit  $O(n)$  und Platz  $O(n)$  bestimmt werden. [LP84]

**Algorithmus 1.4** Kürzester Pfad in einem Polygon (Lee, Preparata)

- Berechne Triangulation  $T$  und dualen Graphen  $T^*$  des Polygons  $P$ .  $O(n)$
- Bestimme die Dreiecke  $D_s$  und  $D_t$  die  $s$  bzw.  $t$  enthalten.  $O(n)$
- Bestimme das Teilpolygon  $P'$ , das nur Dreiecke längs des Pfades von  $D_s$  nach  $D_t$  in  $T^*$  enthält (Tiefensuche).  $O(n)$
- Bestimme induktiv die Kürzesten  $\pi(s, v_{i,1})$  und  $\pi(s, v_{i,2})$ ,  $1 \leq i \leq n$  von  $s$  zu den Endpunkten  $v_{i,1}$  und  $v_{i,2}$  der Diagonale  $d_i$ .  $O(n)$

**1.2.2 Der Algorithmus von Guibas und Hershberger**

Das Ergebnis von Lee und Preparata läßt sich verbessern; wir benötigen dazu folgendes Theorem:

**Theorem 1.12** (Cutting-Theorem, Chazelle, 1982)

Sei  $T$  eine Triangulation eines einfachen Polygons  $P$ , dann existiert eine Diagonale  $d \in T$ , so daß auf jeder Seite von  $d$  mindestens  $\frac{n}{3} - 1$  Dreiecke liegen. [Cha82]

**Beweis.** Übungsaufgabe. Hinweis: Man nutze die Tatsache aus, daß der duale Graph von  $T$  ein Baum ist. □

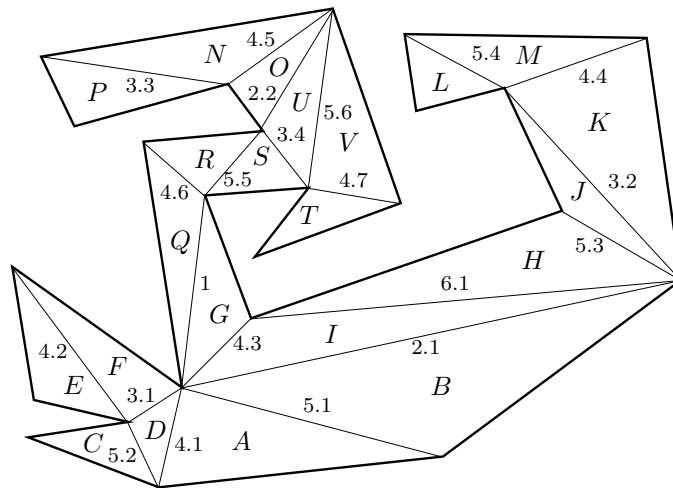


Abbildung 1.22: Balancierte hierarchische Triangulation.

Durch iterierte Anwendung von Theorem 1.12 auf die Teilpolygone zu beiden Seiten von  $d$  entsteht eine balancierte, hierarchische Zerlegung von  $P$  in Dreiecke, vgl. Abbildung 1.22. Diese Zerlegung läßt sich in einem Binärbaum  $\hat{T}$  darstellen, dessen Knoten den Diagonalen in  $T$  entsprechen, vgl. Abbildung 1.23. Die Wurzel von  $\hat{T}$  entspricht also der Diagonalen, die  $P$  in  $P_1$  und  $P_2$  unterteilt, die Kinder der Wurzel sind die Diagonalen der Zerlegungen von  $P_1$  und  $P_2$  etc. Bezeichnet  $P(d)$  das Subpolygon, das von der Diagonalen  $d$  unterteilt wird, so entspricht jedem Knoten in  $\hat{T}$  ein Subpolygon  $P(d)$  von  $P$ ; die Wurzel von  $\hat{T}$  entspricht  $P$ , die Blätter von

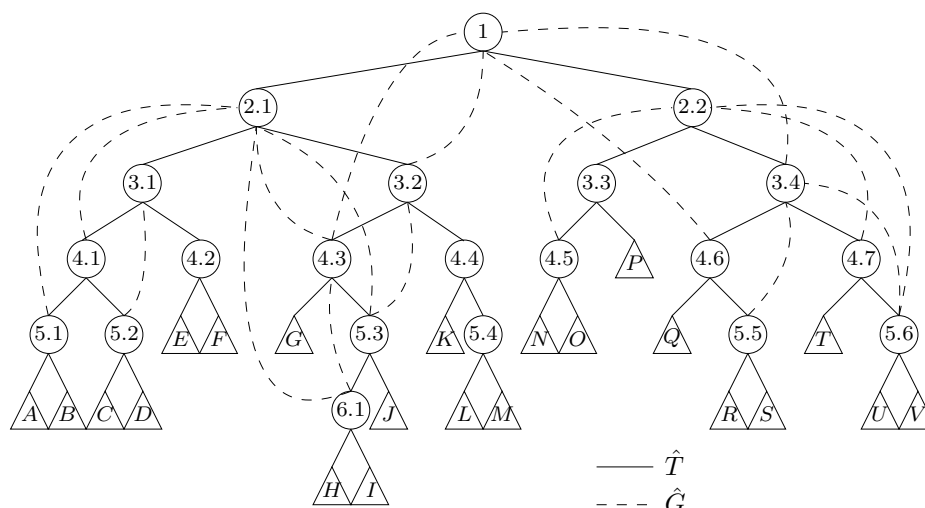


Abbildung 1.23: Baum der balancierten hierarchischen Triangulation.

$\hat{T}$  den Dreiecken der Triangulation von  $P$ . Diese Subpolygone werden von Kanten von  $P$  und Diagonalen berandet.

Zur Illustration des Zusammenhangs zwischen den Pfaden in der Triangulation  $T$  und den Pfaden im Baum  $\hat{T}$  betrachten wir den Pfad von Dreieck  $P$  nach Dreieck  $M$  in Abbildung 1.22 und vergleichen die in  $T$  überquerten Diagonalen mit den in  $\hat{T}$  durchlaufenen Knoten:

$T$	3.3	4.5	2.2	3.4	5.5	4.6	1	4.3	6.1	5.3	3.2	4.4
$\hat{T}$	3.3		2.2				1		2.1		3.2	4.4

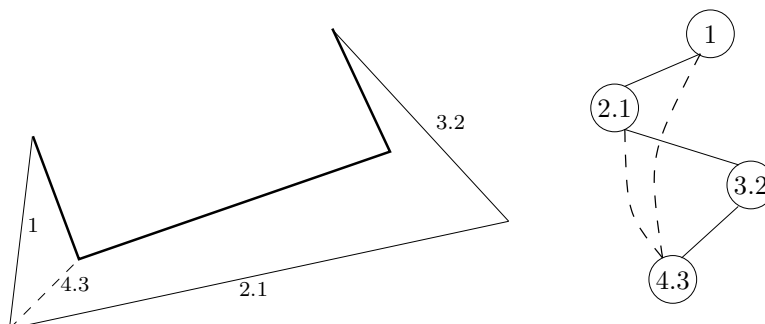
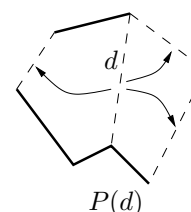


Abbildung 1.24: Subpolygon  $P(d_{4.3})$  und Abkürzungen in  $\hat{G}$ .

Zu beobachten ist, daß der Pfad in  $\hat{T}$  im allgemeinen kürzer ist — da der Baum balanciert ist, ist die Länge des Pfades  $O(\log n)$  —, jedoch enthält er Knoten, die Diagonalen entsprechen, die vom Pfad im Polygon nicht überquert werden, wie die Knoten 2.1 und 5.4 im obigen Beispiel. 5.4 ist dabei nur ein “falsches” Endstück, Probleme bereitet 2.1. Abhilfe schafft die Bereitstellung von Abkürzungen in  $\hat{T}$ : kürzeste Pfade, die ein Subpolygon  $P(d)$  passieren, passieren ein Paar von Diagonalen  $d_1, d_2$ , die auf dem Rand von  $P(d)$  liegen. Um diese Paare systematisch zu erfassen, werden beim Einfügen der Diagonalen  $d$  in  $\hat{T}$  zusätzliche Zeiger von  $d$  auf alle älteren Diagonalen, die das von  $d$  geteilte Polygon  $P(d)$  beranden, hinzugefügt, siehe obige Abbildung. Wird im Beispiel von oben der Knoten 4.3 eingefügt, so werden die zusätzlichen Zeiger von 4.3 auf die Knoten 1 und 2.1 aufgenommen; mit dem Knoten 3.2 ist 4.3 schon in  $\hat{T}$  verbunden, siehe Abbildung 1.24. So



entsteht aus dem Hierarchiebaum<sup>7</sup>  $\hat{T}$  der Triangulation  $T$  der Schichtengraph  $\hat{G}$ , mit folgenden Eigenschaften:

**Lemma 1.13** *Eigenschaften von  $\hat{G}$ :*

(i) Der Weg zwischen zwei beliebigen Dreiecken in  $T$  ergibt sich durch Konkatination von  $O(\log n)$  vielen Kanten im Graphen  $\hat{G}$ .

(ii) Diese Kanten lassen sich in Zeit  $O(\log n)$  in  $\hat{G}$  bestimmen.

(iii) Die Größe von  $\hat{G}$  ist  $O(n)$ .

**Beweis.**

(i) Sei  $\pi^*$  der Weg im dualen Graphen  $T^*$ . Betrachte die von  $\pi^*$  überquerten Diagonalen der Triangulation und ihre Tiefe (Schicht) in  $\hat{T}$ :

Wir konstruieren einen Pfad  $\pi' = d_1, d_2, \dots, d_l, d_{l+1}, \dots, d_{\text{first}}, \dots, d_k$  aus  $\pi^*$ .  $\pi'$  entsteht aus  $\pi^*$  durch Streichen von Diagonalen. Es soll gelten, dass zwischen zwei Diagonalen  $d_l$  und  $d_{l+1}$  in  $\pi^*$  keine Diagonale existiert mit Tiefe kleiner als  $d_l$ . Jedoch soll  $d_{l+1}$  eine kleinere Tiefe haben als  $d_l$ . Bis  $d_{\text{first}}$  wird so die Tiefe immer geringer und ab  $d_{\text{first}}$  entsprechend wieder größer.  $\pi'$  entsteht aus einem Durchlauf von  $d_1$  nach  $d_{\text{first}}$ , bei dem die minimale Tiefe festgehalten wird und alle Diagonalen verworfen werden, die tiefer liegen als das bisherige Minimum, entsprechend für den Weg von  $d_k$  nach  $d_{\text{first}}$ .

$$\begin{array}{ccccccc} \pi^* = d_1 & \dots & d_2 & \dots & d_l & \dots & d_{l+1} & \dots & d_{\text{first}} & \dots & d_{l+2} & \dots & d_k \\ \xrightarrow{\text{tiefer als}} & & \xrightarrow{\text{tiefer als}} & & \xrightarrow{\text{tiefer als}} & & \xrightarrow{\text{tiefer als}} & & \xleftarrow{\text{kleinste Tiefe}} & & \xleftarrow{\text{tiefer als}} & & \xleftarrow{\text{tiefer als}} \end{array}$$

Im Beispiel ergibt sich also für den Weg von Dreieck  $P$  nach Dreieck  $M$  der Pfad  $\pi' = 3.3$   
2.2 1 3.2 4.4.

$\pi'$  besteht bis  $d_{\text{first}}$  aus Kanten mit strikt abnehmender Tiefe, ab  $d_{\text{first}}$  aus Kanten mit strikt zunehmender Tiefe, also können nur  $O(\log n)$  viele Kanten in  $\pi'$  liegen.

**Beh.:** Im Baum  $\hat{T}$  ist  $d_{i+1}$  Vorfahre von  $d_i$ , und im Graphen  $\hat{G}$  sind  $d_i, d_{i+1}$  mit einer Kante verbunden.

**Bew.:**

Unmittelbar nach dem Einfügen von  $d_i$  in die hierarchische Triangulation sind  $d_i$  und  $d_{i+1}$  längs  $\pi'$  direkt benachbart, denn die Diagonalen auf  $\pi'$  zwischen  $d_i$  und  $d_{i+1}$  sind noch nicht eingefügt

$\Rightarrow d_i \subseteq P(d_{i+1})$

$\Rightarrow$  in  $\hat{T}$  ist  $d_i$  im Teilbaum mit Wurzel  $d_{i+1}$  enthalten, d. h.  $d_{i+1}$

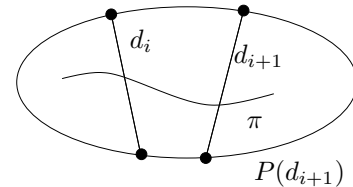
ist Vorgänger von  $d_i$ .

Außerdem ist  $d_{i+1} \subset \partial P(d_i)$ , also sind  $d_i, d_{i+1}$  in  $\hat{G}$  verbunden.  $\diamond$

(ii) Seien  $d_1$  und  $d_m$  die erste bzw. letzte Diagonale auf dem Weg vom Start zum Ziel in  $T$ . Sei  $d_{\text{first}}$  der tiefste gemeinsame Vorfahre in  $\hat{T}$  von  $d_1$  und  $d_m$ . Dieser läßt sich in Zeit  $O(\log n)$  bestimmen, da  $\hat{T}$  balanciert ist.

Laufe in  $\hat{T}$  von  $d_1$  aufwärts nach  $d_{\text{first}}$  und teste für jeden Knoten mit Diagonale  $d$  auf dem Weg in  $\hat{T}$ , ob  $d$  in der Triangulation  $T$  zwischen  $d_1$  und  $d_{\text{first}}$  liegt. Falls ja, nimm die letzte Kante zu  $d$  in  $\hat{G}$  in den Weg auf (d. h. die Kante vom aktuellen Knoten zu dem Knoten, in dem zuletzt eine Kante aufgenommen wurde). Verfahre ebenso für  $d_m$  bis  $d_{\text{first}}$  rückwärts.

Um diesen Test durchführen zu können, betrachte die Kanten des dualen Graphen  $T^*$  und zeichne einen inneren Knoten als Wurzel aus.<sup>8</sup> Offenbar liegt die Diagonale  $d$  genau dann auf



<sup>7</sup>Nicht zu verwechseln mit dem dualen Graph  $T^*$ !

<sup>8</sup>Zur Erinnerung: In  $T^*$  entsprechen die Kanten den Diagonalen der Triangulation; im Gegensatz zu  $\hat{T}$ , wo die Knoten den Diagonalen entsprechen!



dem Pfad von  $d_1$  nach  $d_{\text{first}}$ , wenn die Kante  $d$  in  $T^*$  mit Wurzel  $w$  Vorgänger von *genau einer* der beiden Kanten  $d_1$  oder  $d_{\text{first}}$  ist. Vergleiche hierzu  $d \in \{j, k, l\}$  in Abbildung 1.25. Die Diagonale  $j$  ist weder Vorgänger von  $d_1$  noch von  $d_{\text{first}}$ . Die Diagonale  $k$  ist Vorgänger von  $d_1$  und  $l$  ist Vorgänger von beiden Diagonalen  $d_1, d_{\text{first}}$ .

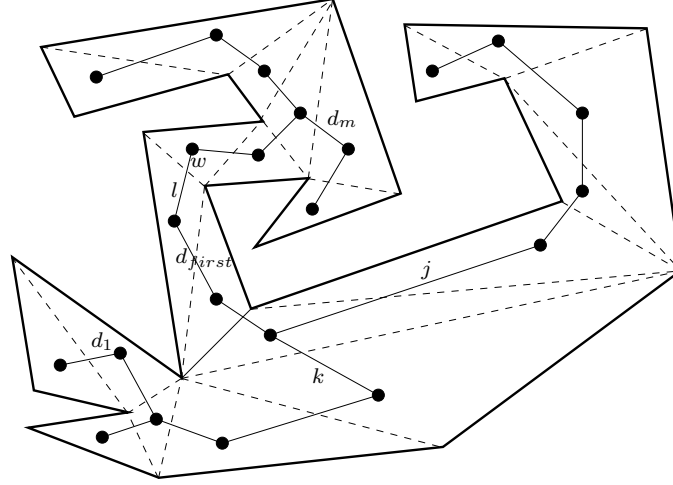


Abbildung 1.25: Diagonalen  $j, k, l, d_1, d_{\text{first}}$  im dualen Baum  $T^*$  mit Wurzel  $w$ .

Identifiziere nun jede Kante  $d$  mit ihrem “unteren”, d. h. von der Wurzel  $w$  weiter entfernten Knoten  $\delta(d)$ .

$$\begin{array}{lll} \delta : & \text{Kanten} & \xrightarrow{\text{bijektiv}} \text{Knoten} \setminus \{w\} \\ \delta : & \text{Kante } d \text{ von } T^* & \mapsto \text{Knoten } \delta(d) \text{ der weiter von } w \text{ entfernt ist.} \end{array}$$

Es gilt:

$$\forall \text{ Kanten } d, d' : d \text{ ist Vorgänger von } d' \iff \delta(d) \text{ ist Vorgänger von } \delta(d').$$

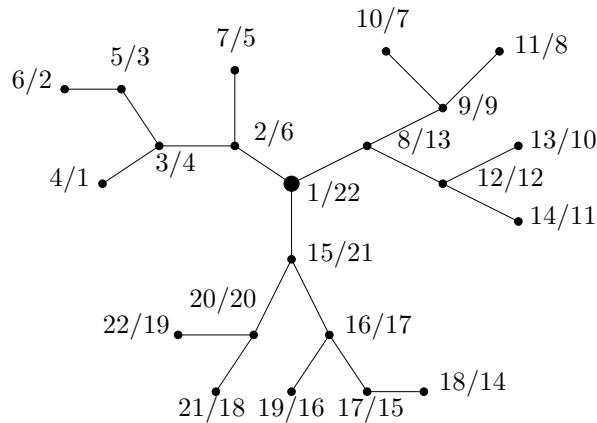


Abbildung 1.26: Prä- und Postorderdurchlauf.

Damit reduziert sich das Problem auf folgende Frage: Wie entscheidet man schnell für zwei Knoten  $a, b$  eines Baumes  $T^*$  mit Wurzel  $w$ , ob  $a$  Vorgänger von  $b$  ist. Dies kann wie in Algorithmus 1.5 beschrieben nach einem Preprocessing in Zeit  $O(n)$  für jedes Knotenpaar in Zeit  $O(1)$  entschieden werden.

(iii) Beim Einfügen berandet jede Diagonale zu jedem Zeitpunkt genau zwei Teilpolygone. Daraus folgt, daß von jedem Knoten in  $\hat{G}$  maximal zwei Kanten zu jeder tieferen Schicht führen. Weiter

**Algorithmus 1.5** Schneller Test auf Vorgängerrelation im Baum**Preprocessing:**

Durchlaufe  $T^*$  einmal in Präorder und einmal in Postorder und beschrifte jeden Knoten  $a$  mit seinen Werten  $\text{pre}(a)$  und  $\text{post}(a)$ .

$O(n)$

**Nun gilt:**

$$a \text{ ist Vorgänger von } b \iff \text{pre}(a) < \text{pre}(b) \wedge \text{post}(a) > \text{post}(b)$$

ist die Anzahl der Schichten unterhalb von  $d$  gleich der Höhe  $h$  von  $d$  in  $\hat{T}$ . Der Teilbaum von  $d$  in  $\hat{T}$  enthält wegen der Balancierung mindestens  $\left(\frac{2}{3}\right)^h$  Blätter (Übungsaufgabe). Da die Teilbäume disjunkt sind, kann es höchstens  $\frac{n}{\left(\frac{2}{3}\right)^h} = \left(\frac{3}{2}\right)^h$  solcher Knoten der Höhe  $h$  geben. Also gilt:

$$|\hat{G}| \leq 2 \sum_d \text{Höhe von } d \leq 2 \sum_{h=1}^{\log_{\frac{3}{2}} n} h \left(\frac{2}{3}\right)^h n \in O(n),$$

da die Reihe  $\sum_h hX^h$  für  $X < 1$  den Konvergenzradius 1 hat. □

Wir wissen nun, daß jeder Pfad in  $P$  zwischen zwei Punkten nur eine logarithmische Anzahl von Subpolygonen passiert. Anstatt Wege der Länge  $O(n)$  im dualen Graphen der Triangulation zu suchen, können wir also Wege der Länge  $O(\log n)$  in  $\hat{G}$  betrachten und daraus mit Hilfe vorab berechneter Informationen den kürzesten Weg ableiten.

Dazu benutzen wir “Sanduhren”, eine Verallgemeinerung der Trichter (Abbildung 1.21) ohne festen Startpunkt, als Struktur zur Repräsentation aller kürzesten Wege zwischen zwei Diagonalen. Mit jeder Kante in  $\hat{G}$  ist eine Sanduhr verbunden. Durch Konkatenation der Sanduhren aller Diagonalen entlang eines Weges  $\pi'$  in  $\hat{G}$  erhält man eine Repräsentation aller kürzesten Wege zwischen den gewünschten Diagonalen.

Eine Sanduhr zwischen zwei Diagonalen  $d_1 = \overline{ab}$  und  $d_2 = \overline{cd}$  —  $a, c, d, b$  seien Eckpunkte von  $P$  im Uhrzeigersinn — wird von den beiden Diagonalen und den kürzesten Pfaden  $\pi(a, c)$  und  $\pi(b, d)$  berandet, vgl. Abbildung 1.27. Wir nennen eine Sanduhr *geschlossen*, falls  $\pi(a, c)$  und  $\pi(b, d)$  gemeinsame Segmente haben. In diesem Fall besteht die Sanduhr aus zwei Trichtern mit einer polygonalen Kette zwischen den Trichteröffnungen  $e$  und  $f$ . Ansonsten nennen wir die Sanduhr *offen*. Offene Sanduhren werden von zwei konvexen, polygonalen Ketten berandet, und die Diagonalen  $d_1$  und  $d_2$  sind gegenseitig sichtbar, d. h. es existiert ein Liniensegment  $\overline{gh}$  mit  $g \in d_1$  und  $h \in d_2$ .

Über die Komplexität der Sanduhren gibt folgendes Lemma Auskunft:

**Lemma 1.14** Die Gesamtkomplexität aller Sanduhren liegt in  $O(n)$ .

**Beweis.** Die Komplexität einer Sanduhr  $S(d_i, d_j)$  hängt von der Höhe  $h$  der in  $\hat{T}$  tieferen Diagonalen ab. Sei  $d_j$  die tiefere Diagonale, dann läßt sich die Komplexität der Sanduhr  $S(d_i, d_j)$  durch die Anzahl der Knoten unterhalb  $d_j$  abschätzen, also durch  $(\text{Höhe von } d_j)^2$ . Da wir beim Einfügen der Diagonalen  $d_j$  den Knoten  $d_j$  in  $\hat{G}$  mit den benachbarten Diagonalen verbinden, gibt es 2 Sanduhren, in denen  $d_j$  der tiefere Knoten ist. Analog zu Lemma 1.13(iii) gibt es  $\left(\frac{2}{3}\right)^h$  Knoten der Höhe  $h$ . Insgesamt erhalten wir:

$$2 \sum_{\text{Diagonale } d} (\text{Höhe von } d)^2 \leq 2 \sum_{h=1}^{\log_{\frac{3}{2}} n} (\# \text{Knoten der Höhe } h) \cdot h^2$$

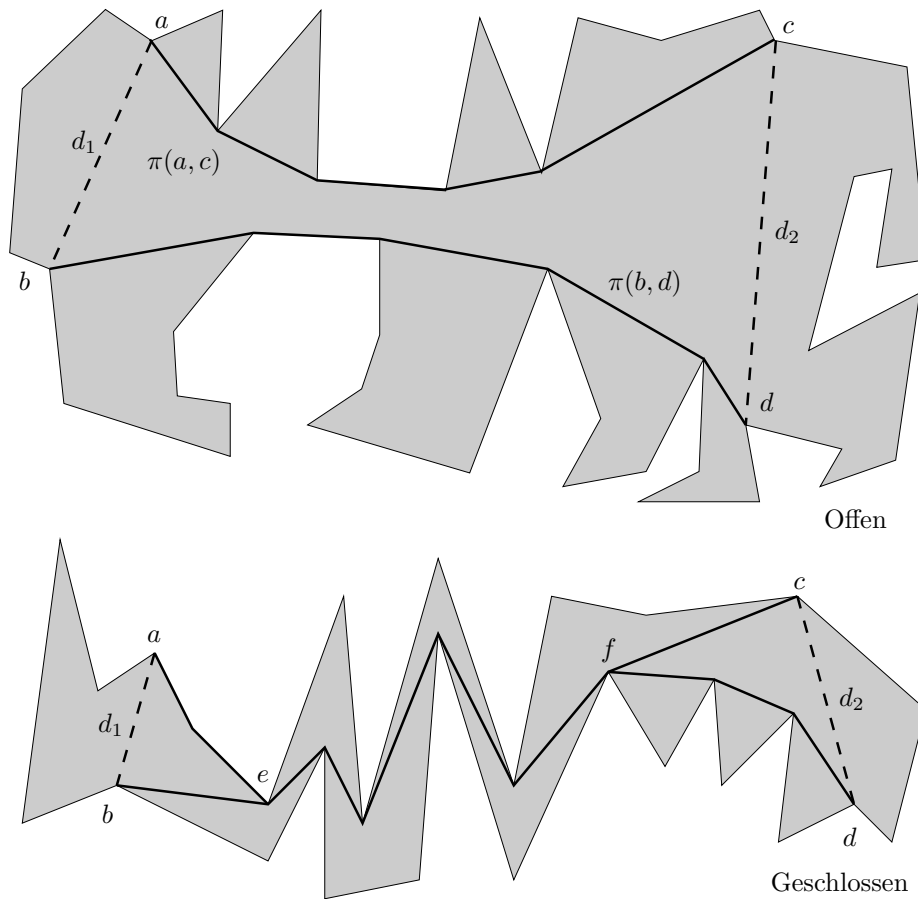


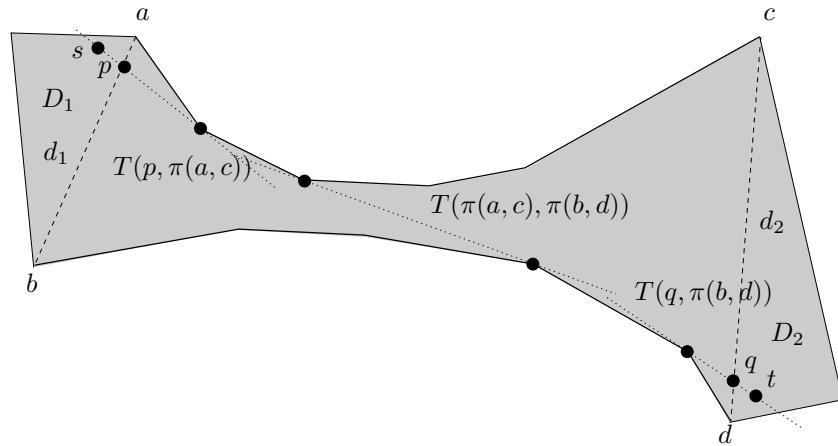
Abbildung 1.27: Offene und geschlossene Sanduhr zwischen  $d_1 = \overline{ab}$  und  $d_2 = \overline{cd}$ .

$$\leq 2n \sum_{h=1}^{\log_3 n} \left(\frac{2}{3}\right)^h \cdot h^2 = 2nC \in O(n)$$

□

Für den kürzesten Pfad zwischen zwei gegebenen Anfragepunkten  $p \in d_1$  und  $q \in d_2$  sind folgende Fälle zu betrachten:

- Im geschlossenen Fall besteht  $\pi(p, q)$  aus der Konkatination der Kürzesten  $\pi(p, e)$ , der polygonalen Kette  $\pi(e, f)$  und  $\pi(f, q)$ .
- Ist die Sanduhr offen, sind folgende Fälle zu betrachten:
  - Sind  $p$  und  $q$  gegenseitig sichtbar, so ist  $\pi(p, q) = \overline{pq}$ .
  - Andernfalls betrachte  $\mathcal{L} = \{\overline{rs} \mid r \in d_1, s \in d_2\}$ . Wenn ein  $\ell \in \mathcal{L}$  existiert, so daß  $p$  und  $q$  auf derselben Seite von  $\ell$  liegen (o. E. auf der Seite von  $b$  und  $d$ ), so besteht  $\pi(p, q)$  aus der Tangente von  $p$  an  $\pi(b, d)$ , der Tangente von  $q$  an  $\pi(b, d)$  und dem Stück von  $\pi(b, d)$ , das beide Tangentenendpunkte verbindet. Die Menge  $\mathcal{L}$  dient dabei nur der Anschauung, natürlich können nicht alle Liniensegmente explizit berechnet werden. Die Lage von  $p$  und  $q$  läßt sich auch durch sukzessive Tangentenbildung bestimmen.
  - Wenn jedes  $\ell \in \mathcal{L}$  zwischen  $p$  und  $q$  liegt — o. E. liege  $p$  auf der Seite von  $a$ ,  $q$  auf der Seite von  $d$  — setzt sich  $\pi(p, q)$  zusammen aus der Tangente von  $p$  an  $\pi(a, c)$ , der Tangente von  $q$  an  $\pi(b, d)$ , der gemeinsamen Tangente zwischen  $\pi(a, c)$  und  $\pi(b, d)$

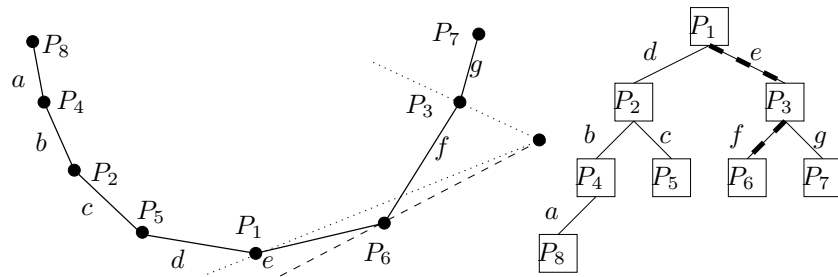
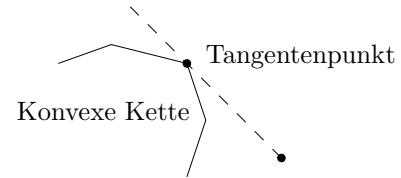


Abbildungung 1.28: Kürzester Pfad in offener Sanduhr.

und den Teilen von  $\pi(a, c)$  und  $\pi(b, d)$ , die die Tangentenendpunkte verbinden, siehe Abbildung 1.28.

Analog lassen sich kürzeste Wege zwischen zwei Punkten  $s \in D_1$  und  $t \in D_2$  finden, die in den  $d_1$  und  $d_2$  aufgesetzten Dreiecken  $D_1$  und  $D_2$  liegen.

Damit reduziert sich die Bestimmung kürzester Wege zwischen Punkten in  $d_1$  und  $d_2$  oder in den angrenzenden Dreiecken auf die Berechnung eines Tangentenpunktes auf konvexen Ketten. Dies läßt sich mit binärer Suche in Zeit  $O(\log n)$  lösen; ein Verfahren, das auch bei der Konstruktion der konvexen Hülle benutzt wird, siehe Abbildung 1.29.



Abbildungung 1.29: Speicherung einer Sanduhr in einem binären Baum.

Eine entscheidende Eigenschaft der Sanduhren ist, daß sie sich in logarithmischer Zeit konkatenieren lassen. Dabei sind einige Fallunterscheidungen zu treffen und Details zu beachten, z. B. kann die Konkatenation zweier offener Sanduhren geschlossen sein. Für Details sei hier auf die Arbeit von Guibas und Hershberger [GH87] verwiesen. Es gilt:

**Lemma 1.15** Eine Sanduhr zwischen zwei Diagonalen  $d_i$  und  $d_j$  werde von  $m_{ij}$  Dreiecken der Triangulation überdeckt. Es existiert eine Datenstruktur zur Speicherung der Sanduhren  $S(d_i, d_j)$ , die folgende Operationen unterstützt:

- (i) Entfernung zwischen zwei Punkten in  $D_i$  und  $D_j$  in Zeit  $O(\log m_{ij})$ .
- (ii) Kürzeste zwischen zwei Punkten in  $D_i$  und  $D_j$  in Zeit  $O(\log m_{ij} + k)$ .
- (iii) Zusammensetzen zweier Sanduhren  $S(d_i, d_j)$  und  $S(d_j, d_\ell)$  zu einer Sanduhr  $S(d_i, d_\ell)$  in Zeit und zusätzlichem Platz  $O(\log m_{ij} + \log m_{j\ell})$ .

Zusammenfassend erhalten wir Algorithmus 1.6 und folgendes Theorem:

**Theorem 1.16** (Guibas, Hershberger, 1987)

Zu einem Polygon  $P$  mit  $n$  Ecken kann nach Triangulation und Vorbereitungszeit  $O(n)$  zwischen zwei beliebigen Punkten  $a$  und  $b$  im Inneren von  $P$  die Entfernung in Zeit  $O(\log n)$  und ein kürzester Pfad in Zeit  $O(\log n + k)$  bestimmt werden, wobei  $k$  die Anzahl der Segmente auf dem kürzesten Pfad ist. Der Speicherbedarf liegt in  $O(n)$ . [GH87, GH89]

Eine einfachere Datenstruktur für Sanduhren findet sich in [Her91]. Offen ist die Frage, ob sich kürzeste Wege im Polygon in Zeit  $O(n)$  ohne Triangulation berechnen lassen.

---

**Algorithmus 1.6** Kürzeste Pfade in einem Polygon (Guibas, Hershberger)

---

**Vorbereitungsphase:**

- Berechne Triangulation  $T$ , dualen Graphen  $T^*$ , hierarchische Triangulation  $\hat{T}$  und Schichtengraph  $\hat{G}$ .
- Bestimme Prä- und Postorderbeschriftungen der Knoten in  $T^*$ .
- Berechne für jede Kante  $(d_i, d_j)$  in  $\hat{G}$  die Sanduhr  $S(d_i, d_j)$ .
- Baue eine Suchstruktur auf, um Punkte schnell in den Dreiecken lokalisieren zu können (Point-Location).
- Bereite den Graphen  $\hat{G}$  für Pfadsuche vor.

Zeit und Platz  $O(n)$ **Anfrage für Punkte  $p, q$ :**

- Bestimme die Dreiecke  $D_p$  und  $D_q$ .  $O(\log n)$
  - Bestimme in  $\hat{G}$  den Pfad zwischen den Diagonalen  $d_p$  und  $d_q$ :
    - Finde tiefsten gemeinsamen Vorfahren  $d_{\text{first}}$
    - Betrachte Pfad  $\pi_{\hat{T}}$  von  $D_p$  nach  $D_q$  in  $\hat{T}$ :  
 Teste für jede Diagonale  $d$  in  $\pi_{\hat{T}}$ , ob  $d$  zwischen  $D_p$  und  $D_q$  in  $T$  liegt. Dies ist dann der Fall, wenn  $d$  Vorgänger von *entweder*  $D_p$  *oder*  $D_q$  in  $T^*$  ist.
  - Liefert Pfad  $\pi_{\hat{G}}$  mit Länge  $O(\log n)$  in Zeit  $O(\log n)$
  - Konstruiere  $S(d_p, d_q)$  aus den Sanduhren längs des Pfades  $\pi_{\hat{G}}$  von  $D_p$  nach  $D_q$  im Graphen  $\hat{G}$ .  $O(\log^2 n)$   
 Durch Berechnung zusätzlicher Sanduhren im voraus in Zeit und Platz  $O(n)$  auch dies auch möglich in Zeit  $O(\log n)$
  - Bestimme die Kürzeste von  $p$  nach  $q$  mit Hilfe von  $S(d_p, d_q)$ .  $O(\log n + k)$
-

### 1.2.3 Anwendung: Berechnung des geodätischen Durchmessers

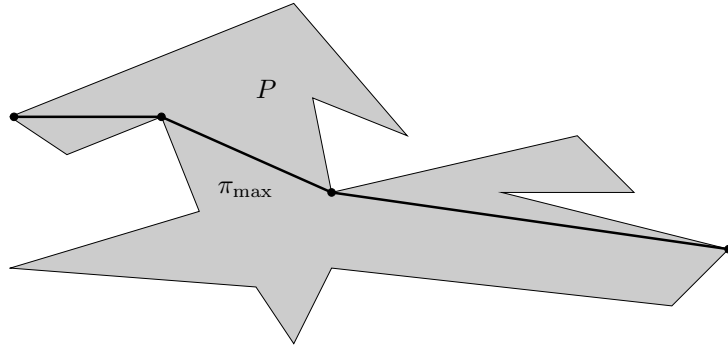


Abbildung 1.30: Ein Polygon und sein geodätischer Durchmesser.

Eine Anwendung von kürzesten Pfaden im Inneren eines Polygons ist die Bestimmung des geodätischen Durchmessers, der im Bereich der Mustererkennung eine Rolle spielt; Abbildung 1.30 zeigt ein Polygon mit seinem geodätischem Durchmesser.

**Definition 1.17** Der geodätische Durchmesser  $\pi_{\max}$  eines Polygons  $P$  ist der längste kürzeste Pfad zwischen zwei Ecken im Inneren von  $P$ :

$$\pi_{\max} := \max_{p_i, p_j \text{ Ecken von } P} d(p_i, p_j).$$

Dabei bezeichnet  $d(p_i, p_j)$  den geodätischen Abstand von  $p_i$  und  $p_j$ : die Länge des kürzesten Pfades von  $p_i$  nach  $p_j$ .

Ein naiver Ansatz wäre, für jedes Paar  $(p_i, p_j)$  den geodätischen Abstand  $d(p_i, p_j)$  nach Theorem 1.16 in Zeit  $O(\log n)$  und Vorbereitungszeit  $O(n)$  zu berechnen und dann das Maximum dieser Abstände zu bestimmen; damit könnten wir den geodätischen Durchmesser in Zeit  $O(n^2 \log n)$  berechnen.

Auch hier läßt sich mehr erreichen, die folgenden beiden Ideen gehen auf Aggarwal, Klawe, Moran, Shor und Wilber (1986) zurück. [AKM<sup>+</sup>87a]

#### 1. Idee:

Nutze aus, daß die paarweisen Abstände  $d(p_i, p_j)$  nicht voneinander unabhängig sind. Konkret gilt für alle Ecken  $p_i, p_j, p_\ell, p_m$ , die in dieser Reihenfolge auf dem Rand des Polygons liegen:

$$d(p_i, p_m) + d(p_j, p_\ell) < d(p_i, p_\ell) + d(p_j, p_m).$$

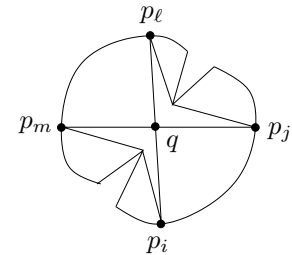
Dies folgt aus den Dreiecksungleichungen

$$d(p_i, p_m) \leq d(p_i, q) + d(q, p_m) \text{ und}$$

$$d(p_j, p_\ell) \leq d(p_j, q) + d(q, p_\ell) \text{ mit}$$

$$d(p_j, p_m) = d(p_j, q) + d(q, p_m) \text{ und}$$

$$d(p_i, p_\ell) = d(p_i, q) + d(q, p_\ell).$$



Seien  $p_1, p_2, \dots, p_n$  die Ecken des Polygons gegen den Uhrzeigersinn. Definiere eine  $n \times n$  Matrix  $A = (a_{i\ell})$  durch:

$$a_{i\ell} = \begin{cases} \ell - n & \text{für } 1 \leq \ell \leq i \\ d(i, \ell) & \text{für } i + 1 \leq \ell \leq n \end{cases}$$

Dabei sei  $d(i, j) := d(p_i, p_j)$ . Die Matrix  $A$  besteht im oberen Dreieck aus Pfadlängen, im unteren Dreieck aus negativem Füllwerk.

$$\begin{array}{c} \begin{array}{cccccc} & 1 & 2 & 3 & & n-1 & n \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ \vdots \\ n-1 \\ n \end{array} & \left( \begin{array}{cccccc} 1-n & \boxed{d(1,2)} & d(1,3) & \dots & d(1,n-1) & d(1,n) \\ 1-n & 2-n & \boxed{d(2,3)} & \dots & d(2,n-1) & d(2,n) \\ 1-n & 2-n & 3-n & \dots & d(3,n-1) & d(3,n) \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 1-n & 2-n & 3-n & \dots & \dots & \boxed{d(n-1,n)} \\ 1-n & 2-n & 3-n & \dots & \dots & 0 \end{array} \right) \end{array}$$

**Definition 1.18** Eine  $n \times m$ -Matrix  $A = (a_{i\ell})$  heißt **monoton**, falls

$$\forall 1 \leq i < j \leq n, 1 \leq k < \ell \leq m : (a_{ik} < a_{i\ell} \Rightarrow a_{jk} < a_{j\ell}).$$

Anschaulich:

$$\begin{array}{c} \begin{array}{cc} & k & \ell \\ i & \left( \begin{array}{cc} a_{ik} < a_{i\ell} \\ \Downarrow \\ a_{jk} < a_{j\ell} \end{array} \right) \\ j & \end{array} \end{array}$$

**Bemerkung.** Sei  $A$  eine monotone Matrix. Dann gilt: je tiefer die Zeile, desto weiter rechts liegt das linkeste Zeilenmaximum.

So kann in folgender monotonen Matrix das Maximum der dritten Zeile nicht links von der vierten Spalte stehen — dem Maximum der zweiten Zeile.

$$\begin{pmatrix} 5 & 4 & 3 & 2 & 1 & 0 \\ 1 & 7 & 3 & \boxed{9} & 4 & 8 \\ 4 & 6 & 5 & \boxed{8} & 8 & 9 \end{pmatrix}$$

**Lemma 1.19** Die oben definierte Matrix  $A$  ist *monoton*.

**Beweis.**

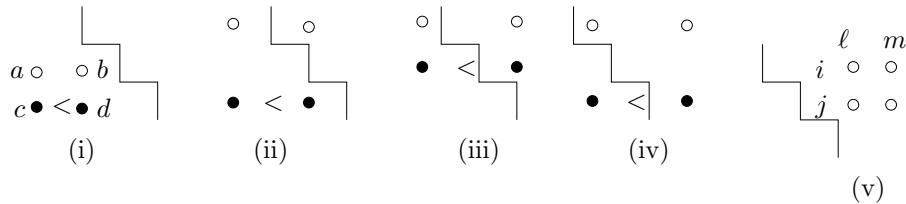


Abbildung 1.31: Fallunterscheidung nach Lage von  $A$  in  $T$ .

Sei  $T = \begin{pmatrix} a_{i\ell} & a_{im} \\ a_{j\ell} & a_{jm} \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$  mit  $i < j, \ell < m$  eine Untermatrix von  $A$ . Je nach Lage von  $T$  in  $A$  unterscheiden wir folgende Fälle, siehe Abbildung 1.31: In Fall (i) liegt  $T$  unterhalb der Treppe, in diesem Fall — und ebenso in den Fällen (ii) bis (iv) gilt  $c < d$ .

Als einzig interessanter Fall bleibt (v). In diesem Fall folgt aus der Definition der Matrix  $A$ :  $i < j < \ell < m$ . Die Ungleichung  $j < \ell$  gilt, da sich  $a_{i\ell}$  oberhalb der Diagonalen der Matrix befindet. Insgesamt folgt aus den Annahmen  $i < j, \ell < m$ , dass  $1 \leq i < j < \ell < m \leq n$  gilt und sich somit die Punkte  $i, j, \ell, m$  in dieser Reihenfolge auf dem Rand des Polygons befinden.

Für die Knoten auf dem Rand des Polygons gilt deshalb  $d(i, m) + d(j, \ell) \leq d(i, \ell) + d(j, m)$ , aus  $d(i, \ell) < d(i, m)$  folgt also  $d(j, \ell) < d(j, m)$ . □

## 2. Idee:

**Theorem 1.20** *Sei  $A$  eine monotone  $n \times m$ -Matrix,  $n \leq m$ . Dann läßt sich das maximale Element in Zeit  $O(m)$ .*

**Beweis.** Berechne mit Algorithmus 1.8 für jede Zeile  $i$  das am weitesten links stehende Maximum  $\max(i)$ . Die Anzahl der Vergleiche in Algorithmus 1.7 ist dabei beschränkt durch das Maximum der Anzahl Zeilen (Anzahl der Elemente auf der Diagonalen) und dem der Anzahl der Spalten (maximale Anzahl der Spaltenstreichungen), also  $m$ .

Damit ist nur noch die Laufzeit von Algorithmus 1.8 zu bestimmen: Sei  $T(n)$  die Anzahl der Zugriffe auf Matrixelemente der Prozedur Zeilenmaxima. Sei  $n = 2^k$ :

$$\begin{aligned}
 T(n) &\leq T\left(\frac{n}{2}\right) + C \cdot n + O(m) \\
 &\leq T\left(\frac{n}{4}\right) + C \cdot \frac{n}{2} + C \cdot n + O(m) \\
 &\leq \dots \\
 &\leq T(1) + C \left( \frac{1}{2^{k-1}} + \frac{1}{2^{k-2}} + \dots + \frac{1}{2} + 1 \right) \cdot n + O(m) \\
 &\in O(n + m) \\
 &\in O(m)
 \end{aligned}$$

□

Bemerkung: Im ersten Rekursionsschritt entstehen  $O(m)$  Kosten für die Spaltenreduktion zu einer quadratischen Matrix. In den weiteren Rekursionsschritten sind die Kosten der Spaltenreduktion in der Konstante  $C$  enthalten. Analog entstehen am Ende einmalig zusätzlich  $O(m)$  Kosten für die Berechnung der Maxima der ungeraden Zeilen.

Da wir nach Theorem 1.20 die Zeilenmaxima in Zeit  $O(m)$  bestimmen können, folgt direkt, dass wir nur  $O(m)$  Matrixelemente miteinander vergleichen müssen.

Mit Theorem 1.20 können wir zu einer *gegebenen* Matrix schnell das maximale Element ausrechnen. Allerdings würde es uns schon Zeit  $O(n^2)$  kosten, die Matrix  $A$  aufzustellen! Hier hilft uns die Tatsache, daß wir gar nicht alle  $O(n^2)$  Einträge der Matrix brauchen, sondern mit  $O(n)$  Vergleichen auskommen. Wir berechnen die Matrix  $A$  also nicht explizit, sondern die benötigten Matrixelemente erst dann, wenn sie gebraucht werden. Damit gilt:

**Theorem 1.21** *Der geodätische Durchmesser eines einfachen Polygons mit  $n$  Ecken läßt sich in Zeit  $O(n \log n)$  bestimmen. Es läßt sich sogar für jede Ecke der am weitesten entfernte Partner in dieser Zeit feststellen.* [AKM<sup>+</sup>87a]

Durch bessere Kombination von Matrixsuche und Kürzestenbestimmung ist dies sogar in Zeit  $O(n)$  möglich (Hershberger, Suri, 1993, [HS93, HS97]).



---

**Algorithmus 1.7** Spaltenreduktion

---

**Input:** monotone  $n \times m$ -Matrix  $A$ ,  $n \leq m$ .**Output:** monotone  $n \times n$ -Matrix  $A'$  mit denselben Zeilenmaxima wie  $A$ ; aus der Position von  $\max(i)$  in  $A'$  ist  $\max(i)$  in  $A$  rekonstruierbar.

Beginnend mit  $a_{1,1}$  teste, ob  $a_{i,i} < a_{i,i+1}$  ist. Ist dies der Fall, streiche die  $i$ -te Spalte, da die Elemente der  $i$ -ten Spalte wegen der Monotonieeigenschaft keine Zeilenmaxima mehr sein können. Vergleiche dann  $a_{i-1,i-1}$  mit  $a'_{i-1,i} = a_{i-1,i+1}$  usw. Ist  $a_{i,i} \geq a_{i,i+1}$ , inkrementiere  $i$ ; vergleiche also als nächstes  $a_{i+1,i+1}$  mit  $a_{i+1,i+2}$ .

Lautet die Antwort für alle Elemente auf der Diagonalen “nein”, so sind alle Elemente der  $n$ -ten Spalte größer gleich den Elementen der Spalte  $(n+1)$ , also können wir die  $(n+1)$ -ten Spalte streichen und mit dem Test  $a_{n,n} \stackrel{?}{<} a_{n,n+2}$  fortfahren.

Als Output erhalten wir eine quadratische Matrix mit der Eigenschaft, dass die Elemente rechts von der Diagonalen in jeder Zeile eine monoton fallende Folge bilden, d.h.  $a_{i,j} \geq a_{i,j+1} \forall j \geq i$ .

---



---

**Algorithmus 1.8** Zeilenmaxima

---

**Input:** monotone  $n \times m$ -Matrix  $B$ ,  $n \leq m$ **Output:** alle linkesten Zeilenmaxima  $\max(i), 1 \leq i \leq n$ 

- $C :=$  alle Zeilen von  $B$  mit geradem Zeilenindex;  $O(n)$
  - $C' := \text{Spaltenreduktion}(C)$ ;  $O(m)$
  - $\text{Zeilenmaxima}(C')$ ;     /\*  $C'$  ist  $\frac{n}{2} \times \frac{n}{2}$ -Matrix \*/  $T(\frac{n}{2})$
  - Rekonstruiere Zeilenmaxima von  $C$ ;     /\* = gerade Zeilen von  $B$  \*/  $O(n)$
  - Berechne die Maxima der ungeraden Zeilen von  $B$ :  
Wegen Bemerkung nach Definition 1.18 liegt  $\max(2i+1)$  zwischen den Spalten, in denen  $\max(2i)$  und  $\max(2i+2)$  liegen. Da sich die Intervalle, in denen nach den Maxima der ungeraden Zeilen gesucht werden muß, nicht überlappen, ist dies möglich in Zeit  $O(m)$
-

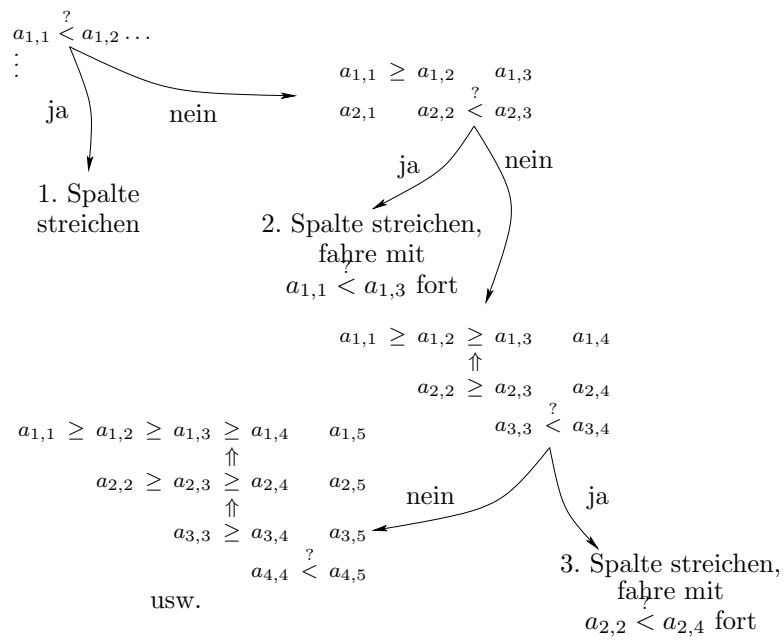


Abbildung 1.32: Spaltenreduktion.

### 1.2.4 Shortest Watchman Routes

Eine andere Anwendung von kürzesten Pfaden innerhalb eines Polygons ist die Berechnung der kürzesten Wächterroute oder Shortest Watchman Route (SWR). Gesucht ist der kürzeste Weg in einem Polygon, den ein Wachmann oder Roboter mit unbeschränkter Sicht gehen kann und auf dem Weg jeden Punkt in dem Polygon mindestens einmal sieht.<sup>9</sup> Eine Route ist stets ein geschlossener Weg, d. h. der Roboter kehrt zum Startpunkt zurück. Bei der einfacheren Variante des Shortest Watchman Problems ist der Startpunkt des Roboters und damit der Route gegeben, bei der schwierigeren Variante sucht man die kürzeste Route, die in einem beliebigen Punkt des Polygons startet. Ebenfalls schwieriger zu berechnen als eine SWR ist eine Shortest Watchman Tour, also ein Pfad, der jeden Punkt im Polygon einmal sieht, jedoch nicht zum Startpunkt zurückkehrt. Das Shortest Watchman Problem wurde 1986 von Chin und Ntafos vorgestellt [CN86]. Seitdem wurden zahlreiche Arbeiten zur Berechnung der SWR veröffentlicht, von denen sich jedoch viele als fehlerhaft herausgestellt haben oder bzgl. der Laufzeit verbessert wurden, siehe z. B. [CN91, THI93, TH93, HN97, THI99, CJS99, CJ95].

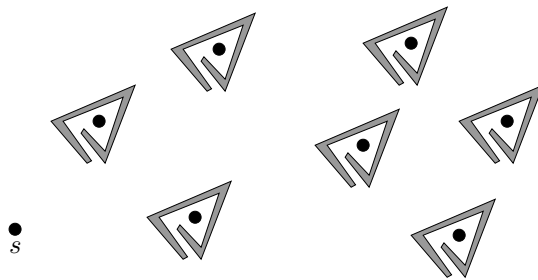


Abbildung 1.33: Die Berechnung einer SWR liefert eine Besuchsreihenfolge der Punkte.

Im allgemeinen ist die Berechnung einer SWR schwierig:<sup>10</sup>

**Theorem 1.22** (Chin, Ntafos, 1986)

Die Berechnung einer Shortest Watchman Route in einer Szene mit polygonalen Hindernissen (Polygon mit Löchern) ist NP-hard. [CN86, CN88]

**Beweis.** In Polygonen mit Löchern läßt sich das Traveling Salesman Problem auf das Shortest Watchman Problem reduzieren: die Bestimmung der kürzesten Besuchsreihenfolge der polygonalen Löcher entspricht dem TSP. Wir umgeben alle Punkte aus der Eingabe für das TSP mit einem Hindernis, das die Sicht auf alle anderen Punkte abschneidet, siehe Abbildung 1.33. Jedes dieser Hindernisse hat  $O(1)$  Kanten und die Konstruktion der gesamten Szene ist in Zeit  $O(n)$  möglich. Eine Lösung für das Shortest Watchman Problem in dieser Szene liefert eine kürzeste Besuchsreihenfolge der Punkte und damit eine Lösung für das TSP.  $\square$

Wir beschränken uns daher auf einfache Polygone, d. h. Polygone ohne Selbstschnitte und Löcher, und auch in dieser Klasse von Polygonen zunächst nur auf zwei Spezialfälle, siehe Abbildung 1.34.

**Definition 1.23** Ein Polygon  $P$  heißt **monoton**, wenn eine Gerade  $\ell$  existiert, so daß für jede zu  $\ell$  orthogonale Gerade  $\ell'$  der Schnitt von  $P$  mit  $\ell'$  zusammenhängend ist, d. h. der Schnitt ist ein Liniensegment, ein Punkt oder leer. Ist die Gerade  $\ell$  eine Parallele zur Y-Achse, so heißt  $P$  **y-monoton**.

$P$  heißt **rechtwinklig**, wenn jeder innere Winkel  $90^\circ$  oder  $270^\circ$  beträgt.

Den einfachsten Fall stellen rechtwinklige, monotone Polygone dar:

<sup>9</sup>Ein Weg  $\pi$  sieht das gesamte Polygon  $P$  : $\Leftrightarrow$  zu jedem Punkt  $p \in P$  existiert ein Punkt  $p' \in \pi$ , so daß das Liniensegment  $\overline{pp'}$  vollständig in  $P$  liegt.

<sup>10</sup>Zu den Begriffen NP-hard und NP-vollständig siehe Anhang.

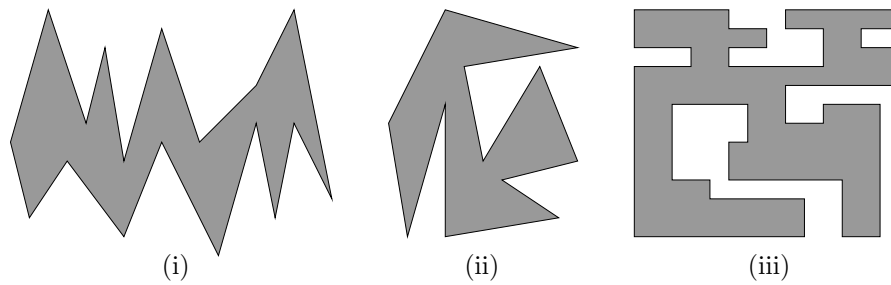


Abbildung 1.34: (i) X-monotones Polygon, (ii) nicht-monotones Polygon, (iii) rechtwinkliges Polygon.

**Theorem 1.24** (Chin, Ntafos, 1986)

In einem rechtwinkligen, monotonen Polygon kann die Shortest Watchman Route in Zeit  $O(n)$  berechnet werden. [CN86, CN88]

**Beweis.** Übungsaufgabe. □

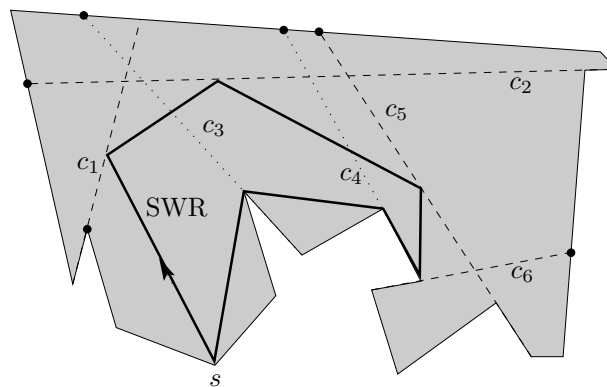


Abbildung 1.35: Polygon mit notwendigen Cuts (gepunktet), wesentlichen Cuts (gestrichelt) und Shortest Watchman Route.

Auch in rechtwinkligen, nicht-monotonen Polygonen kann die SWR recht einfach berechnet werden. Wir brauchen dazu einige Begriffe, die wir gleich für allgemeine — d. h. nicht-rechtwinklige — Polygone einführen.

**Definition 1.25** Die Verlängerungen der Kanten von reflexen Ecken<sup>11</sup> in das Polygon hinein sind die **Cuts** des Polygons. Von den beiden Cuts einer reflexen Ecke ist nur einer interessant, nämlich derjenige, der bzgl. des Startpunktes  $s$  die Sicht blockiert.<sup>12</sup> Wir nennen diese Cuts **notwendige Cuts**. Ein Cut  $c_1$  **dominiert** einen Cut  $c_2$ , wenn jeder Weg von  $s$  zu  $c_1$  den Cut  $c_2$  schneidet. Notwendige Cuts, die von keinem anderen notwendigen Cut dominiert werden, heißen **wesentliche Cuts**.

Ein Cut  $c_i$  teilt das Polygon in zwei Teile. Die zu  $c_i$  gehörende **Tasche**  $P_{c_i}$  ist der Teil, der den Startpunkt  $s$  nicht enthält. Damit können wir definieren — und man kann leicht sehen, daß beide Definitionen äquivalent sind —, daß ein Cut  $c_1$  einen anderen Cut  $c_2$  dominiert, wenn  $P_{c_1} \subset P_{c_2}$  gilt.

Cuts, die von anderen dominiert werden, werden sozusagen nebenbei exploriert und müssen nicht weiter berücksichtigt werden. Abbildung 1.35 zeigt ein Beispiel für ein Polygon mit seinen notwendigen und wesentlichen Cuts.  $c_3$  und  $c_4$  sind nicht wesentlich, da jeder Weg zu  $c_5$  über diese Cuts führt.  $c_5$  dominiert  $c_3$  und  $c_4$ . Die SWR (und jede andere Explorationstour) berührt

<sup>11</sup>Polygonecke, deren Innenwinkel  $> \pi$  ist.

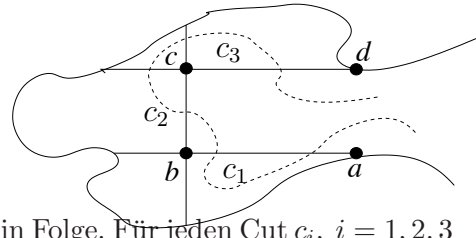
<sup>12</sup>Anschaulich der Cut, den wir überschreiten müssen, um hinter die Ecke schauen zu können.

alle wesentlichen Cuts und die Menge der wesentlichen Cuts ist die kleinste Menge von Cuts, die besucht werden müssen, um das ganze Polygon zu sehen.

In einem Polygon mit gegebenem Startpunkt  $s$  können wir die Cuts entsprechend dem im Uhrzeigersinn ersten Schnittpunkt zwischen Cut und Polygonrand ordnen, siehe Abbildung 1.35. Es gilt:

**Lemma 1.26** *In einem rechtwinkligen Polygon existiert stets eine SWR, die die wesentlichen Cuts gemäß der Ordnung auf dem Rand besucht.*

**Beweis.** Wir zeigen zunächst, dass eine SWR nie in eine Tasche hineinläuft bzw. es wird kein wesentlicher Cut überschritten. Kritisch sind nur Situationen, wo sich wesentliche Cuts schneiden. Dies können bei rechtwinkligen Polygonen maximal vier in Folge sein. Wir betrachten die Situation mit drei Cuts in Folge. Für jeden Cut  $c_i$ ,  $i = 1, 2, 3$  gilt: Wenn man den Weg in der Tasche von  $c_i$  durch die orthogonale Projektion auf  $c_i$  ersetzt, werden die gleichen Cuts besucht wie vorher. Der Weg wird dabei nur kürzer. Daher reicht es, die Cuts  $c'_1 = ab$ ,  $c'_2 = bc$ ,  $c'_3 = cd$  zu besuchen, ohne in die Taschen zu laufen. Eine kürzeste Tour darf keine Selbstschnitte haben, sonst könnte man die Tour lokal abkürzen. Es folgt somit, dass nur durch Besuchen der Cuts entlang der Ordnung auf dem Rand eine schnittfreie und somit kürzeste Tour entsteht.  $\square$




---

**Algorithmus 1.9** Shortest Watchman Route in einfachen Polygonen

---

- Bestimme die wesentlichen Cuts  $c_1, \dots, c_k$ .  $O(n)$
  - Schneide die Teile des Polygons ab, die bzgl. des Startpunktes hinter wesentlichen Cuts liegen.  $O(n)$
  - Trianguliere das resultierende Polygon  $P'$ .  $O(n)$
  - Konstruiere  $P''$  durch "Roll-Out" von  $P'$ :  $O(n)$ 
    - $P^{(1)}$  seien die relevanten Dreiecke von  $P'$  auf dem Weg im dualen Graphen  $T^*$  von  $s$  nach  $c_1$ .
    - Für jeden Cut  $c_i$ ,  $i = 2, \dots, k$ : Erweitere  $P^{(i-1)}$  zu  $P^{(i)}$  durch alle relevanten Dreiecke entlang des Randes von  $P'$  auf dem Weg von  $c_{i-1}$  zu  $c_i$ , gespiegelt an den Cuts  $c_1, \dots, c_{i-1}$  und angesetzt an  $c_{i-1}$ .
    - Erweitere  $P^{(k)}$  zu  $P''$  wie oben durch alle relevanten Dreiecke auf dem Weg von  $c_k$  nach  $s$ . Dabei entsteht eine gespiegelte Kopie  $s'$  des Zielpunktes.
  - Berechne in  $P''$  den kürzesten Weg  $\pi$  von  $s$  nach  $s'$ .  $O(n)$
  - Die SWR entsteht durch "zurückfalten" der Liniensegmente in  $\pi$ .
- 

Lemma 1.26 liefert uns den Algorithmus für die Berechnung einer SWR:

**Theorem 1.27** (*Chin, Ntafos, 1986*)

*In einem rechtwinkligen Polygon kann die Shortest Watchman Route in Zeit  $O(n)$  berechnet werden.* [CN86, CN88]

**Beweis.** Algorithmus 1.9 berechnet die SWR in einem rechtwinkligen Polygon. Abbildung 1.36 zeigt ein Beispiel.

Die wesentlichen Cuts lassen sich in Zeit  $O(n)$  berechnen (ohne Beweis). Damit bleibt zu zeigen, daß  $P''$  nicht mehr als  $O(n)$  Kanten enthält. Dazu betrachten wir den zur Triangulation dualen Graphen  $T^*$ . Dem sukzessiven Besuch der Cuts  $c_1, \dots, c_n$  entspricht ein Depth-First Durchlauf von  $T^*$ . Jede Kante von  $T^*$  wird dabei zweimal durchlaufen. Damit wird auch jedes Dreieck der Triangulation maximal zweimal in  $P''$  übernommen und die Anzahl der Kanten von  $P''$  liegt in  $O(n)$ .  $\square$

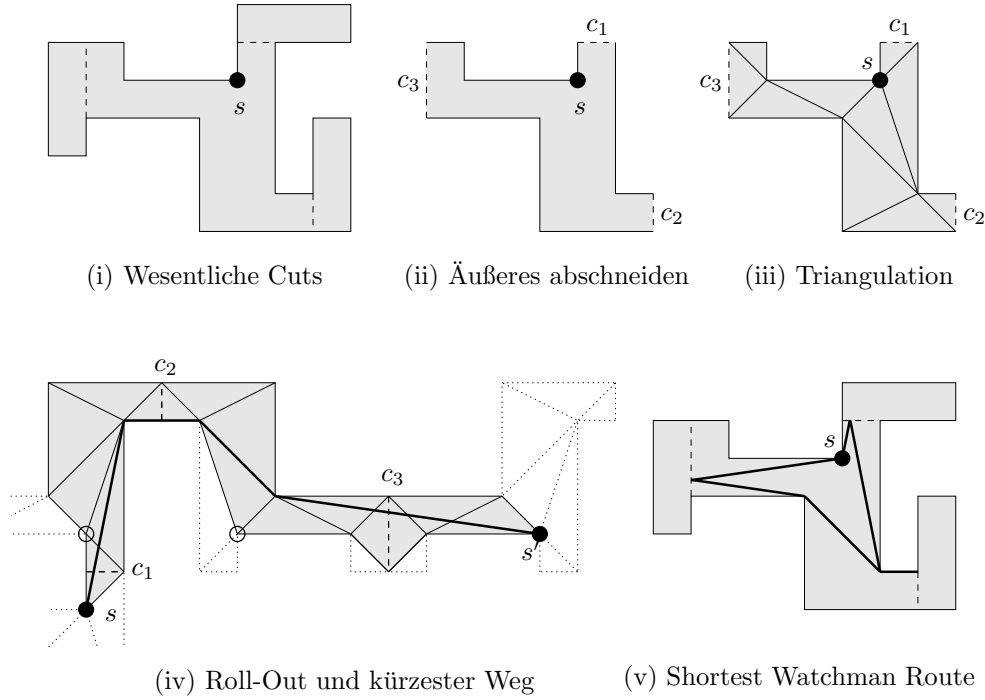


Abbildung 1.36: Berechnung der SWR in einem rechtwinkligen Polygon.

Algorithmus 1.9 wäre mit leichten Modifikationen bei gleicher Laufzeit in Polygonen anwendbar, in denen sich je maximal zwei wesentliche Cuts gegenseitig schneiden, da auch Polygone dieser Art Lemma 1.26 erfüllen. In allgemeinen Polygonen können sich allerdings mehr als zwei Cuts gegenseitig schneiden. Wir nennen solche Mengen von Cuts eine “Corner”. In solchen Situationen ist nicht von vornherein klar, in welcher Reihenfolge die Cuts besucht werden.

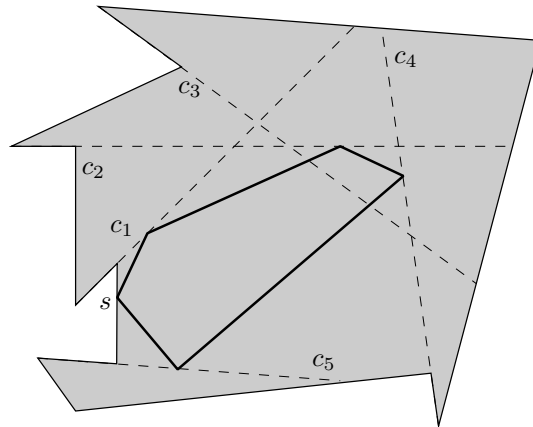


Abbildung 1.37: Eine “Corner”: die Cuts  $c_1, c_2, c_3$  und  $c_4$  schneiden sich gegenseitig.

**Definition 1.28** Eine “Corner” ist eine geordnete Menge von Cuts  $c_i, c_{i+1}, \dots, c_j$ , für die gilt

- (i)  $c_k$  schneidet  $c_{k+1}$  für  $k \in \{i, \dots, j-1\}$ ,
- (ii)  $c_i$  schneidet  $c_{i-1}$  *nicht* und
- (iii)  $c_j$  schneidet  $c_{j+1}$  *nicht*.

Betrachte Abbildung 1.37: die Cuts  $c_1, c_2, c_3$  und  $c_4$  bilden eine Corner und die Reihenfolge der Cuts wird von der SWR nicht eingehalten:  $c_3$  wird überschritten, bevor  $c_2$  erreicht wird. Die Besuchsreihenfolge der Corners ist jedoch fest:

**Lemma 1.29** *Die Shortest Watchman Route besucht die Corners eines Polygons gemäß der Ordnung entlang des Randes.*

Die von der SWR berührten Abschnitte der Cuts zwischen zwei Schnittpunkten werden aktive Segmente genannt. Sind die aktiven Segmente gegeben, kann man die SWR wie oben berechnen, das große Problem ist jedoch, die Menge der aktiven Segmente zu berechnen. Erste Ansätze waren, mit einer Menge von aktiven Segmenten eine Watchman Route  $R'$  zu berechnen und diese durch lokale Anpassungen, d. h. durch Austausch der aktiven Segmente, zu verkürzen. Einen anderen Ansatz verfolgen Dror et. al. die die folgende Verallgemeinerung des Shortest Watchman Problems und ähnlicher Probleme lösen.

**Definition 1.30**

- (i) Beim einfachen **Touring Polygon Problem** (TPP) ist eine Sequenz von einfachen, konvexen, disjunkten Polygonen  $P_1, P_2, \dots, P_k$  mit insgesamt  $n$  Kanten, ein Startpunkt  $s$  und ein Zielpunkt  $t$  gegeben. Gesucht ist der kürzeste Pfad  $\pi$ , der in  $s$  startet, in  $t$  endet und alle Polygone  $P_i$  in der gegebenen Reihenfolge besucht.
- (ii) Beim allgemeinen TPP verlangen wir zusätzlich, daß die Pfade zwischen zwei Polygonen  $P_i$  und  $P_{i+1}$  ( $i = 0, \dots, k$ ;  $P_0 := s$ ;  $P_{k+1} := t$ ) innerhalb des **Zaunes**  $F_i$  verlaufen. Ein Zaun  $F_i$  ist gegeben durch ein einfaches Polygon, das  $P_i$  und  $P_{i+1}$  enthält. Die Polygone dürfen sich überlappen und nur die **Fassade** eines Polygons  $P_i$  muß konvex sein. Die Fassade eines Polygons  $P_i$  ist der Teil des Randes von  $P_i$ , der nicht auf dem Rand des Zaunes  $F_{i-1}$  liegt:  $\text{Fassade}(P_i) := \partial P_i \setminus \partial F_{i-1}$ .

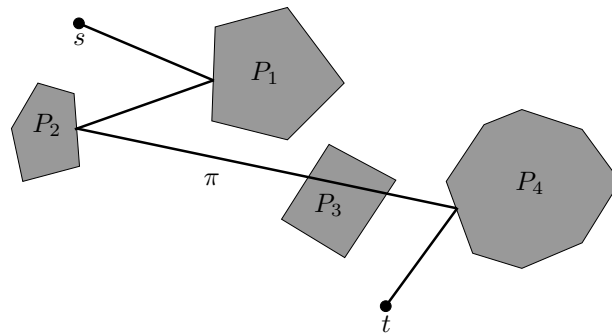


Abbildung 1.38: Ein Beispiel für das einfache Touring Polygon Problem.

Beim TPP wird nicht ausgeschlossen, daß ein Polygon  $P_i$  vor einem Polygon  $P_j$  mit  $j < i$  besucht wird, allerdings muß das Polygon  $P_i$  dann ein weiteres mal besucht werden. Der Besuch eines Polygons  $P_i$  ist erst dann gültig, wenn vorher alle anderen Polygone  $P_1, \dots, P_{i-1}$  besucht wurden, andernfalls gilt das Polygon nach wie vor als unbesucht. Abbildung 1.38 zeigt ein Beispiel für das einfache TPP, Abbildung 1.39 für das allgemeine TPP. Der gestrichelte Teil des Randes von  $P_4$  ist die Fassade von  $P_4$ . Beachte, daß  $P_5$  vor  $P_4$  besucht wird, aber "gewertet" wird erst der Besuch von  $P_5$  nach dem Besuch von  $P_4$ .

Beschränken wir uns zunächst auf das einfache TPP. Klar ist, daß der kürzeste Weg eine polygonale Kette sein muß, andernfalls ließe sich der Weg verkürzen. Die erste wichtige Idee zur

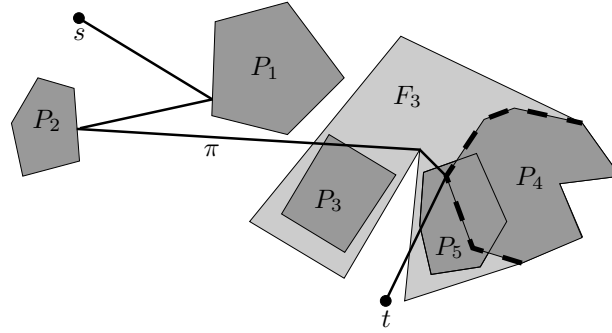
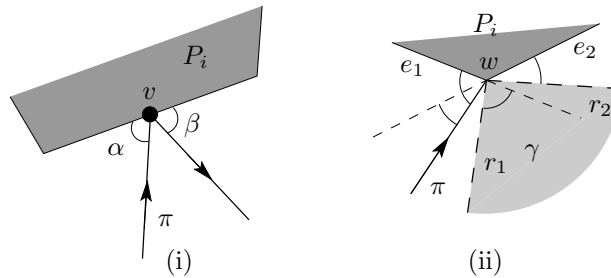


Abbildung 1.39: Ein Beispiel für das allgemeine Touring Polygon Problem.

Abbildung 1.40: Lokale Optimalität: (i) bei Reflexion an einer Kante gilt  $\alpha = \beta$ , (ii) bei Reflexion an einer Ecke liegt die Ausgangskante im Winkelbereich  $\gamma$ .

Berechnung des Pfades  $\pi$  ist, lokale Optimalität auszunutzen; eine Idee, die wir schon bei der Berechnung der kürzesten Pfade in Abschnitt 1.2 benutzt haben:

### Lemma 1.31

- (i) Ein Pfad  $\pi$  ist lokal optimal, wenn die Ecken von  $\pi$  auf dem Rand eines der Polygone  $P_i$  liegen. Liegt die Ecke  $v$  von  $\pi$  auf einer Polygonkante, so sind die Winkel zwischen der Polygonkante und den beiden zu  $v$  inzidenten Kanten gleich (Einfallswinkel = Ausfallswinkel, siehe Abbildung 1.40(i)). Liegt die Ecke  $v$  auf einer Polygonecke  $w$ , so liegt die ausgehende Kante von  $\pi$  in dem Winkelbereich  $\gamma$ , der durch die Reflexionen  $r_1$  und  $r_2$  der eingehenden Kante an den beiden zu  $w$  inzidenten Polygonkanten  $e_1$  und  $e_2$  begrenzt wird, siehe dazu auch Abbildung 1.40(ii)).
- (ii) Für jedes  $p \in \mathbb{R}^2$  und jedes  $i \in \{1, \dots, k\}$  gibt es einen eindeutigen, lokal optimalen Pfad  $\pi_i(p)$ , der in  $s$  startet, die Polygone  $P_1$  bis  $P_i$  in der gegebenen Reihenfolge besucht und in  $p$  endet. Einen solchen Pfad nennen wir im folgenden den **kürzesten  $i$ -Pfad** zu  $p$ .
- (iii) Ein lokal optimaler  $i$ -Pfad ist global optimal.

### Beweis.

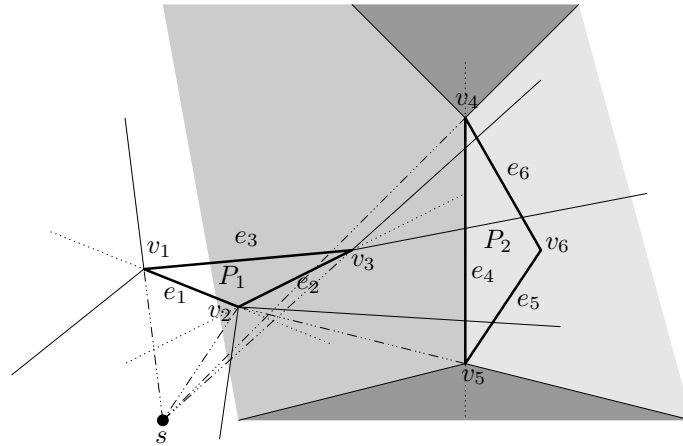
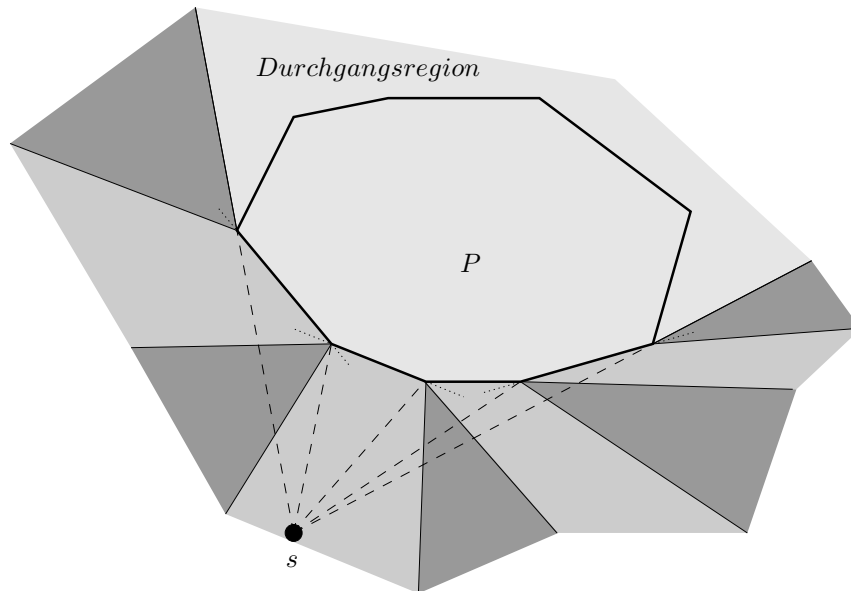
- (i) Ist die Bedingung verletzt, wäre der Pfad lokal verkürzbar.
- (ii) Dies folgt später aus der Konstruktion der lokal optimalen Pfade.
- (iii) Folgt aus (ii).

□

Die zweite Idee ist, die Ebene in Regionen mit kombinatorisch gleichen Pfaden einzuteilen, d.h. wir zerlegen die Ebene  $\mathbb{R}^2$  disjunkt in Regionen  $\mathcal{R}_j$ , so daß für alle Punkte  $p \in \mathcal{R}_j$  der kürzeste  $k$ -Pfad von  $p$  zu  $s$  über die gleichen Polygonecken/Polygonkanten führt bzw. an den gleichen Polygonecken/Polygonkanten reflektiert wird. Auch diese Idee haben wir bereits in Form der Shortest Path Map benutzt. Abbildung 1.41 zeigt ein Beispiel für eine vollständige kombinatorische Shortest Path Map für unser Problem. Die Beschriftungen der Regionen geben





Abbildung 1.43: Last Step Shortest Path Map zu  $P_2$ .Abbildung 1.44: Last Step Shortest Path Map zu  $P$ .

ein kürzester  $i$ -Pfad das Polygon  $P_i$  nach dem Besuch von  $P_1, \dots, P_{i-1}$  zum ersten Mal besuchen kann. Die von  $T_i$  ausgehenden, reflektierten Strahlen bezeichnen wir als den **Stern**  $R_i$  von  $T_i$ .

Im Gegensatz zur vollständigen kombinatorischen Shortest Path Map kann eine Last Step Shortest Path Map keine hohe Komplexität haben, wie folgendes Lemma zeigt:

**Lemma 1.32**

- (i)  $T_i$  ist zusammenhängend.
- (ii) Die Strahlen  $R_i$  sind disjunkt und jeder Punkt  $p \in \mathbb{R}^2$  außerhalb der Durchgangsregion wird von genau einem Strahl von  $R_i$  getroffen.
- (iii) Die Komplexität einer  $\mathcal{S}_i$  liegt in  $O(|P_i|)$ .

**Beweis.**

- (i) Übungsaufgabe
- (ii) Wir zeigen induktiv, daß die Strahlen  $R_i$  disjunkt sind. Für  $\mathcal{S}_1$  gilt dies offensichtlich. Nehmen wir an, die Aussage gilt für  $\mathcal{S}_1, \dots, \mathcal{S}_{i-1}$ . Zu jedem Knoten  $v$  von  $P_i$  läßt sich eindeutig die Region von  $\mathcal{S}_{i-1}$  bestimmen, in der  $v$  liegt. Damit ist auch der kürzeste

$(i - 1)$ -Pfad zu  $v$  eindeutig, insbesondere die letzte Kante dieses Pfades. Damit sind alle letzten Kanten zu Knoten aus  $P_i$  disjunkt bis auf einen gemeinsamen Startpunkt. Es ist bekannt, daß die Reflektionen disjunkter Liniensegmente an einem konvexen Polygon ebenfalls disjunkt sind.

- (iii) Die Kanten in  $\mathcal{S}_i$  lassen sich in zwei Klassen unterteilen: die Kanten von  $T_i$  und die Strahlen, die von den Ecken von  $T_i$  ausgehen.  $T_i$  ist zusammenhängend und — da die  $P_j$  disjunkt sind — Teil des Randes von  $P_i$ . Daher besteht  $T_i$  aus maximal  $|P_i|$  Kanten. Die von den Ecken von  $T_i$  ausgehenden Strahlen sind Teile von  $R_i$  und damit disjunkt. Ihr Beitrag zur Komplexität von  $\mathcal{S}_i$  liegt damit ebenfalls in  $O(|P_i|)$ .

□

Damit haben wir eine Vorstellung, wie eine Karte aussehen kann, mit Hilfe derer wir unsere Pfade berechnen können. Es stellen sich nun die Fragen, wie wir die  $\mathcal{S}_i$  effizient berechnen und wie wir aus den  $\mathcal{S}_i$  einen kürzesten Pfad konstruieren können. Die letzte Frage beantwortet folgendes Lemma:

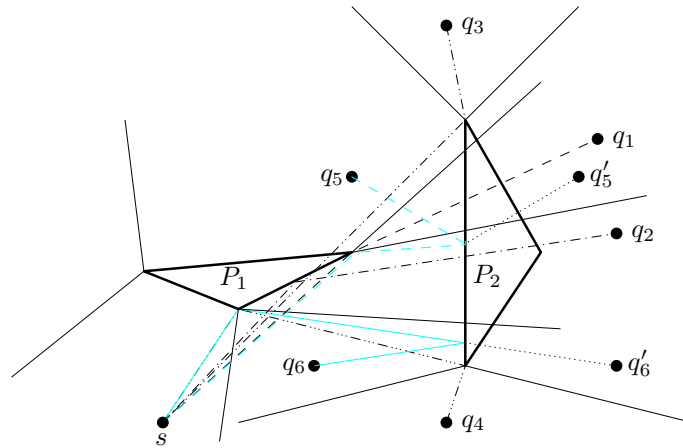
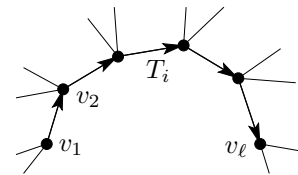


Abbildung 1.45: Queries beim einfachen TPP:  $q_1$  und  $q_2$  liegen in der Durchgangsregion,  $q_3$  und  $q_4$  in einer Zelle zu einer Ecke,  $q_5$  und  $q_6$  in der Zelle einer Kante.

**Lemma 1.33** Sind für das einfache TPP die Last Step Shortest Path Maps  $\mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_i$  gegeben, so kann zu allen  $q \in \mathbb{R}^2$  der kürzeste  $i$ -Pfad  $\pi_i(q)$  in Zeit  $O(k \cdot \log \frac{n}{k})$  berechnet werden.

**Beweis.** Algorithmus 1.10 liefert zu einem Anfragepunkt  $q \in \mathbb{R}^2$  den Pfad  $\pi_i(q)$ , Abbildung 1.45 zeigt einige Beispiele für kürzeste  $i$ -Pfade.

Wir wissen aus Lemma 1.32, daß die Komplexität einer LSSPM  $\mathcal{S}_i$  in  $O(|P_i|)$  liegt. Die Karte kann daher in einer Datenstruktur gespeichert werden, die in Zeit  $O(\log |P_i|)$  die Zelle liefert, in der ein Anfragepunkt liegt. Wir könnten dazu eine Point-Location Datenstruktur benutzen; es reicht jedoch, die Knoten der konvexen Kette  $T_i$  in einem Suchbaum oder Array zu speichern. Zu jedem Knoten  $v$  halten wir die beiden von  $v$  ausgehenden Strahlen ebenfalls fest.



Durch die rekursiven Anfragen liegt die Zeit, um einen kürzesten  $i$ -Pfad zu bestimmen, in  $O(\sum_{j=1}^k \log |P_j|)$ . Diese Summe nimmt bei  $|P_j| = \frac{n}{k}$  ihr Maximum an. □

Mit Hilfe der Queries lassen sich die Last Step Shortest Path Maps recht einfach sukzessiv konstruieren, siehe Algorithmus 1.11. Die Zeit zur Konstruktion einer einzelnen  $\mathcal{S}_i$  liegt in

---

**Algorithmus 1.10** Einfaches TPP: Query

---

Bestimmt zu gegebenen Last Step Shortest Path Maps  $\mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_i$  und Anfragepunkt  $q \in \mathbb{R}^2$  den kürzesten  $i$ -Pfad von  $s$  zu  $q$ .

- Bestimme die Region von  $q$  in  $\mathcal{S}_i$ :  
Durch prüfen der Lage (rechts oder links) von  $q$  bzgl. der konvexen Kette aus  $T_i$  und den beiden Strahlen am Rand der Durchgangsregion wird bestimmt, ob  $q$  in der Durchgangsregion oder in einer Reflektionszelle liegt. Liegt  $q$  in einer Reflektionszelle, so kann diese durch binäre Suche mit Rechts/Linkstests in den Strahlen, die von den Knoten von  $T_i$  ausgehen, bestimmt werden.
  - Falls  $q$  in der Durchgangsregion liegt: es gilt  $\pi_i(q) = \pi_{i-1}(q)$ , bestimme rekursiv  $\pi_{i-1}(q)$ .
  - Falls  $q$  in der Zelle zu einer Ecke  $v$  liegt: das letzte Segment des kürzesten  $i$ -Pfades von  $s$  zu  $q$  ist  $\overline{vq}$ . Bestimme rekursiv  $\pi_{i-1}(v)$ .
  - Falls  $q$  in der Zelle zu einer Kante  $e$  liegt: sei  $q'$  die Spiegelung von  $q$  an  $e$ . Berechne rekursiv  $\pi_{i-1}(q')$  und ersetze in diesem Pfad das Segment von  $e$  zu  $q'$  durch das Segment von  $e$  zu  $q$ .
- 

---

**Algorithmus 1.11** Einfaches TPP: Konstruktion der  $\mathcal{S}_i$ 

---

- Berechne  $\mathcal{S}_1$  durch Reflektionen an den Knoten von  $P_1$ .
  - Berechne sukzessiv  $\mathcal{S}_i$  aus  $\{\mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_{i-1}\}$  wie folgt: Berechne zu jedem Knoten  $v \in P_i$  den kürzesten  $(i-1)$ -Pfad  $\pi_{i-1}(v)$ . Schneidet das letzte Stück dieses Pfades das Innere von  $P_i$ , so liegt  $v$  in der Durchgangsregion von  $\mathcal{S}_i$ . Andernfalls liegt  $v$  in  $T_i$  und die beiden Reflektionen des letzten Segmentes von  $\pi_{i-1}(v)$  an den zu  $v$  inzidenten Kanten bestimmen die Strahlen, die die Zelle von  $v$  begrenzen.
-

$O(|P_i|k \log \frac{n}{k})$ , insgesamt also

$$O\left(\sum_{i=1}^k |P_i|k \log \frac{n}{k}\right) = O(nk \log \frac{n}{k}).$$

Mit Lemma 1.32 und Lemma 1.33 folgt:

**Theorem 1.34** (Dror, Efrat, Lubiw, Mitchell, 2003)

Für das einfache TPP mit  $k$  Polygonen mit insgesamt  $n$  Kanten läßt sich in Zeit  $O(kn \log \frac{n}{k})$  eine Suchstruktur der Größe  $O(n)$  aufbauen, mit Hilfe derer der kürzeste  $i$ -Pfad zu einem Punkt  $q \in \mathbb{R}^2$  in Zeit  $O(k \log \frac{n}{k})$  berechnet werden kann. [DELM03]

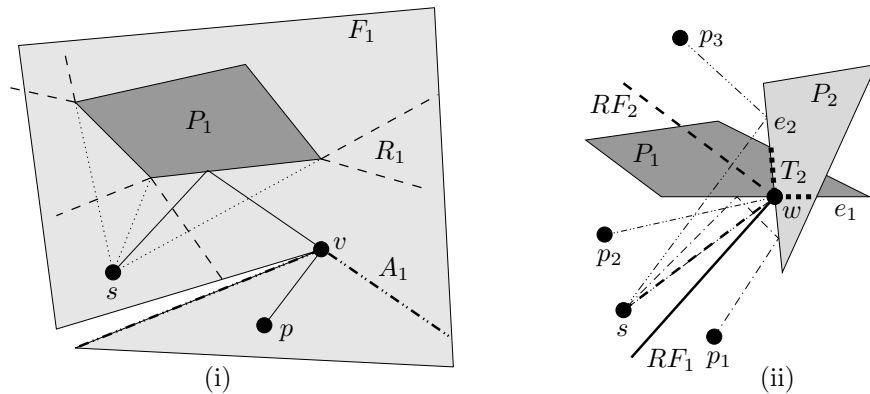


Abbildung 1.46: (i) Kürzeste Wege können an reflexen Ecken der Zäune abknicken, (ii) mögliche Kombinationen bei überlappenden Polygonen.

Im allgemeinen TPP müssen wir die Zäune  $F_i$  berücksichtigen, d. h. der kürzeste  $i$ -Pfad zu einem Punkt  $p$  muß zwischen  $P_j$  und  $P_{j+1}$  in  $F_j$  liegen. Kürzeste Pfade können nun zusätzlich an reflexen Ecken eines Zaunes abknicken, wie z. B. der Weg von  $s$  nach  $p$  in Abbildung 1.46(i). Dadurch gibt es eine weitere Klasse von Zellen in der Zerlegung der Ebene: Zellen, die einer reflexen Ecke eines Zaunes zugeordnet sind. Z. B. führen kürzeste  $i$ -Pfade aus Punkten der Zelle unterhalb der strichpunktierten Linien in Abbildung 1.46(i) über den Knoten  $v$ .

Die zweite Verallgemeinerung ist, daß sich die Polygone überlappen dürfen, wodurch weitere Arten von kombinatorisch kürzesten Wegen ins Spiel kommen. Betrachte Abbildung 1.46(ii): der Strahl  $RF_1$  entsteht durch Punktspiegelung von  $\overline{sw}$  an  $w$  und anschließender Spiegelung an den Kanten  $e_1$  und  $e_2$ , also den Kanten von  $P_1$  und  $P_2$ , auf denen der Schnittpunkt  $w$  liegt.  $RF_2$  entsteht durch Reflektion von  $\overline{sw}$  an  $e_2$ . Der Punkt  $p_1$  liegt unterhalb des Strahles  $RF_1$  und der kürzeste Weg von  $s$  nach  $p_1$  wird erst an  $P_1$ , danach an  $P_2$  reflektiert.  $p_2$  liegt zwischen  $RF_1$  und  $RF_2$  und sein kürzester Weg verläuft direkt über  $w$ . Schließlich liegt  $p_3$  oberhalb von  $RF_2$  und in der Durchgangsregion von  $P_1$  und der kürzeste Weg wird lediglich durch eine Reflektion an  $P_2$  gebildet.

Wie bereits im einfachen Fall sei  $T_i$  der Reflektionsbereich, also der Bereich von  $P_i$ , in dem die letzten Kanten eingehender  $i$ -Pfade reflektiert werden. Da sich die  $P_j$  überlappen können, ist  $T_i$  hier nicht mehr unbedingt ein Teil des Randes von  $P_i$  sondern kann auch im Inneren von  $P_i$  liegen, siehe z. B.  $T_2$  (gepunktet) in Abbildung 1.46(ii).  $R_i$  bezeichne wie oben die Menge der Strahlen, die von  $T_i$  ausgehen. Berücksichtigen wir auch den Zaun  $Z_j$ , so müssen wir folgendes beachten: Die Strahlen  $R_j$ , die an einer Ecke des Zaunes abknicken, um eine sonst unerreichbare Region zu durchqueren nennen wir  $A_j$ , siehe Abbildung 1.46. Wir können folgendes für das allgemeine TPP festhalten (ohne Beweis):

**Lemma 1.35**

- (i) Für ein Polygon  $P_i$  ist der Reflektionsbereich  $T_i$  ein Baum.
- (ii) Der Stern  $R_i$  besteht aus disjunkten Strahlen außerhalb der Durchgangsregion (ohne Berücksichtigung der  $F_i$ ) und jeder Punkt  $p \in \mathbb{R}^2$  wird von einem eindeutigen Strahl getroffen.
- (iii) Ein lokal optimaler Pfad ist eindeutig und lokale Optimalität garantiert globale Optimalität.

Wie beim einfachen TPP können wir nun sukzessive die  $T_i$ ,  $R_i$  und  $A_i$  sowie die zugehörigen Last Step Shortest Path Maps berechnen. Wir wollen hier nur das Ergebnis festhalten:

**Theorem 1.36** (Dror, Efrat, Lubiw, Mitchell, 2003)

Für das allgemeine TPP mit  $k$  Polygonen und  $k+1$  Zäunen mit insgesamt  $n$  Kanten können wir in Zeit  $O(k^2 n \log n)$  eine Suchstruktur der Größe  $O(kn)$  aufbauen, mit Hilfe derer der kürzeste  $i$ -Pfad zu einem Punkt  $q \in \mathbb{R}^2$  in Zeit  $O(i \log n + m)$  berechnet werden kann, wobei  $m$  die Anzahl der Segmente des Pfades angibt. [DELM03]

Kommen wir nun zurück zu unserem eigentlichen Problem, die Shortest Watchman Route zu berechnen.

**Theorem 1.37** (Dror, Efrat, Lubiw, Mitchell, 2003)

In einem einfachen Polygon mit  $n$  Ecken und Startpunkt  $s$  läßt sich die SWR in Zeit  $O(n^3 \log n)$  berechnen. [DELM03]

**Beweis.** Aus der Eingabe  $(P, s)$  zum SWR können wir wie folgt eine Eingabe gegeben durch  $(P_1, \dots, P_k, F_1, \dots, F_k, s', t)$  für das allgemeine TPP berechnen: Start- und Zielpunkt des TPP-Pfades ist der Startpunkt  $s$  der SWR. Als Polygone  $P_i$  wählen wir die Taschen  $P_{c_i}$  von  $P$ , siehe Definition 1.25, als Zäune legen wir  $P$  fest ( $F_i := P$ ). Für die Laufzeit des TPP ist nur die Komplexität der Fassade der  $P_i$  entscheidend, und diese besteht in unserem Fall nur aus 2 Ecken. Somit liegt die Komplexität aller Fassaden in  $O(n)$ . Die Komplexität aller Zäune liegt in  $O(n^2)$  und wir erhalten eine Laufzeit von  $O(n^3 \log n)$ . □

## 1.3 Kürzeste Pfade im Raum

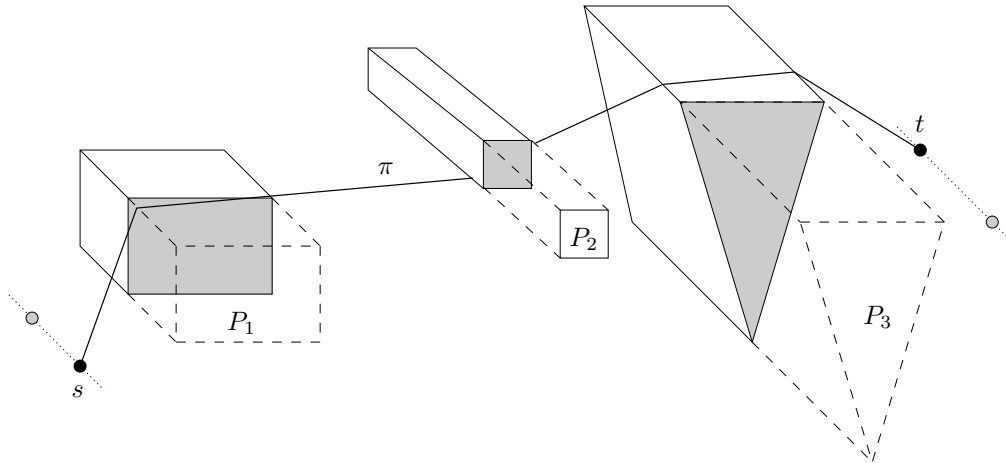


Abbildung 1.47: Kürzester Pfad im 3D.

Verlassen wir nun die Ebene und betrachten kürzeste Pfade im dreidimensionalen Raum: Gegeben seien  $h$  Polyeder im Raum mit insgesamt  $n$  Ecken, ein Startpunkt  $s$  und ein Endpunkt  $t$ . Gesucht ist der kürzeste Pfad  $\pi$  von  $s$  nach  $t$ , der außerhalb der Polyeder verläuft.

Eine erste (ernüchternde) Erkenntnis ist, daß dieses Problem nicht mehr diskret ist, da kürzeste Pfade über innere Punkte von Kanten verlaufen können. Wir haben zwei Teilprobleme zu lösen:

1. Bestimme die Folge der Kanten bzw. Ecken, die von einer Kürzesten berührt werden. (kombinatorisches Problem)
2. Bestimme die Berührungspunkte auf den Kanten. (algebraisch-numerisches Problem)

Leider ist bereits das erste Teilproblem NP-hard:

**Theorem 1.38** (Canny, Reif, 1987)

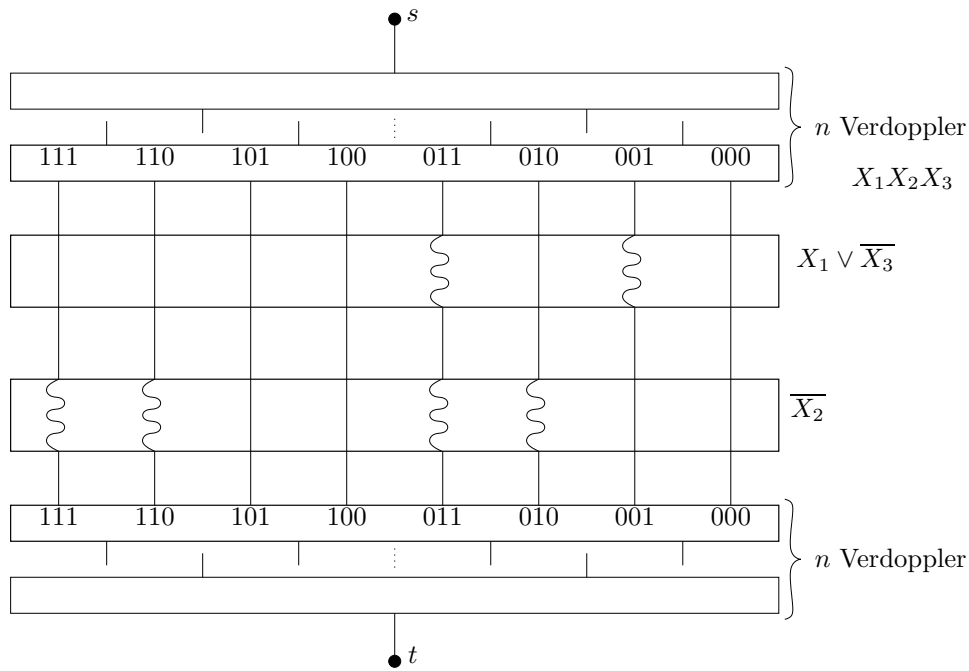
Die Bestimmung der Kantenfolge einer Kürzesten im  $\mathbb{R}^3$  ist NP-hard. Bereits die Frage, ob es zu gegebenem  $k$  eine Kantenfolge gibt, die einen Weg  $\pi$  mit  $|\pi| \leq k$  liefert, ist NP-hard. [CR87]

**Beweis.**

Zeige: 3-SAT ist auf die Kantenbestimmung reduzierbar. Dazu bilden wir in polynomieller Zeit den booleschen Ausdruck  $\alpha = \bigwedge_{i=1}^m (\Pi_{i1} \vee \Pi_{i2} \vee \Pi_{i3})$  mit  $\Pi_{ij} \in \{X_k, \overline{X_k}\}$ ,  $1 \leq k \leq n$ , auf einen Parcours  $P_\alpha$  im Raum mit Startpunkt  $s$  und Endpunkt  $t$  ab. Die Geodätischen von  $s$  nach  $t$  in  $P_\alpha$  zerfallen dabei in Klassen  $G_w$ , die den  $2^n$  möglichen Wahrheitswertbelegungen  $w$  der Variablen  $X_i$  entsprechen. Die Klasse einer Geodätischen läßt sich aus der Folge der von ihr besuchten Kanten in  $P_\alpha$  ablesen.  $P_\alpha$  wird so konstruiert, daß für eine Kürzeste  $\pi$  von  $s$  nach  $t$ , die in einer Klasse  $G_w$  liegt, gilt: entweder die Belegung  $w$  erfüllt den Ausdruck  $\alpha$ , oder  $\alpha$  ist unerfüllbar.

Abbildung 1.48 zeigt die wesentliche Idee: die vom Startpunkt  $s$  ausgehende Geodätische wird  $n$  mal verdoppelt, so daß für jede der  $2^n$  möglichen Variablenbelegungen ein Pfad existiert. Diejenigen Pfade, die die Klausel nicht erfüllen, werden verlängert — in der Abbildung durch  $\sim\sim\sim$  angedeutet — und schließlich wieder zu einer Geodätischen zusammengefaßt, wobei darauf zu achten ist, daß die Konstruktion des Parcours in polynomieller Zeit möglich sein soll!

Der Parcours besteht im wesentlichen aus Platten der Dicke  $\varepsilon$  mit Schlitz, durch die die Geodätischen verlaufen können. Legt man mehrere Platten in geeigneter Weise mit einem

Abbildung 1.48: Beispiel: Filter für Klausel  $(X_1 \vee X_3) \wedge \overline{X_2}$ .

Abstand  $\varepsilon$  übereinander, läßt sich der Verlauf der Geodätischen beeinflussen. Zur Konstruktion eines solchen Parcours benötigen wir die im folgenden erklärten Bausteine. In jedem Baustein — bis auf die Barrieren im Literalfilter — werden alle Klassen von Geodätischen um den gleichen Betrag verlängert.

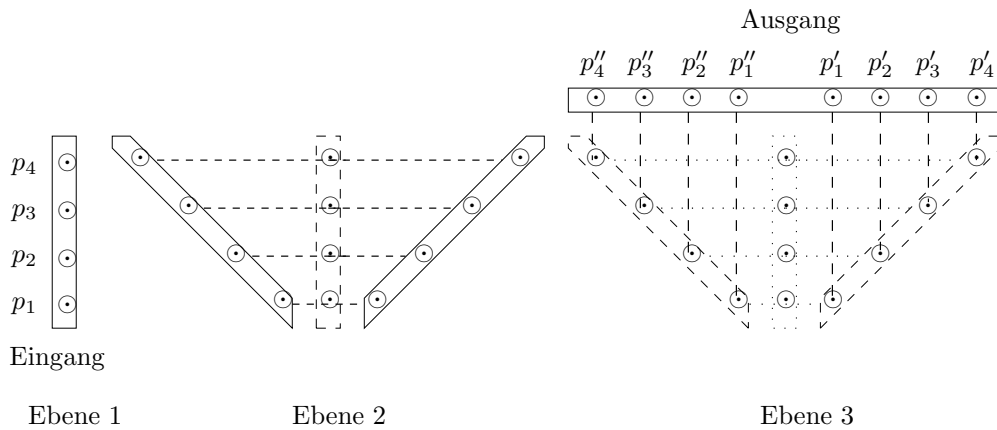


Abbildung 1.49: Die drei Ebenen eines Verdopplers.

### Verdoppler

Ein Verdoppler dient dazu, seine aus  $n$  Klassen von Geodätischen bestehende Eingabe in  $2n$  Klassen aufzuspalten. Er besteht aus drei Ebenen: die unterste Ebene enthält einen vertikalen Schlitz, durch den die Eingabe in den Verdoppler geführt wird. Die Eingabe verzweigt nach rechts und links und wird über zwei diagonale Schlitzte in der zweiten Ebene zu einem horizontalen Schlitz in der dritten Ebene geführt, siehe Abbildung 1.49. Das Zeichen  $\odot$  symbolisiert in den Abbildungen eine Geodätische, die von einer unteren Ebene in eine obere Ebene läuft.



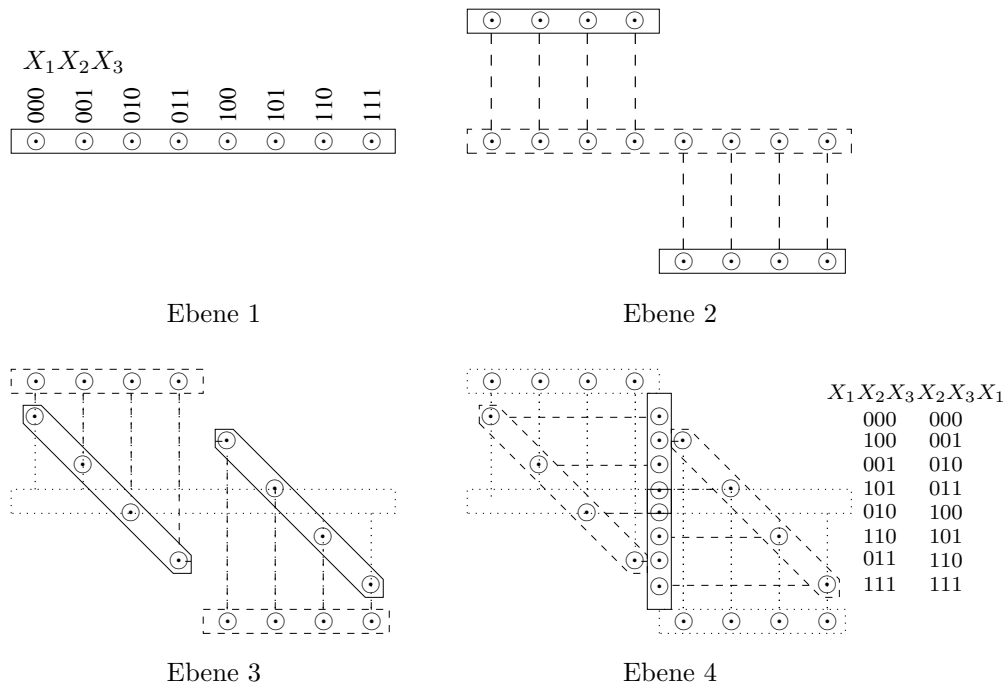


Abbildung 1.50: Die vier Ebenen eines Mischers.

### Mischer

Ein Mischer besteht aus vier Ebenen. Die Eingabe wird durch einen horizontalen Schlitz in der untersten Ebene geführt, in zwei Hälften geteilt und durch zwei horizontale Schlitze in der zweiten Ebene geführt, vgl. Abbildung 1.50. In der dritten Ebene liegen zwei Reihen mit diagonalen Schlitzen, die die gesplittete Eingabe in die Vertikale umlenken, wo sie durch eine Reihe vertikaler Schlitze in der vierten Ebene ausgegeben wird. Die Schlitze in den Diagonalen sind dabei so versetzt, daß sich jeweils eine Klasse der ersten Hälfte und eine Klasse der zweiten Hälfte abwechseln. Die Eingabe  $X_i X_j X_k \dots$  wird derart permutiert, daß die Klassen mit  $X_j = 1$  in einer Hälfte liegen, die Klassen mit  $X_j = 0$  in der anderen Hälfte. Anders ausgedrückt ergibt sich eine Linksrotation der Variablen,  $X_j$  wird zum signifikantesten Bit.

### Literalfilter

Eine Klasse von Geodätischen kodiert eine Binärzahl  $w = (w_1, \dots, w_n) \in \{0, 1\}^n$ , die einer Belegung der Variablen entspricht. Der Literalfilter dient nun dazu, die Klassen von Geodätischen zu verlängern, deren Belegung das Literal nicht erfüllt. Um also alle Pfade herauszufiltern, bei denen  $w_i = 0$  den Ausdruck erfüllt, müssen alle Pfade mit  $w_i = 1$  mit einer Barriere verlängert werden. Dabei dürfen wir die Barrieren allerdings nicht in die ursprüngliche Kodierung einfügen, wie dies in Abbildung 1.48 gezeigt ist, da dies evtl. zu exponentiell vielen Barrieren führen würde; um z. B.  $w_n$  zu filtern, wäre eine Barriere in jedem zweiten Pfad nötig! Also setzen wir Mischer ein, um die zu filternde Variable zur signifikantesten Variable zu machen, und können dann mit *einer* Barriere auf der rechten bzw. linken Seite alle Pfade mit  $w_i = 0$  bzw.  $w_i = 1$  verlängern, siehe Abbildung 1.51. Da ein Mischer eine Linksrotation der Variablen bewirkt, ist  $w_i$  nach  $i - 1$  Mischerdurchläufen das signifikanteste Bit. Anschließend benötigen wir noch  $n - i + 1$  Mischer, um wieder die ursprüngliche Kodierung zu erhalten.

### Klauselfilter

Der Klauselfilter dient schließlich zur Umsetzung einer Klausel der Form  $C_i = \Pi_{i1} \vee \Pi_{i2} \vee \Pi_{i3}$  mit  $\Pi_{ij} \in \{X_k, \overline{X_k}\}$ ,  $1 \leq k \leq n$ . Dazu benötigen wir drei "parallel geschaltete" Literalfilter:

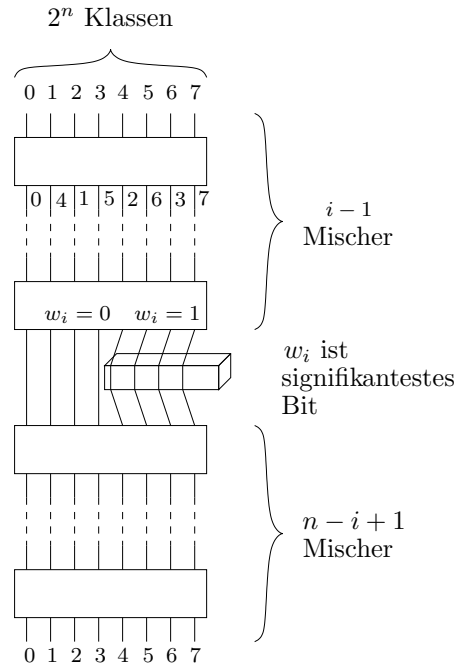


Abbildung 1.51: Literalfilter.

die Eingabe von  $2^n$  Klassen von Geodätischen wird allen Literalfiltern zugeführt. In den Literalfiltern werden die Geodätischen verlängert, die die Klausel nicht erfüllen. Durch die Parallelschaltung ergibt sich eine Disjunktion der Literale, siehe Abbildung 1.52.

Zur Konstruktion eines Parcours aus einer 3-SAT Klauselmeng

$$\alpha = \bigwedge_{i=1}^m C_i$$

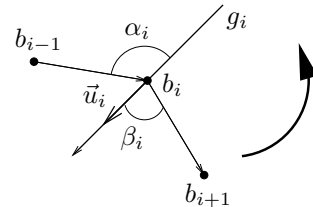
über  $n$  Variablen benötigen wir zunächst einen Block aus  $n$  Verdopplern, um aus der vom Startpunkt ausgehenden Geodätischen  $2^n$  Klassen für jede mögliche Variablenbelegung zu erzeugen. Diese Klassen werden durch  $m$  hintereinander geschaltete Klauselfilter und schließlich durch  $n$  invers benutzte Verdoppler geführt, die die gefilterten Klassen von Geodätischen wieder zu einer zusammenfassen, die zum Zielpunkt  $t$  führt.  $\square$

Damit bleibt das zweite Teilproblem: Wie bestimmt man zu gegebener Kantenfolge die richtigen Berührungspunkte, also den kürzesten Pfad, der die Kanten in dieser Reihenfolge besucht?

Schon das einfachere Problem, zu Geraden  $g_i$  im Raum und fest vorgegebener Besuchsreihenfolge die Berührungspunkte des Pfades so zu verschieben, daß seine Länge minimal wird, ist schwierig. Ein lokales Optimalitätskriterium erhält man durch Hochklappen der durch  $g_i$  und  $b_{i+1}$  gegebene Ebene in die durch  $g_i$  und  $b_{i-1}$  gegebene Ebene: liegen  $b_{i-1}$ ,  $g_i$  und  $b_{i+1}$  in einer Ebene, muß der kürzeste Pfad zwischen  $b_{i-1}$  und  $b_{i+1}$  ein Liniensegment durch  $b_i$  sein. Dies gilt genau dann, wenn die beiden Winkel  $\alpha_i$  und  $\beta_i$  gleich sind. Wären beide Winkel nicht gleich, so ließe sich der Pfad zwischen  $b_{i-1}$  und  $b_{i+1}$  verkürzen.

Sei  $\vec{u}_i$  ein Vektor in Richtung  $g_i$  mit  $|\vec{u}_i| = 1$ , dann gilt:<sup>13</sup>

$$\alpha_i = \beta_i \iff \frac{\overrightarrow{b_{i-1}b_i} \cdot \vec{u}_i}{|\overrightarrow{b_{i-1}b_i}|} = \frac{\overrightarrow{b_ib_{i+1}} \cdot \vec{u}_i}{|\overrightarrow{b_ib_{i+1}}|} \quad (*).$$



<sup>13</sup>  $\vec{a} \cdot \vec{b}$  bezeichnet dabei das Skalarprodukt zweier Vektoren mit eingeschlossenem Winkel  $\gamma$ :  $\vec{a} \cdot \vec{b} = |\vec{a}||\vec{b}| \cos \gamma$ .

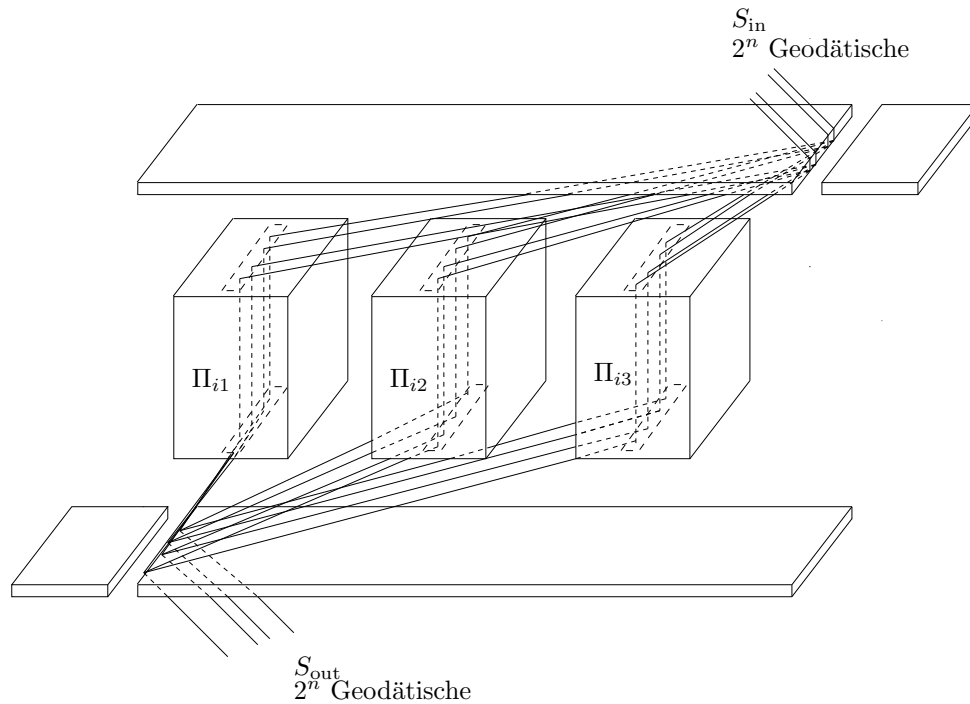


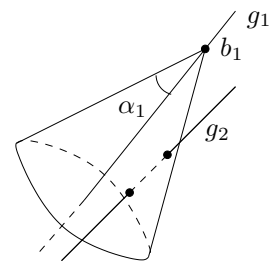
Abbildung 1.52: Klauselfilter.

Die Gleichung (\*) muß für jede Gerade  $g_i$  mit  $1 \leq i \leq n$  gelten. Ein naiver Ansatz wäre nun, den ersten Berührungspunkt  $b_1$  hinter  $s$  zu raten und mit der Winkelbedingung die Fortsetzung des Pfades zu bestimmen. Das Problem dabei ist, daß  $g_2$  den  $\alpha_1$ -Kegel um  $g_1$  zweimal schneiden kann (siehe Abbildung), die Fortsetzung des Pfades ist also nicht eindeutig, es bleiben  $2^{n-1}$  Möglichkeiten.

Der Versuch, die Gleichungen (\*) mit algebraischen Methoden exakt zu lösen, führt zu Weglängen, die durch algebraische Zahlen<sup>14</sup> mit exponentiell in  $n$  wachsendem Grad dargestellt werden. [Baj85]

Das Problem zu entscheiden, ob ein Pfad einer Länge  $\leq k$  in einer Umgebung mit  $n$  Ecken existiert, kann in Zeit  $2^{n^{O(1)}}$  und Platz  $n^{O(\log n)}$  Platz gelöst werden. [RS94]

Es gibt polynomiale  $(1 + \varepsilon)$ -Approximationsschemata<sup>15</sup> in Zeit  $O(n^2 \log^c n \cdot \frac{1}{\varepsilon^4})$ . [Cla87]



<sup>14</sup>Eine algebraische Zahl ist eine komplexe Zahl, die Nullstelle eines Polynoms  $a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0$  mit rationalen oder ganzzahligen Koeffizienten ist. Nicht algebraisch sind z. B.  $\pi$  oder  $e$ .

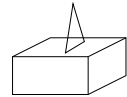
<sup>15</sup>D.h. der kürzeste Weg läßt sich beliebig genau approximieren, je kleiner das  $\varepsilon$ , desto genauer die Approximation, aber desto höher die Rechenzeit.

## 1.4 Kürzeste Pfade auf der Oberfläche eines Polyeders

Da sich die Bestimmung kürzester Pfade im Raum im allgemeinen als sehr schwierig erwiesen hat, betrachten wir den Spezialfall, bei dem sowohl  $s$  als auch  $t$  auf der Oberfläche desselben Polyeders mit  $n$  Kanten im Raum liegen. Auch der kürzeste Weg soll auf der Oberfläche des Polyeders verlaufen und nicht frei durch den Raum, auch wenn die ‐Luftlinie‐ kürzer wäre. Wir interessieren uns also z. B. für den Weg, den ein Wanderer vom Gipfel eines Berges zum nächsten Gipfel nehmen würde: ohne weitere Hilfsmittel muß er durch das Tal zwischen beiden Gipfeln wandern.

Ein Polyeder habe dabei folgende Eigenschaften:

- Es ist eine kompakte, zusammenhängende Teilmenge des  $\mathbb{R}^3$ .
- Der Rand besteht aus Polygonen.
- In jeder Umgebung eines Randpunktes liegen sowohl innere als auch äußere Punkte. Obige Abbildung zeigt also *keinen* Polyeder in unserem Sinn.



Weiterhin nehmen wir an, die Oberfläche des Polyeders sei bereits trianguliert;<sup>16</sup> aus der Euler-Formel ergibt sich, daß die Anzahl der Dreiecke in  $O(n)$  liegt.

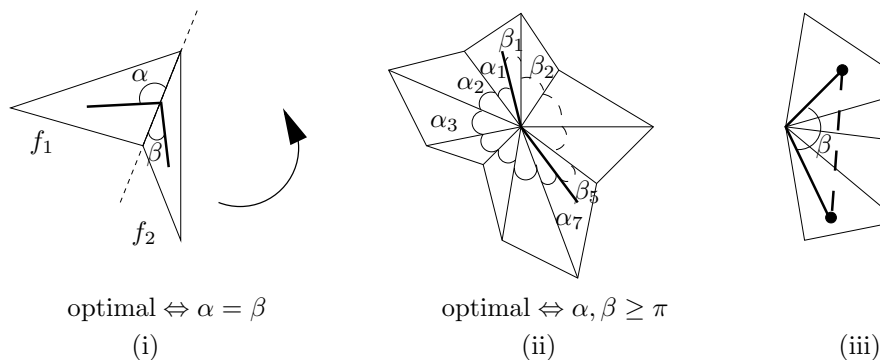


Abbildung 1.53: Eigenschaften von Kürzesten auf Polyedern.

Halten wir zunächst einige Eigenschaften von kürzesten Pfaden auf Polyedern fest:

Eine lokale Eigenschaft ist, daß in Abbildung 1.53(i) der Pfad genau dann optimal ist, wenn  $\alpha = \beta$  gilt. Dreht man nämlich eine der Flächen um ihre gemeinsame Kante in die von der anderen Fläche aufgespannte Ebene, so daß die Flächen rechts und links von der Kante liegen und nicht übereinander, so muß der kürzeste Pfad eine Gerade in der Ebene sein. Dies ist mit der Bedingung  $\alpha = \beta$  äquivalent.

In Abbildung 1.53(ii) verläuft die Kürzeste (fett gezeichnet) durch einen Eckpunkt. Dies ist dann und nur dann möglich, wenn sowohl die Summe der Winkel  $\alpha = \sum \alpha_i$  als auch die Summe der Winkel  $\beta = \sum \beta_i$  größer gleich<sup>17</sup>  $\pi$  ist. Wäre z. B.  $\beta < \pi$ , wie in der Abwicklung in Abbildung 1.53(iii) gezeigt, kann die fett gezeichnete Linie nicht die Kürzeste sein, da die gestrichelte Linie kürzer ist. Da zudem gilt:  $p$  konvex<sup>18</sup>  $\Rightarrow \alpha + \beta < 2\pi$  haben wir folgendes gezeigt:

<sup>16</sup>Ansonsten ließe sich das Polyeder in Zeit  $O(n)$  triangulieren.

<sup>17</sup>An dieser Stelle ist natürlich die Zahl  $\pi = 3,1415\dots$  gemeint, nicht der Pfad  $\pi$ .

<sup>18</sup>Eine Ecke  $v$  im  $\mathbb{R}^3$  ist konvex, wenn eine Ebene existiert an der das Polyeder ‐durchgeschnitten‐ werden kann, so daß der  $v$  enthaltende Teil des Polyeders ein konvexes Polyeder ist.

**Lemma 1.39** Eine Geodätische  $\pi$  auf einem Polyeder hat folgende lokale Eigenschaften:

- (i) An jeder von  $\pi$  überschrittenen Polyederkante müssen die Scheitelwinkel gleich sein.
- (ii)  $\pi$  kann nicht über eine konvexe Ecke von  $P$  führen.

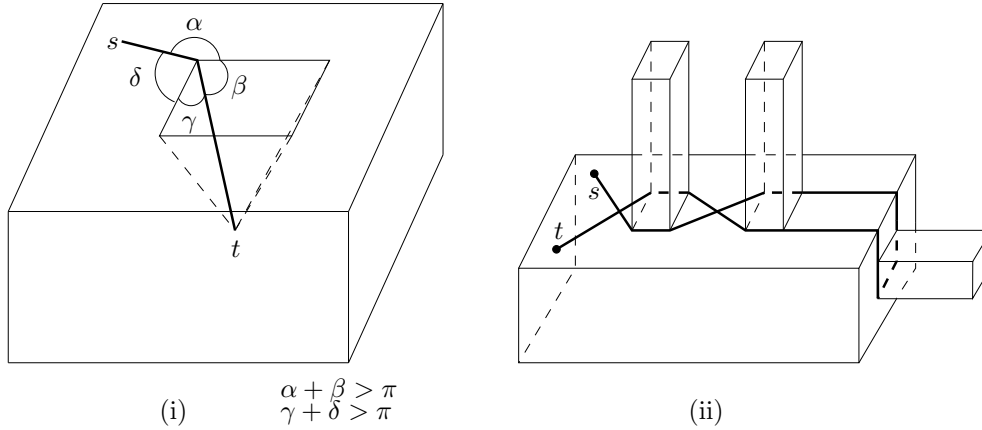


Abbildung 1.54: (i) Kürzeste über nichtkonvexe Ecke, (ii) Geodätische von  $s$  nach  $t$ , aber keine Kürzeste.

Wie das Beispiel in Abbildung 1.54(i) zeigt, kann eine Kürzeste sehr wohl über nichtkonvexe Ecken laufen.

Neben den lokalen Eigenschaften können wir auch einige globale Eigenschaften festhalten, die jedoch *nicht* für Geodätische gelten. Abbildung 1.54(ii) zeigt eine Geodätische, für die alle drei Eigenschaften nicht gelten!<sup>19</sup>

**Lemma 1.40** Kürzeste Wege  $\pi_i, \pi_j$  von  $a$  nach  $b_i, b_j$  auf einem Polyeder haben folgende globale Eigenschaften:

- (i)  $\pi_i$  ist einfach, d. h.  $\pi_i$  enthält keine Selbstschnitte.
- (ii)  $\pi_i$  schneidet jede Fläche des Polyeders höchstens einmal.
- (iii)  $\pi_i$  und  $\pi_j$  können sich nicht im Inneren einer Fläche kreuzen.

**Beweis.**(Skizze) Bei allen Eigenschaften kann man sich leicht klar machen, daß  $\pi_i$  verkürzbar wäre, wenn eine Eigenschaft verletzt wird.  $\square$

Die erste Arbeit, die sich mit diesem Problem befaßte, lieferte dieses Ergebnis:

**Theorem 1.41** (Sharir, Schorr, 1984)

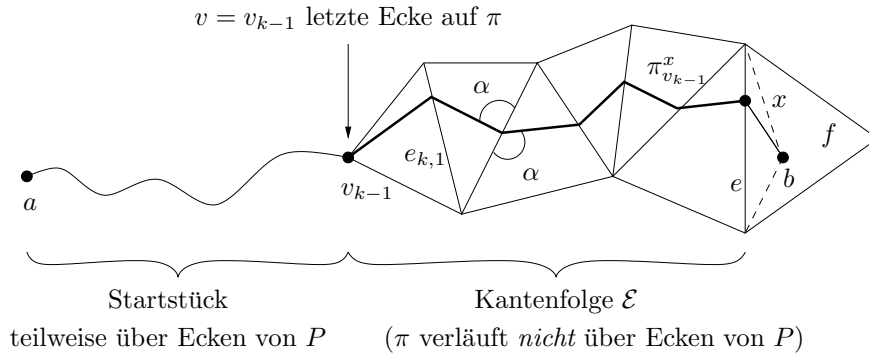
Sei  $P$  ein konvexes Polyeder,  $a \in P$  fest (o. E. eine Ecke von  $P$ ) und  $b \in P$  gegeben. Die Entfernung bzw. die Kürzeste von  $a$  nach  $b$  in  $P$  läßt sich nach Vorbereitungszeit  $O(n^3 \log n)$  in Zeit  $O(n)$  und Platz  $O(n^2)$  berechnen. [SS84, SS86]

Im folgenden werden wir uns mit einem anderen Ansatz befassen, der das Resultat von Theorem 1.41 verbessert, insbesondere auf beliebige Polyeder erweitert hat. Dazu untersuchen wir zunächst, wie die Kandidaten für Kürzeste von  $a$  nach  $b$  aussehen: Es sind Kantenzüge  $\pi$  auf  $P$ , die durch nichtkonvexe Ecken von  $P$  laufen und Folgen von Kanten von  $P$  schneiden, vgl. Abbildung 1.55:

$$\pi = \underbrace{v_0}_{=a}, \underbrace{e_{1,1}, \dots, e_{1,m_1}}_{\text{Kanten}}, \underbrace{v_1}_{\text{2. Ecke}}, \underbrace{e_{2,1}, \dots, e_{2,m_2}}_{\text{Kanten}}, \dots, \underbrace{v_{k-1}}_{\text{letzte Ecke}}, \underbrace{e_{k,1}, \dots, e_{k,m_k}}_{\text{Kanten}}, \underbrace{v_k}_{=b}$$

Aus Lemma 1.40 folgt, daß  $\pi$  einfach ist und jede Kante  $e_{ij}$  nur einmal kreuzt.

<sup>19</sup>Zur Erinnerung: eine Geodätische ist lokal unverkürzbar, eine Kürzeste auch global.

Abbildung 1.55: Pfad  $\pi$  von  $a$  nach  $b$  über  $x \in e$ .

Eine erste Idee ist nun, zunächst nur die Punkte auf den Rändern der Dreiecke des Polyeders zu betrachten und zu einer Kante  $e$  eines Dreiecks  $f$  für alle  $x \in e$  den kürzesten Weg  $\pi_a^x$  von  $a$  nach  $x$  mit  $\pi_a^x \cap \overset{\circ}{f} = \emptyset$  (d.h.  $\pi_a^x$  verläuft außerhalb von  $f$ ) zu berechnen. Hier wären jedoch überabzählbar viele Wege zu berechnen! Abhilfe schafft die Idee, Regionen von Punkten mit kombinatorisch gleichen Kürzesten zum Punkt  $x$  zu bilden. Es genügt, die Kantenfolge  $\mathcal{E} = e_{k,1}, \dots, e_{k,m_k}$  die der Weg ab der "letzten Ecke"  $v = v_{k-1}$  durchlaufen hat zu betrachten, da wir nur *eine* Kürzeste berechnen wollen. Die verschiedenen Kürzesten von  $a$  nach  $v$  brauchen nicht unterschieden zu werden, wir müssen uns nur eine solche merken.

Wegen der Winkelbedingung in Lemma 1.39(i) lassen sich  $v$  und alle Dreiecke ab  $v$ , die der Weg  $\pi_v^x$  durchquert, in die  $f$ -Ebene aufklappen.<sup>20</sup> Dabei wird  $\pi_v^x$  zu einem Liniensegment zwischen  $a$  und  $x$ . Im folgenden bezeichne  $\bar{v}$  das in die  $f$ -Ebene aufgeklappte Bild des Punktes  $v$ .

Fassen wir also alle  $x \in e$  zusammen, für die ein kürzester Weg von  $a$  über  $v$  und  $\mathcal{E}$  den Punkt  $x$  erreicht:

**Definition 1.42** Sei  $e$  eine fest gewählte Kante des Dreiecks  $f$ ,  $v$  eine beliebige Ecke von  $P$  und  $\mathcal{E} = (e_1, \dots, e_m)$  eine beliebige Folge von Kanten von  $P$ . Das Optimalitätsintervall von  $v$  und  $\mathcal{E}$  bzgl.  $e$  und  $f$  ist definiert als

$$I(v, \mathcal{E}) := \left\{ x \in e \mid \exists \text{ Kürzeste } \delta \text{ von } a \text{ nach } x \text{ mit} \right. \\
\left. \begin{array}{l} \bullet \delta \cap \overset{\circ}{f} = \emptyset \\ \bullet \delta \text{ endet mit } v, e_1, \dots, e_m, x \end{array} \right\}.$$

**Lemma 1.43** *Eigenschaften von  $I(v, \mathcal{E})$ :*

- (i) Jede solche Menge  $I$  ist ein Intervall auf  $e$  (evtl. leer).
- (ii) Zwei verschiedene Intervalle können sich nicht überlappen.
- (iii)  $e$  wird von Intervallen  $I(v, \mathcal{E})$  ganz überdeckt.

**Beweis.**

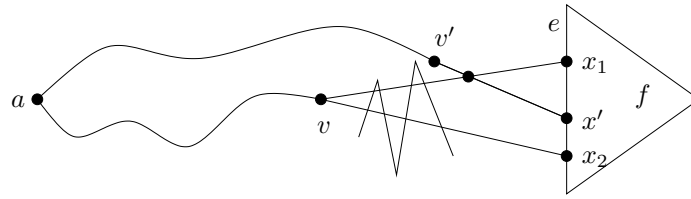
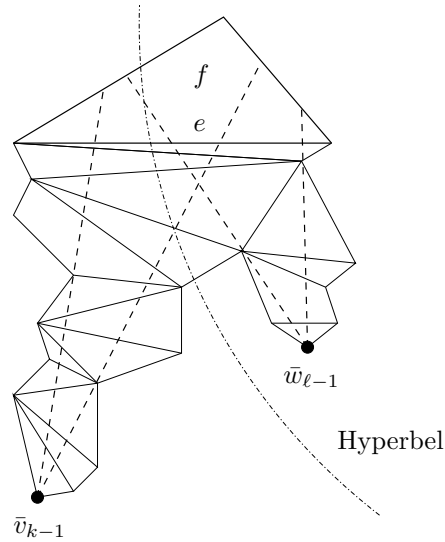
- (i) Zu zeigen:  $I(v, \mathcal{E})$  ist zusammenhängend. Seien  $x_1, x_2 \in e$  in  $I(v, \mathcal{E})$  enthalten, und ein Punkt  $x'$  liege zwischen  $x_1$  und  $x_2$  auf  $e$ , vgl. Abbildung 1.56. Zeige:  $x' \in I(v, \mathcal{E})$ .

Angenommen,  $x' \notin I(v, \mathcal{E})$ , dann klappe die Dreiecke entlang der Kantenfolge  $\mathcal{E}$  in die Fläche von  $f$  auf

$\Rightarrow v' \neq v$ , sogar liegt  $v'$  außerhalb des Dreiecks  $\triangle(v, x_1, x_2)$

$\Rightarrow$  die beiden Kürzesten von  $a$  nach  $x_1/x_2$  und  $x'$  kreuzen sich im Inneren einer Fläche  $\nabla$ .

<sup>20</sup>D.h. eines von je zwei Dreiecken wird solange um die gemeinsame Kante gedreht, bis beide in einer Ebene, eins rechts und eins links der Kante liegen.

Abbildung 1.56:  $I(v, \mathcal{E})$  muß zusammenhängend sein.Abbildung 1.57: Der Bisektor von  $\bar{v}_{k-1}$  und  $\bar{w}_{\ell-1}$  ist eine Hyperbel.

- (ii) Angenommen, zwei Intervalle überlappen sich. Sei  $x$  im Durchschnitt der Intervalle, dann ergibt sich beim Weg von  $a$  nach  $x$  über  $\bar{v}_{k-1}$  eine Weglänge von  $|\pi_a^{v_{k-1}}| + |\bar{v}_{k-1} - x|$ , beim Weg über  $\bar{w}_{\ell-1}$  eine von  $|\pi_a^{w_{\ell-1}}| + |\bar{w}_{\ell-1} - x|$ . Für jeden Punkt  $x$  im Durchschnitt der beiden Intervalle müssen beide Weglängen gleich sein, also muß gelten:

$$|\pi_a^{v_{k-1}}| + |\bar{v}_{k-1} - x| = |\pi_a^{w_{\ell-1}}| + |\bar{w}_{\ell-1} - x|.$$

Durch Umformen dieser Gleichung erhält man die Gleichung einer Hyperbel, die den Bisektor zwischen  $\bar{v}_{k-1}$  und  $\bar{w}_{\ell-1}$  beschreibt, also die Menge der Punkte, für die beide Alternativen gleich sind. Diese Hyperbel hat mit  $e$  jedoch nur einen Schnittpunkt, nämlich den Punkt, an dem sich beide Intervalle berühren. Zwei Intervalle können sich also nicht überlappen.

- (iii) Dies ist klar, da es für jedes  $x \in e$  eine Kürzeste von  $a$  gibt.

□

**Achtung:** Es kann mehrere Intervalle  $I(v_i, \mathcal{E}_j)$  mit derselben “letzten” Ecke  $v_i$  geben, d. h.  $v_i$  kann mehrere unterschiedlich abgewinkelte Kopien  $\bar{v}_{i,k}$  haben.

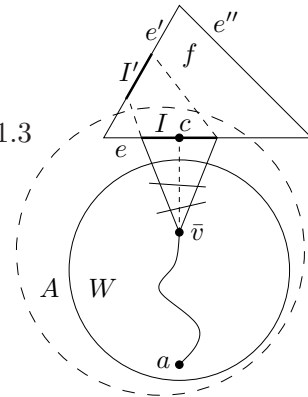
**Algorithmus 1.12** Continuous Dijkstra zur Berechnung der  $I(v, \mathcal{E})$ 

**Datenstruktur:** Prioritätswarteschlange  $W$  von Intervallen mit bisher bekannten Kürzesten von  $a$ .

**Iterationsschritt:**

- Wähle das Intervall  $I = I(v, \mathcal{E})$  aus  $W$ , für das  $d(I) = d(a, v) + |\bar{v} - c|$  minimal ist, wobei  $c$  der zu  $\bar{v}$  nächstgelegene Punkt auf  $I$  ist.
- Propagiere dieses Intervall  $I$  durch das Dreieck auf die beiden gegenüberliegenden Kanten  $e'$  und  $e''$ : Berechne das Intervall  $I'$  auf Kante  $e'$  mit  $I' = \{x \in e' \mid \text{der Weg von } a \text{ über } v \text{ und } I \text{ durch das Dreieck } f \text{ ist kürzer als der kürzeste bisher bekannte Weg zu dem Intervall } I'(v', \mathcal{E}') \text{ auf Kante } e', \text{ das } x \text{ enthält}\}$  und füge dies in die Intervallliste von  $e'$  ein; analog mit  $e''$ .

Im ersten Schritt werden also zu jeder Fläche  $f$  die Intervalle  $I(v, \mathcal{E})$  aller drei Kanten von  $f$  berechnet. Dazu benutzen wir eine Methode, die als Continuous Dijkstra bekannt ist. Allgemein wird bei dieser Methode wie in Algorithmus 1.3 (Dijkstra) auf Seite 14 eine Welle mit einem Ausläufer verwaltet, die sich vom Startpunkt ausbreitet. Erreicht die Welle einen Punkt  $x$  zum ersten Mal, wird  $x$  beschriftet, und die Welle breitet sich von  $x$  weiter aus und propagiert dabei den Wert von  $x$ . In unserem Fall genügen eine endliche Zahl solcher Wellenausbreitungen. Die Welle enthält dabei alle Intervalle mit bekannten Kürzesten, der Ausläufer die Intervalle mit vorläufigen Kürzesten. Propagiert wird das “nächstgelegene” Intervall auf die anderen Dreiecksseiten, dort konkurriert es mit den vorhandenen vorläufigen Intervallen, siehe Algorithmus 1.12,  $I'$  ist dabei wirklich nur *ein* Intervall!



Für jede Kante  $e$  werden die Intervalle  $I$  in einem balancierten Blattsuchbaum gespeichert. Dies sind maximal  $O(n)$  Intervalle je Kante. Um ein neues Intervall  $I'$  der Kante  $e'$  hinzuzufügen, benötigen wir zunächst  $O(\log n)$  Zeit um ein Intervall  $J$  zu finden, für das es Punkte in  $J$  gibt, so dass der kürzeste Weg über  $I(v, \mathcal{E})$  kürzer ist als der bisherige Weg zu diesen Punkten. Von diesem Intervall aus suchen wir alle Nachbarintervalle, die sich ändern bzw. verschwinden. Das können  $O(k)$  viele sein. Weil wir einen Blattsuchbaum verwenden benötigen wir dafür  $O(k)$  Schritte und für das Entfernen jeweils  $O(\log n)$  Zeit. Am Ende wird ein neues Intervall eingefügt.

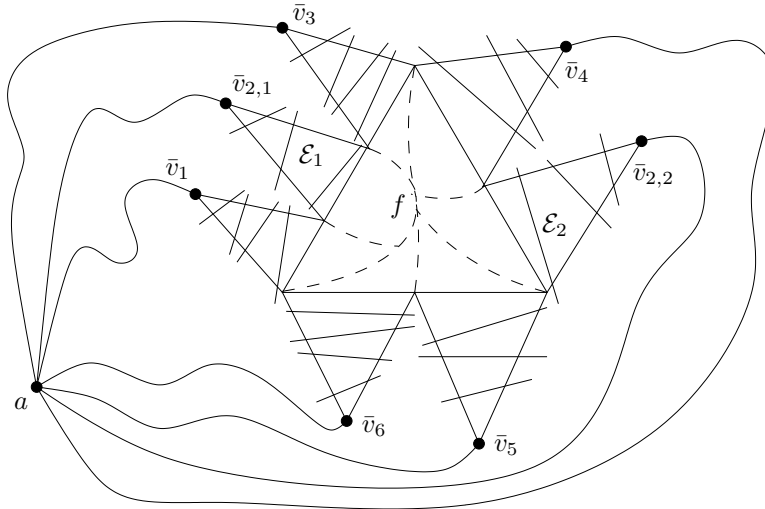
Der Dijkstra-Algorithmus wählt stets das Intervall mit dem momentan kürzesten Weg zu einem Intervall aus, und dieses Intervall wird nie mehr verschwinden. Pro Schritt werden nun maximal 2 neue Intervalle eingefügt. Insgesamt kann es nur  $O(n^2)$  Intervalle geben, also kann der Dijkstra insgesamt nicht mehr als  $O(n^2)$  Intervalle erzeugen. Das Einfügen und Entfernen von Intervallen wie oben beschrieben benötigt also insgesamt  $O(n^2 \log n)$  Zeit. Das Ablaufen der Intervalllisten der Blattsuchbäume benötigt nicht mehr als  $O(n^2)$  Zeit. Alle Blattsuchbäume zusammen haben einen Speicherplatzbedarf von  $O(n^2)$ .

Im zweiten Schritt werden die durch diese Intervalle vorhandenen Weginformationen in das Innere der Dreiecke fortgesetzt. Die Intervalle auf den drei Kanten von  $f$  — genauer: ihre “Quellen”  $\bar{v}_i$  — teilen sich das Innere von  $f$  nach einer gewichteten Distanzfunktion auf: Setze  $a_i := d(a, v_i)$ . Ein Punkt  $x$  auf der Dreiecksfläche wird dem Punkt  $\bar{v}_i$  zugeordnet, über den der Weg von  $a$  nach  $x$  minimal wird, für den also gilt:

$$a_i + |\bar{v}_{i,k} - x| < \min_{\substack{j \neq i \\ \forall \ell \neq k}} \left\{ a_j + |\bar{v}_{j,\ell} - x| \right\}.$$

Die  $\bar{v}_{i,k}$  teilen also die Fläche  $f$  untereinander auf. Die Grenzen der von den  $\bar{v}_{i,k}$  definierten



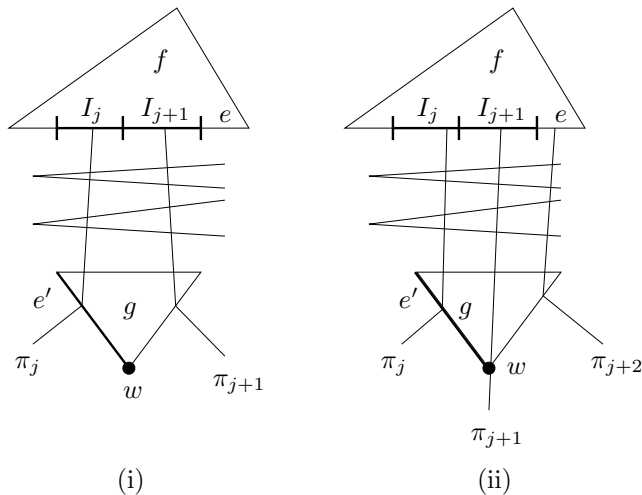
Abbildung 1.58: Dreiecksfläche  $f$  mit Intervallen  $I(v, \mathcal{E})$ .

Regionen von  $f$ , die Bisektoren, sind Hyperbelbögen, da die Punkte auf dem Bisektor zwischen  $\bar{v}_{i,k}$  und  $\bar{v}_{j,\ell}$  gleiche Weglängen haben müssen; durch die Gleichung  $a_i + |\bar{v}_{i,k} - x| = a_j + |\bar{v}_{j,\ell} - x|$  wird eine Hyperbel beschrieben.

Diese Unterteilung der Fläche  $f$  entspricht einem Voronoi-Diagramm der Ecken  $\bar{v}_{i,k}$  mit additiven Gewichten  $a_i$ , siehe Anhang. Dies kann für  $\ell$  Intervalle auf den Kanten von  $f$  in Zeit  $O(\ell \log \ell)$  und Speicherplatz  $O(\ell)$  konstruiert werden. Doch wieviele nicht-leere Intervalle kann es auf einer Kante geben?

**Lemma 1.44** Die Anzahl  $\ell$  der nicht-leeren Intervalle auf einer Kante  $e$  ist in  $O(n)$ .

**Beweis.**

Abbildung 1.59: Ein Dreieck  $g$  mit Ecke  $w$  fungiert höchstens zweimal als Trenner für zwei Intervalle.

Seien  $I_j = I_j(v_j, \mathcal{E}_j)$  und  $I_{j+1} = I_{j+1}(v_{j+1}, \mathcal{E}_{j+1})$  zwei benachbarte Intervalle auf  $e$ . Wähle je einen inneren Punkt pro Intervall und verfolge die definierenden Kürzesten  $\pi_j$  und  $\pi_{j+1}$ , die zu  $I_j$  und  $I_{j+1}$  gehören, rückwärts.  $\pi_j$  und  $\pi_{j+1}$  können zunächst dieselben Kanten im Inneren kreuzen, d. h.  $\mathcal{E}_j$  und  $\mathcal{E}_{j+1}$  können gemeinsame Endstücke haben, müssen sich jedoch wegen  $(v_j, \mathcal{E}_j) \neq (v_{j+1}, \mathcal{E}_{j+1})$  einmal trennen. Betrachte das Dreieck  $g$  und die Ecke  $w$ , an der sich die Trennung vollzieht: das Paar  $(g, w)$  fungiert höchstens zweimal als Trenner, siehe Abbildung 1.59(ii), und

jedem Paar läßt sich eindeutig eine Kante  $e'$  zuordnen, die  $g$  berandet und im Uhrzeigersinn  $w$  verläßt. Wir haben also eine Abbildung  $(I_j, I_{j+1}) \mapsto (g, w)$ , die “fast injektiv” ist — jedes  $(g, w)$  hat höchstens 2 Urbilder — und eine injektive Abbildung  $(g, w) \mapsto e'$ . Also gilt  $\ell \leq 2n \in O(n)$ .  $\square$

Der Beweis zu Lemma 1.44 benutzt die häufig angewandte Technik, die Kosten für eine Berechnung den geometrischen Objekten, hier den Kanten, zuzuordnen.

---

**Algorithmus 1.13** Berechnung kürzester Wege auf Polyedern
 

---

**Input:** Polyeder  $P$  mit  $n$  Ecken, Oberfläche sei trianguliert, Startecke  $a$ .

**Output:** Für beliebigen Anfragepunkt  $b \in \partial P$ : Entfernung und kürzesten Weg von  $a$  nach  $b$ .

**Vorbereitungsphase:**

- Berechne für jede Dreieckskante die Intervalle  $I(v, \mathcal{E})$  mit Continuous Dijkstra.  $O(n^2 \log n)$
- Berechne für jede Dreiecksfläche das durch die Intervalle  $I(v, \mathcal{E})$  der drei Kanten bestimmte Voronoi-Diagramm mit additiven Gewichten (siehe Anhang A.3.4).  $O(n^2 \log n)$
- Bereite die Voronoi-Diagramme auf den Flächen für Lokalisierung von Punkten vor (z. B.: Trapezzerlegung, [Sei91]).  $O(n^2)$ , Platz  $O(n^2)$

**Query** für Zielpunkt  $b$  in Dreieck  $f$ :

- Bestimme die Voronoi-Region in  $f$ , die den Punkt  $b$  enthält  
 $\implies$  Abstand zu  $a$  ist bekannt.  $O(\log n)$
- Setze die Kürzeste von  $a$  nach  $b$  zusammen aus:
  - einer Kürzesten von  $a$  nach  $v$  (Nebenprodukt des Continuous Dijkstra: in  $v$  verankerte, verkettete Liste)
  - dem “geraden” Weg durch die Kantenfolge  $e$  zum Punkt  $x \in I$
  - dem Liniensegment  $\overline{xb}$ .

Bei  $k$  Segmenten auf dem kürzesten Weg:  $O(k)$

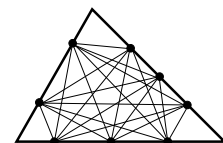
---

Zusammenfassend erhalten wir Algorithmus 1.13 und folgendes Theorem:

**Theorem 1.45** (Mitchell, Mount, Papadimitriou, 1986)

Sei  $s \in P$  fest, o. E. eine Ecke von  $P$ , gegeben  $b \in P$ . Die Entfernung bzw. die Kürzeste von  $a$  nach  $b$  in  $P$  läßt sich nach Vorbereitungszeit  $O(n^2 \log n)$  in Zeit  $O(\log n + k)$  und Platz  $O(n^2)$  berechnen; dabei ist  $k$  die Anzahl der Segmente auf dem kürzesten Pfad. [MMP87]

Ein praxistauglicher Ansatz zur  $(1+\varepsilon)$ -Approximation kürzester Wege auf Polyedern stammt von J. Sack.<sup>21</sup> Hierbei werden in die Kanten der Dreiecke geeignet viele und adäquat verteilte Stützpunkte eingefügt und daraus ein vollständiger Graph gebildet, auf dem der Algorithmus von Dijkstra angewendet wird. Dies liefert gute Approximationsergebnisse und erlaubt die Einhaltung von Nebenbedingungen wie maximale Steigung, maximaler Böschungswinkel, z. B. bei der Berechnung von Wegen für Fahrzeuge. [ALMS98]




---

<sup>21</sup><http://www.scs.carleton.ca/~sack/>

## 1.5 Kürzeste Pfade für Liniensegmente

Bisher hatten wir nur kürzeste Pfade für Punkte betrachtet. Zum Abschluß des ersten Kapitels wollen wir kurz auf die Probleme eingehen, die bei der Berechnung kürzester Pfade für ausgedehnte Objekte entstehen. Schon für Liniensegmente im  $\mathbb{R}^2$  ohne Hindernisse ist das Problem nicht trivial. Die Bewegung von Liniensegmenten im  $\mathbb{R}^2$  kann man als Modell für einen Balken oder eine Leiter ansehen, die von zwei Trägern an je einem Ende getragen wird.

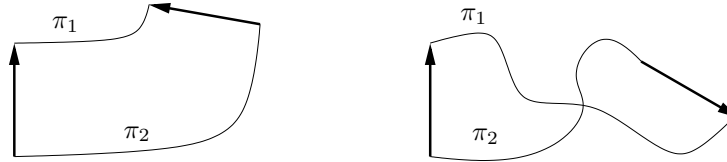


Abbildung 1.60: Bewegung eines Liniensegmentes.

Bei ausgedehnten Objekten ist zunächst einmal festzulegen, was eine Kürzeste ist bzw. wie sich die Kosten für eine Bewegung berechnen. In der Anschauung mit dem Balken sind sicherlich die Wege interessant, die die beiden Träger zurücklegen.<sup>22</sup> Eine Möglichkeit ist also, die Längen der von den Endpunkten zurückgelegten Bahnen zu summieren und diese Summe zu minimieren. Für Liniensegmente gilt damit für eine Bewegung  $B$ :

$$\text{Kosten}(B) := \text{Länge}(\pi_1) + \text{Länge}(\pi_2).$$

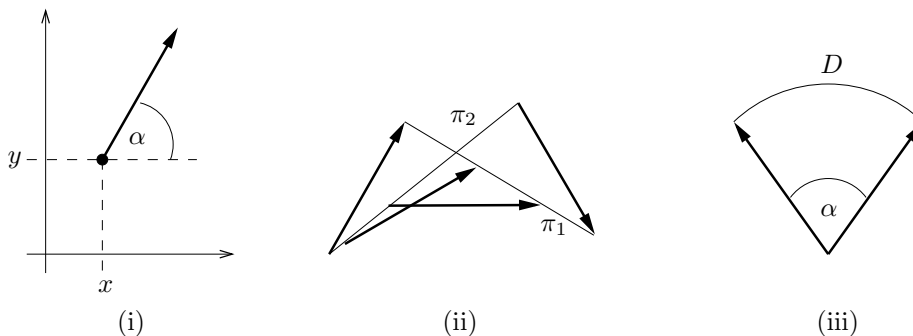


Abbildung 1.61: (i) Freiheitsgrade eines Liniensegmentes, (ii) geradlinige Bewegung (iii) Rotation.

Ein Liniensegment hat drei Freiheitsgrade, siehe Abbildung 1.61(i): die Koordinaten seines Startpunktes und sein Drehwinkel; seine Position läßt sich also durch das Tripel  $A = (x, y, \alpha) \in \mathcal{C}$  mit  $\mathcal{C} = \mathbb{R} \times \mathbb{R} \times [0, 2\pi)$  beschreiben.

Unser Problem läßt sich damit folgendermaßen beschreiben: gegeben zwei Positionen  $A, A' \in \mathcal{C}$ , bestimme eine Bewegung von  $A$  nach  $A'$  mit minimalen Kosten.

Doch welche Bewegungen sind minimal? Die geradlinige Bewegung in Abbildung 1.61(ii) ist sicherlich optimal, da  $\pi_1$  und  $\pi_2$  Geraden sind und nicht kürzer sein können. Auch die Rotation in Abbildung 1.61(iii) ist optimal, zum Beweis dieser vermeintlich einfachen Tatsache brauchen wir das folgende Theorem:

**Theorem 1.46** (*Surface–Area Formel von Cauchy*)  
Sei  $C$  eine geschlossene konvexe Kurve in der Ebene. Mit

$$h_C(\alpha) := \sup \{ x \cos \alpha + y \sin \alpha \mid (x, y) \in C \},$$

$$\text{dia}_C(\alpha) := h_C(\alpha) + h_C(\alpha + \pi), \quad \alpha \in [0, \pi)$$

<sup>22</sup>Je schwerer der Balken ist, umso größer dürfte das Interesse der beiden Träger an kürzesten Wegen sein.

gilt für die Länge von  $C$ :

$$\text{Länge}(C) = \int_0^\pi \text{dia}_C(\alpha) d\alpha.$$

Dabei ist  $\text{dia}_C(\alpha)$  anschaulich die Messung von  $C$  mit einer um den Winkel  $\alpha$  zur X-Achse gedrehten Schieblehre, siehe Abbildung 1.62(i).

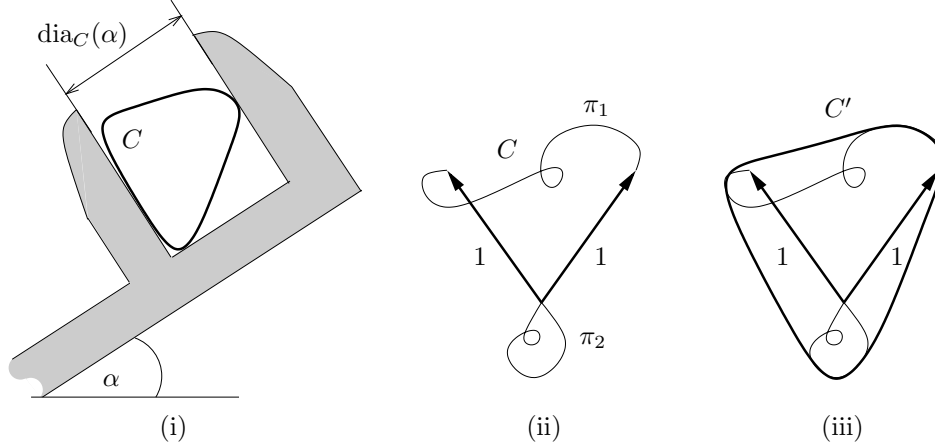


Abbildung 1.62: (i)  $\text{dia}_C(\alpha)$ , (ii) Bewegung  $B$ , (iii) konvexe Hülle von  $C$ .

Nehmen wir nun an, eine beliebige andere Bewegung  $B$  mit Pfaden  $\pi_1$  und  $\pi_2$  sei optimal, vgl. Abbildung 1.62(ii). Sei  $C$  die Verknüpfung der beiden Kurven  $\pi_1$  und  $\pi_2$  und der beiden Liniensegmente (o. E. sei die Länge der Liniensegmente 1).  $C'$  sei die konvexe Hülle von  $C$ , siehe Abbildung 1.62(iii).  $D$  sei die Kurve der Rotation (Abbildung 1.61(iii)). Dann gilt:

$$\begin{aligned} 2 + \text{Kosten}(B) &= \text{Länge}(C) \\ &\geq \text{Länge}(C') \\ &= \int_0^\pi \text{dia}_{C'}(\alpha) d\alpha \quad \text{nach Theorem 1.46} \\ &\geq \int_0^\pi \text{dia}_D(\alpha) d\alpha \quad \text{wegen } \text{dia}_{C'}(\alpha) \geq \text{dia}_D(\alpha) (*) \\ &= \text{Länge}(D) \\ &= 2 + \text{Kosten}(R) \end{aligned}$$

Begründung (\*): Im *kritischen* Winkelbereich für  $\alpha$  gilt  $\text{dia}_D(\alpha) = 1$ . Da auch bei beliebiger Bewegung  $C$  der Winkel  $\alpha$  (oder  $-\alpha$ ) angenommen werden muss (irgendwo muss das Segment ja liegen), gilt für alle Winkel in diesem Bereich  $\text{dia}_{C'}(\alpha) \geq 1$ , siehe Abbildung 1.63.

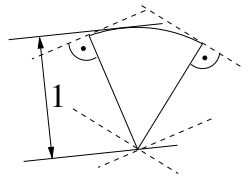


Abbildung 1.63:  $\text{dia}_D$  im *kritischen* Winkelbereich.

Also ist eine Rotation um einen Winkel  $\leq \pi$  optimal.

Für sich betrachtet sind also Rotation um einen Punkt auf dem Liniensegment und Translation optimal, doch wie verhält es sich mit Kombinationen davon? Es gilt:

**Theorem 1.47** (*Icking, Rote, Welzl, Yap, 1989*)

*Zwischen je zwei Positionen von Liniensegmenten gibt es eine optimale Bewegung von einem der folgenden Typen:*

- (i) maximal drei Rotationen,*
- (ii) maximal zwei Rotationen und eine geradlinige Bewegung,*
- (iii) eine Rotation zwischen zwei geradlinigen Bewegungen.*

*Die Teilbewegungen lassen sich effektiv berechnen.*

*[IRWY93]*

Das Problem der kürzesten Pfade von Liniensegmenten wurde von Icking et. al. wiederentdeckt. Erwähnt wurde es bereits von Dubovitskij [Dub85] und Gurevich [Gur75].

Fassen wir abschließend die Ergebnisse aus Kapitel 1 zusammen:

- Kürzeste Pfade für Punkte im  $\mathbb{R}^2$  lassen sich effizient berechnen.
- Kürzeste Pfade für Punkte im  $\mathbb{R}^d$ ,  $d > 2$  zu finden ist NP-hard.
- Kürzeste Pfade für ausgedehnte Objekte zu finden ist schwierig.

Deshalb werden wir uns im folgenden Kapitel auf Kollisionsfreiheit konzentrieren.



## Kapitel 2

# Kollisionsfreie Bahnen für polygonale Roboter

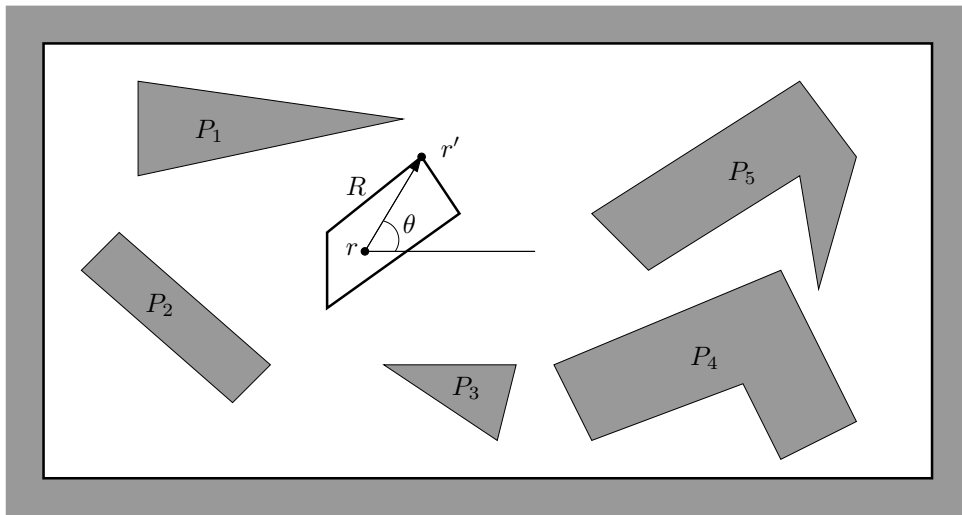


Abbildung 2.1: Roboter  $R$  in einem Arbeitsraum.

Wir betrachten nun nicht länger punktförmige Roboter, sondern ausgedehnte Objekte. Wir modellieren den Roboter  $R$  durch ein abgeschlossenes Polygon — konvex oder allgemein — mit  $m$  Kanten. Der Arbeitsraum des Roboters sei eine beschränkte Umgebung mit  $h$  polygonalen Hindernissen  $P_i$  mit insgesamt  $n$  Ecken, siehe Abbildung 2.1.

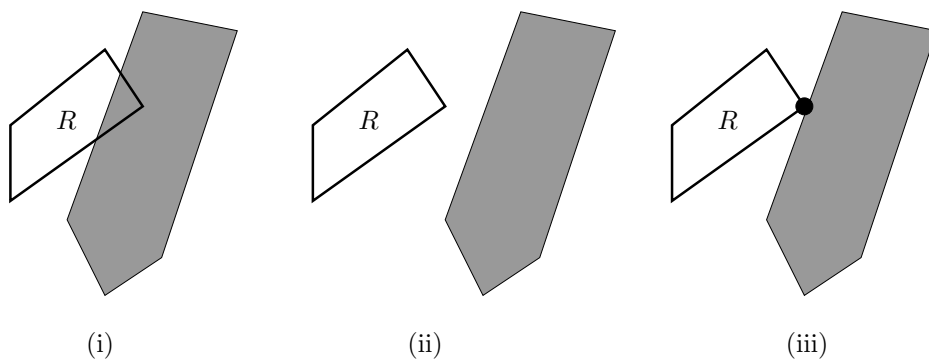


Abbildung 2.2: (i) Verbotene Konfiguration, (ii) freie Platzierung (erlaubt), (iii) halbfreie Platzierung (erlaubt).

Zur Diskussion kollisionsfreier Bahnen wollen wir einige Begriffe definieren:

**Definition 2.1**

- (i) Eine **Plazierung** oder **Konfiguration** ist eine Position des Roboters im Arbeitsraum. Zur eindeutigen Beschreibung der Position legen wir einen Referenzpunkt  $r = (x, y)$  fest, sowie einen zweiten Punkt  $r'$  als Winkelreferenz. Die Plazierung läßt sich dann durch die Angabe des Referenzpunktes  $r$  und des Winkels  $\theta$  zwischen der X-Achse und dem Vektor  $\overrightarrow{rr'}$  als  $R(x, y, \theta)$  mit  $(x, y, \theta) \in \mathbb{R} \times \mathbb{R} \times [0, 2\pi)$  angeben. Der Roboter hat damit drei Freiheitsgrade (Degree of Freedom, DOF).
- (ii) Der **Konfigurationsraum** ist der Raum der möglichen Plazierungen des Roboters, hier  $\mathcal{C} = \mathbb{R} \times \mathbb{R} \times [0, 2\pi)$ .
- (iii) Die **verbotenen Konfigurationen** sind solche, bei denen der Roboter in ein Hindernis eindringt. Das Berühren einer Hinderniswand dagegen ist erlaubt (**halbfreie Plazierung**), vgl. Abbildung 2.2.<sup>1</sup>

$$\mathcal{C}_{\text{verb}} := \left\{ c \in \mathcal{C} \mid R(c) \cap \bigcup \overset{\circ}{P}_i \neq \emptyset \right\}$$

- (iv) Der **Raum der freien Plazierungen** enthält alle erlaubten Konfigurationen

$$\mathcal{C}_{\text{frei}} := \mathcal{C} \setminus \mathcal{C}_{\text{verb}}$$

Damit haben wir folgendes Bahnplanungsproblem: Gegeben  $s, t \in \mathcal{C}_{\text{frei}}$ . Gibt es einen Weg von  $s$  nach  $t$ , der ganz in  $\mathcal{C}_{\text{frei}}$  verläuft? Wenn ja, bestimme einen.

Die Frage nach der Existenz eines Weges ist äquivalent zu der Frage, ob  $s$  und  $t$  in derselben Zusammenhangskomponente von  $\mathcal{C}_{\text{frei}}$  liegen. Zur Erinnerung:

**Definition 2.2** Eine Menge  $Z$  (Teilmenge eines topologischen Raumes  $T$ ) heißt genau dann **weg-zusammenhängend**, wenn

$$\forall a, b \in Z : \exists \text{ Pfad } \pi \text{ von } a \text{ nach } b \text{ in } Z.$$

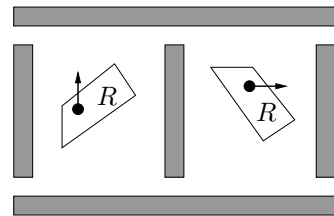
Der Pfad  $\pi$  ist dabei gegeben durch eine stetige Abbildung  $\pi : [0, 1] \rightarrow Z$  mit  $\pi(0) = a$  und  $\pi(1) = b$ .

**Lemma 2.3** Sei  $A \subset T$  beliebig, dann wird für alle  $a, b \in A$  durch

$$a \sim b :\Leftrightarrow \exists \text{ Pfad } \pi \subset A \text{ von } a \text{ nach } b \text{ in } A \text{ mit } \pi(0) = a \text{ und } \pi(1) = b$$

eine Äquivalenzrelation definiert. Die Äquivalenzklassen  $Z_i$  von  $A$  bzgl.  $\sim$  heißen **Zusammenhangskomponenten** von  $A$ . Sie sind die größten zusammenhängenden Teilmengen und überdecken  $A$ :  $A = \bigcup \overset{\circ}{Z}_i$ .<sup>2</sup>

Natürlich kann  $\mathcal{C}_{\text{frei}}$  aus mehreren Zusammenhangskomponenten bestehen. In nebenstehender Abbildung kann  $R$  sich in einem der beiden Räume bewegen,  $\mathcal{C}_{\text{frei}}$  hat zwei Zusammenhangskomponenten. Das Bahnplanungsproblem ist unlösbar, wenn  $s \in Z_i$  und  $t \in Z_j$  mit  $i \neq j$ .



Somit stellen sich uns folgende algorithmische Probleme:

- Bestimme die Zusammenhangskomponenten von  $\mathcal{C}_{\text{frei}}$  bzw. die Komponente  $Z_s$ , die  $s$  enthält.
- Bestimme, ob  $t$  in  $Z_s$  liegt. Falls ja, berechne einen Weg  $\pi \subseteq Z_s$  mit  $\pi(0) = a$  und  $\pi(1) = b$ .

<sup>1</sup>Zur Erinnerung:  $\overset{\circ}{P}$  bezeichnet das Innere des Polygons.

<sup>2</sup> $A \overset{\circ}{\cup} B$  bezeichne die **disjunkte Vereinigung**  $A \overset{\circ}{\cup} B = A \cup B$  mit  $A \cap B = \emptyset$ .



## 2.1 Reine Translationsbewegungen für konvexe Roboter

Beschränken wir uns zunächst auf den Spezialfall eines *konvexen* Roboters mit  $m$  Kanten, der sich in einer Umgebung aus  $h$  Polygonen  $P_i$  mit insgesamt  $n$  Kanten und  $\forall i \neq j : \overset{\circ}{P}_i \cap \overset{\circ}{P}_j = \emptyset$  bewegt. Außerdem beschränken wir uns auf reine Translationsbewegungen, der Roboter darf also nur verschoben, aber nicht gedreht werden. Die Ausrichtung des Roboters ist stets gleich, damit hat der Roboter hier nur zwei Freiheitsgrade, und seine Platzierung wird durch  $R(x, y) \in \mathcal{C} = \mathbb{R} \times \mathbb{R}$  eindeutig beschrieben.

### 2.1.1 Voronoi-Diagramme, Translationsbewegungen mit Sicherheitsabstand

Sei zunächst der Roboter  $R$  durch einen Kreis gegeben. Kreisförmige Roboter haben die Vorteile, daß man keine Rotationsbewegungen zu betrachten hat, wenn als Referenzpunkt der Kreismittelpunkt gewählt wird, und sie außerdem als Approximation für beliebige Roboter dienen können, siehe nebenstehende Abbildung. Dabei werden allerdings nicht alle Wege gefunden; man kann sich leicht Umgebungen vorstellen, in denen der Roboter  $R$  eine Lücke zwischen zwei Hindernissen passieren kann, während der kreisförmige Roboter zu groß dafür ist.

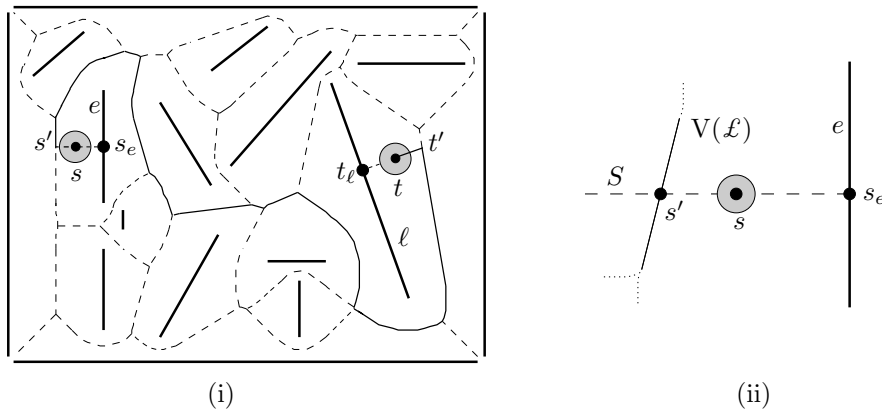
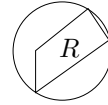


Abbildung 2.3: (i) Bahnplanung im Voronoi-Diagramm für Liniensegmente (ii) Konstruktion von  $s'$ .

Das Problem läßt sich lösen, indem wir, wie in Abschnitt A.3.3 beschrieben, das Voronoi-Diagramm von Liniensegmenten bzgl. der  $L_2$ -Metrik mit den Kanten der Hindernisse als Liniensegmente konstruieren, siehe Abbildung 2.3. Wie in Algorithmus 2.1 beschrieben reduziert sich damit die Suche nach einem kollisionsfreien Weg von  $s$  nach  $t$  auf die Suche in dem planaren Graphen  $V(\mathcal{L})$  von  $s'$  nach  $t'$ . Wegen  $s, t \in \mathcal{C}_{\text{frei}}$  existiert eine Bewegung von  $s$  nach  $s'$  und von  $t'$  nach  $t$ .

Mit dieser Methode erhält man Pfade mit maximalem Sicherheitsabstand. Außerdem läßt sie sich auf die Translation konvexer Roboter verallgemeinern, indem man mit dem Spiegelbild von  $R$  eine konvexe Distanzfunktion definiert und das Voronoi-Diagramm von Liniensegmenten bzgl. dieser konvexen Distanzfunktion berechnet. Nach einer Vorbereitungszeit von  $O(mn \log n)$  ermöglicht dies, eine Anfrage in Zeit  $O(mn)$  zu beantworten, siehe [Ma00]. Auf nicht-konvexe Roboter, Roboter mit mehreren Freiheitsgraden oder höhere Dimensionen läßt sich diese Methode nicht effektiv verallgemeinern. Das Prinzip der Reduktion auf einen eindimensionalen Graphen läßt sich jedoch verallgemeinern, wir werden später darauf zurückkommen. Der Ansatz, in einer erzeugten Karte einen Pfad zu suchen, ist als Roadmap-Approach bekannt.

**Algorithmus 2.1** Bahnplanung mit Sicherheitsabstand für kreisförmige Roboter**Vorbereitung:**

- Konstruiere das Voronoi-Diagramm  $V(\mathcal{L})$  mit den Kanten der Hindernisse als Liniensegmente.  $O(n \log n)$
- Bereite das Voronoi-Diagramm für Point-Location vor.  
mit Trapezzerlegung  $O(n \log n)$

**Query** für zwei Punkte  $s, t$ :

- Lokalisier  $s, t$  im Voronoi-Diagramm.  $O(\log n)$
- Sei  $s \in VR(e, \mathcal{L})$ ,  $t \in VR(\ell, \mathcal{L})$ :
- Bestimme den zu  $s$  nächstgelegenen Punkt  $s_e$  auf  $e$ .
- Bestimme Strahl  $S$  von  $s_e$  durch  $s$ .
- $s'$  sei Schnittpunkt von  $S$  mit  $V(\mathcal{L})$ ; analog  $t'$ .
- Berechne mit Breitensuche in  $V(\mathcal{L})$  einen Pfad von  $s'$  nach  $t'$  mit zulässigem Mindestabstand oder berichte, daß es keinen solchen gibt.  $O(n)$

**2.1.2 Berechnung von  $\mathcal{C}_{\text{frei}}$** 

Ein anderer Ansatz zur Berechnung reiner Translationsbewegungen konvexer Roboter ist, den Raum der freien Konfigurationen  $\mathcal{C}_{\text{frei}}$  explizit zu berechnen. Nehmen wir zunächst an, die Hindernisse seien konvex. Betrachten wir ein Hindernis  $P_i$  und untersuchen, welche Positionen  $R(x, y)$  aufgrund der  $P_i$  verboten sind. Von Lozano-Pérez stammt die Idee, die Hindernisse so zu expandieren, daß in der entstehenden Szene das Bahnplanungsproblem für einen Punkt zu lösen ist [LP83].

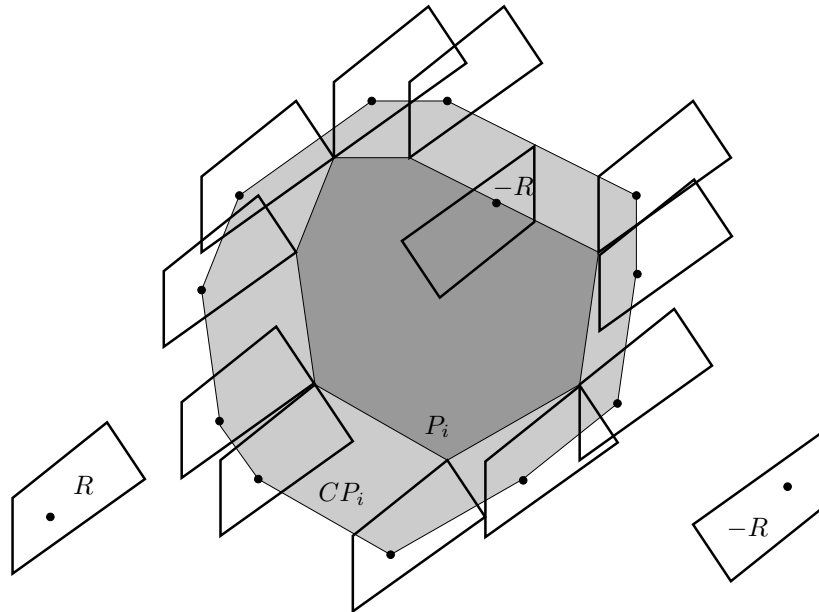


Abbildung 2.4: Hindernis, Roboter und Konfigurationsraum-Hindernis.

Das zu  $P_i$  gehörende expandierte Hindernis  $CP_i$  (Konfigurationsraum-Hindernis), ist definiert als

$$CP_i := \{c \in \mathcal{C} \mid R(c) \cap P_i \neq \emptyset\}.$$

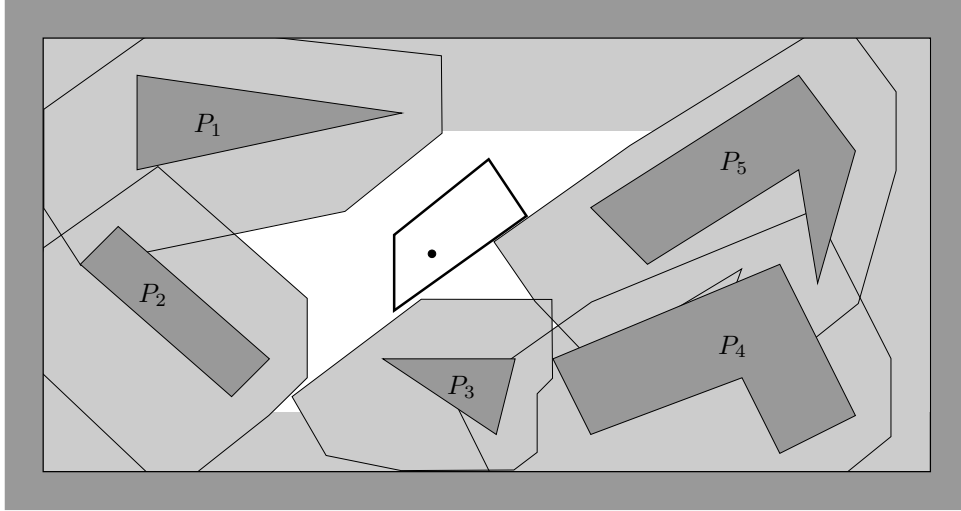


Abbildung 2.5: Konfigurationsraum-Hindernisse zu Abbildung 2.1.

Anschaulich ist der Rand von  $CP_i$  die Bahn des Referenzpunktes die entsteht, wenn der Roboter  $R$  um  $P_i$  wandert, vgl. Abbildung 2.4.<sup>3</sup> Abbildung 2.5 zeigt die Konfigurationsraum-Hindernisse zu Abbildung 2.1. Man beachte, daß  $C_{\text{frei}}$  hier aus zwei Komponenten besteht! Anstatt den Rand von  $R$  um den Rand von  $P_i$  herumzuschieben, kann man auch den Referenzpunkt des am Ursprung gespiegelten Roboters  $-R = -R(0, 0)$  um den Rand von  $P_i$  schieben. Das Konfigurationsraum-Hindernis zu  $P_i$  ergibt sich dann aus der Vereinigung von  $P_i$  und allen Positionen von  $-R$ , bei denen der Referenzpunkt auf dem Rand von  $P_i$  liegt. Dies entspricht genau der Minkowski-Summe von  $P_i$  und  $-R$ .

**Definition 2.4** Die **Minkowski-Summe** zweier Mengen  $A, B \subset \mathbb{R}^2$  ist

$$A \oplus B := \{a + b \mid a \in A, b \in B\}.$$

**Bemerkung 2.5** *Minkowski-Summen sind*

- (i) *kommutativ:*  $A \oplus B = B \oplus A$ ,
- (ii) *assoziativ:*  $A \oplus (B \oplus C) = (A \oplus B) \oplus C$ ,
- (iii) *distributiv bzgl. der Vereinigung:*  $A \oplus (B \cup C) = (A \oplus B) \cup (A \oplus C)$ .

**Lemma 2.6** *Das zu  $P_i$  gehörende Konfigurationsraum-Hindernis bzgl.  $R$  ist*

$$CP_i = P_i \oplus (-R(0, 0)).$$

**Beweis.**

Zeige:  $\forall x, y : R(x, y) \cap P_i \neq \emptyset \iff (x, y) \in P_i \oplus (-R(0, 0))$ .

“ $\implies$ ” Sei  $q = (q_x, q_y) \in R(x, y) \cap P_i$

$$\begin{aligned} \Rightarrow (q_x, q_y) &\in R(x, y) \\ \Rightarrow (q_x - x, q_y - y) &\in R(0, 0) \\ \Rightarrow (-q_x + x, -q_y + y) &\in -R(0, 0). \end{aligned}$$

Außerdem gilt  $(q_x, q_y) \in P_i$ , insgesamt also

$$(x, y) = (-q_x + x, -q_y + y) + (q_x, q_y) \in -R(0, 0) \oplus P_i.$$

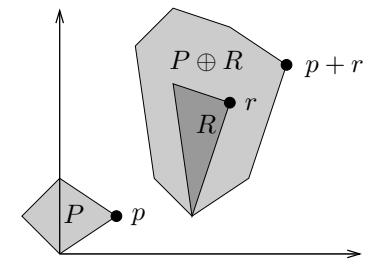
<sup>3</sup>In Anhang B findet sich eine Seite mit Robotern zum Ausschneiden und Nachvollziehen der in diesem Kapitel vorgestellten Beispiele.

“ $\Leftarrow$ ” Sei  $(x, y) \in P_i \oplus -R(0, 0)$   
 $\Rightarrow \exists (q_x, q_y) \in P_i, (r_x, r_y) \in R(0, 0) :$   
 $(x, y) = (q_x, q_y) - (r_x, r_y) = (q_x - r_x, q_y - r_y)$   
 $\Rightarrow (q_x, q_y) = (x, y) + \underbrace{(r_x, r_y)}_{\in R(0, 0)} \in R(x, y) \cap P_i. \quad \square$

Minkowski-Summen treten also bei der Bewegungsplanung in natürlicher Weise auf. Betrachten wir deshalb einige Eigenschaften von Minkowski-Summen.

**Definition 2.7** Gegeben sei ein Richtungsvektor  $d$  mit  $|d| = 1$ . Ein Punkt  $p \in P$  ist **Extremalpunkt** von  $P$  in Richtung  $d$ , wenn folgendes gilt: Die zu  $d$  orthogonale Gerade  $g$  durch  $p$  teilt die Ebene in zwei Halbebenen, von denen die eine zusammen mit  $g$  ganz  $P$  enthält und die Andere den Punkt  $p + d$ .

**Bemerkung 2.8** *Extremalpunkte der Minkowski-Summe  $P \oplus R$  sind Summen von Extremalpunkten von  $P$  und  $R$ .*

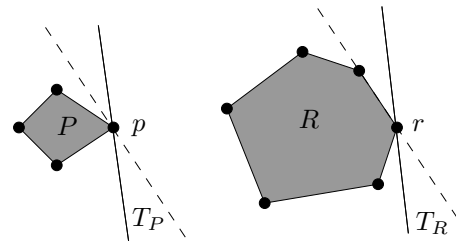


**Lemma 2.9** Seien  $P$  und  $R$  konvexe Polygone mit  $n$  bzw.  $m$  Kanten. Dann ist  $P \oplus R$  konvex, hat maximal  $n + m$  Kanten und kann in Zeit  $O(n + m)$  berechnet werden.

**Beweis.**

Die Konvexität folgt aus der Definition der Minkowski-Summe.

Rotiere simultan zwei parallele Tangenten  $T_P, T_R$  um  $P$  und  $R$ . Wenn  $T_P$  eine Ecke  $p$  von  $P$  berührt und gleichzeitig  $T_R$  eine Ecke  $r$  von  $R$ , dann ist  $p + r$  eine Ecke von  $P \oplus R$ . Alle Ecken von  $P \oplus R$  entstehen auf diese Weise. Da alle Ecken von  $P$  und  $R$  genau einmal betrachtet werden, braucht die Rotation Zeit  $O(n + m)$ .



□

**Definition 2.10** Zwei durch Jordankurven<sup>4</sup> berandete Mengen  $A, B \subseteq \mathbb{R}^2$  heißen ein Paar **Pseudokreise** (pseudo disks), wenn ihre Ränder entweder höchstens zwei echte Kreuzungen oder höchstens einen Berührungspunkt aufweisen.

Abbildung 2.6(i) und (ii) veranschaulicht die Definition. Beispiele für Pseudokreise sind Kreise, Quadrate und achsenparallele Rechtecke mit festem Seitenverhältnis (Abbildung 2.7(i)–(iii)). Beliebige Rechtecke sind im allgemeinen keine Pseudokreise (Abbildung 2.7(iv)). Folgendes Lemma gibt eine äquivalente Charakterisierung von Pseudokreisen; die Konvexität ist dabei eine notwendige Forderung, wie Abbildung 2.6(iii) zeigt.

**Lemma 2.11** Seien  $A, B$  konvex.  $A, B$  sind genau dann Pseudokreise, wenn  $A \setminus B$  und  $B \setminus A$  zusammenhängend sind.

**Beweis.** Übungsaufgabe. □

<sup>4</sup>Eine geschlossene Kurve, die die Ebene in zwei Gebiete unterteilt.

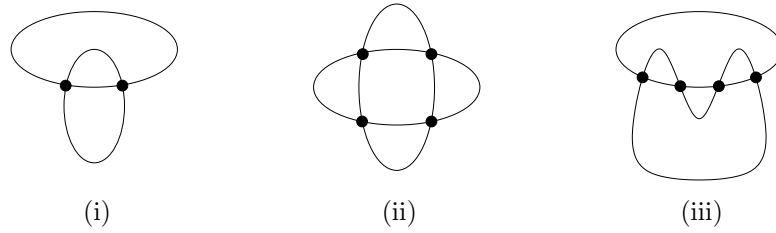


Abbildung 2.6: (i) Pseudokreise, (ii) keine Pseudokreise, (iii) die Konvexität ist eine notwendige Forderung.

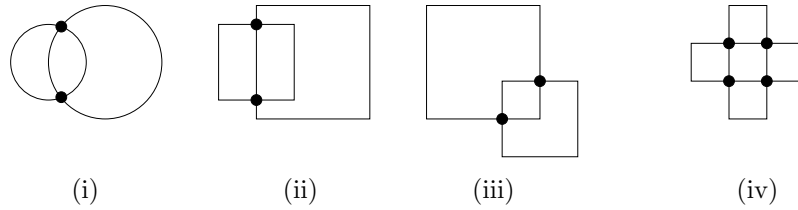


Abbildung 2.7: (i)–(iii) Beispiele für Pseudokreise, (iv) Rechtecke sind im allgemeinen keine Pseudokreise.

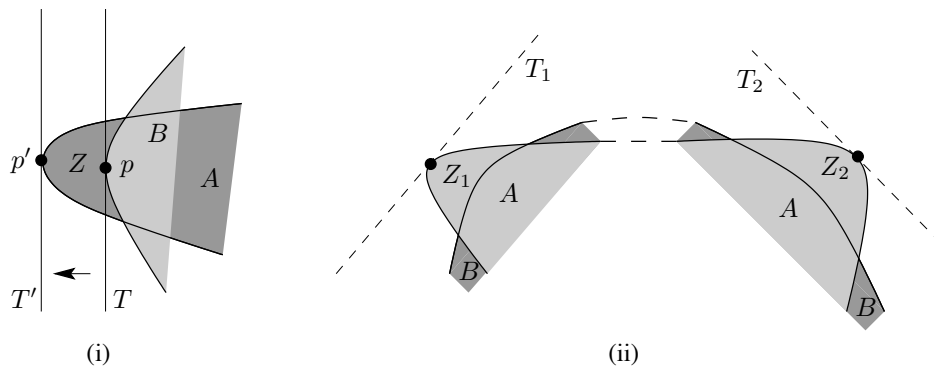


Abbildung 2.8: (i)  $p$  muß Element von  $Z$  sein, (ii)  $A \cup B$  kann nicht aus zwei Zusammenhangskomponenten bestehen.

**Lemma 2.12** Seien  $P_1, P_2, R$  konvexe Polygone mit  $\overset{\circ}{P}_1 \cap \overset{\circ}{P}_2 = \emptyset$ . Dann sind  $A = P_1 \oplus R$  und  $B = P_2 \oplus R$  ein Paar konvexer Pseudokreise. [KLPS86]

### Beweis.

Die Konvexität folgt aus der Definition von  $\oplus$ . Wegen der Symmetrie genügt es zu zeigen:  $A \setminus B$  ist zusammenhängend. Jede Zusammenhangskomponente  $Z$  von  $A \setminus B$  trägt zum Rand der konvexen Hülle  $\text{ch}(A \cup B)$  bei: Falls  $Z = A$  ist, also  $A \cap B = \emptyset$ , müssen auch Punkte von  $A$  auf  $\partial \text{ch}(A \cup B)$  liegen; sonst wäre  $A \subset \text{ch}(A \cup B) = \text{ch}(B) = B$ . Ist  $Z$  eine *echte* Teilmenge von  $A$ , so muß  $\partial Z$  einen Punkt  $p \in \partial B$  enthalten. Lege die Tangente  $T$  durch  $p$  an  $B$ , dann liegt  $B$  vollständig auf einer Seite von  $T$ ; siehe Abbildung 2.8(i). Verschiebe jetzt  $T$  parallel in die  $B$  entgegengesetzte Richtung, bis der letzte Randpunkt  $p'$  von  $A$  erreicht wird. Die Tangente  $T'$  durch  $p'$  an  $A$  ist auch Tangente an  $A \cup B$ , also gilt  $p' \in \partial A \cap \partial \text{ch}(A \cup B)$ .

Nehmen wir nun an, es gäbe zwei Zusammenhangskomponenten  $Z_1, Z_2$  in  $A \setminus B$ .

$\Rightarrow$  Es gibt zwei Richtungen  $\vec{d}_1, \vec{d}_2$ , in denen  $A$  extremer als  $B$  ist, d. h. die Tangente an  $\text{ch}(A \cup B)$  berührt  $A$ , siehe Abbildung 2.8(ii).

$\Rightarrow$  Es gibt zwei Richtungen  $\vec{d}_1, \vec{d}_2$ , in denen  $P_1$  extremer als  $P_2$  ist.  
Bem. 2.8

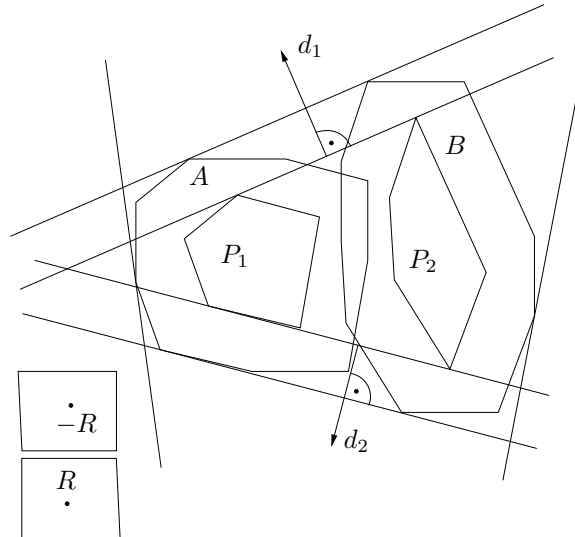


Abbildung 2.9: Um  $A \cup B$  rotierende Tangenten wechseln zweimal zwischen  $A$  und  $B$ , weil  $P_1$  und  $P_2$  disjunkt sind.

- $\Rightarrow$  Da  $P_1, P_2$  konvex und nicht leer sind, und  $\overset{\circ}{P}_1 \cap \overset{\circ}{P}_2 = \emptyset$  gilt, ist  $P_1$  entweder in allen Richtungen zwischen  $\vec{d}_1$  und  $\vec{d}_2$  oder in allen Richtungen zwischen  $\vec{d}_2$  und  $\vec{d}_1$  extremer als  $P_2$ , siehe Abbildung 2.9.
- $\Rightarrow$  Bem. 2.8  $A$  ist entweder in allen Richtungen zwischen  $\vec{d}_1$  und  $\vec{d}_2$  oder in allen Richtungen zwischen  $\vec{d}_2$  und  $\vec{d}_1$  extremer als  $B$ .
- $\Rightarrow$  Eine um  $A \cup B$  rotierende Tangente berührt erst  $A$ , dann  $A \cup B, B, A \cup B$  und schließlich wieder  $A$ , siehe Abbildung 2.9.
- $\Rightarrow$  Es gibt nur eine Zusammenhangskomponente  $\zeta$ .

□

**Theorem 2.13** (Kedem, Livne, Pach, Sharir, 1986)

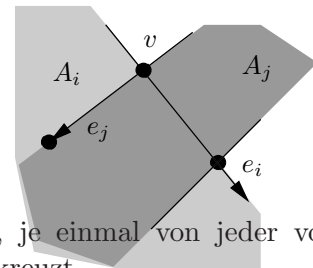
Sei  $A_1, \dots, A_k$  eine Familie von polygonalen Pseudokreisen mit insgesamt  $n$  Kanten. Dann hat  $\partial \cup A_i$  die Komplexität  $O(n)$ . [KLPS86]

**Beweis.**

Auf  $\partial \cup A_i$  gibt es zwei Typen von Ecken: Ecken von Pseudokreisen — davon gibt es maximal  $n$  Stück — und Schnittpunkte von Kanten von Pseudokreisen. Sei  $v$  ein Schnittpunkt zweier Pseudokreis-Kanten  $e_i$  und  $e_j$ . Folge der Kante  $e_i$  ins Innere von  $A_j$ , dann gibt es wiederum zwei Fälle:

1. Fall:  $e_i$  endet in  $A_j$ , dann stelle  $v$  dem Endpunkt von  $e_i$  in  $A_j$  in Rechnung.

2. Fall:  $e_i$  geht durch  $A_j$  hindurch. Auf  $e_i$  liegen dann zwei Schnittpunkte von  $A_i$  und  $A_j$ . Da  $A_i$  und  $A_j$  Pseudokreise sind, kann es keinen weiteren Schnittpunkt geben. Also muß  $e_j$  im Inneren von  $A_i$  enden, und wir können  $v$  dem Endpunkt von  $e_j$  in  $A_i$  in Rechnung stellen.



Auf diese Weise wird jede Ecke  $w$  höchstens zweimal belastet, je einmal von jeder von  $w$  ausgehenden Kante, falls diese sich mit den Rändern anderer  $A_\ell$  kreuzt. □

Theorem 2.13 gilt auch allgemeiner, ist jedoch in dieser Form viel schwieriger zu beweisen:

**Theorem 2.14** Sei  $A_1, \dots, A_k$  eine Familie von Pseudokreisen. Dann hat  $\partial \cup A_i$  die Komplexität  $O(n)$ . [KLPS86]

Welche Komplexität hat nun  $P \oplus R$ , wenn nicht beide Polygone konvex sind?

**Lemma 2.15** Seien  $P_1, P_2$  Polygone mit  $n$  bzw.  $m$  Ecken.

- (i) Sind  $P_1, P_2$  konvex, so ist  $P_1 \oplus P_2$  konvex und hat die Komplexität  $O(m+n)$ .
- (ii) Ist nur  $P_2$  konvex, so hat  $P_1 \oplus P_2$  die Komplexität  $O(mn)$ .
- (iii) Ist kein  $P_i$  konvex, so hat  $P_1 \oplus P_2$  die Komplexität  $O(m^2n^2)$ .

Alle Schranken sind scharf.

**Beweis.**

- (i) Bereits gezeigt.
- (ii) Trianguliert man  $P_1$ , so entstehen  $n-2$  Dreiecke  $D_i$  mit paarweise disjunktem Inneren und

$$P_1 = \bigcup_{i=1}^{n-2} D_i.$$

$$P_1 \oplus P_2 = \left( \bigcup_{i=1}^{n-2} D_i \right) \oplus P_2 = \bigcup_{i=1}^{n-2} \underbrace{(D_i \oplus P_2)}_{O(m) \text{ Kanten}}.$$

Dies ist nach Lemma 2.12 eine Familie von Pseudokreisen mit  $O(mn)$  Kanten, also hat  $P_1 \oplus P_2$  die Komplexität  $O(mn)$  nach Theorem 2.13.

- (iii) Trianguliere  $P_1$  und  $P_2$ ,  $P_1 = \bigcup_{i=1}^{n-2} D_i$  und  $P_2 = \bigcup_{j=1}^{m-2} D'_j$ .

$$P_1 \oplus P_2 = \bigcup_{i=1}^{n-2} \bigcup_{j=1}^{m-2} \underbrace{D_i \oplus D'_j}_{O(1) \text{ Kanten}}.$$

Insgesamt erhalten wir  $O(mn)$  Kanten von denen sich jedes Paar kreuzen kann, also  $O(m^2n^2)$  viele Schnittpunkte. Theorem 2.13 ist hier nicht anwendbar, da  $\bigcup D_i \oplus D'_j$ ,  $1 \leq i \leq n-2, 1 \leq j \leq m-2$  keine Familie von Pseudokreisen ist (Übungsaufgabe).

Abbildung 2.10 zeigt Beispiele, in denen die Schranken scharf sind. □

Damit können wir folgende Aussage beweisen:

**Theorem 2.16** Sei  $R$  ein polygonaler, konvexer Roboter mit  $m$  Ecken, und seien  $P_i$  polygonale Hindernisse mit insgesamt  $n$  Ecken; dann hat  $\mathcal{C}_{\text{frei}}$  die Komplexität  $O(mn)$  und kann in Zeit  $O(mn \log^2(mn))$  berechnet werden.

**Beweis.**

Es genügt,  $\mathcal{C}_{\text{verb}}$  zu berechnen.  $P_i$  habe  $n_i$  Ecken. Triangulieren wir die  $P_i$ , so entsteht eine Familie von Pseudokreisen, für die gilt:

$$\mathcal{C}_{\text{verb}} = \underbrace{\bigcup_i \underbrace{P_i \oplus -R(0,0)}_{\text{Komplexität } O(m \cdot n_i)}}_{\text{Komplexität } O(m \cdot \sum n_i) = O(mn)}.$$

$\mathcal{C}_{\text{verb}}$  kann mit dem Divide-and-Conquer Algorithmus 2.2 berechnet werden. Die Lösung der dabei entstehenden Rekursion  $T(k) \leq 2T(\frac{k}{2}) + Ck \log k$  mit  $k = mn$  liegt in  $O(k \log^2 k)$ :

$$\begin{aligned} T(k) &\leq 2T\left(\frac{k}{2}\right) + Ck \log k \\ &\leq 4T\left(\frac{k}{4}\right) + 2C\frac{k}{2} \log \frac{k}{2} + Ck \log k \\ &\leq \dots \\ &\leq 2^{\log k} T(1) + \sum_{i=0}^{\log k - 1} Ck \log \frac{k}{2^i} \\ &\leq Ck + \sum_{i=0}^{\log k - 1} Ck \log k \\ &= Ck + Ck \log^2 k \in O(k \log^2 k) \end{aligned}$$

□

---

**Algorithmus 2.2** Berechnung von  $\mathcal{C}_{\text{verb}}$ 

---

- Unterteile die Hindernisse beliebig in gleichgroße Teilmengen  $\mathcal{P}_1$  und  $\mathcal{P}_2$ .
  - Berechne rekursiv  $\mathcal{C}_{\text{verb}}(\mathcal{P}_1)$  und  $\mathcal{C}_{\text{verb}}(\mathcal{P}_2)$  mit je  $O(mn)$  Kanten.
  - Berechne mit Sweep die Vereinigung  $\mathcal{C}_{\text{verb}}(\mathcal{P}_1) \cup \mathcal{C}_{\text{verb}}(\mathcal{P}_2)$  in Zeit  $O(mn \log(mn))$ .
- 

Damit haben wir das Bahnplanungsproblem für konvexe Roboter und Translationen zwischen polygonalen Hindernissen auf das Finden von Pfaden für Punkte im zweidimensionalen Raum  $\mathcal{C}_{\text{frei}}$  — d. h. zwischen polygonalen Hindernissen mit Gesamtgröße  $O(mn)$  — reduziert. Dies läßt sich mit Algorithmen zur Berechnung kürzester Pfade oder mit dem in Algorithmus 2.3 beschriebenen Roadmap-Verfahren lösen.

---

**Algorithmus 2.3** Roadmap-Verfahren

---

**Vorbereitung:**

- Berechne die Trapezzerlegung von  $\mathcal{C}_{\text{frei}}$  und die zugehörige Suchstruktur für Point-Location.  $\text{randomisiert } O(mn \log(mn))$
- Entferne die Trapeze, die im Inneren eines Hindernisses liegen.
- Platziere je einen Knoten im Inneren eines jeden Trapez und auf jeder vertikalen Verlängerung.  $O(mn)$
- Verbinde adjazente Knoten (siehe Abbildung 2.11)  
 $\leadsto$  Zusammenhangsgraph von  $\mathcal{C}_{\text{frei}}$  (Roadmap).

**Query:**

- Lokalisier  $s$  und  $t$  in den Trapezen.  $O(\log(mn))$
  - Falls  $s$  und  $t$  im selben Trapez liegen: verbinde  $s$  und  $t$ .
  - Ansonsten verbinde  $s$  und  $t$  mit den in den Trapezen liegenden Knoten und führe Breitensuche in der Roadmap durch.  $O(mn)$
- 

Insgesamt erhalten wir folgendes Theorem:

**Theorem 2.17** *Translationsbewegungen eines konvexen, polygonalen Roboters mit  $m$  Ecken in einer Umgebung mit polygonalen Hindernissen mit insgesamt  $n$  Ecken können nach  $O(mn \log^2(mn))$  (randomisiert) Vorbereitungszeit in Zeit  $O(mn)$  geplant werden.*

Algorithmus 2.3 ist vergleichsweise einfach, jedoch nicht optimal.



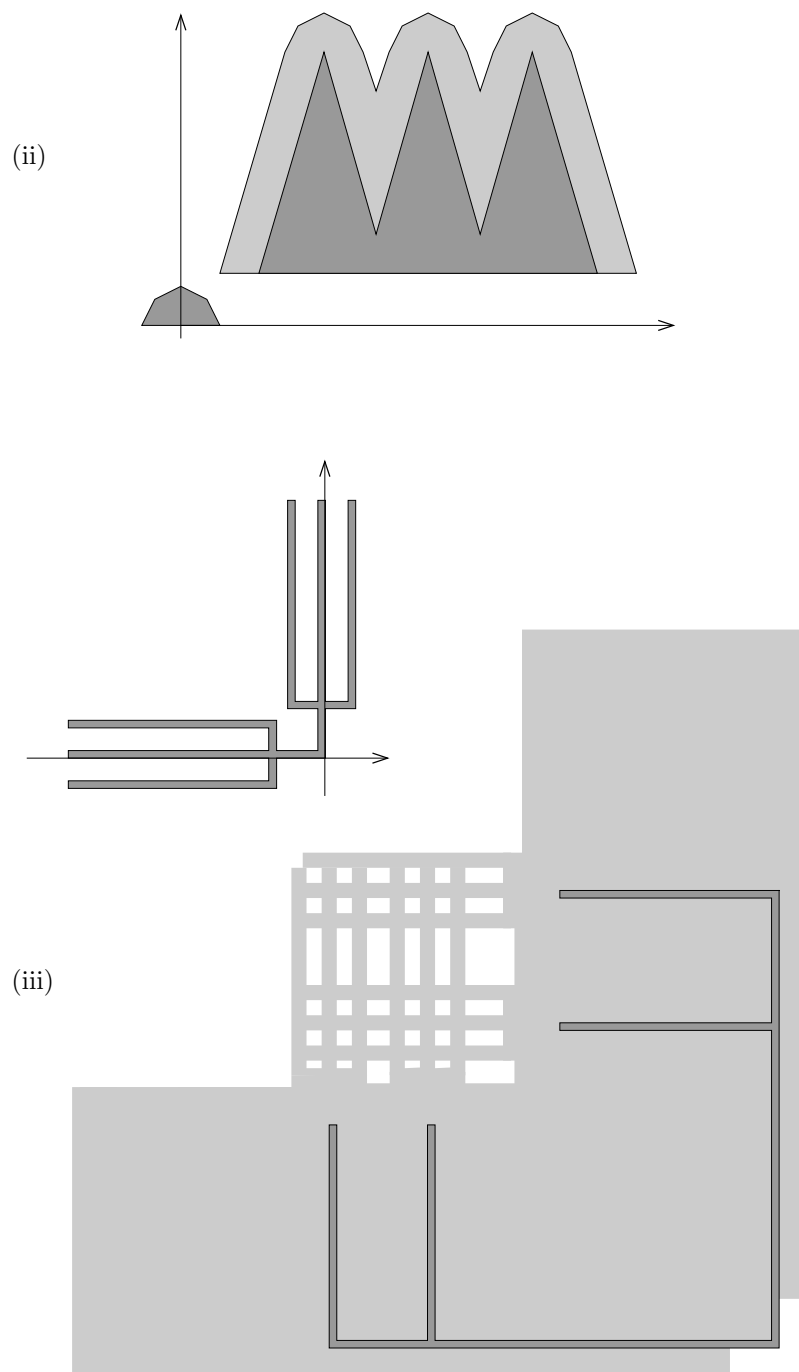


Abbildung 2.10: (ii)  $P_1 \oplus P_2$  hat  $\Theta(mn)$  Ecken, wenn nur eines der Polygone konvex ist, (iii)  $P_1 \oplus P_2$  hat  $\Theta(m^2n^2)$  Ecken, wenn keines der Polygone konvex ist.

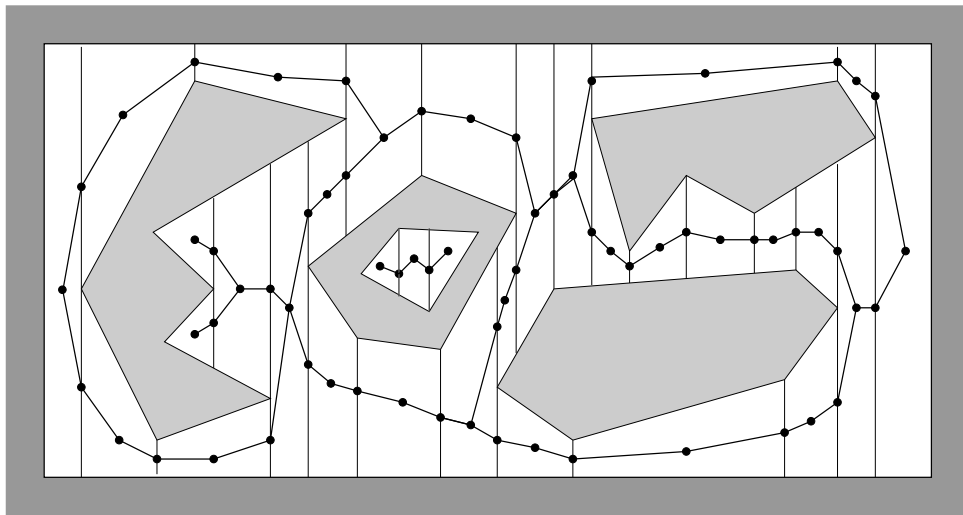


Abbildung 2.11: Konfigurationsraum-Hindernisse und Roadmap.

## 2.2 Reine Translationsbewegungen für beliebige Roboter

Betrachten wir nun einen allgemeinen, polygonalen Roboter mit  $m$  Ecken, der sich wie gehabt in einer Umgebung mit polygonalen Hindernissen  $P_i$  mit insgesamt  $n$  Ecken bewegt.

Eine mögliche Vorgehensweise wäre, die konvexe Hülle des Roboters zu betrachten und wie in Abschnitt 2.1 vorzugehen. Das Problem dabei ist uns schon bei der Approximation durch kreisförmige Roboter begegnet: es werden evtl. keine Bahnen gefunden, obwohl welche existieren.

Da der Roboter  $R$  nicht mehr konvex ist, hat  $\mathcal{C}_{\text{frei}}$  hier nach Lemma 2.15(iii) die Komplexität  $O(m^2n^2)$ . Die Berechnung des gesamten Raumes  $\mathcal{C}_{\text{frei}}$  wäre also sehr aufwendig. Die Idee ist nun, nur noch diejenige Zusammenhangskomponente  $Z_s$  von  $\mathcal{C}_{\text{frei}}$  zu berechnen, die die Startposition  $s$  enthält. Dies genügt, denn wenn die Zielposition  $t$  nicht in  $Z_s$  liegt, ist das Bahnplanungsproblem nicht lösbar, andernfalls ist ein Pfad von  $s$  nach  $t$  in  $Z_s$  zu berechnen.

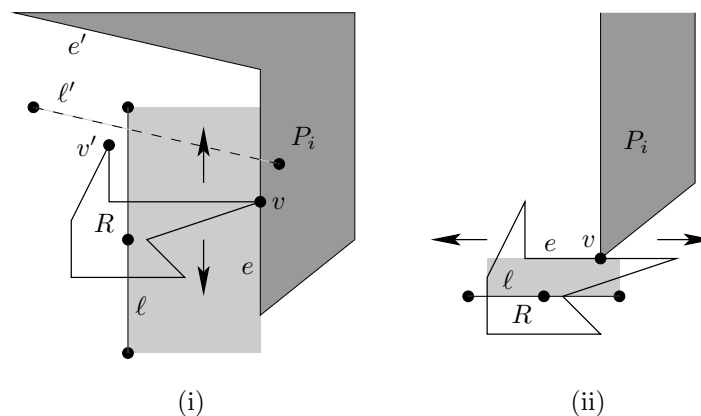


Abbildung 2.12: Paare von Ecken und Kanten von  $P_i$  und  $R$  erzeugen im Konfigurationsraum ein Liniensegment.

Untersuchen wir einmal genauer, wodurch  $\mathcal{C}_{\text{frei}}$  eingeschränkt wird: Für jede Ecke  $v$  des Roboters und jede Kante  $e$  eines Hindernisses erzeugt das Ecke/Kante-Paar  $(v, e)$  im Konfigurationsraum ein Liniensegment  $\ell$ , über das der Referenzpunkt des Roboters nicht hinweg bewegt werden darf. Dieses Liniensegment entsteht durch Bewegung der Roboterecke  $v$  entlang der Hinderniskante  $e$ . Andere Ecke/Kante-Paare werden dabei außer acht gelassen. In Abbildung 2.12(i) erzeugen die Ecke/Kante-Paare  $(v, e)$  und  $(v', e')$  auf diese Weise die Liniensegmente  $\ell$  und  $\ell'$ . Weiterhin erzeugt jedes Kante/Ecke-Paar  $(e, v)$  einer Kante  $e$  des Roboters und einer Hindernisecke  $v$  ein Liniensegment  $\ell$ , siehe Abbildung 2.12(ii).

Wir brauchen daher nicht zu jedem Hindernis das Konfigurationsraumhindernis zu berechnen, sondern können jedes Ecke/Kante- und Kante/Ecke-Paar einzeln betrachten. Für jedes dieser Paare erhalten wir ein Liniensegment.  $Z_s$  ist dann eine Zelle in einem Arrangement von  $2mn$  Liniensegmenten.

### 2.2.1 Komplexität und Berechnung einzelner Zellen

Wie kompliziert kann eine solche Zelle sein? Das folgende Theorem beantwortet die Frage etwas allgemeiner, nämlich für die Komplexität einer Zelle in einem Arrangement von Kurvenstücken ohne Selbstschnitte:

**Theorem 2.18** (Guibas, Sharir, Sifrony, 1989)

Sei  $\mathcal{A}$  ein Arrangement von  $n$  Kurvenstücken, von denen sich je zwei höchstens  $s$  mal schneiden. Dann hat jede Zelle in  $\mathcal{A}$  die Komplexität  $O(\lambda_{s+2}(n)) \subseteq O(n \log^* n)$ . [GSS89]

$\lambda_s(n)$  bezeichnet die maximale Länge einer Davenport-Schinzel-Sequenz der Ordnung  $s$  über einem Alphabet mit  $n$  Buchstaben, siehe Abschnitt A.4.

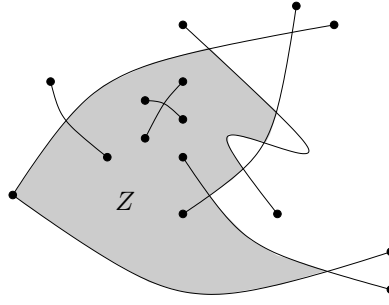


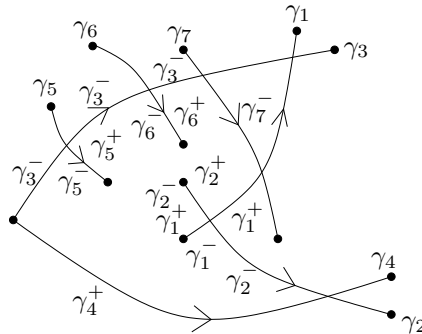
Abbildung 2.13: Zelle in Arrangement von Kurvenstücken.

Für den Beweis von Theorem 2.18 betrachten wir eine Zelle. Der Rand  $\partial Z$  von  $Z$  zerfällt in Zykel. Wir betrachten nur den äußeren Zyklus,  $C$ , und nehmen an, daß  $C$  entgegen den Uhrzeigersinn orientiert ist. Wie man sich leicht überlegen kann, reicht für die Komplexitätsbetrachtung der Zelle die Betrachtung des äußeren Zyklus aus (Übungsaufgabe).

Nun orientieren wir jeden Bogen, z.B. von links nach rechts, wodurch ein positiver und negativer Durchlaufsinne festgelegt wird. Den Zyklus  $C$  aus Abbildung 2.14 können wir so rein kombinatorisch durch die zyklische Folge

$$S = \langle \gamma_4^+ \gamma_2^- \gamma_1^- \gamma_1^+ \gamma_2^- \gamma_2^+ \gamma_1^+ \gamma_7^- \gamma_3^- \gamma_6^+ \gamma_6^- \gamma_3^- \gamma_5^+ \gamma_5^- \gamma_3^- \rangle$$

beschreiben.

Abbildung 2.14: Die orientierten Segmente von  $C$ .

**Lemma 2.19** (*Konsistenzlemma*)

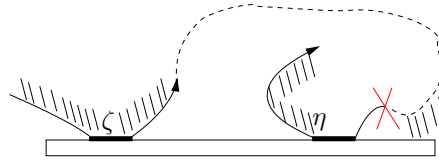
Die Segmente eines orientierten Bogens  $\gamma_i^+$  ( $\gamma_i^-$ ) erscheinen in  $C$  in derselben Reihenfolge wie längs  $\gamma_i^+$  ( $\gamma_i^-$ ).

**Beweis.** Zur Repräsentation von  $\gamma_i^+$  und  $\gamma_i^-$  erweitern wir  $\gamma_i$  zu einem Schlauch ohne die Topologie von  $C$  zu verändern. Diesen Schlauch nennen wir nun ebenfalls  $\gamma_i$ .

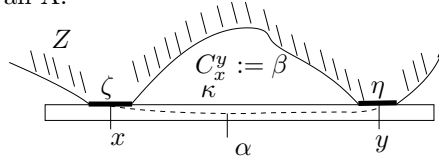
Seien  $\zeta$  und  $\eta$  zwei Segmente auf einer Seite von  $\gamma_i$ , die in  $C$  konsekutiv sind, das heißt, kein Stück von  $\gamma_i$  taucht zwischen  $\zeta$  und  $\eta$  entlang  $C$  auf. Unterscheiden wir je nach Verlauf der Kurve  $C$ , so können wir einige Fälle ausschließen. In Abbildung 2.15 liegt die Zelle  $Z$  auf der linken Seite von  $C$ . Da sie in  $\eta$  auf der Außenseite des Schlauches liegen muss, ist dieser Fall nicht möglich. Die Zelle  $Z$  läge hier auf beiden Seiten von  $C$ .

Dadurch ergeben sich zwei Fälle, siehe Abbildung 2.16.

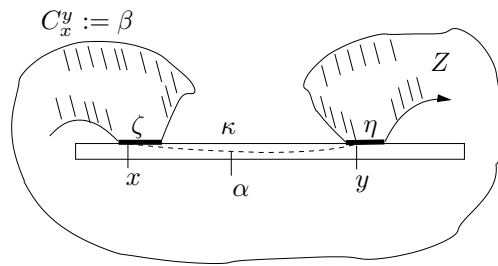
In beiden Fällen können wir innerhalb des Schlauches eine Kurve  $\alpha$  von  $x \in \zeta$  nach  $y \in \eta$  legen, die  $\zeta$  und  $\eta$  verbindet. Das Kurvenstück  $C_x^y$  nennen wir im folgenden  $\beta$ . Die geschlossene Kurve  $\alpha \cup \beta$  liegt im Komplement des Inneren von  $Z$ .

Abbildung 2.15: Die Zelle  $Z$  läge hier auf beiden Seiten von  $C$ .

Fall A:



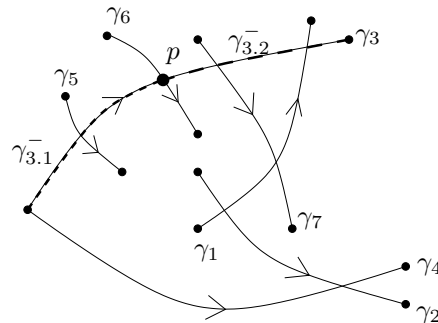
Fall B:

Abbildung 2.16: Kein Stück von  $\gamma_i$  taucht zwischen  $\zeta$  und  $\eta$  entlang  $C$  auf.

In beiden Fällen liegt  $Z$  links von der gegen den Uhrzeigersinn orientierten, geschlossenen Kurve  $C$ . Das bedeutet, daß  $\alpha \cup \beta$  die Kurve  $C$  vom Segment  $\kappa$  zwischen  $\zeta$  und  $\eta$  separiert. Kein Punkt  $z \in \kappa$  liegt somit auf  $C$ .  $\square$

Um die Komplexitäten der im Anhang beschriebenen Davenport–Schinzel–Sequenzen anwenden zu können, müssen wir uns von der zyklischen Betrachtung lösen und eine lineare Ordnung der betrachteten Segmente erschaffen.

Dabei kann der Fall auftreten, daß in der Sequenz  $s$  bezüglich eines Startpunktes  $p$  einige orientierte Bogensegmente nicht mehr in der Reihenfolge entlang der Segmente besucht werden, siehe zum Beispiel  $\gamma_3^+$  in Abbildung 2.17, daher verdoppeln wir diese Segmente einfach. Aus  $\gamma_3^-$  wird  $\gamma_{3,1}^-$  und  $\gamma_{3,2}^-$ . Es entsteht dabei eine lineare Sequenz  $s^*$  mit  $4n$  Segmenten.

Abbildung 2.17: Das Segment  $\gamma_3^-$  wird in der nicht-zyklischen Sequenz nicht in der Reihenfolge gemäß der Orientierung besucht und wird deshalb am Startpunkt  $p$  verdoppelt.

Bemerkung: Es kann tatsächlich vorkommen, dass wir jedes Segment teilen müssen, siehe Abbildung 2.18.

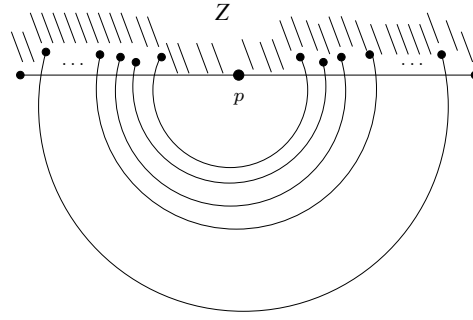


Abbildung 2.18: Wird der Startpunkt  $p$  gewählt, so müssen alle Segmente geteilt werden.

Mit diesen Vorbereitungen zeigen wir die folgende Behauptung, mit der Theorem 2.18 bewiesen wird.

**Lemma 2.20** *Die Sequenz  $s^*$  ist eine  $(4n, s + 2)$  Davenport–Schinzel–Sequenz.*

**Beweis.** Es ist klar, daß nie zwei Segmente  $\gamma_{i,j}^+$  ( $\gamma_{i,j}^-$ ) hintereinander stehen, da ein Schnittpunkt dazwischen sein muß. Es bleibt zu zeigen, daß sich keine zwei verschiedenen Bogensegmente  $\zeta$  und  $\eta$  in der Sequenz  $s^*$  mehr als  $s + 2$  mal abwechseln. Dazu nehmen wir an, es gäbe eine solche (Teil-)Sequenz  $s^*$  mit  $s + 3$  Wechseln und führen dies zu einem Widerspruch, indem wir zeigen, daß sich  $\zeta$  und  $\eta$  dann mehr als  $s$  mal schneiden.

Für eine gegebene Sequenz

$$s^* = (\cdots \zeta_1 \cdots \eta_1 \cdots \zeta_2 \cdots \eta_2 \cdots \cdots \zeta_j \cdots \eta_j \cdots \zeta_k \cdots (\eta_k \cdots))$$

mit den signifikanten  $\zeta_i$  und  $\eta_i$  fassen wir je 4 sukzessive Segmente zu einem Quadrupel wie folgt zusammen.

$$\begin{pmatrix} \zeta_1 & , & \eta_1 & , & \zeta_2 & , & \eta_2 \\ \eta_1 & , & \zeta_2 & , & \eta_2 & , & \zeta_3 \\ \zeta_2 & , & \eta_2 & , & \zeta_3 & , & \eta_3 \\ \vdots \end{pmatrix}$$

Der Index  $k$  und die Existenz von  $\eta_k$  hängt von  $s$  ab: wenn  $s + 3$  ungerade ist, dann existiert  $\eta_k$  und es gilt  $k = \frac{s+2}{2} + 1$ . Wenn  $s + 3$  gerade ist, entfällt das letzte  $\eta_k$  und der Wert von  $k$  ist eine leichte Übungsaufgabe.

Sei  $s + 3$  ungerade. In diesem Fall erhalten wir insgesamt  $s + 1$  Quadrupel:

$$(\zeta_1, \eta_1, \zeta_2, \eta_2), (\eta_1, \zeta_2, \eta_2, \zeta_3), (\zeta_2, \eta_2, \zeta_3, \eta_3), \dots, (\zeta_{\frac{s+2}{2}}, \eta_{\frac{s+2}{2}}, \zeta_{\frac{s+2}{2}+1}, \eta_{\frac{s+2}{2}+1}).$$

Die Anzahl der Quadrupel ergibt sich aus  $2 \cdot (\frac{s+2}{2}) - 1$  Quadrupeln angeführt von je  $\frac{s+2}{2}$  Elementen aus  $\zeta$  und  $\frac{s+2}{2} - 1$  Elementen aus  $\eta$ .

Wir zeigen, daß für jedes Quadrupel der Form  $(\zeta_i, \eta_i, \zeta_{i+1}, \eta_{i+1})$  (bzw.  $(\eta_i, \zeta_{i+1}, \eta_{i+1}, \zeta_{i+2})$ ) genau ein Schnittpunkt zwischen  $\zeta_i$  und  $\eta_i$  (bzw. zwischen  $\eta_i$  und  $\zeta_{i+1}$ ) liegen muß. Damit hätten wir insgesamt  $s + 1$  verschiedene Schnittpunkte im Widerspruch zu der Annahme, daß sich  $\zeta$  und  $\eta$  nicht mehr als  $s$  mal schneiden. Ergo existieren höchstens  $s + 2$  Wechsel.

Es bleibt zu zeigen, daß für ein beliebiges Quadrupel, nennen wir es o. B. d. A.  $(\zeta_i, \eta_i, \zeta_{i+1}, \eta_{i+1})$  der besagte Schnittpunkt existiert. Dazu wählen wir  $x \in \zeta_i$ ,  $y \in \zeta_{i+1}$ ,  $z \in \eta_i$  und  $w \in \eta_{i+1}$  so, daß  $C$  diese Punkte in der Reihenfolge  $x, z, y, w$  besucht. Die Teilbögen  $\beta_x^z, \beta_z^y, \beta_y^w$  und  $\beta_w^x$  aus  $C$  schneiden sich nur in den Endpunkten, weil  $C$  eine geschlossene, einfache Kurve ist. Wir betrachten weiterhin die Bögen  $\beta_x^y \in \zeta$  und  $\beta_z^w \in \eta$ , siehe Abbildung 2.19.

Wir wollen zeigen, daß sich  $\beta_x^y$  und  $\beta_z^w$ , wie Abbildung 2.19 durch den Bogen  $\beta_{z'}^{w'}$  angedeutet wird.

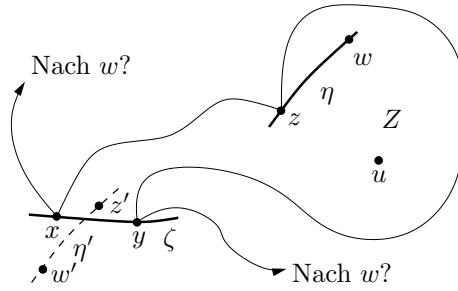


Abbildung 2.19: Die paarweise disjunkten Bögen  $\beta_x^z, \beta_z^y, \beta_y^w, \beta_w^x, \beta_x^y$  und  $\beta_z^w$ .

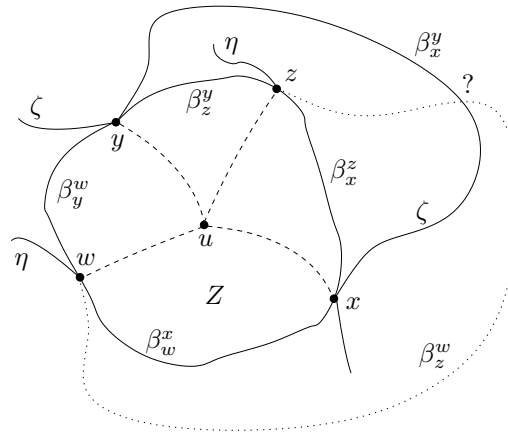


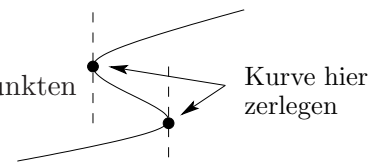
Abbildung 2.20: Eine kreuzungsfreie Einbettung, wie sie von der angenommenen Lage der Bögen induziert wird, existiert nicht.

Wir wollen einen formalen Widerspruchsbeweis führen. Dazu nehmen wir an, daß sich  $\beta_x^y$  und  $\beta_z^w$  nicht schneiden. Wählen wir einen Punkt  $u$  im Inneren von  $Z$ , dann können wir  $x, y, z$  und  $w$  kreuzungsfrei mit  $u$  verbinden. Da die Bögen  $\beta_x^z, \beta_z^y, \beta_y^w, \beta_w^x, \beta_x^y$  und  $\beta_z^w$  allesamt kreuzungsfrei sind, hätten wir den vollständigen Graphen  $K_5$  kreuzungsfrei in die Ebene eingebettet. Das ist aber nicht möglich: wie in Abbildung 2.20 zeigt, kann man nicht, jeden der Punkte  $u, w, x, y, z$  mit jedem anderen verbinden, ohne daß sich zwei Bögen kreuzen. Diese Tatsache läßt sich auch mit Hilfe der Eulerformel und einer weiteren Abschätzung formal beweisen, siehe z. B. [Kle97].

□

Wie können wir nun eine einzelne Zelle  $Z$  im Arrangement von  $n$  Kurvenstücken berechnen, die durch einen in  $Z$  enthaltenen Punkt — den Referenzpunkt des Roboters — bestimmt ist? Zur Berechnung gehen wir von folgenden Annahmen aus:<sup>5</sup>

- (i) Je zwei Bögen schneiden sich höchstens  $s$  mal.
- (ii) Jeder Bogen hat nur konstant viele vertikale Tangenten. Damit können wir ohne Einschränkung annehmen, daß alle Bögen X-monoton sind, nicht X-monotone Bögen können an den Tangentenpunkten der vertikalen Tangenten in X-monotone Teilbögen zerlegt werden, vgl. Abbildung.

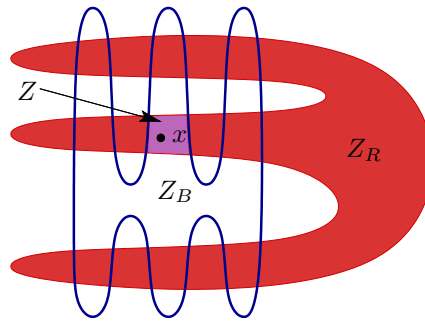


Ein erster Ansatz zur Lösung dieses Problems ist der Divide-and-Conquer Algorithmus 2.4. Das Problem dabei ist, daß  $Z_R \cap Z_B$  sehr komplex werden kann und die explizite Berechnung zu hohe Kosten verursacht. Abbildung 2.21 zeigt ein Beispiel, in dem  $Z_R \cap Z_B$  recht komplex ist,  $Z$  selbst jedoch recht einfach. Wir brauchen also einen effizienteren Ansatz für den Merge-Schritt; die Frage ist nun, wie man die Zelle  $Z$  berechnen kann, ohne alle Zusammenhangskomponenten von  $Z_R \cap Z_B$  zu berechnen.

<sup>5</sup>Diese Forderungen sind erfüllt, wenn die Bögen durch algebraische Gleichungen vom Grad  $O(1)$  beschrieben werden.

**Algorithmus 2.4** Berechnung einer Zelle im Arrangement von  $n$  Bögen**Gegeben:** Punkt  $x$ ,  $n$  Bögen.**Gesucht:** Die Zelle  $Z$  im Arrangement der Bögen, die  $x$  enthält.

- Zerlege die Menge der Bögen in gleichgroße Teilmengen  $R$  und  $B$ .
- Berechne rekursiv im Arrangement  $\mathcal{A}(R)$  die Zelle  $Z_R$ , die  $x$  enthält.
- Berechne rekursiv im Arrangement  $\mathcal{A}(B)$  die Zelle  $Z_B$ , die  $x$  enthält.
- Berechne die Zusammenhangskomponente  $Z$  von  $Z_R \cap Z_B$ , die  $x$  enthält und berichte diese (Red–Blue–Merge).

Abbildung 2.21:  $Z_R \cap Z_B$  kann sehr komplex sein, obwohl  $Z$  recht einfach ist.

Betrachten wir einen etwas allgemeineren Fall; gegeben sind:

- Ein “rotes” Arrangement  $\mathcal{R}$  mit ausgezeichneten Zellen  $R_1, \dots, R_{m_R}$  mit insgesamt  $r$  Ecken (Bogensegmenten).
- Ein “blaues” Arrangement  $\mathcal{B}$  mit ausgezeichneten Zellen  $B_1, \dots, B_{m_B}$  mit insgesamt  $b$  Ecken (Bogensegmenten).<sup>6</sup>
- Eine Menge von Punkten  $p_i \in R_{\mu_i} \cap B_{\nu_i}$ ,  $1 \leq i \leq k$  — d.h. jeder Punkt ist sowohl in einer roten als auch in einer blauen Zelle enthalten —, wobei nicht jede Menge  $R_\mu \cap B_\nu$  ein  $p_i$  enthalten muß und verschiedene  $p_i$  in derselben Menge  $R_\mu \cap B_\nu$  liegen können.

Betrachte nun das “violette” Arrangement  $\mathcal{V}$ , das durch Überlagerung des roten Arrangements  $\mathcal{R}$  und des blauen Arrangements  $\mathcal{B}$  entsteht. In  $\mathcal{V}$  liegt jeder Punkt  $p_i$  in einer Zelle  $Z_i$ . Wir wollen nun diese  $Z_i$  berechnen, jedoch jede nur einmal, auch wenn sie mehrere Punkte enthält. Über die Komplexität dieser Zellen, d.h. die Anzahl Segmente auf dem Rand, gibt das folgende Lemma Auskunft:

**Lemma 2.21** (*Kombinationslemma; Guibas, Sharir, Sifrony, 1989*)Für die Komplexität der Zellen  $Z_1, \dots, Z_\ell$ ,  $\ell \leq k$  gilt:<sup>7</sup>

[GSS89]

$$|Z_1| + |Z_2| + \dots + |Z_\ell| \in O(r + b + k)$$

Eine stärkere Version dieses Lemmas stammt von Edelsbrunner, Guibas und Sharir [EGS90], zu finden auch in [SA95]. Die Aussage von Lemma 2.21 ist keineswegs trivial: ein Bogen kann zu vielen violetten Zellen, ja sogar mehrere Stücke zum Rand einer Zelle beitragen, siehe Abbildung 2.23. Wir geben hier keinen Beweis für Lemma 2.21.

<sup>6</sup>Wir bezeichnen im Folgenden die Segmente der Zellen  $R_i$  und  $B_j$  als Bögen, um sie von den neuen Segmenten zu unterscheiden, die beim Schnitt der Zellen entstehen.

<sup>7</sup> $\ell \leq k$ , da mehrere Punkte in derselben Zelle liegen können.



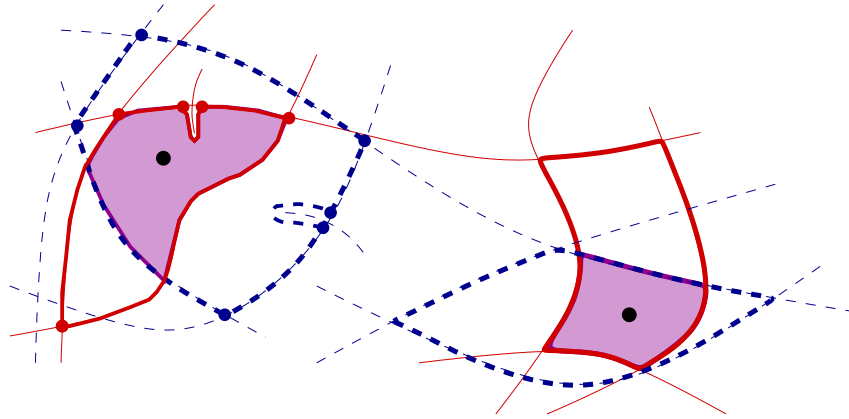


Abbildung 2.22: Arrangement von roten und blauen Zellen und Überlagerung. Komplexität der linken blauen (linken roten) Zelle: 6 (5) Segmente.

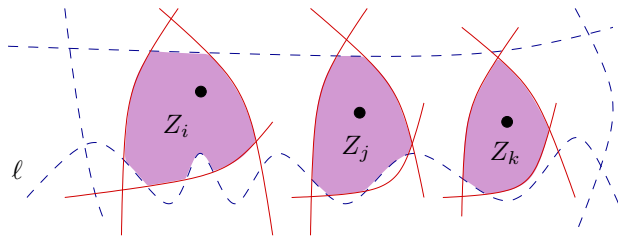


Abbildung 2.23: Der Bogen  $\ell$  trägt zu vielen violetten Zellen bei.

Wie lassen sich nun die Zellen  $Z_i$  berechnen?

**Theorem 2.22** Seien  $Z_1, \dots, Z_\ell$  diejenigen Zellen, welche die Punkte  $p_1, \dots, p_k$  enthalten. Dann lassen sich die Zellen  $Z_1, \dots, Z_\ell$  in Zeit  $O((r + b + k) \log(r + b + k))$  berechnen.

**Beweis.** Algorithmus 2.5 berechnet etwas mehr als verlangt, nämlich alle violetten Zellen, die einen Punkt  $q \in Q$  im Abschluß enthalten. Wie bereits erwähnt, liegt die Schwierigkeit darin, *nur* diese Zellen zu berechnen, nicht alle Zusammenhangskomponenten von  $R_\mu \cap B_\nu$ . Die beiden Sweeps sind notwendig, da z. B. in Abbildung 2.25(i) während des ersten Sweeps nicht erkannt werden kann, daß in  $a$  eine violette Region beginnt, da noch kein  $q \in Q$  entdeckt wurde.  $a$  liegt zwar auf dem Rand einer Zelle, ist jedoch kein Endpunkt eines Bogens. Erst wenn die Sweepline  $q_1$  erreicht, wird eine violette Teilregion begonnen. Analog kann während des zweiten Sweeps der Beginn der violetten Region noch nicht im Punkt  $b$  sondern erst bei  $q_3$  erkannt werden.

Die Idee bei dem Sweep (Algorithmus 2.6) ist die Verwendung von je einem oberen und unteren “Scout” je violetter Teilregion. Diese Scouts bewegen sich mit der Sweepline weiter und helfen bei der Erkennung und Behandlung von Ereignissen, die Einfluß auf den Verlauf des Randes der violetten Region haben. Für die Abschätzung des Arbeitsaufwands bei der Ereignisbehandlung ist wichtig, daß jeder Bogen von höchstens einem oberen Scout und höchstens einem unteren Scout bewacht wird, der sich auf einem Bogen der anderen Farbe bewegt. Es treten zwei Arten von Aktualisierungen der Ereignisstruktur auf:

- A:** berechne den in X-Richtung nächsten Schnittpunkt zwischen Scout und beobachtetem Bogen.
- B:** berechne den in X-Richtung nächsten Schnittpkt. zwischen beiden Scouts.

<sup>8</sup>Die Schnittpunkte gleichfarbiger Bögen gehören zwar zur Ereignisstruktur, jedoch nicht zu  $Q$ . Ansonsten würde in Abbildung 2.25(i) bereits in Punkt  $a$  eine Region beginnen! Diese Schnittpunkte sind aus dem letzten Rekursionsschritt bekannt, müssen also nicht explizit berechnet werden.

**Algorithmus 2.5** Red–Blue–Merge

- 
- Sei  $Q := \{p_1, \dots, p_k\} \cup \{\text{Endpunkte von auf dem Rand der roten oder blauen Zellen liegenden Bögen}\}$ .  $n := r + b + k$ ,  $|Q| \in O(n)$ .
  - Für jedes  $q \in Q$ : berechne mit gewöhnlichem Sweep in den Arrangements  $\mathcal{R}$  und  $\mathcal{B}$  die vertikalen Segmente von  $q$  nach oben und unten zum nächsten roten und blauen Bogen. Diese werden wie zusätzliche Bögen behandelt (Abbildung 2.24(i)).  $O(n \log n)$
  - Sortiere die  $q \in Q$  und alle Ecken<sup>8</sup> der Zellen  $R_\mu$  und  $B_\nu$  nach auf- und absteigenden X-Koordinaten (Ereignisstrukturen).  $O(n \log n)$
  - Sweep von links nach rechts:  
Berechne die Teile der violetten Teilzellen, die rechts vom linkensten darin enthaltenen  $q \in Q$  liegen.
  - Sweep von rechts nach links:  
Berechne die Teile der violetten Teilzellen, die links vom rechtensten darin enthaltenen  $q \in Q$  liegen.
  - Setze die Teilregionen zusammen.  $O(n)$
- 

Folgende Ereignisse können auftreten:

1. Eine neue Region startet. Zeit  $O(1)$  für das Anlegen einer neuen Region und zwei Scouts, Aktualisierungen **A** und **B**.
2. Schnittpunkt von Bögen gleicher Farbe oder Endpunkt eines Bogens. Evtl. muss ein beobachtender Scout einen neuen Bogen beobachten, Aktualisierung **A**.
3. Der Scoutbogen und der beobachtete Bogen treffen aufeinander. Der Scout wechselt auf den beobachteten Bogen und beobachtet den ehemaligen Scoutbogen, Aktualisierung **A** und **B**.
4. Die Scouts treffen aufeinander. Eine violette Teilzelle endet, die Scouts verschwinden, Zeit  $O(1)$ .
5. Ein Punkt  $q \in Q$  wird in einer aktiven Region angetroffen. Die aktive Region endet und es entstehen ein oder zwei neue Teilzellen. Wir brauchen  $O(1)$  Zeit um festzustellen, daß  $q$  zwischen den Scouts liegt und  $O(1)$  Zeit für den Abschluß der alten Teilzelle und den Beginn der neuen Teilzelle(n) wie unter 1.

Pro Ereignis tragen wir konstant viele neue Schnittpunkte in die Ereignisstruktur ein, das Einfügen und die Aktualisierung der Sweep-Status-Struktur kostet uns jeweils  $O(\log n)$ . Wir haben nach Lemma 2.21  $O(r + b + k) = O(n)$  viele Ereignisse. Damit können alle violetten Regionen in Zeit  $O(n \log n)$  berechnet werden. □

Abbildung 2.27 zeigt zwei Beispiele für die Aufgabenteilung der Scouts:

- (i) In Punkt  $a$  wechselt der Scout  $O_1$  von einem roten auf einen blauen Bogen und bewacht jetzt den roten Bogen, auf dem er vorher gelaufen ist.  $O_2$  muß aktiv werden und den blauen Rand bewachen.
- (ii) In Punkt  $b$  ist das Ende einer violetten Teilregion erreicht. Der obere und untere Scout der Teilregion treffen zusammen und verschwinden, der Scout  $O_2$  wird aktiv und bewacht den blauen Rand.

In beiden Fällen wird der aktiv werdende Scout  $O_2$  durch binäre Suche in der Sweep-Status-Struktur gefunden.

Insgesamt können wir festhalten:

**Theorem 2.23** *Gegeben seien  $n$  Bögen mit  $O(1)$  senkrechten Tangenten; je zwei Bögen schneiden sich höchstens  $s$  mal. Die Zelle  $Z$  im Arrangement, die einen gegebenen Punkt  $x$  enthält, kann in Zeit  $O(\lambda_{s+2}(n) \log^2 n)$  berechnet werden.<sup>9</sup>*

**Beweis.** Die Laufzeit ergibt sich aus der Rekursionsgleichung  $T(n) = 2T(\frac{n}{2}) + \text{Merge}(n)$  für den Divide-and-Conquer Ansatz. Die Kosten für einen Merge betragen nach Theorem 2.22  $O((r+b+k) \log(r+b+k))$ . In unserem Fall gilt  $k = 1$  und mit Theorem 2.18  $r + b \in O(\lambda_{s+2}(n))$ . Insgesamt also  $\text{Merge}(n) \in O(\lambda_{s+2}(n) \log(\lambda_{s+2}(n))) \subseteq O(\lambda_{s+2}(n) \log n)$ . Damit ergibt sich

$$T(n) = 2^\ell T(1) + C \cdot \sum_{i=1}^{\ell} \lambda_{s+2}(n) \log n \in O(\lambda_{s+2}(n) \log^2 n).$$

□

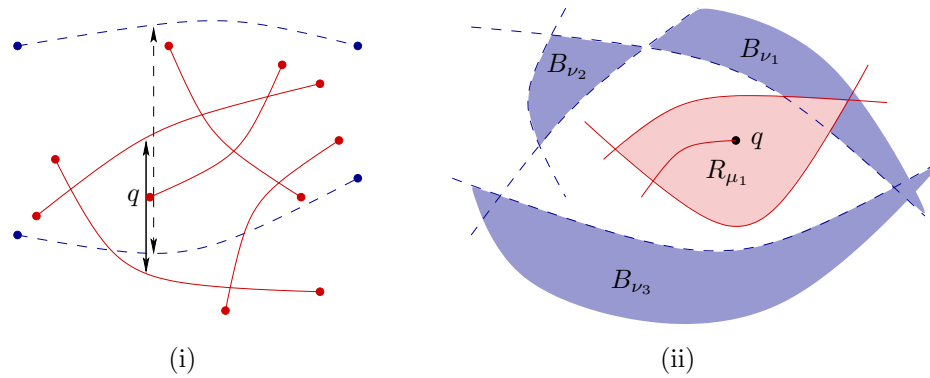


Abbildung 2.24: (i) Zusätzliche vertikale Segmente, (ii) bei  $q$  beginnt keine violette Region.

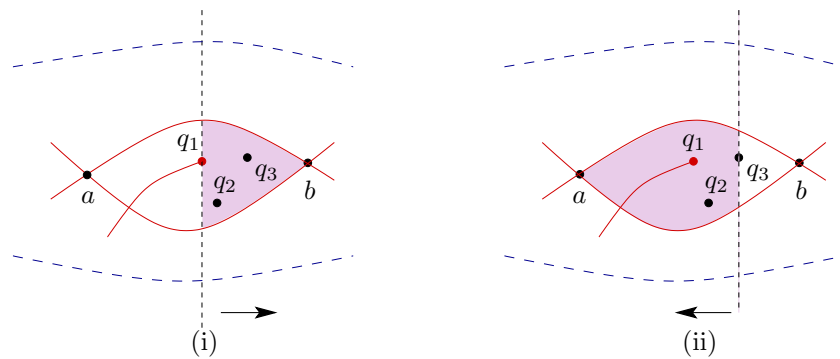


Abbildung 2.25: (i) Sweep von links nach rechts, (ii) Sweep von rechts nach links.

<sup>9</sup> $\log^2(n) = \log(n) \cdot \log(n)$ .

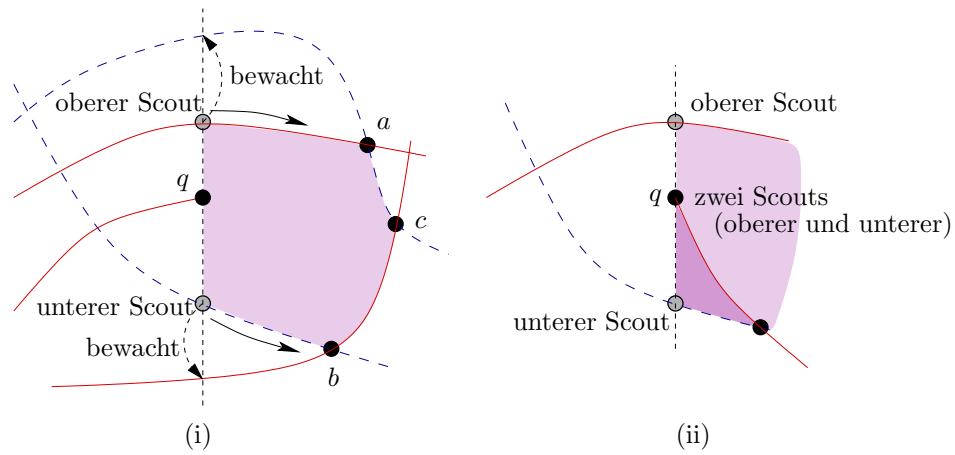


Abbildung 2.26: In  $q$  starten (i) eine oder (ii) zwei violette Teilregionen mit Scouts.

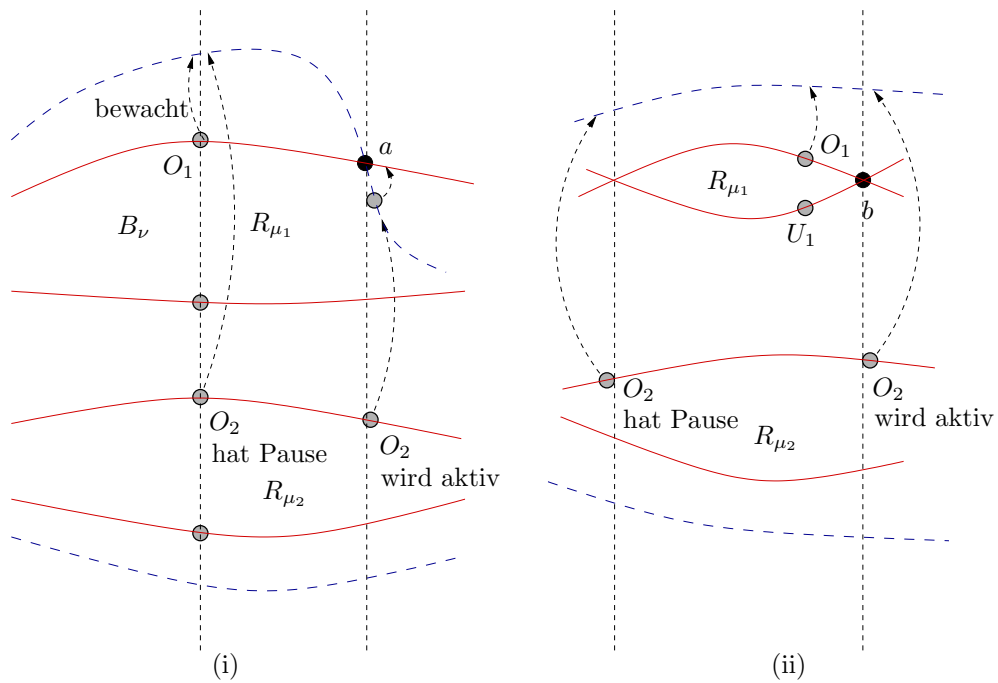


Abbildung 2.27: Beispiele für die Aufgaben der Scouts.

---

**Algorithmus 2.6** Sweep zur Berechnung der Teilzellen

---

(hier: Sweep von links nach rechts, rechts nach links analog)

**Ereignisstruktur:** Priority-Queue  $W$  sortiert nach X-Koordinaten, initialisiert mit den Punkten aus  $Q$  und allen Knoten auf dem Rand der blauen und roten Regionen.

**Sweep-Status-Struktur:** enthält die Bögen entlang der Sweepline, nach Y-Koordinaten sortiert. Für jeden Bogen wird ein Zeiger auf den beobachtenden Scout gespeichert und pro Scout ein Zeiger auf den Bogen, auf dem sich der Scout bewegt und den Bogen, den der Scout beobachtet.

**Ereignis:** Wenn ein Punkt  $q \in Q$  angetroffen wird, benutze die vertikalen Segmente von  $q$  um festzustellen, ob bei  $q$  eine violette Teilregion beginnt. Dies ist genau dann der Fall, wenn  $q$  sowohl in einem  $R_\mu$  als auch in einem  $B_\nu$  enthalten ist. In Abbildung 2.24(ii) beginnt in  $q$  keine violette Region, da  $q \notin \bigcup B_\nu$ . Falls ja, beginne bei  $q$  eine oder zwei violette Teilregionen, und starte am oberen und unteren Rand jeder Teilregion je einen Scout, siehe Abbildung 2.26.

**Aufgaben der Scouts:**

Hier die Aufgaben des oberen Scouts  $O$ , der untere Scout verhält sich symmetrisch.

- Der Scout startet auf dem zu  $q$  nächsthöheren Bogen. Hier starte  $O$  o.B.d.A. auf einem roten Bogen  $\varrho$ .
  - Der Scout bewegt sich mit der Sweepline.
  - Der Scout bewacht den nächsthöheren blauen Bogen  $\beta$ :
    - Berechne den in X-Richtung nächsten Schnittpunkt  $x$  zwischen  $\varrho$  und  $\beta$  und trage ihn in  $W$  ein (Abbildung 2.26(i), Punkte  $a$  und  $b$ ).
    - Wenn die Sweepline den Punkt  $x$  erreicht, wechselt  $O$  auf  $\beta$  und bewacht  $\varrho$ .
    - Wenn  $\beta$  endet oder einen anderen blauen Bogen schneidet, beobachte analog den nächsthöheren Bogen  $\beta'$ .
  - “Arbeitsteilung”: falls zwei Scouts denselben Bogen bewachen, ist nur der oberste aktiv. Längs der Sweepline gilt also: der Rand einer blauen Region wird nur von dem obersten darunter befindlichen Scout auf einem roten Bogen bewacht. Diese Invariante muß bei jedem Ereignis aktualisiert werden.
  - $O$  beobachtet den unteren Scout, d.h. ihr Schnittpunkt wird berechnet und falls existent in  $W$  eingetragen. Wenn beide sich treffen, ist das Ende der violetten Region erreicht und beide Scouts verschwinden (Abbildung 2.26(i), Punkt  $c$ ). Die violette Region endet evtl. schon eher, falls nämlich ein weiterer Punkt aus  $Q$  zwischen den Scouts erreicht wird.
-

### 2.2.2 Anwendung: Translation eines beliebigen Polygons

Mit den Ergebnissen des vorherigen Abschnittes können wir nun unser Bahnplanungsproblem lösen:

**Theorem 2.24** *Sei  $R$  ein polygonaler Roboter mit  $m$  Ecken, der sich in einer Umgebung mit polygonalen Hindernissen  $P_i$  mit insgesamt  $n$  Ecken bewegt. Gegeben seien Start- und Zielposition  $s, t$ . Dann läßt sich in Zeit  $O(mn \alpha(mn) \log^2(mn))$  eine kollisionsfreie Translation von  $s$  nach  $t$  bestimmen oder feststellen, daß keine solche existiert.*

**Beweis.** Die Laufzeit von Algorithmus 2.7 folgt aus Theorem 2.23 mit  $\lambda_3(mn) \in O(mn \alpha(mn))$  (Theorem A.9 auf Seite 152).  $\square$

Wenn man bedenkt, daß  $\alpha(mn)$  fast konstant ist und auch  $\log^2(mn)$  nur sehr langsam wächst, so bleibt im wesentlichen eine Laufzeit von  $O(mn)$ . Angesichts dieses doch recht komplexen Problems ein bemerkenswertes Ergebnis!

---

#### Algorithmus 2.7 Translation eines beliebigen Roboters

---

##### Vorbereitung:

- Betrachte das Arrangement  $\mathcal{A}$  der  $2mn$  Liniensegmente, die sich aus den Bedingungen der Ecke/Kante–Paare ergeben.
- Konstruiere in  $\mathcal{A}$  die Zelle  $Z$ , die  $s$  enthält.  
Komplexität  $O(mn \alpha(mn))$ , Laufzeit  $O(mn \alpha(mn) \log^2(mn))$
- Führe Trapezzerlegung in  $Z$  aus, und konstruiere den Zusammenhangsgraph (Sweep).  
 $O(mn \alpha(mn) \log(mn))$

**Query:** für gegebenes  $t$ :

- Bestimme das Trapez, das  $t$  enthält.  $O(\log(mn))$
  - Finde einen Pfad von  $s$  nach  $t$  im Zusammenhangsgraph.  $O(mn \alpha(mn))$
- 

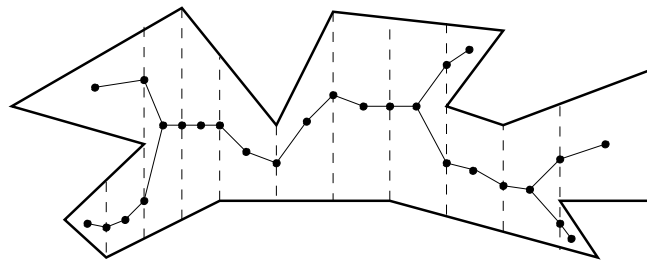
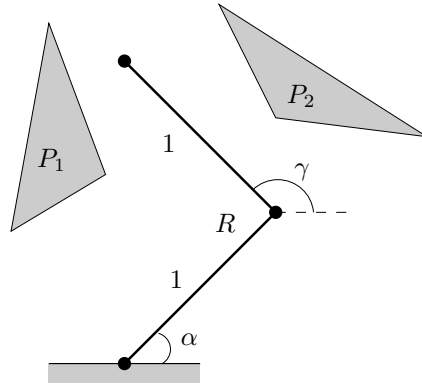


Abbildung 2.28: Trapezzerlegung und Zusammenhangsgraph einer Zelle.

### 2.2.3 Anwendung: Allgemeine Systeme mit zwei Freiheitsgraden

Bisher haben wir nur die Translation von Polygonen betrachtet, die eindeutig durch die Position ihres Referenzpunktes bestimmt sind, also zwei Freiheitsgrade (DOF) haben. Der Ansatz des vorherigen Abschnittes ist jedoch allgemein genug, um andere Systeme mit zwei Freiheitsgraden zu behandeln! Abbildung 2.29 zeigt einen Roboter mit zwei Armen, dessen Position durch die beiden Winkel  $\alpha$  und  $\gamma$  eindeutig beschrieben wird. Sein Konfigurationsraum ist also  $\mathcal{C} = [0, 2\pi) \times [0, 2\pi)$ .

Abbildung 2.29: Roboter mit zwei Armen und Freiheitsgraden  $\alpha$  und  $\gamma$ .

Wie sieht hier der Rand des Raumes der freien Konfigurationen  $\mathcal{C}_{\text{frei}}$  aus?

Ein Segment auf dem Rand von  $\mathcal{C}_{\text{frei}}$  entsteht z. B. durch den Kontakt einer Hindernisecke mit dem vorderen Arm (siehe Abbildung). Da die Hindernisecke nicht den Endpunkt von  $R$  berührt, muß ein  $\lambda \in [0, 1]$  existieren, so daß für den Kontaktpunkt  $(v, w)$  gilt:

$$\begin{pmatrix} v \\ w \end{pmatrix} = \begin{pmatrix} \cos \alpha \\ \sin \alpha \end{pmatrix} + \lambda \begin{pmatrix} \cos \gamma \\ \sin \gamma \end{pmatrix}$$

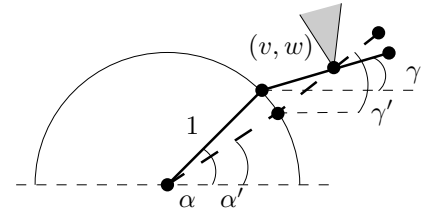
$$\Rightarrow \lambda = \frac{v - \cos \alpha}{\cos \gamma} = \frac{w - \sin \alpha}{\sin \gamma}$$

mit  $x := \cos \alpha, y := \cos \gamma$ :

$$(v - x)\sqrt{1 - y^2} = (w - \sqrt{1 - x^2})y$$

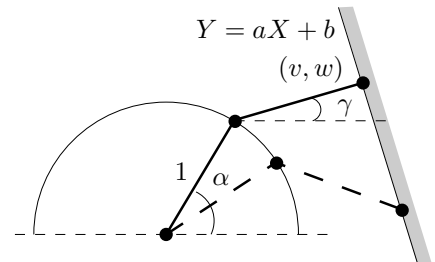
$$\Leftrightarrow (v - x)^2(1 - y^2) = (w^2 - 2w\sqrt{1 - x^2} + 1 - x^2)y^2.$$

Aus der letzten Gleichung hebt sich  $x^2 y^2$  fort. Löst man nach der Wurzel auf und quadriert die Gleichung, so ergibt sich eine Gleichung vom Grad<sup>10</sup> 6 in  $x, y$ . Wegen  $0 \leq \lambda \leq 1$  beschreibt diese ein Bogenstück vom Grad 6. Es handelt sich dabei um Bögen im Konfigurationsraum des Roboters, die Bögen sind daher nicht direkt in der Anschauung ablesbar.



Ein anderer Typ von Segmenten auf dem Rand von  $\mathcal{C}_{\text{frei}}$  entsteht durch den Kontakt des Roboterkopfes mit einer Wand. Die Wand läßt sich durch eine Gerade  $Y = aX + b$  beschreiben, und für den Kontaktpunkt  $(v, w)$  gilt:

$$\begin{aligned} (v, w) &= (\cos \alpha + \cos \gamma, \sin \alpha + \sin \gamma) \in \{Y = aX + b\} \\ \Leftrightarrow a(\cos \alpha + \cos \gamma) + b &= \sin \alpha + \sin \gamma \\ \Leftrightarrow a(x + y) + b &= \sqrt{1 - x^2} + \sqrt{1 - y^2} \\ \Leftrightarrow a(x + y)^2 + b - 2 + x^2 + y^2 &= \sqrt{1 - x^2} \sqrt{1 - y^2} \end{aligned}$$



Durch Quadrieren der letzten Gleichung ergibt sich ein Bogenstück vom Grad 4.

Alle Konfliktbögen haben einen Grad kleiner gleich 6, mit Bezouts Theorem<sup>11</sup> folgt, daß je zwei

<sup>10</sup>Der Grad eines Polynoms mit mehreren Variablen ist die Summe der Grade (d. h. der höchsten Potenzen) der einzelnen Variablen.

<sup>11</sup>Zwei algebraische Kurven von Grad  $m$  und  $n$  schneiden sich in höchstens  $m \cdot n$  Punkten.

Bögen höchstens  $6^2$  Schnitte haben. Mit Theorem 2.23 auf Seite 81 folgt, daß für dieses Problem kollisionsfreie Bewegungen in Zeit  $O(\lambda_{38}(n) \log^2 n) \subseteq O(n \log^{2+\varepsilon} n)$  geplant werden können.



## 2.3 Rotations- und Translationsbewegungen für konvexe Roboter

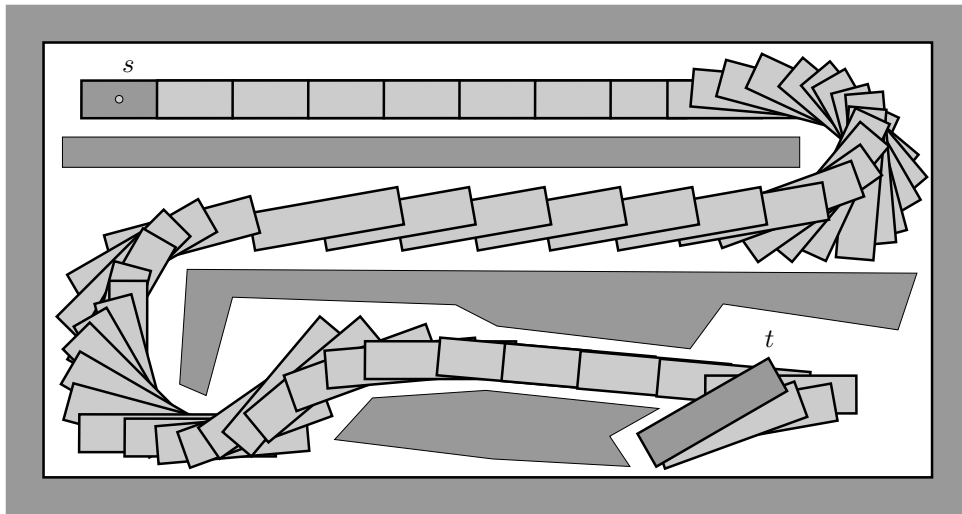


Abbildung 2.30: Bahn eines Roboters mit Rotationen und Translationen.

Bisher hatten wir nur Translationsbewegungen des Roboters betrachtet. Für kreisförmige Roboter ist dies keine Einschränkung, um jedoch einen langgezogenen Roboter zu bewegen, kann es sehr wohl nötig sein, diesen zu drehen, um enge Passagen zu überwinden oder Hindernissecken zu umschiffen.

In diesem Abschnitt wollen wir Bewegungen betrachten, die sowohl aus Translationen als auch aus Rotationen bestehen können, jedoch beschränken wir uns auf den Fall eines konvexen Roboters. Ein Beispiel für Roboter, die beliebige Translations- und Rotationsbewegungen ausführen können, sind Fahrzeuge mit Mecanum-Rädern.<sup>12</sup>

$R$  sei — wie gehabt — durch ein konvexes Polygon mit  $m$  Ecken gegeben und bewege sich in einer Umgebung mit polygonalen Hindernissen  $P_i$  mit insgesamt  $n$  Ecken. Ein klassischer Spezialfall dieses Problems ist die Bewegung einer Leiter (Piano-Movers-Problem, Abbildung 2.31); im Gegensatz zu Abschnitt 1.5 interessieren uns hier jedoch keine kürzesten Pfade.

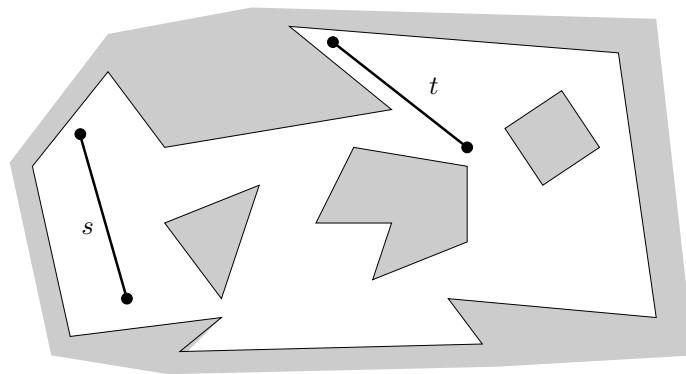


Abbildung 2.31: Kann die Leiter von  $s$  nach  $t$  bewegt werden?

### Theorem 2.25 (Ke, O'Rourke, 1987)

Die Bewegung eines Liniensegmentes mit Translation und Rotation in einer Umgebung mit polygonalen Hindernissen mit insgesamt  $n$  Ecken kann  $\Omega(n^2)$  viele Schritte erfordern. [KO88]

<sup>12</sup>Räder, auf deren Umfang mehrere Rollen in einem Winkel von  $45^\circ$  zur Orientierung angeordnet sind. Durch geeignete Ansteuerung aller Räder sind beliebige Translations- und Rotationsbewegungen möglich. [<http://prt.fernuni-hagen.de/forschung/mecanum-de.html>]

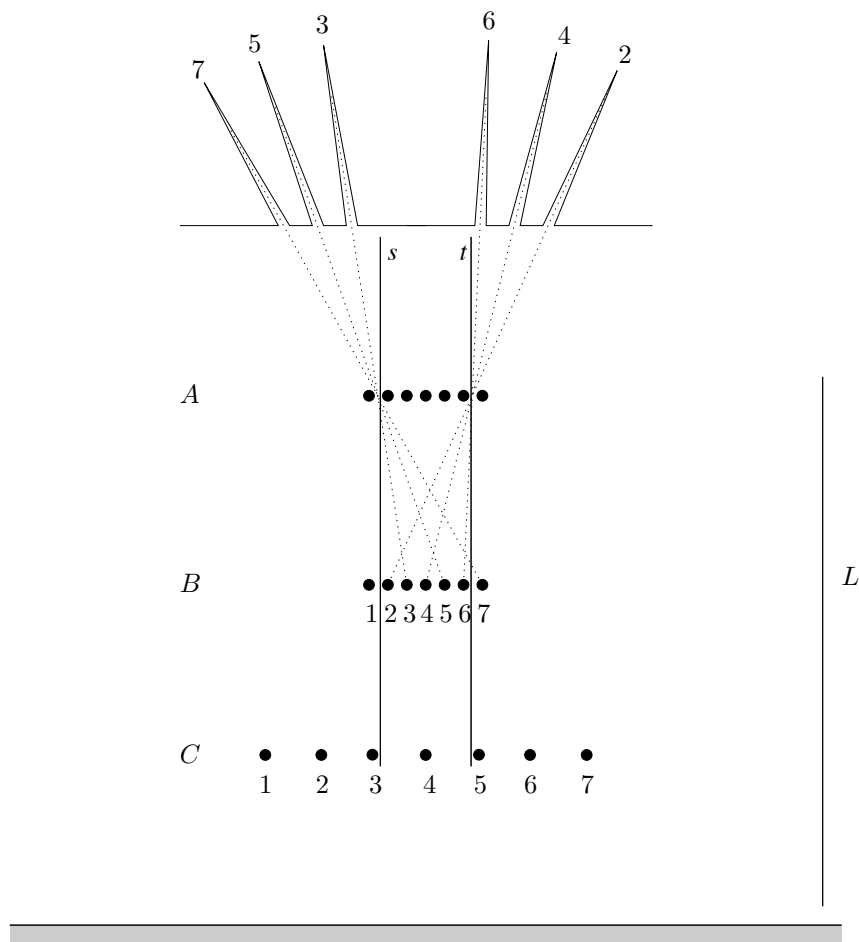


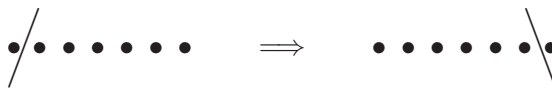
Abbildung 2.32: Die Bewegung eines Liniensegments kann  $\Omega(n^2)$  viele Schritte erfordern.

**Beweis.** Abbildung 2.32 zeigt eine Umgebung, in der  $\Omega(n^2)$  viele Schritte gebraucht werden, um das Liniensegment  $L$  von  $s$  nach  $t$  zu bewegen.<sup>13</sup>

- Sei  $(i, i+1)$  eine feste Lücke zwischen den B-Pflocken. Um den B-Pflock  $i+1$  zu überwinden, muß  $L$  in die Kerbe  $i+1$  hineingeschoben werden.
- Die Kerben sind so konstruiert, dass das Überwinden eines B-Pflocks mit ungeradem (geradem) Index nur in der ersten (letzten) Lücke zwischen den A-Pflocken möglich ist.
- Schiebt man  $L$  ganz nach unten, so kann man eine konstante Anzahl von A-Pflocken überwinden, bis  $L$  den nächsten C-Pflock berührt.
- Um einen C-Pflock zu überwinden, schiebt man  $L$  hoch bis zur oberen Wand.
- Um in einer festen Lücke zwischen den B-Pflocken die Orientierung von  $L$  folgendermassen zu ändern, sind  $\Omega(n)$  Züge nötig:



- Zur Überwindung aller B-Pflocke von  $s$  nach  $t$



<sup>13</sup>Besser als jede Erklärung ist es sich das Applet *Bewegung einer Leiter in 2 Dimensionen* anzuschauen, dass unter <http://www.geometrylab.de/LowerBound2D/index.html> (Stand 2009) zu finden ist.

müssen abwechselnd rechte und linke Kerben aufgesucht werden, was je einen Durchlauf durch die Reihe der A-Pföcke erfordert, insgesamt also  $\Omega(n^2)$  Züge.  $\square$

Bemerkenswert ist, daß in Theorem 2.25 nicht mit der Komplexität von  $\mathcal{C}_{\text{frei}}$  argumentiert wird, die Aussage bezieht sich vielmehr auf die Anzahl der notwendigen Schritte. Die Planung der Bahn kann nicht einfacher sein.

**Bemerkung 2.26** *Im  $\mathbb{R}^3$  kann die Bewegung eines Liniensegments  $\Omega(n^4)$  Züge erfordern und in Zeit  $O(n^6 \log n)$  geplant werden.* [K088]

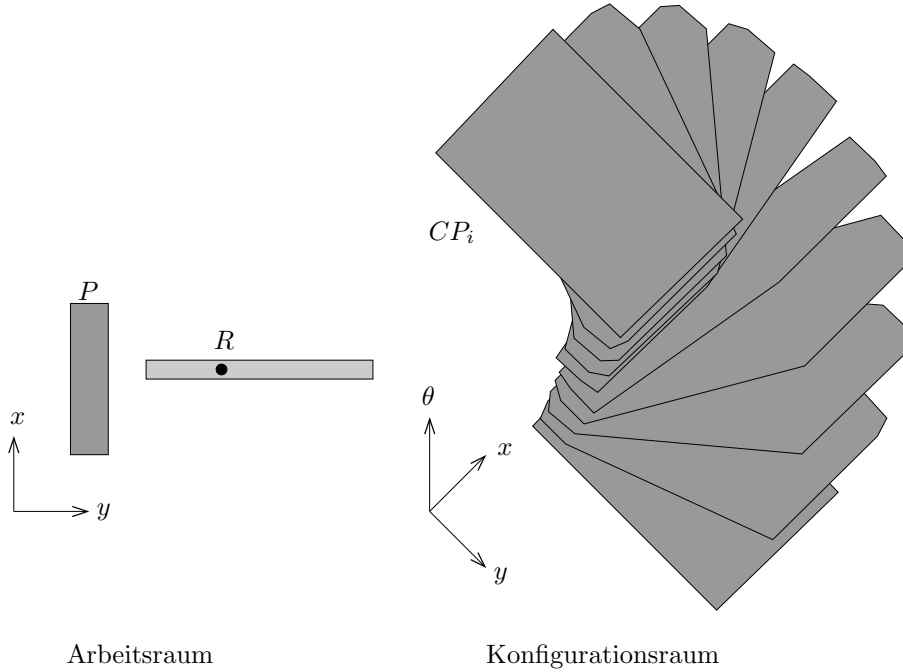


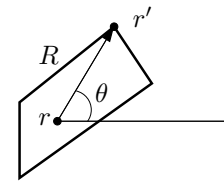
Abbildung 2.33: Konfigurationsraum-Hindernis bei Rotation und Translation.

Wir wollen nun im  $\mathbb{R}^2$  Rotations- und Translationsbewegungen für beliebige konvexe Roboter planen. Der Roboter hat drei Freiheitsgrade:  $(x, y, \theta) \in \mathcal{C} = \mathbb{R}^2 \times [0, 2\pi)$ , vereinfacht auch  $\mathcal{C} = \mathbb{R}^3$ .

Die Konfigurationsraum-Hindernisse  $CP_i$  sind hier keine Polyeder mehr, sie haben gekrümmte Flächen und Kanten. Es sind durch Regelflächen<sup>14</sup> berandete, dreidimensionale Objekte. Betrachtet man den Schnitt eines  $CP_i$  mit einer Ebene mit konstantem  $\theta_i$ , so ist der Schnitt genau die Minkowski-Summe  $P_i \oplus -R(0, 0, \theta_i)$ . Man kann sich ein Konfigurationsraum-Hindernis also als den Stapel der Minkowski-Summen über alle  $\theta_i$  vorstellen, siehe Abbildung 2.33.

Im Prinzip würde der Ansatz, den Raum  $\mathcal{C}_{\text{frei}}$  in Zellen zu zerlegen und einen Zusammenhangsgraph zu konstruieren, hier auch funktionieren, wäre aber sehr schwierig zu implementieren.

Ein naiver Ansatz ist, mit fester Orientierung  $\theta_i$  für  $\theta_i = i \cdot \frac{360^\circ}{k}$ ,  $0 \leq i \leq k-1$  mit bisher bekannten Verfahren Translationsbahnen zu planen und die 2-DOF-Lösungen geeignet zusammenzusetzen. Dies liefert jedoch nicht immer korrekte Lösungen: die Rotation zwischen zwei Schichten  $\theta_i$  und  $\theta_{i+1}$  kann eine Kollision mit einem Hindernis verursachen, wenn nämlich die Konfigurationen  $(x, y, \theta_i)$  und  $(x, y, \theta_{i+1})$  in  $\mathcal{C}_{\text{frei}}$  liegen, die Konfiguration  $(x, y, \theta')$  mit einem Winkel  $\theta'$  zwischen  $\theta_i$  und  $\theta_{i+1}$  jedoch verboten ist.

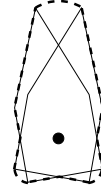


<sup>14</sup>Flächen, die bei der Bewegung eines Liniensegments im Raum entstehen.

Eine mögliche Abhilfe wäre, den Roboter  $R$  um  $\frac{360^\circ}{k}$  zu drehen und die dabei überstrichene Fläche als Roboter  $R'$  für die Translationsplanung zu betrachten. Die mit diesem Ansatz berichteten Bahnen sind dann stets kollisionsfrei, jedoch wird für  $R'$  evtl. keine Bahn gefunden, obwohl für  $R$  eine Bahn existiert.

Der naive Ansatz widerspricht also unserem Ziel korrekter Bahnplanung. Trotzdem werden wir die Idee, nur endlich viele Orientierungen zu verwenden, weiter verfolgen; wir müssen nur die richtige Auswahl an Orientierungen finden.

Zur Vorbereitung betrachten wir im folgenden Abschnitt kritische Plazierungen des Roboters.



### 2.3.1 Kritische Plazierungen

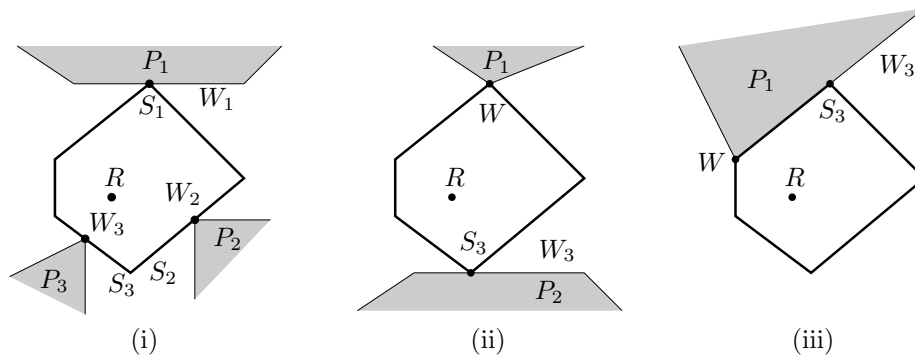


Abbildung 2.34: Kritische Plazierungen.

**Definition 2.27** Sei  $R$  ein konvexer Roboter mit  $m$  Ecken, und seien  $P_i$  polygonale Hindernisse. Ein **Kontaktpaar**  $O = (W, S)$  liegt vor, wenn

- (a)  $W$  ist eine Hinderniskante und  $S$  eine Roboterecke (Typ I) oder
- (b)  $W$  ist eine Hindernisecke und  $S$  eine Roboterseite (Typ II) oder
- (c)  $W$  ist eine Hindernisecke und  $S$  eine Roboterecke (Typ III)

und  $W$  berührt  $S$ .

Eine halbfreie<sup>15</sup> Platzierung  $(x, y, \theta)$  von  $R$  heißt **kritische Platzierung** genau dann, wenn an der Platzierung

- (A) drei paarweise verschiedene Kontaktpaare vom Typ I oder II oder
- (B) ein Kontaktpaar vom Typ III und ein Kontaktpaar vom Typ I oder II beteiligt sind.

Abbildung 2.34 zeigt Beispiele für kritische Plazierungen. (i) zeigt den "klassischen" Fall, in dem der Roboter an drei Stellen ein Hindernis berührt, also drei Kontaktpaare vom Typ I oder II beteiligt sind. (ii) und (iii) zeigen Beispiele für kritische Plazierungen, an denen ein Kontaktpaar vom Typ III beteiligt ist. Kritische Plazierungen spielen eine wichtige Rolle, wie folgendes Lemma zeigt:

**Lemma 2.28** In jeder Zusammenhangskomponente von  $C_{\text{frei}}$  gibt es mindestens eine kritische Platzierung.

**Beweis.** Starte mit freier Platzierung von  $R$  und bewege  $R$  nach unten, bis  $R$  ein Hindernis berührt. Nehmen wir o. E. an, eine Ecke von  $R$  berühre die Kante eines  $P_i$ , siehe Abbildung 2.35(i). Verschiebe dann  $R$  längs dieser Kante, bis einer der folgenden Fälle eintritt:

<sup>15</sup>Zur Erinnerung: Eine Platzierung heißt **halbfreie Platzierung** wenn der Roboter eine Hinderniswand berührt.

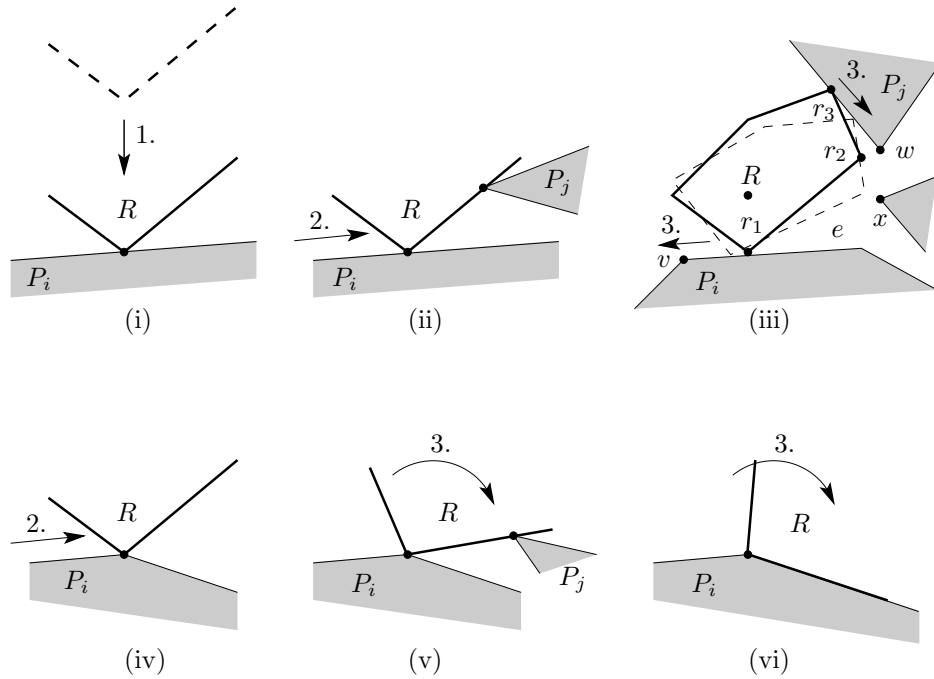


Abbildung 2.35: Beweis: In jeder Zusammenhangskomponente gibt es mindestens eine kritische Plazierung.

1. Es entsteht ein weiterer Kontakt zu einem Hindernis  $P_j$  (Abb. (ii)). Falls eine Ecke  $R$  eine Ecke von  $P_j$  berührt, hat  $R$  eine kritische Plazierung erreicht, ansonsten bewege  $R$  unter Aufrechterhaltung der Kontakte zu  $P_i$  und  $P_j$ . Man kann sich leicht klarmachen, daß dabei einer der folgenden Fälle eintreten muß (Abbildung 2.35(iii)):

- (a) Eine der beiden Kontaktecken von  $R$  erreicht eine Hindernisecke ( $r_1$  erreicht  $v$  oder  $r_3$  erreicht  $w$ ).
- (b) Eine Kante von  $R$  erreicht eine Hinderniskante ( $r_2$  erreicht  $e$ ).
- (c) Es entsteht ein weiterer Hinderniskontakt ( $R$  berührt die Hindernisecke  $x$ . Diese kann zu einem dritten Hindernis  $P_k$  gehören oder zu  $P_i$  oder  $P_j$ ).

In allen Fällen ist ein dritter Kontakt entstanden und  $R$  hat eine kritische Plazierung erreicht.

2. Es wird ein Endpunkt  $v$  erreicht (Abbildung 2.35(iv)). Rotiere dann  $R$  um  $v$ , bis

- (a)  $R$  hat Kontakt zu weiterem Hindernis  $P_j$  (Abb. (v)).
- (b)  $R$  hat weiteren Kontakt zu Hindernis  $P_i$  (Abb. (vi)).

In beiden Fällen hat  $R$  eine kritische Plazierung erreicht. □

Im folgenden wollen wir die Anzahl der kritischen Plazierungen vom Typ A abschätzen. Die Anzahl möglicher Plazierungen vom Typ B ist dagegen mit  $O(m^2n^2)$  sehr einfach anzugeben (Übungsaufgabe).

**Bemerkung 2.29** Wenn man  $R$  unter Aufrechterhaltung von zwei Kontaktpaaren bewegt, beschreibt der Referenzpunkt eine Kurve vom Grad  $\leq 4$ .

**Beweis.** Siehe [SS83a]. □

**Korollar 2.30** Für je drei Kante/Ecke-Paare<sup>16</sup>  $O_i = (W_i, S_i)$  gibt es höchstens vier halbfreie Plazierungen von  $R$ , bei denen die Kontakte  $O_i$  entstehen. [SS83a]

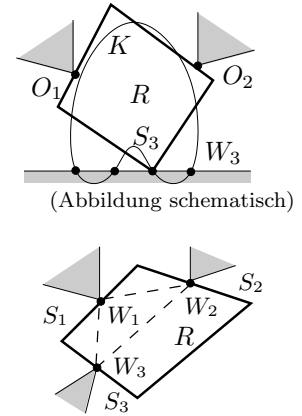
<sup>16</sup>Zur Erinnerung:  $W_i$  bezeichnet den am Kontakt beteiligten Teil des Hindernisses,  $S_i$  den des Roboters.

**Beweis.** Angenommen, bei  $O_3 = (W_3, S_3)$  ist  $S_3$  eine Ecke von  $R$ . Lege den Referenzpunkt auf  $S_3$ . Halte  $O_1$  und  $O_2$  fest und bewege  $R$ .

$\Rightarrow S_3$  beschreibt Kurve  $K$  vom Grad  $\leq 4$ .

$\Rightarrow K$  hat  $\leq 4$  Schnitte mit  $W_3$ .

Angenommen, bei allen  $O_i$  ist  $S_i$  eine Kante von  $R$ . Dann halte  $R$  fest und bewege die Hindernisse, d.h. bewege das Dreieck  $W_1W_2W_3$  der Hindernisecken unter Aufrechterhaltung der Kontakte mit den Roboteranten  $S_1, S_2$  und  $S_3$ . Damit kann wie oben gefolgert werden.  $\square$



Es gibt  $2mn$  mögliche Kontaktpaare  $O_i$  vom Typ I oder II, von denen wir drei auswählen. Insgesamt kann es also höchstens  $4 \cdot \binom{2mn}{3} \in \Theta(m^3n^3)$  viele kritische Plazierungen geben. Von der Konvexität des Roboters haben wir dabei noch keinen Gebrauch gemacht!

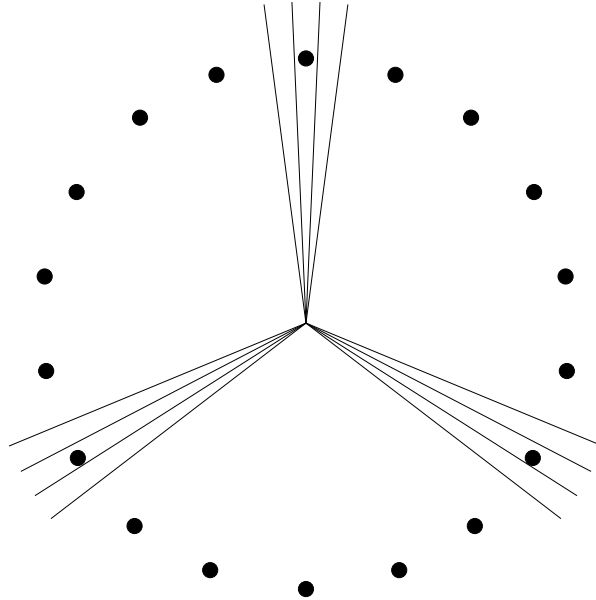


Abbildung 2.36: Ein nichtkonvexer Roboter mit  $\Theta(m^3n^3)$  kritischen Plazierungen.

**Theorem 2.31** Für nichtkonvexe Roboter kann es  $\Theta(m^3n^3)$  viele kritische Plazierungen geben.

**Beweis.** Siehe Abbildung 2.36.  $\square$

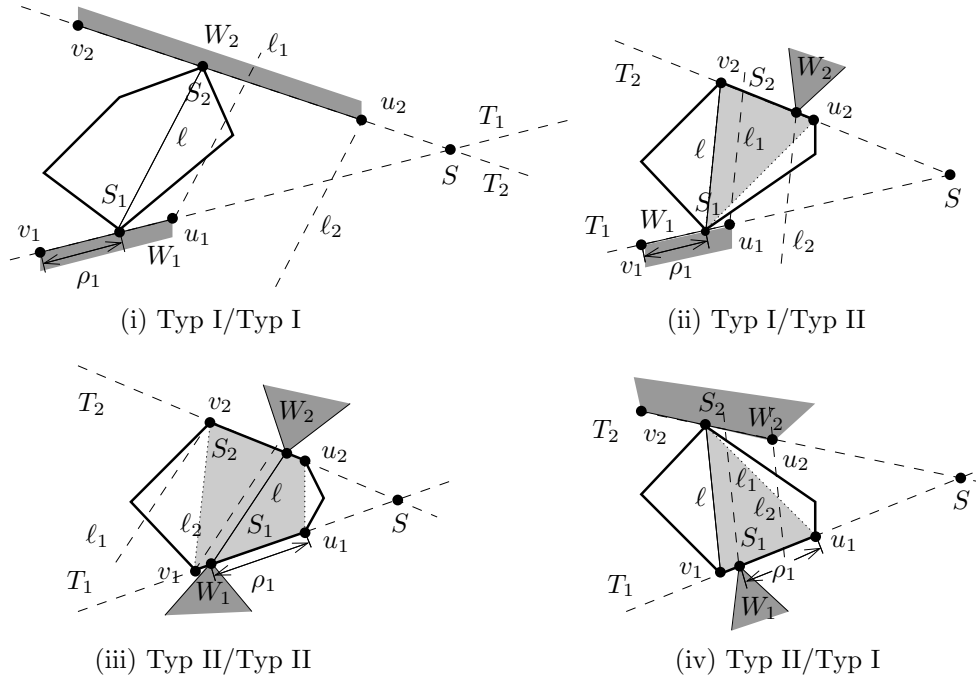
Für konvexe Roboter gilt jedoch:

**Theorem 2.32** (Leven, Sharir, 1987)

Die Anzahl der kritischen Plazierungen eines konvexen  $m$ -Ecks  $R$  zwischen Polygonen mit insgesamt  $n$  Ecken liegt in  $O(mn \lambda_6(mn))$ . [LS87]

Dabei gilt  $\lambda_6(mn) \in O(mn \log^*(mn))$ , die Anzahl der kritischen Plazierungen ist also kaum mehr als quadratisch.

Zum Beweis dieses Theorems sind einige Vorüberlegungen nötig. Betrachten wir zunächst alle möglichen Arten von zwei Kontaktpaaren  $O_1, O_2$  zwischen  $R$  und den Hindernissen, siehe Abbildung 2.37. Nehmen wir an, die beiden Tangenten  $T_1$  und  $T_2$ , die durch Verlängerung der

Abbildung 2.37:  $O_2$  beschränkt  $O_1$ .

beiden Kontaktpaare entstehen, seien nicht parallel. Verschiebt man nun  $R$  längs der Tangente  $T_1$  (o. E.) auf den Schnittpunkt  $S$  mit der anderen Tangente zu, so entsteht an der Kontaktstelle  $O_2$  ein verbotener Schnitt. Falls dieser verbotene Durchschnitt bis zum Ende der Translation entlang der Kante andauert, so sagen wir,  $O_1$  wird von  $O_2$  bei Winkel  $\theta$  beschränkt. Genauer:

**Definition 2.33** Seien  $O_1, O_2$  zwei Kontaktpaare,  $T_1$  und  $T_2$  die Verlängerungen der beteiligten Kanten.  $O_2$  **beschränkt**  $O_1$  bei Winkel  $\theta$ , wenn eine Platzierung  $R(x, y, \theta)$  mit den Kontaktpaaren  $O_1$  und  $O_2$  existiert, so daß  $R' = \text{ch}(S_1 \cup S_2)$  stets  $W_2$  schneidet, wenn  $R$  bei fester Orientierung  $\theta$  entlang der Tangente  $T_1$  unter Beibehaltung des Kontaktes  $O_1$  in Richtung des Schnittpunktes  $S$  von  $T_1$  und  $T_2$  verschoben wird.

Die konvexe Hülle  $\text{ch}(S_1 \cup S_2)$  ist in Abbildung 2.37 hellgrau dargestellt, in (i) gilt  $\text{ch}(S_1 \cup S_2) = \ell$ . Im folgenden bezeichnen  $u_{1/2}$  die Endpunkte der am Kontakt beteiligten Kanten in Richtung  $S$ ,  $v_{1/2}$  die Endpunkte der Kanten in die andere Richtung und  $x_i$  sei der Berührungspunkt des Kontaktpaares.

Die Definition der Beschränkung setzt voraus, daß sich die Tangenten  $T_1$  und  $T_2$  schneiden. Es gibt jedoch Fälle, in denen  $T_1$  und  $T_2$  parallel sind, z. B. wenn eine Roboterkante ( $S_1 = S_2$ ) zwei Hindernisecken berührt oder beide involvierten Kanten parallel sind. In allen anderen Fällen gilt:

**Lemma 2.34** Für zwei Kontaktpaare  $O_1$  und  $O_2$  gilt — von degenerierten Fällen abgesehen — mindestens eine der beiden Aussagen:

- (i)  $O_1$  beschränkt  $O_2$  bei  $\theta$ .
- (ii)  $O_2$  beschränkt  $O_1$  bei  $\theta$ .

**Beweis.**(Skizze)

Betrachte die Fallunterscheidung in Abbildung 2.37. In den Fällen (i) und (iii) sei  $\ell$  die Verbindung zwischen den Kontaktpunkten  $x_1$  und  $x_2$ , in den Fällen (ii) und (iv) die Verbindung zwischen der beteiligten Roboterecke  $S_1$  bzw.  $S_2$  und der Ecke  $v_2$  bzw.  $v_1$  der beteiligten Hinderniskante. Weiter sei

- (i)  $\ell_1$  die Parallele zu  $\ell$  durch  $u_1$ ,  $\ell_2$  die durch  $u_2$ .
- (ii)  $\ell_1$  die Parallele zu  $\ell$  durch  $u_1$ ,  $\ell_2$  die durch  $x_2$ .
- (iii)  $\ell_1$  die Parallele zu  $\ell$  durch  $v_2$ ,  $\ell_2$  die durch  $v_1$ .
- (iv)  $\ell_1$  die Parallele zu  $\ell$  durch  $x_1$ ,  $\ell_2$  die durch  $u_2$ .

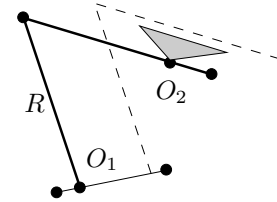
Dann gilt

- (i)  $\ell_1$  schneidet  $W_2$  oder  $\ell_2$  schneidet  $W_1$ .
- (ii)  $\ell_1$  schneidet  $\overline{x_2v_2}$  oder  $\ell_2$  schneidet  $\overline{x_1u_1}$ .
- (iii)  $\ell_1$  schneidet  $S_1$  oder  $\ell_2$  schneidet  $S_2$ .
- (iv)  $\ell_1$  schneidet  $\overline{x_2u_2}$  oder  $\ell_2$  schneidet  $\overline{x_1u_1}$ .

In allen Fällen schneidet  $R'$  während der Bewegung  $W_1$  oder  $W_2$  und es gilt:  $O_2$  beschränkt  $O_1$  oder  $O_1$  beschränkt  $O_2$ .  $\square$

Dabei ist die Konvexität von  $R$  entscheidend. In nebenstehender Abbildung mit nichtkonvexem  $R$  ist nicht klar, ob  $O_1$  von  $O_2$  beschränkt wird oder  $O_2$  von  $O_1$ .

Es gibt auch Fälle, in denen beides gilt, allerdings ist diese Platzierung nicht notwendigerweise frei:



**Bemerkung 2.35** Seien  $O_1, O_2$  fest. Dann gibt es genau eine Orientierung  $\theta$  von  $R$ , für die  $\ell_1$  und  $\ell_2$  zusammenfallen; für die also gilt:

$O_1$  beschränkt  $O_2$  bei  $\theta$  **und**  $O_2$  beschränkt  $O_1$  bei  $\theta$ .

Damit können wir unser Theorem beweisen:

**Beweis.** (Theorem 2.32)

Wir wollen einen Zusammenhang zwischen der Anzahl kritischer Platzierungen und der unteren Kontur von Funktionen<sup>17</sup> herstellen. Sei

$$\Pi_{O_1O_2} := \{ \theta \mid O_2 \text{ beschränkt } O_1 \text{ bei Winkel } \theta \}.$$

Mit den oben eingeführten Bezeichnungen  $u_{1/2}$ ,  $v_{1/2}$  und  $x_i$  sei

$$\rho_i := \begin{cases} |\overline{x_i u_i}|, & \text{falls } S_i \text{ Roboterseite} \\ |\overline{v_i x_i}|, & \text{falls } S_i \text{ Roboterecke} \end{cases}$$

$\rho_i$  gibt die Distanz an, um die in Definition 2.33 bei konstantem  $\theta$  und unter Aufrechterhaltung des Kontaktes  $O_i$  geschoben werden kann, ohne daß ein verbotener Schnitt entsteht, vgl. auch Abbildung 2.37.

Definiere damit die Funktion

$$\begin{aligned} f_{O_1O_2} : \Pi_{O_1O_2} &\longrightarrow \mathbb{R}^{>0} \\ \theta &\longmapsto \rho_1 \end{aligned}$$

Eine entscheidende Eigenschaft ist: Sei  $(x, y, \theta)$  eine halbfreie Platzierung von  $R$  mit Kontaktpaaren  $O_1$  und  $O_2$  mit  $O_2$  beschränkt  $O_1$  mit Abstand  $\rho_1 = f_{O_1O_2}(\theta)$ . Für  $\rho > \rho_1$  ist keine (halb)freie Platzierung mit Kontaktpaaren  $O_1, O_2$  und Abstand  $\rho$  möglich, da  $O_2$  einen verbotenen Schnitt liefert.

Sei nun  $(x, y, \theta)$  eine kritische Platzierung mit Kontaktpaaren  $O_1, O_2, O_3$  und  $\mathcal{A}(O_i)$  das Arrangement der Funktionsgraphen  $f_{O_iO_k}$  mit  $O_k$  beschränkt  $O_i$ . Folgende Fälle können eintreten:

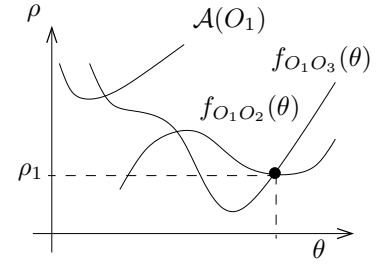
<sup>17</sup>Siehe Abschnitt A.4.2 auf Seite 153.



**1. Fall:** Zwei Kontaktpaare beschränken das dritte, o. E.  $O_2$  und  $O_3$  beschränken  $O_1$ .

Betrachte das Arrangement  $\mathcal{A}(O_1)$  der Funktionsgraphen von  $f_{O_1O}$  mit  $O$  beschränkt  $O_1$ . Dann ist der Punkt  $(\theta, \rho_1)$  mit  $\rho_1 = f_{O_1O_2}(\theta) = f_{O_1O_3}(\theta)$  ein Eckpunkt der unteren Kontur im Arrangement  $\mathcal{A}(O_1)$ . Denn wäre  $f_{O_1O_2}(\theta) \neq \min_O f_{O_1O}(\theta)$ , z. B.  $f_{O_1O_2}(\theta) > f_{O_1O'}(\theta)$ , so könnte  $(x, y, \theta)$  wegen zu großem

$\rho$ -Wert im Vergleich zu  $f_{O_1O'}(\theta)$  nicht erlaubt sein. Ebenso für  $f_{O_1O_3}(\theta)$ .

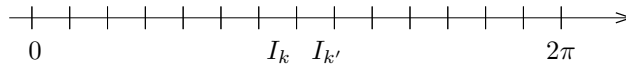


Wir haben hierbei angenommen, daß bei  $O_1$  stets derselbe Endpunkt der involvierten Kanten zur Definition von  $\rho_1 = f_{O_1O}(\theta)$  dient. Es kann aber auch der andere Endpunkt sein, in diesem Fall muß man zwei Klassen  $\mathcal{A}_L(O_1)$  und  $\mathcal{A}_R(O_1)$  bilden und eine Fallunterscheidung durchführen.

Das Arrangement  $\mathcal{A}(O_1)$  besteht aus  $2mn$  Funktionen, von denen je zwei höchstens vier Schnittpunkte haben. Nach Theorem A.11 auf Seite 153 hat die untere Kontur die Komplexität  $O(\lambda_6(mn))$ . Für ein festes  $O_1$  gibt es also höchstens  $O(\lambda_6(mn))$  solcher Plazierungen, damit insgesamt höchstens  $O(mn \lambda_6(mn))$ .

**2. Fall:** In  $O_1, O_2, O_3$  gibt es keine zwei Paare, die das dritte beschränken. Dann liegt nach Lemma 2.34 ein Zyklus vor, nehmen wir o. E. an  $O_1$  beschränkt  $O_3$ ,  $O_3$  beschränkt  $O_2$  und  $O_2$  beschränkt  $O_1$ .

Betrachte alle Knoten der unteren Konturen in sämtlichen Arrangements  $\mathcal{A}(O)$ . Diese liefern eine Zerlegung des Intervalls  $[0, 2\pi)$  in  $O(mn \lambda_6(mn))$  viele Teilintervalle  $I_k$ , innerhalb derer die untere Kontur eines jeden Arrangements eindeutig von einer der Funktionen bestimmt wird.



Sei  $O_1$  ein beliebiges Kontaktpaar. Die Frage ist nun: gibt es über  $I_k$  ein zyklisches Tripel von Kontaktpaaren, an dem  $O_1$  beteiligt ist? Und falls ja, bilden sie auch eine kritische Platzierung?

Zur Beantwortung dieser Frage, gehe wie folgt vor:

1. Sehe in  $\mathcal{A}(O_1)$  nach, welche Funktion  $f_{O_1O_2}$  die untere Kontur über  $I_k$  bildet.
2. Sehe in  $\mathcal{A}(O_2)$  nach, welche Funktion  $f_{O_2O_3}$  die untere Kontur über  $I_k$  bildet.
3. Sehe in  $\mathcal{A}(O_3)$  nach, ob die Funktion  $f_{O_3O_1}$  die untere Kontur über  $I_k$  bildet. Wenn ja, ist ein Zyklus gefunden.

Dies zeigt: in einem Intervall  $I_k$  kann es für jedes  $O_1$  höchstens ein zyklisches Tripel geben, an dem  $O_1$  beteiligt ist, insgesamt also  $O(mn)$  viele.

Sei nun  $I_{k'}$  ein Nachbarintervall von  $I_k$  und  $\theta^*$  der gemeinsame Endpunkt. In  $\theta^*$  sind drei Kontaktpaare  $O_1, O_2$  und  $O_3$  beteiligt. Nur in den Arrangements dieser Kontaktpaare können sich die unteren Konturen beim Übergang von  $I_k$  zu  $I_{k'}$  ändern, also kann es über  $I_{k'}$  höchstens  $O(1)$  neue zyklische Tripel geben. Wir haben  $O(mn \lambda_6(mn))$  viele Teilintervalle, in jedem kommen  $O(1)$  viele Tripel hinzu. Für jedes zyklische Tripel kann es höchstens vier kritische Platzierungen geben, insgesamt also  $O(mn \lambda_6(mn))$ .  $\square$

**Bemerkung 2.36**

- (i) Wir haben für  $m = 2$  (Leiter) in Abbildung 2.32 auf Seite 88 ein Beispiel mit  $\Theta(n^2)$  möglichen kritischen Platzierungen gesehen.
- (ii) Man kann auch für konvexe Roboter mit  $m > 2$  Ecken Hindernisse mit  $n$  Ecken so gestalten, daß  $\Theta(m^2 n^2)$  viele kritische Platzierungen entstehen.
- (iii) Die Frage, ob es sogar  $\Theta(mn \lambda_6(mn))$  viele kritische Platzierungen geben kann, ist offen.

**Definition 2.37**

Sei  $T^+$  die Menge aller Platzierungen  $(x, y, \theta)$ ,

- (i) die eindeutig den Ecken der unteren Konturen der  $O(mn)$  vielen Arrangements  $\mathcal{A}(O_i)$  entsprechen oder
- (ii) die einem zyklischen Tripel von Kontaktpaaren  $(O_1, O_2, O_3)$  entsprechen, mit  
 $O_1$  beschränkt  $O_3$ ,  
 $O_3$  beschränkt  $O_2$  und  
 $O_2$  beschränkt  $O_1$ .

Jedes Tripel liefert  $\leq 4$  Platzierungen, und jede der 4 möglichen Platzierungen ist durch  $(\rho_1, \theta)$  (oder  $(\rho_2, \theta)$ ,  $(\rho_3, \theta)$ ) eindeutig bestimmt. Dies sind Nullstellen einer Kurve 4. Grades.

$T^+$  enthält alle kritischen Platzierungen, aber eventuell auch verbotene Platzierungen: es könnte z. B. ein Kontaktpaar  $O_4$  einen verbotenen Schnitt bei Kontakt in  $O_1$  liefern, der in  $\mathcal{A}(O_1)$  nicht sichtbar ist, weil  $O_4$  nicht  $O_1$  beschränkt. Wir werden diese Platzierungen später herausfiltern; vorerst sind wir daran interessiert, die Menge  $T^+$  zu berechnen.

**Theorem 2.38** (Leven, Sharir, 1987)

Die Menge  $T^+$  läßt sich in Zeit  $O(mn \lambda_6(mn) \log(mn))$  berechnen.

[LS87]

**Beweis.** Algorithmus 2.8 betrachtet insgesamt  $O(mn \lambda_6(mn))$  Intervalle mit je  $O(\log(mn))$  Einfügeaufwand und insgesamt  $O(mn \lambda_6(mn))$  Löschoptionen aus  $W \Rightarrow$  Berechnungszeit  $O(mn \lambda_6(mn) \log(mn))$ .  $\square$

Im nächsten Abschnitt werden wir den Konfigurationsraum des Roboters für feste Orientierungen  $\theta$  betrachten und beobachten, wie dieser sich für fortlaufende  $\theta$  ändert. Die Orientierungen an den kritischen Platzierungen sind dabei genau die Orientierungen, an denen sich die Struktur des Konfigurationsraums ändert.

---

**Algorithmus 2.8** Berechnung von  $T^+$ 

---

- Berechne für jedes Kontaktpaar  $O_i$ , über welchen Intervallen  $O_i$  von  $O_k$  beschränkt wird. Bilde daraus die Arrangements  $\mathcal{A}_{L/R}(O_i)$ .  $O(n^2m^2)$
- Für jedes der  $O(mn)$  Arrangements  $\mathcal{A}(O_i)$  berechne die untere Kontur (Divide-and-Conquer mit Sweep im Merge-Schritt). Da sich zwei Funktionsgraphen  $f_{O_1O_2}, f_{O_1O_3}$  höchstens viermal schneiden:  $O(\lambda_6(mn) \log(mn))$  pro Arrangement
- Kritische Platzierungen, Fall 1:  
Berichte die Schnittpunkte von Funktionen auf den unteren Konturen in den Arrangements.
- Kritische Platzierungen, Fall 2:
  - Sortiere die  $O(mn)$  Listen der Eckpunkte der unteren Konturen zu einer Liste  $L = I_0, I_1, \dots$ .  
Wegen  $\log(\lambda_6(mn)) \leq \log(m^2n^2) = 2 \log(mn)$ :  
 $mn \lambda_6(mn) \log(mn \lambda_6(mn)) \in O(mn \lambda_6(mn) \log(mn))$
  - Beginne mit dem linken Intervall  $I_0 \in L$ .  
Für jedes  $O_1$  teste, ob über  $I_0$  ein Zyklus  $(O_1, O_2, O_3)$  existiert. Wenn ja, berechne die kritischen Platzierungen wie folgt:
    - Berechne die Winkel  $\theta_{11}, \theta_{12}, \dots$  der maximal vier zugehörigen kritischen Platzierungen, an denen der Zyklus beteiligt ist.
    - Berichte sofort für die in  $I_0$  enthaltenen Orientierungen  $\theta_i$  die Platzierungen  $(\theta_i, (O_1, O_2, O_3))$ .
    - Füge die übrigen Platzierungen  $(\theta_j, (O_1, O_2, O_3))$  in eine nach Winkeln sortierte Warteschlange  $W$  ein.
  - Bearbeite von links nach rechts die übrigen Intervalle  $I_k \in L$ .
    - Beim Übergang von  $I_{k-1}$  nach  $I_k$  kann nur in einem Arrangement  $\mathcal{A}(O_1)$  ein Wechsel der unteren Kontur stattfinden.  $f_{O_1O_2}$  werde von  $f_{O_1O_4}$  abgelöst.
    - Falls  $O_1$  über  $I_{k-1}$  einen Zyklus mit  $O_2$  hatte, entferne aus  $W$  die zu  $(O_1, O_2, O_3)$  gehörenden kritischen Platzierungen; diese sind nicht mehr aktuell.
    - Teste, ob  $O_1, O_4$  einen Zyklus  $(O_1, O_4, O_5)$  über  $I_k$  bilden. Wenn ja, berechne wie oben die kritischen Platzierungen.
    - Berichte alle in  $W$  gespeicherten Platzierungen  $(\theta, (O_i, O_k, O_\ell))$  mit  $\theta \in I_k$ , und entferne sie aus  $W$ .

$$O(mn \lambda_6(mn) \log(mn))$$


---

### 2.3.2 Knotengraph und Kantengraph

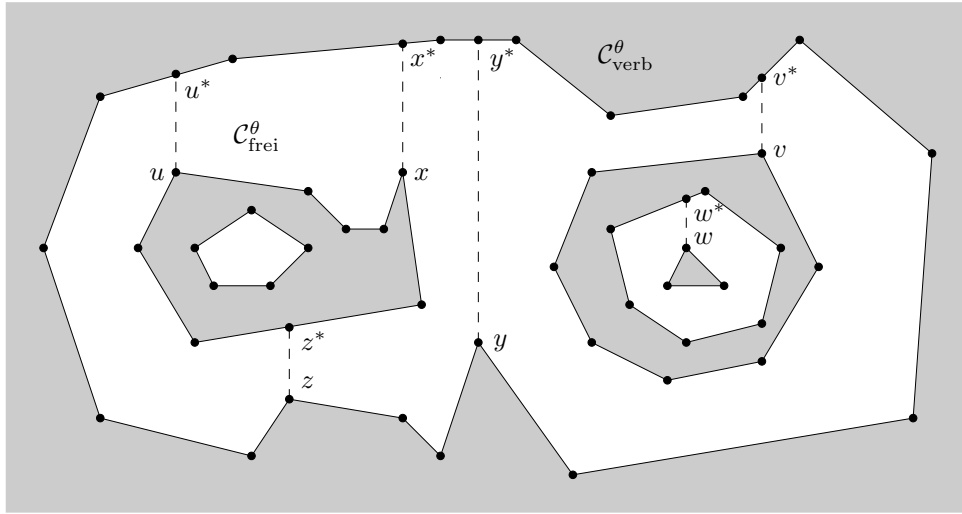


Abbildung 2.38:  $\mathcal{C}_{\text{frei}}^\theta$ ,  $\mathcal{C}_{\text{verb}}^\theta$  und Hilfsknoten von  $V^\theta$ .

Sei zunächst die Orientierung  $\theta$  fest. Wir betrachten den Schnitt  $\mathcal{C}_{\text{frei}}^\theta$  durch den Konfigurationsraum  $\mathcal{C}_{\text{frei}}$ . In  $\mathcal{C}_{\text{frei}}^\theta$  brauchen wir nur Translationen zu betrachten, und wir wissen aus den vorherigen Abschnitten, daß für einen konvexen Roboter  $\mathcal{C}_{\text{frei}}^\theta$  die Komplexität  $O(mn)$  hat und sich in Zeit  $O(mn \log^2(mn))$  berechnen läßt. Dabei hatten wir die Hindernisse in konvexe Teile, z. B. Dreiecke zerlegt. Zur diskreten Darstellung des Konfigurationsraumes hatten wir die Trapezzerlegung benutzt; diese Methode läßt sich jedoch nicht gut auf drei Dimensionen erweitern. Deshalb benutzen wir hier zur Darstellung von  $\mathcal{C}_{\text{frei}}^\theta$  den Knotengraph  $V^\theta$ .

Wir gehen im folgenden davon aus, daß alle Zusammenhangskomponenten von  $\mathcal{C}_{\text{frei}}$  beschränkt sind, sich also Wände um die Szene befinden; siehe Abbildung 2.38. Der Rand von  $\mathcal{C}_{\text{frei}}^\theta$  besteht dann aus endlich vielen Polygonen.

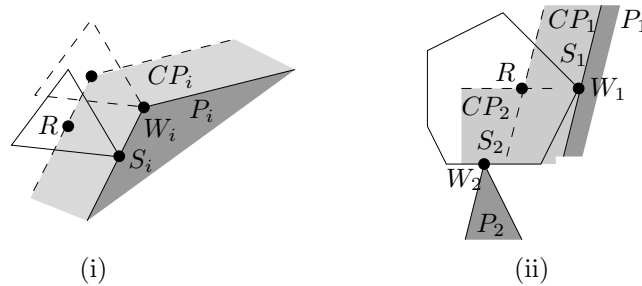


Abbildung 2.39: Konvexe und konkave Ecken in  $\partial\mathcal{C}_{\text{verb}}^\theta$ .

In  $\mathcal{C}_{\text{verb}}^\theta$  gibt es zwei Typen von Ecken: Konvexe Ecken entstehen durch Ecke/Ecke-Kontakte  $O_i = (W_i, S_i)$  von  $R$  mit einer konvexen Hindernisecke (Kontaktpaar vom Typ III), konkave Ecken durch zwei Ecke/Kante-Kontaktpaare (Typ I oder II), siehe Abbildung 2.39.

**Definition 2.39** Zu einer festen Orientierung  $\theta$  enthält der **Knotengraph**  $V^\theta$ :

- Alle Knoten der Polygone in  $\mathcal{C}_{\text{verb}}^\theta$ .
- Für jeden lokal y-maximalen Knoten  $u$  auf  $\partial\mathcal{C}_{\text{verb}}^\theta$  den ersten Punkt  $u^*$  ("Hilfsknoten") oberhalb  $u$ .
- Kanten zwischen adjazenten Knoten (einschließlich Hilfsknoten) und zwischen y-maximalen Knoten und entsprechenden Hilfsknoten, vgl. Abbildung 2.38.

- Folgende Beschriftungen:
  - Konvexer Knoten von  $\mathcal{C}_{\text{verb}}^\theta$ : Ecke/Ecke-Paar  $O_i = (W_i, S_i)$ .
  - Konkaver Knoten von  $\mathcal{C}_{\text{verb}}^\theta$ : zwei Ecke/Kante-Paare  $O_i, O_j$ .
  - Hilfsknoten  $u^*$ : die Bezeichnung “Hilfsknoten” und die Beschriftung des erzeugenden Knoten  $u$ .
  - Kante von  $\mathcal{C}_{\text{verb}}^\theta$ : Ecke/Kante-Paar  $O_i = (W_i, S_i)$ . Vergleiche  $(S_1, W_1)$  bzw.  $(S_2, W_2)$  in Abbildung 2.39 (ii).

Bei festem  $\theta$  sind die Knoten durch ihre Beschriftungen eindeutig beschrieben. Wir verwalten zusätzlich eine Suchstruktur, die es erlaubt, anhand einer Beschriftung den zugehörigen Knoten aufzufinden.

#### Bemerkung 2.40

- (i)  $V^\theta$  ist ein planarer Graph mit  $O(mn)$  Knoten und Kanten.
- (ii) Jeder Knoten hat einen Grad  $\leq 3$ , falls sich die Ecken der Hindernisse in allgemeiner Lage befinden, d. h. keine Ecken mit gleichen  $X$ -Koordinaten.
- (iii)  $V^\theta$  läßt sich in Zeit  $O(mn \log(mn))$  konstruieren.
- (iv) Zu gegebener Beschriftung läßt sich der zugehörige Knoten in Zeit  $O(\log(mn))$  finden.

Den Zusammenhang zwischen dem Knotengraph und der Bewegungsplanung stellt folgendes Lemma her:

**Lemma 2.41** Seien  $u, v$  zwei Knoten von  $V^\theta$ . Dann gilt:

- $u, v$  liegen in derselben Zusammenhangskomponente von  $V^\theta$
- $\iff u, v$  liegen auf dem Rand derselben Zusammenhangskomponente von  $\mathcal{C}_{\text{frei}}^\theta$ .

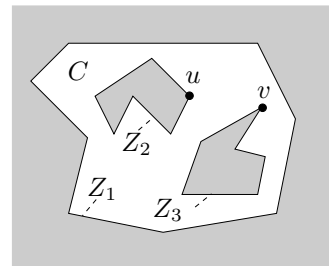
#### Beweis.

“ $\implies$ ” Falls  $(u, v)$  eine Kante von  $V^\theta$  bildet, liegen nach Definition 2.39  $u$  und  $v$  auf dem Rand derselben Zusammenhangskomponente von  $\mathcal{C}_{\text{frei}}^\theta$ ; auch falls  $v = u^*$ .

Falls  $u, v$  in  $V^\theta$  über mehrere Kanten verbunden sind, folgt die Behauptung durch Induktion.

“ $\impliedby$ ” Angenommen,  $u$  und  $v$  liegen auf dem Rand derselben Zusammenhangskomponente  $C$  von  $\mathcal{C}_{\text{frei}}^\theta$ .

Dieser Rand kann selbst in mehrere Komponenten  $Z_1, \dots, Z_s$  zerfallen, siehe Abbildung. Liegen  $u$  und  $v$  in derselben Komponente, so sind sie nach Definition 2.39 in  $V^\theta$  durch einen Kantenzug verbunden. Ansonsten nehmen wir an,  $u \in Z_2$ ,  $v \in Z_3$  und  $Z_1$  sei die äußere Komponente von  $\partial C$ . Seien  $V_u^\theta, V_v^\theta$  die Zusammenhangskomponenten von  $u$  und  $v$  in  $V^\theta$ .



**Behauptung:**  $V_u^\theta, V_v^\theta$  enthalten Knoten von  $Z_1$ .

**Beweis:**

- Angenommen,  $V_u^\theta$  enthält keinen Knoten von  $Z_1$ . Sei  $z$  der höchstgelegene Knoten in  $V_u^\theta$ .
- $\Rightarrow$   $z$  ist lokal  $y$ -maximal.
  - $\Rightarrow$  Der Strahl von  $z$  aufwärts muß durch  $\mathcal{C}_{\text{frei}}^\theta$  verlaufen, da  $z$  nicht auf der äußeren Komponente von  $\partial C$  liegt.
  - $\Rightarrow$  Es gibt einen Hilfsknoten  $z^*$ .
  - $\Rightarrow$   $z^* \in V_u^\theta$   $\nmid$  Maximalität von  $z$ .  $\diamond$

Also enthalten beide Komponenten Knoten von  $Z_1 \Rightarrow V_u^\theta = V_v^\theta$ .  $\square$

Die Frage ist nun, was mit  $V^\theta$  passiert, wenn sich  $\theta$  ändert. Lassen wir  $\theta$  stetig wachsen, so verändern sich auch die Knoten und Kanten von  $V^\theta$  stetig, bis ein bestimmtes  $\theta_k$  erreicht wird, an dem sich auch die Struktur von  $V^\theta$  ändert. Eine Orientierung, an der in  $V^\theta$  eine Strukturänderung auftritt, nennen wir eine **kritische Orientierung**.

Die Abbildungen 2.40 bis 2.43 zeigen kritische Orientierungen. Die dunkel schraffierten Teile sind Hindernisse, die hell schraffierten Teile Konfigurationsraum-Hindernisse für die Translation von  $R$ . Die gestrichelten Segmente sind die Kanten der Konfigurationsraum-Hindernisse. Dargestellt ist, wie sich  $\mathcal{C}_{\text{frei}}$  ändert, wenn  $R$  gedreht wird. Dabei können alle Ereignisse auch “rückwärts” ablaufen, z. B. kann in (i) auch eine neue Kante von  $\partial \mathcal{C}_{\text{frei}}^\theta$  entstehen. Die kritischen Orientierungen (i)–(iv) entstehen aus den Orientierungen der kritischen Platzierungen, (v) und (vi) müssen wegen der Aktualisierung von Hilfsknoten beachtet werden.

- (i) Drei Ecke/Kante-Kontaktpaare: eine Kante von  $\partial \mathcal{C}_{\text{frei}}^\theta$  verschwindet, ein “wanderndes” Kontaktsegment verändert  $\mathcal{C}_{\text{frei}}^\theta$ .
- (ii) Drei Ecke/Kante-Kontaktpaare: eine Komponente von  $\mathcal{C}_{\text{frei}}^\theta$  verschwindet; ein “wanderndes” Kontaktsegment hat  $\mathcal{C}_{\text{frei}}^\theta$  verkleinert.
- (iii) Ein Ecke/Ecke-Kontaktpaar, ein Ecke/Kante-Kontaktpaar: ein lokaler Zusammenhang von  $\mathcal{C}_{\text{frei}}^\theta$  verschwindet, eine “wandernde” konvexe Ecke hat den Durchgang abgeschnitten.

Die Anzahl der kritischen Orientierungen von Typ (i)–(iii) ist kleiner gleich der Anzahl der kritischen Platzierungen, also in  $O(mn \lambda_6(mn))$ . Die Aktualisierung von  $V^\theta$  ist pro Ereignis in  $O(\log(mn))$  Suchzeit +  $O(1)$  Änderungsaufwand möglich. Die folgenden Fälle verursachen mehr Aufwand.

- (iv) Ein Ecke/Ecke-Kontaktpaar und ein Ecke/Kante-Kontaktpaar (inzident): eine Roboter-kante wird parallel zu einer Hinderniskante und zwei benachbarte Kanten werden kollinear, siehe Abbildung 2.41.

In diesem Fall ändern sich die Beschriftungen der Kanten des erweiterten Hindernisses wie folgt:

$$\begin{aligned} (W, S_R) &\mapsto (W_L, S) \\ (W_R, S) &\mapsto (W, S_L) \end{aligned}$$

- Jede Hinderniskante kann  $O(mn)$  Kanten zu  $V^\theta$  beitragen
- $\Rightarrow$  Such- und Aktualisierungsaufwand in  $V^\theta$ :  $O(mn \log(mn))$ .
- $\Rightarrow$  Dieses Ereignis kann maximal  $O(mn)$  mal auftreten, der Gesamtaufwand liegt also in  $O((mn)^2 \log(mn))$ .

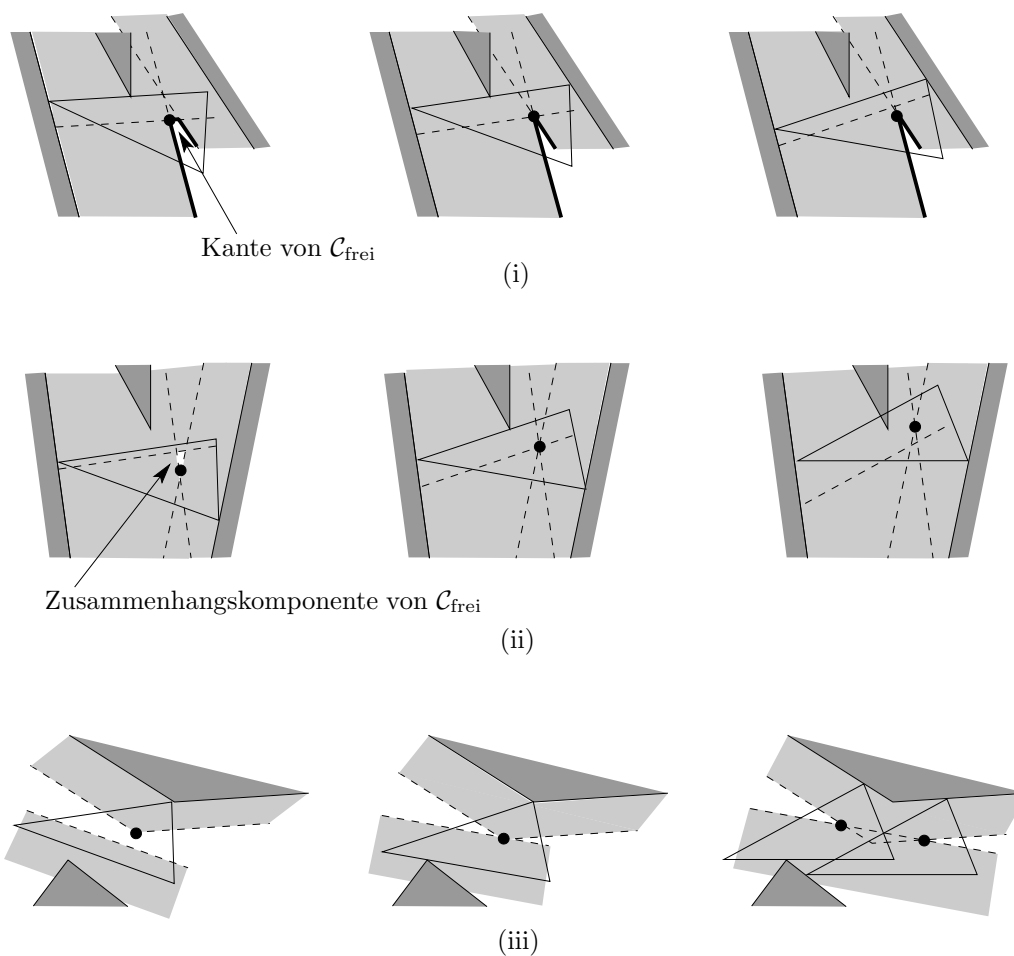


Abbildung 2.40: (i) eine Kante verschwindet, (ii) eine Komponente von  $\mathcal{C}_{\text{frei}}$  verschwindet, (iii) ein lokaler Zusammenhang verschwindet.

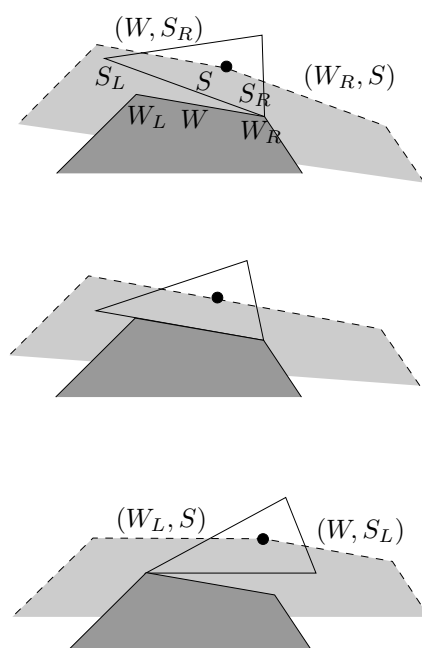


Abbildung 2.41: Zwei benachbarte Kanten werden kollinear.

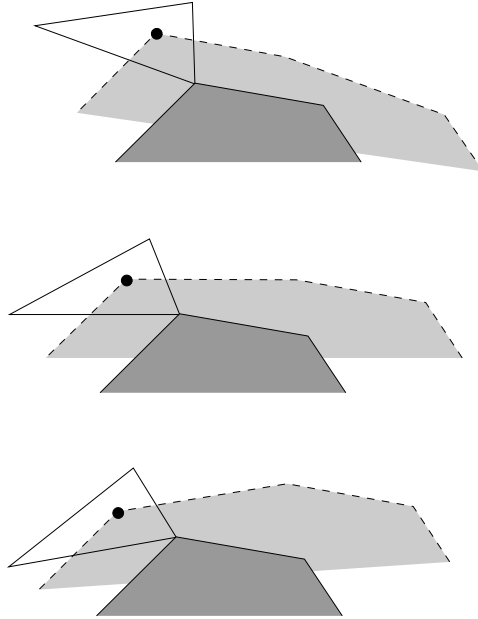


Abbildung 2.42: Eine Kante wird waagrecht, das lokale Y-Maximum ändert sich.

(v) Ein Ecke/Kante-Kontaktpaar: eine Kante von  $R$  und damit eine Kante eines Konfigurationsraumhindernisses wird waagrecht. Dabei kann sich die Menge der konvexen,  $y$ -maximalen Knoten ändern.

- Es können sich maximal  $n$  Knoten ändern.
- Das Ereignis kann  $m$  mal auftreten (für jede Kante von  $R$ ).
- Für jeden Knoten  $u$ , der nicht länger  $y$ -maximal ist, ist der Hilfsknoten  $u^*$  aus  $V^\theta$  zu entfernen sowie die neuen  $y$ -maximalen Knoten  $u'$  und ihre Hilfsknoten  $u'^*$  zu bestimmen und in  $V^\theta$  einzufügen (Sweep über  $\mathcal{C}_{\text{frei}}^\theta$  in Zeit  $O(mn \log(mn))$ ).

Der Zeitbedarf ist also insgesamt  $O(m^2 n \log(mn))$ .

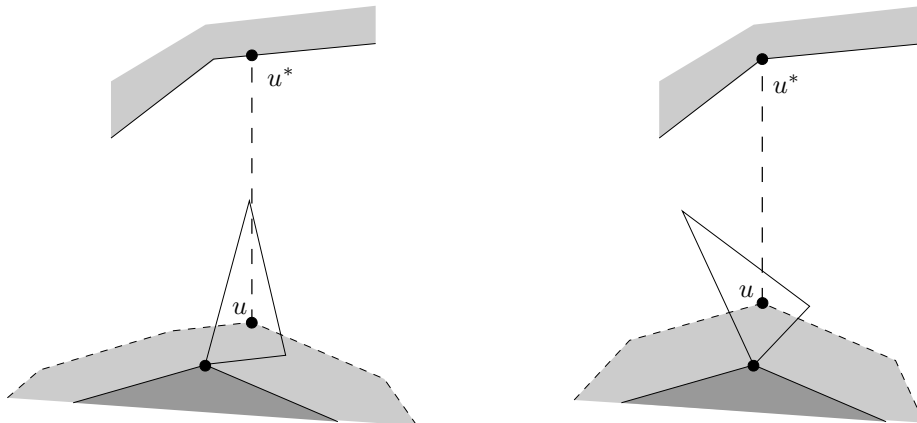


Abbildung 2.43: Ein Hilfsknoten wandert auf eine Ecke.

(vi) Ein Kontaktpaar Ecke/Ecke: eine konvexe,  $y$ -maximale Ecke liegt direkt unterhalb einer Ecke. Der Hilfsknoten wandert auf die Nachbarkante. Die Aktualisierung des Hilfsknotens ist nach  $O(\log(mn))$  Suchzeit in Zeit  $O(1)$  möglich. Dieses Ereignis tritt maximal  $O(mn \lambda_6(mn))$  mal auf; dies läßt sich mit einem ähnlichen Beweis wie für die Anzahl der kritischen Platzierungen zeigen.



Damit wissen wir:

**Bemerkung 2.42** Der Gesamtaufwand für die Aktualisierung von  $V^\theta$  beträgt  $O(mn \lambda_6(mn) \log(mn))$ .

Wir haben jetzt einen Graph  $V^\theta$  definiert, der für die Berechnung von Translationsbewegungen benutzt werden kann, kritische Orientierungen  $\theta$  untersucht, bei denen sich die Struktur von  $V^\theta$  ändert, und betrachtet, wie  $V^\theta$  aktualisiert werden muß. Jetzt wollen wir einen weiteren Graphen definieren, den Kantengraph  $E$ , der die gesamten Informationen über alle  $V^\theta$ ,  $\theta \in [0, 2\pi)$ , enthält und zur Bewegungsplanung verwendet werden kann.

Die Idee dabei ist diese: für wachsende Orientierung  $\theta$  beschreibt ein Knoten von  $\mathcal{C}_{\text{frei}}^\theta$  eine — möglicherweise gekrümmte — Kurve im Raum. Stelle nun in  $E$  diese Kurve als erweiterten Knoten mit seiner “Lebensdauer” als  $\theta$ -Intervall dar. Genauer:

**Definition 2.43** Sei  $\theta$  eine nicht-kritische Orientierung; sei  $u$  ein Knoten in  $V^\theta$  (bzw. seine Beschriftung, also die Namen der verursachenden Kontaktpaare).

- (i) Die **Lebensdauer** von  $u$  sei das maximale Intervall  $L(u) = (\theta_1, \theta_2)$ , in dem der Knoten  $u$  in jedem Knotengraph  $V^\theta$ ,  $\theta \in L(u)$ , vorkommt und nicht mit einem anderen Knoten  $u'$  zusammenfällt.
- (ii) Der **Kantengraph**  $E$  bestehe aus den erweiterten Knoten  $(u, L(u))$ . Zwei Knoten  $(u, L(u))$  und  $(v, L(v))$  sind in  $E$  genau dann durch eine Kante verbunden, wenn es ein  $\theta \in L(u) \cap L(v)$  gibt, so daß in  $V^\theta$  die Kante  $(u, v)$  existiert. Auch für eine Kante  $e$  kann eine Lebensdauer  $L(e)$  definiert werden.

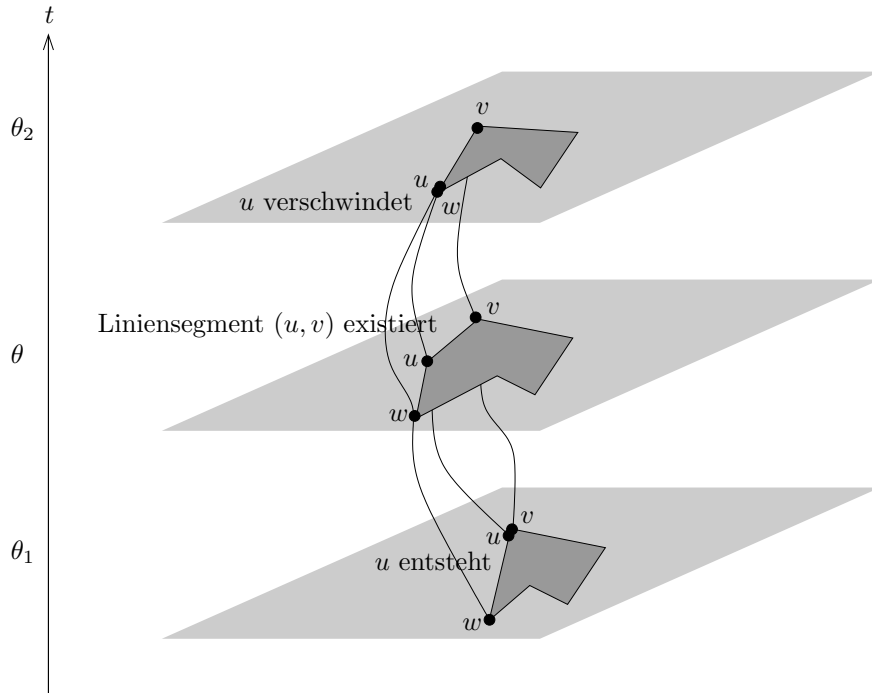


Abbildung 2.44: Kantengraph  $E$ .

Anschaulich entspricht jeder Kante von  $\mathcal{C}_{\text{frei}}$  ein erweiterter Knoten  $(u, L(u))$  in  $E$  und jedem Stück einer Fläche von  $\mathcal{C}_{\text{frei}}$  (zwischen den  $\theta$ -Werten von zwei Knoten) eine Kante  $[(u, L(u)), (v, L(v))]$  von  $E$  in folgendem Sinn: Seien  $e_u, e_v$  die Kanten von  $\mathcal{C}_{\text{frei}}$ , die den Knoten  $(u, L(u))$  und  $(v, L(v))$  entsprechen, dann verbindet für jedes  $\theta \in L((u, v))$  die Kante  $(u, v)$  in  $\mathcal{C}_{\text{frei}}^\theta$  zwei Punkte von  $e_u$  und  $e_v$ . Wir haben also eine komplexe, räumliche Struktur auf eine einfachere, kombinatorische Struktur abgebildet.

## Kapitel 3

# Bewegungsplanung für allgemeine Systeme

In den vorherigen Kapiteln haben wir betrachtet, wie wir Bahnen für einen Roboter planen können, der durch ein Polygon beschrieben werden kann. Als Spezialfälle haben wir die Bahnplanung für Punkte, Liniensegmente und Kreise untersucht. Nun stellen wir uns der Frage, ob und wie das Bahnplanungsproblem für allgemeine Systeme gelöst werden kann, also für Roboter, die sich nicht mehr durch Polygone beschreiben lassen, wie z. B. Systeme mit mehreren Armen, die mehrere Teleskop- und Drehgelenke enthalten können.

### 3.1 Eine untere Schranke für das allgemeine Bahnplanungsproblem

Folgendes Theorem zeigt, daß das allgemeine Bahnplanungsproblem bereits mit einer leichten Einschränkung sehr schwierig ist:

**Theorem 3.1** *Ein Roboter  $R$  sei gegeben durch eine Menge von  $m$  unterschiedlich großen, achsenparallelen Rechtecken  $R_i$ , die nicht miteinander verbunden sind. Seine Umgebung sei ein einfaches Polygon  $P$  mit achsenparallelen Kanten. Start- und Zielposition  $s$  und  $t$  seien halbfreie Positionen der  $R_i$  in  $P$ . Zu entscheiden, ob es in diesem Modell eine kollisionsfreie Bewegung von  $s$  nach  $t$  gibt, ist NP-hart.*

**Beweis.** Wir zeigen, daß sich das als NP-vollständig bekannte Problem *Partition* auf die Bahnplanung reduzieren läßt [GJ79]. Bei dem Problem *Partition* ist eine Menge  $X = \{a_1, \dots, a_m\} \subset \mathbb{N}$  gegeben. Zu entscheiden ist, ob es eine Teilmenge  $Y \subset X$  gibt, so daß gilt:

$$\sum_{a_i \in Y} a_i = \sum_{a_j \in X \setminus Y} a_j.$$

Sei  $A := \sum_{i=1}^m a_i$  und  $a_{\max} := \max_{1 \leq i \leq m} a_i$ . Konstruiere nun in polynomieller Zeit zur gegebenen Menge  $X = \{a_1, \dots, a_m\}$  eine Szene für das Bahnplanungsproblem wie folgt (vgl. Abbildung 3.1):

- Die Umgebung besteht aus einer “Halle” der Breite  $m$  mit zwei Seitenarmen der Breite  $\frac{A}{2}$  und der Höhe 1, die sich in einer Höhe von  $2a_{\max}$  über dem unteren Rand der Halle befinden.
- Für jedes  $a_i$  konstruiere ein Rechteck  $R_i$  des Roboters mit der Höhe  $a_i$  und Breite 1, und positioniere  $R_i$  am unteren Rand der Halle.
- Der Roboter  $R$  bestehe aus allen Rechtecken  $R_i$  und einem weiteren Rechteck  $B$  der Breite  $m$  und Höhe  $> 1$ , das sich am oberen Rand der Halle befindet.

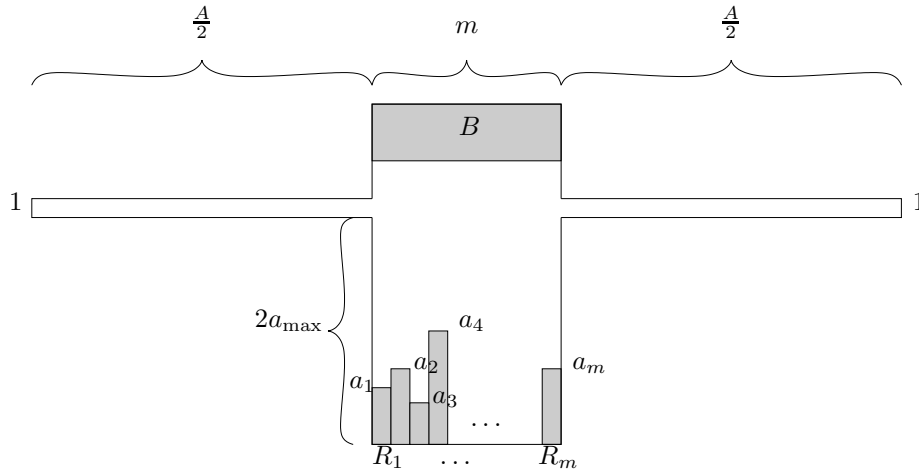


Abbildung 3.1: Umgebung zur Reduktion von Partition auf das allgemeine Bahnplanungsproblem.

- In der Zielkonfiguration  $t$  sollen sich die  $R_i$  am oberen Rand befinden,  $B$  am unteren Rand.

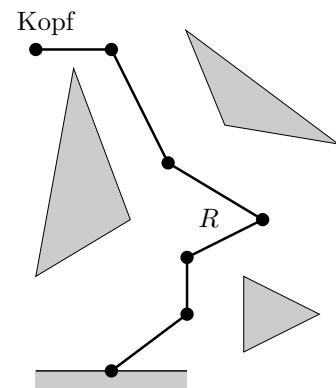
Eine Bewegung von  $s$  nach  $t$  ist genau dann möglich, wenn alle  $R_i$  in die Seitenarme bewegt werden können (genau dann läßt sich  $B$  vom oberen Rand zum unteren Rand bewegen). Dies wiederum ist genau dann möglich, wenn eine Partition der Menge  $X$  existiert.  $\square$

**Bemerkung 3.2** *Man kann sogar zeigen, daß dieses Problem PSPACE-hart<sup>1</sup> ist (Hopcroft, Schwartz, Sharir, [HSS84]).*

<sup>1</sup>Zur Definition von PSPACE-hart siehe Definition A.3 auf Seite 142.

Auch die folgenden Probleme sind NP-hart:

- Gegeben sei ein Roboterarm mit  $m > 3$  Drehgelenken in einer Umgebung mit polygonalen Hindernissen. Gibt es eine kollisionsfreie Bewegung des Roboterkopfes?
- Die Bewegung eines Würfels zwischen polyhedralen Hindernissen. (Wir haben gesehen, daß für ein Rechteck in der Ebene das Bahnplanungsproblem effizient lösbar ist, aber bereits eine Dimension höher wird das Problem NP-hart).



Dies zeigt, daß wir im allgemeinen mit exponentiellen Laufzeiten rechnen müssen.

### 3.2 Beschreibung des Konfigurationsraumes als semialgebraische Menge

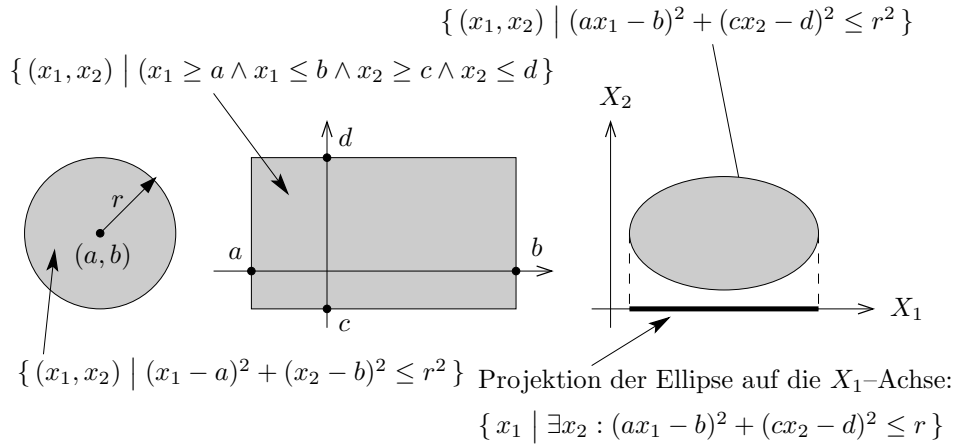


Abbildung 3.2: Beispiele für semialgebraische Mengen.

Zur Lösung des Bahnplanungsproblems suchen wir eine Möglichkeit, den Konfigurationsraum mit einfachen und handhabbaren Mitteln zu beschreiben. Solch eine Beschreibungsmöglichkeit sind semialgebraische Mengen. Einfach beschreibbare Objekte zeigt Abbildung 3.2, zur genauen Definition von semialgebraischen Mengen benötigen wir die folgende Klasse von Ausdrücken:

**Definition 3.3** (Rekursiv) Ein **Tarski-Ausdruck**  $T$  ist:

- Jede polynomiale Ungleichung

$$P(X_1, X_2, \dots, X_n) \leq 0$$

mit einem Polynom  $P(X_1, X_2, \dots, X_n) \in \mathbb{Q}[X_1, X_2, \dots, X_n]$ , d. h.

$$P(X_1, X_2, \dots, X_n) = \sum_{(i_1, \dots, i_n)=0}^{(d_1, \dots, d_n)} a_{i_1, \dots, i_n} X_1^{i_1} X_2^{i_2} \dots X_n^{i_n}, \quad a_{(\dots)} \in \mathbb{Q}.$$

- Eine logische Verknüpfung von zwei Tarski-Ausdrücken  $T_1, T_2$ :

$$\neg T_1, \quad T_1 \vee T_2, \quad T_1 \wedge T_2, \quad T_1 \rightarrow T_2.$$

- Mit einem Tarski-Ausdruck  $T(X)$ , in dem die Variable  $X$  frei vorkommt,<sup>2</sup> auch die Ausdrücke:

$$\exists X : T(X) \quad \text{und} \quad \forall X : T(X).$$

Damit können wir semialgebraische Mengen wie folgt definieren:

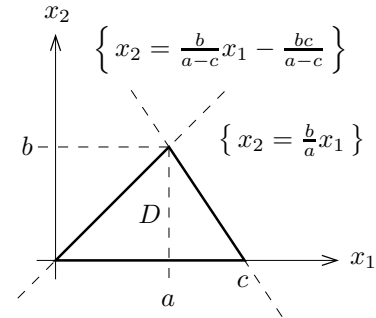
**Definition 3.4** Eine Teilmenge  $S \subseteq \mathbb{R}^n$  heißt **semialgebraisch**, falls es einen Tarski-Ausdruck  $T(X_1, \dots, X_n)$  gibt, in dem die Variablen  $X_1, \dots, X_n$  frei vorkommen, so daß gilt:

$$S = \{ (x_1, \dots, x_n) \in \mathbb{R}^n \mid T(x_1, \dots, x_n) \text{ ist erfüllt} \}.$$

<sup>2</sup>Variablen heißen *frei*, wenn sie nicht an einen Quantor ( $\forall, \exists$ ) gebunden sind, ansonsten heißen sie *gebunden*.

Die Menge der semialgebraischen Mengen ist abgeschlossen gegen  $\bigcap^{<\infty}$ ,  $\bigcup^{<\infty}$ , Komplement und Projektion. Bei algebraischen Mengen sind nur Ausdrücke mit “=”, keine Ungleichungen erlaubt. Die Nullstellen von Polynomen sind algebraische Mengen.

Im ersten Schritt beschreiben wir nun  $R$  und die  $P_i$  als semialgebraische Mengen. Da die Oder-Verknüpfung von Tarski-Ausdrücken wieder ein Tarski-Ausdruck ist, genügt es, ein Dreieck  $D$  mit Hilfe eines Tarski-Ausdrucks zu beschreiben. Dazu gibt es mehrere Möglichkeiten. Wir können z. B.  $D$  durch Summen von Vektoren beschreiben (mit Quantoren):



$$\begin{aligned} D &= \{ (x_1, x_2) \mid \exists \lambda \exists \mu (\lambda \geq 0 \wedge \mu \geq 0 \wedge \lambda + \mu \leq 1 \\ &\quad \wedge (x_1, x_2) = \lambda(c, 0) + \mu(a, b)) \} \\ &= \{ (x_1, x_2) \mid \underbrace{\exists \lambda \exists \mu (\lambda \geq 0 \wedge \mu \geq 0 \wedge \lambda + \mu \leq 1 \\ &\quad \wedge x_1 = \lambda c + \mu a \wedge x_2 = \mu b)}_{T_1(x_1, x_2)} \}. \end{aligned}$$

Eine Alternative ist die Beschreibung von  $D$  als Schnitt von Halbebenen; in diesem Fall kommen wir sogar ohne Quantoren aus:

$$D = \{ (x_1, x_2) \mid \underbrace{x_2 \geq 0 \wedge x_2 \leq \frac{b}{a}x_1 \wedge x_2 \leq \frac{b}{a-c}x_1 - \frac{bc}{a-c}}_{T_2(x_1, x_2)} \}.$$

Triangulieren wir die Hindernisse, so zerfällt ein Hindernis  $P_i$  in Dreiecke  $D_{ij}$  mit  $D_{ij} = \{ (x_1, x_2) \mid T_{ij}(x_1, x_2) \}$  mit adäquatem Tarski-Ausdruck  $T_{ij}(X_1, X_2)$ . Das Hindernis  $P_i$  läßt sich dann als Disjunktion dieser Tarski-Ausdrücke darstellen:

$$P_i(X_1, X_2) \equiv T_{i,1}(X_1, X_2) \vee T_{i,2}(X_1, X_2) \vee \dots \vee T_{i,k}(X_1, X_2)$$

Wir können also die Menge der Hindernisse durch einen Tarski-Ausdruck  $H(X_1, X_2)$  beschreiben. Ebenso kann der Roboter  $R$  in seiner Ausgangslage  $R(0, 0, 90^\circ)$  durch einen Tarski-Ausdruck  $R(X_1, X_2)$  beschrieben werden.

Wie beschreiben wir nun die Bewegungen des Roboters?

**Translation:** Durch Addition eines Translationsvektors  $\begin{pmatrix} a_1 \\ a_2 \end{pmatrix}$ .

**Rotation um  $\theta$ :** Eine Rotation läßt sich durch Anwendung einer Rotationsmatrix  $\begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix}$  beschreiben. Das Problem dabei ist, daß die Funktionen  $\sin$  und  $\cos$  nicht algebraisch sind. Da der Winkel  $\theta$  jedoch nicht explizit benötigt wird, reicht es aus, eine Rotationsmatrix  $\begin{pmatrix} S_1 & -S_2 \\ S_2 & S_1 \end{pmatrix}$  mit  $S_1^2 + S_2^2 = 1$  zu benutzen.

Damit können wir Bewegungen als Tarski-Ausdruck

$$B(X_1, X_2, X'_1, X'_2, A_1, A_2, S_1, S_2)$$

beschreiben, mit  $B(x_1, x_2, x'_1, x'_2, a_1, a_2, s_1, s_2) :\iff$  die Zielposition  $\begin{pmatrix} x'_1 \\ x'_2 \end{pmatrix}$  geht aus der Startposition  $\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$  durch Rotation des Roboters mit  $\begin{pmatrix} s_1 & -s_2 \\ s_2 & s_1 \end{pmatrix}$  und anschließender Translation um  $\begin{pmatrix} a_1 \\ a_2 \end{pmatrix}$  hervor, also wenn gilt:

$$\begin{pmatrix} x'_1 \\ x'_2 \end{pmatrix} = \begin{pmatrix} s_1 & -s_2 \\ s_2 & s_1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} a_1 \\ a_2 \end{pmatrix}.$$

Der Roboter darf nicht von einem Ausgangspunkt  $(x_1, x_2)$  durch eine Bewegung  $B(\dots)$  in einen Zielpunkt  $(x'_1, x'_2)$  überführt werden, der in einem Hindernis liegt. Der Raum der verbotenen Konfigurationen  $\mathcal{C}_{\text{verb}}$  läßt sich damit beschreiben als:

$$\begin{aligned} \mathcal{C}_{\text{verb}}(A_1, A_2, S_1, S_2) := \\ \exists X_1, X_2, X'_1, X'_2 \left( R(X_1, X_2) \wedge H(X'_1, X'_2) \right. \\ \left. \wedge B(X_1, X_2, X'_1, X'_2, A_1, A_2, S_1, S_2) \right). \end{aligned} \quad (*)$$

Durch Negation dieses Ausdrucks erhalten wir eine Beschreibung des Raumes der freien Konfigurationen als semialgebraische Menge:

$$\mathcal{C}_{\text{frei}}(A_1, A_2, S_1, S_2) := \neg \mathcal{C}_{\text{verb}}(A_1, A_2, S_1, S_2)$$

Dies zeigt, daß die Lösung des Bahnplanungsproblems mit den Gültigkeitsbereichen von Tarski-Ausdrücken zusammenhängt. Diese Technik läßt sich auf Roboter mit mehreren Freiheitsgraden verallgemeinern; dabei werden nur die benutzten Ausdrücke komplizierter [Lat91].

### 3.3 Collins' zylindrische algebraische Zerlegung

Wir haben bereits erkannt, daß wir keinen effizienten Algorithmus für das allgemeine Bewegungsplanungsproblem erwarten dürfen. Wir wissen auch — siehe Abschnitt A.2 —, daß es Probleme gibt, die so schwierig sind, daß sie überhaupt nicht algorithmisch lösbar sind. In diesem Abschnitt wollen wir uns mit der Entscheidbarkeit zweier Probleme befassen. Die dabei benutzte Beweistechnik können wir danach auf unser Bewegungsplanungsproblem anwenden.

#### Problem 1:

Die *Theorie der reellen Zahlen* ist die Menge aller Tarski-Ausdrücke ohne freie Variablen, die über ganz  $\mathbb{R}$  gelten:

$$\Omega = \{ T \mid T \text{ ist Tarski-Ausdruck ohne freie Variablen} \}$$

$$Th(\mathbb{R}) = \{ T \in \Omega \mid T \text{ gilt über } \mathbb{R} \}$$

Beispiele:

$$\begin{array}{ll} \forall x (x^2 \leq 0 \rightarrow x = 0) & \in Th(\mathbb{R}) \\ \forall x \exists y (x \geq 0 \rightarrow x = y^2) & \in Th(\mathbb{R}) \\ \exists x (x^2 = -1) & \notin Th(\mathbb{R}) \end{array}$$

#### Theorem 3.5 (Tarski, 1951)

Die Theorie der reellen Zahlen ist entscheidbar.

[Tar51]

#### Problem 2:

Wir haben Beispiele von semialgebraischen Mengen gesehen, die sich durch Tarski-Ausdrücke beschreiben lassen, in denen keine Quantoren auftreten. Das ist kein Zufall! Denn es gilt:

**Theorem 3.6** Jede semialgebraische Menge läßt sich durch einen quantorenfreien Tarski-Ausdruck definieren.

Beide Theoreme werden wir im folgenden beweisen, dazu benötigen wir allerdings noch etwas Handwerkszeug:

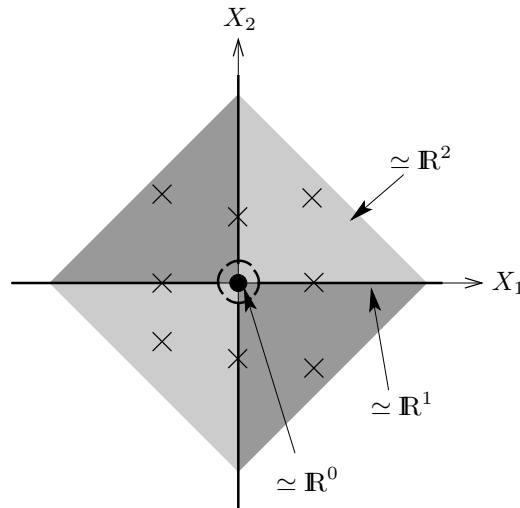


Abbildung 3.3: Beispiel für eine zylindrische algebraische Zerlegungen.

**Definition 3.7** Eine **Algebraische Zerlegung** des  $\mathbb{R}^n$  ist ein endliches System  $\mathcal{K}_n$  disjunkter Teilmengen (Zellen) des  $\mathbb{R}^n$  für die gilt:

- Jede Zelle  $C_j \in \mathcal{K}_n$  ist semialgebraisch.
- Jede Zelle  $C_j \in \mathcal{K}_n$  ist homöomorph zu einem  $\mathbb{R}^k$ ,  $k \leq n$ , d. h. jede Zelle ist zusammenhängend und hat keine Löcher. Für  $k = 0$  ist  $C_j$  ein algebraischer Punkt.<sup>3</sup>
- $\mathcal{K}_n$  ist eine disjunkte Zerlegung des  $\mathbb{R}^n$ :  $\mathbb{R}^n = \bigcup_{j=1}^m C_j$ .

Eine **Probefunktion**  $\sigma_n$  ist eine Funktion  $\sigma_n : \mathcal{K}_n \rightarrow \mathbb{R}^n$ , die jeder Zelle  $C_j$  einen Probepunkt  $\sigma_n(C_j) \in C_j$  zuordnet.

Die **Projektion**  $\pi_n$  sei die Abbildung

$$\begin{aligned} \pi_n : \mathbb{R}^n &\longrightarrow \mathbb{R}^{n-1} \\ (X_1, \dots, X_n) &\longmapsto (X_1, \dots, X_{n-1}) \end{aligned}$$

Die Projektion einer Menge  $\mathcal{M}$  ist die Projektion ihrer Elemente:  $\pi_n(\mathcal{M}) := \{y \mid y = \pi_n(x), x \in \mathcal{M}\}$

**Definition 3.8** Ein System  $(\mathcal{K}_n, \sigma_n)$  heißt **zylindrische algebraische Zerlegung** (cylindrical algebraic decomposition, CAD, hier: ZAZ) des  $\mathbb{R}^n$  falls gilt:

- $\mathcal{K}_n$  ist eine algebraische Zerlegung mit Probefunktion  $\sigma_n$ .
- Für  $n > 1$  läßt sich der  $\mathbb{R}^{n-1}$  rekursiv zerlegen, d. h. es existiert eine ZAZ  $(\mathcal{K}_{n-1}, \sigma_{n-1})$  des  $\mathbb{R}^{n-1}$  mit:
  - Die Probefunktion zu  $\mathcal{K}_{n-1}$  ist die Projektion der Probefkt. zu  $\mathcal{K}_n$ :

$$\begin{aligned} \sigma_{n-1} : \mathcal{K}_{n-1} &\longrightarrow \mathbb{R}^{n-1} \\ \pi_n(C) &\longmapsto \pi_n(\sigma_n(C)) \end{aligned}$$

<sup>3</sup>Ein Punkt, dessen Koordinaten sich durch algebraische Zahlen darstellen lassen.



- Zu jeder Zelle  $C_j \in \mathcal{K}_n$  existiert eine Zelle  $C'_j \in \mathcal{K}_{n-1}$  mit  $C'_j = \pi_n(C_j)$ , die Projektion einer Zelle in  $\mathcal{K}_n$  ist eine Zelle in  $\mathcal{K}_{n-1}$ .
- Für Zellen  $C_i, C_j \in \mathcal{K}_n$  mit  $\pi_n(C_i) = \pi_n(C_j)$  gilt  $\pi_n(\sigma_n(C_i)) = \pi_n(\sigma_n(C_j))$ .
- Für  $n = 1$  ist  $\mathcal{K}_1$  eine Zerlegung von  $\mathbb{R}$  in eine endliche Menge algebraischer Zahlen, den (endlichen und unendlichen) offenen Intervallen zwischen diesen Zahlen.
- Der **Zylinder** über  $C'$  sind alle Zellen  $C \in \mathcal{K}_n$  mit  $\pi_n(C) = C'$ .

Abbildung 3.3 zeigt ein Beispiel für eine zylindrische algebraische Zerlegungen. Die Zerlegung besteht aus vier Viertelebenen (homöomorph zum  $\mathbb{R}^2$ ), vier Halbgraden ( $\simeq \mathbb{R}^1$ ) und einem Punkt ( $\simeq \mathbb{R}^0$ ). Die Probepunkte sind mit  $\times$  gekennzeichnet.

**Definition 3.9** Sei  $\mathcal{P} \subseteq \mathbb{Q}[X_1, \dots, X_n]$  eine endliche Menge von Polynomen über  $\mathbb{Q}[X_1, \dots, X_n]$ . Eine zylindrische algebraische Zerlegung  $(\mathcal{K}_n, \sigma_n)$  heißt  **$\mathcal{P}$ -invariant**, wenn gilt:

$$\forall f(X_1, \dots, X_n) \in \mathcal{P} : \forall C \in \mathcal{K}_n : \text{sign}\left(f\Big|_C\right) = \text{const},$$

d.h. für alle Punkte  $(x_1, \dots, x_n)$  derselben Zelle  $C$  hat  $f(x_1, \dots, x_n)$  das gleiche Vorzeichen.

Für den Beweis von Theorem 3.5, Theorem 3.6 und für das Bewegungsplanungsproblem ist folgendes Theorem entscheidend. In der Bewegungsplanung entspricht die Dimension der Anzahl der Freiheitsgrade des Robotersystems.

**Theorem 3.10** (Collins, 1975)

Man kann eine  $\mathcal{P}$ -invariante zylindrische algebraische Zerlegung effektiv konstruieren. Die Laufzeit ist polynomiell in Anzahl und Grad der Polynome in  $\mathcal{P}$ , doppelt exponentiell in der Anzahl der Dimensionen. [Col75]

Die Idee ist, die Polynome in  $\mathcal{P}$  zu einem Polynom  $P$  zusammenzufassen und die Zerlegung schrittweise mit wachsender Dimension zu konstruieren. Für eine Dimension  $n$  fassen wir  $P$  als Polynom in einer Variablen auf und bestimmen dessen Nullstellen, die als Funktionen über  $n-1$  Variablen dargestellt werden. Diese Nullstellen dienen als Intervallgrenzen bei der Konstruktion des Zylinders. Genauer:

**Beweis.** Durch Induktion über  $n$ . Sei  $P(X_1, \dots, X_n) := \prod_{P_i \in \mathcal{P}} P_i$ .

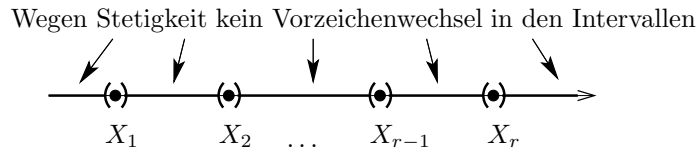


Abbildung 3.4: Zylindrische algebraische Zerlegung für  $n = 1$ .

$n = 1$ :  $P(X_1)$  habe den Grad  $m \Rightarrow P(X_1)$  hat  $r \leq m$  viele verschiedene reelle Nullstellen. Die ZAZ in Abbildung 3.4 ist nach dem Zwischenwertsatz  $\mathcal{P}$ -invariant.

$n > 1$ : Fasse  $P(X_1, \dots, X_n) := \prod_{P \in \mathcal{P}} P(X_1, \dots, X_n)$  als Polynom in  $X_n$  mit Koeffizienten in  $\mathbb{Q}[X_1, \dots, X_{n-1}]$  auf.

Sei  $(x_1, \dots, x_{n-1}) \in \mathbb{R}^{n-1}$  ein festes Tupel,  $\hat{P}(X_n) := P_{x_1, \dots, x_{n-1}}(X_n)$  und sei  $r := \text{grad}_{X_n}(\hat{P}(X_n))$  der Grad der Variablen  $X_n$ . Dann ist die Anzahl der verschiedenen reellen

und komplexen Nullstellen von  $\hat{P}(X_n)$  gleich

$$r - \text{grad}_{X_n} \left( \text{ggT} \left( \hat{P}(X_n), \underbrace{\hat{P}'(X_n)}_{\text{Ableitung nach } X_n} \right) \right).$$

Begründung:

$$\begin{aligned} & \alpha \text{ ist } k\text{-fache Nullstelle von } f(x) \\ \Rightarrow & f(x) = (x - \alpha)^k \cdot g(x) \text{ mit } g(\alpha) \neq 0 \\ \Rightarrow & f'(x) = k(x - \alpha)^{k-1} \cdot g(x) + (x - \alpha)^k \cdot g'(x) \text{ hat } \alpha \text{ als } (k-1)\text{-fache Nullstelle.} \end{aligned}$$

Ist also  $f(x) = \prod_{i=1}^j (x - \alpha_i)^{k_i}$ , so ist  $\text{ggT}(f(x), f'(x)) = \prod_{i=1}^j (x - \alpha_i)^{k_i-1}$ , und der Grad von  $f$  ist um  $j$  höher als der des  $\text{ggT}$ .

Man kann nun polynomielle Bedingungen dafür aufstellen, daß sich  $\text{grad}_{X_n}(\hat{P}(X_n))$  und  $\text{grad}_{X_n} \left( \text{ggT}(\hat{P}(X_n), \hat{P}'(X_n)) \right)$  nicht ändern.<sup>4</sup> Wendet man die Induktionsvoraussetzung auf diese polynomiellen Bedingungen an, dann existiert eine ZAZ  $(\mathcal{K}_{n-1}, \sigma_{n-1})$  des  $\mathbb{R}^{n-1}$ , bei der  $\text{grad}_{X_n}(\hat{P}(X_n))$  und  $\text{grad}_{X_n}(\text{ggT}(\hat{P}(X_n), \hat{P}'(X_n)))$  auf jeder Zelle  $C$  konstant sind.

$$\begin{aligned} \Rightarrow & \text{Die Anzahl der verschiedenen reellen Nullstellen von } \hat{P}(X_n) \text{ ist auf jeder Zelle konstant.} \\ \Rightarrow & \text{Die verschiedenen reellen Nullstellen lassen sich auf jeder Zelle } C \text{ durch stetige Funktionen } f_1(X_1, \dots, X_{n-1}) < f_2(X_1, \dots, X_{n-1}) < \dots < f_j(X_1, \dots, X_{n-1}) \text{ darstellen.} \end{aligned}$$

Man benutzt diese  $f_j$  auf folgende Weise, um den Zylinder  $C \times \mathbb{R}$  in Zellen zu zerlegen, auf denen  $P(X_1, \dots, X_n)$  ein konstantes Vorzeichen hat (siehe Abbildung 3.5):

$$C_{0,1} = \{ (x_1, \dots, x_{n-1}, y) \mid (x_1, \dots, x_{n-1}) \in C, y < f_1(x_1, \dots, x_{n-1}) \}$$

...

$$C_i = \{ (x_1, \dots, x_{n-1}, y) \mid (x_1, \dots, x_{n-1}) \in C, y = f_i(x_1, \dots, x_{n-1}) \}$$

$$C_{i,i+1} = \{ (x_1, \dots, x_{n-1}, y) \mid (x_1, \dots, x_{n-1}) \in C, f_i(x_1, \dots, x_{n-1}) < y < f_{i+1}(x_1, \dots, x_{n-1}) \}$$

...

$$C_{j,\infty} = \{ (x_1, \dots, x_{n-1}, y) \mid (x_1, \dots, x_{n-1}) \in C, f_j(x_1, \dots, x_{n-1}) < y \}$$

Die Probepunkte werden entsprechend definiert. So entsteht  $(\mathcal{K}_n, \sigma_n)$ .

□

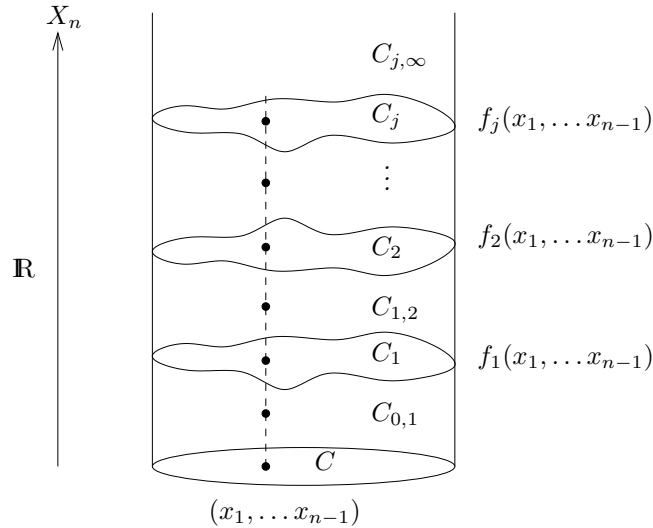
Damit kann man die Entscheidbarkeit der Theorie der reellen Zahlen (Theorem 3.5) beweisen. Wir beschränken uns hier auf ein signifikantes

**Beispiel:** Ist die Tarski-Formel  $\forall X \exists Y : T(X, Y)$  gültig?

Berechne eine  $T$ -invariante ZAZ  $(\mathcal{K}_2, \sigma_2)$  des  $\mathbb{R}^2$ , d. h. die Polynome in  $X$  und  $Y$ , die in  $T$  vorkommen, haben auf jeder Zelle ein konstantes Vorzeichen. Sei  $(\mathcal{K}_1, \sigma_1)$  die projizierte eindimensionale ZAZ, dann gilt:

$\forall X \exists Y : T(X, Y)$  ist gültig  $\iff$  für jede Zelle  $C' \in \mathcal{K}_1$  gibt es eine Zelle  $C \in \mathcal{K}_2$  im Zylinder über  $C'$  mit  $T(x, y)$  ist wahr für  $(x, y) = \sigma_2(C)$ .

<sup>4</sup>Dazu braucht man Kenntnisse aus der Resultantentheorie, die wir später erläutern.

Abbildung 3.5: Zerlegung des Zylinders  $C \times \mathbb{R}$ .

“ $\Rightarrow$ ” Sei  $C' \in \mathcal{K}_1$  beliebig. Betrachte  $x = \sigma_1(C') \in C'$ . Nach Voraussetzung gibt es ein  $\tilde{y}$  mit  $T(x, \tilde{y})$ . Sei  $C$  die Zelle über  $C'$  mit  $(x, \tilde{y}) \in C$ . Wegen der  $T$ -Invarianz gilt dann auch  $T(x, y)$  für  $(x, y) = \sigma_2(C)$ .

“ $\Leftarrow$ ” Sei  $\tilde{x}$  beliebig. Sei  $C'$  die Zelle in  $\mathcal{K}_1$  mit  $\tilde{x} \in C'$ . Nach Voraussetzung gibt es ein  $C \in \mathcal{K}_2$  über  $C'$  mit  $T(x, y)$  für  $(x, y) = \sigma_2(C)$ . Es existiert ein  $\tilde{y}$  mit  $(\tilde{x}, \tilde{y}) \in C$ , da  $C$  im Zylinder von  $C'$  liegt, wegen der  $T$ -Invarianz gilt dann auch  $T(\tilde{x}, \tilde{y})$ .

Dies ist ein endliches Problem und kann in endlich vielen Schritten getestet werden. Der Beweis für andere Quantoren und Formeln mit längerem Präfix ist analog.  $\square$

Auch die Eliminierung von Quantoren (Theorem 3.6) kann man nun beweisen:

**Beispiel:** Sei  $S = \{x \mid \exists Y : H(x, Y)\}$  eine semialgebraische Menge, wobei  $H$  ein Tarski-Ausdruck ist. Darin ist  $x$  eine freie,  $Y$  eine gebundene Variable.

Wie oben sei  $(\mathcal{K}_2, \sigma_2)$  eine ZAZ für die Polynome in  $H$ . Für eine Zelle  $C$  sei  $\alpha_C$  der quantorenfreie Tarski-Ausdruck, welcher  $C$  definiert (nach Collins existiert ein solcher).

Betrachte

$$\mathcal{C} := \{C' \in \mathcal{K}_1 \mid \exists C \in \mathcal{K}_2 \text{ über } C' \text{ mit } H(x, y) \text{ für } (x, y) = \sigma_2(C)\}$$

und setze  $\alpha(x) := \bigvee_{C' \in \mathcal{C}} \alpha_{C'}(x)$ . Beachte:  $\alpha(x)$  ist quantorenfrei!

**Behauptung:**  $S = \{x \mid \exists Y H(x, Y)\} \stackrel{!}{=} \{x \mid \alpha(x)\}$ .

“ $\subseteq$ ” Gelte  $H(\tilde{x}, \tilde{y})$ . Dann sei  $\tilde{x} \in \tilde{C}$ , und  $C$  sei die Zelle über  $\tilde{C}$  mit  $(\tilde{x}, \tilde{y}) \in C$ . Dann gilt auch  $H(x, y)$  für  $(x, y) = \sigma_2(C)$ . Also ist  $\tilde{C} = C' \in \mathcal{C}$  und wegen  $x, \tilde{x} \in C'$  folgt  $\alpha(\tilde{x})$ .

“ $\supseteq$ ” Gelte  $\alpha(\tilde{x})$ . Dann gilt auch  $\alpha_{C'}(\tilde{x})$  für ein  $C' \in \mathcal{C}$ , also auch  $\tilde{x} \in C'$ . Sei  $C$  die Zelle über  $C'$  mit  $H(x, y)$  für  $(x, y) = \sigma_2(C)$ . Dann existiert ein  $\tilde{y}$  mit  $(\tilde{x}, \tilde{y}) \in C$ , da  $C$  im Zylinder über  $C'$  liegt. Dann gilt auch  $H(\tilde{x}, \tilde{y})$ .  $\square$

Im Folgenden wollen wir zeigen, welche polynomiellen Bedingungen gestellt werden müssen um die zylindrische algebraische Zerlegung rekursiv durchzuführen. Eine Bedingung war unter anderem, dass für das Polynom  $\hat{P}(X_n)$  gilt, dass  $\text{grad}_{X_n}(\text{ggT}(\hat{P}(X_n), \hat{P}'(X_n)))$  konstant ist.

Allgemeiner bestimmen wir die Anzahl gemeinsamer Nullstellen zweier Polynome  $A(X)$  und  $B(X)$  mit Grad  $a$  respektive Grad  $b$ . Wir betrachten dazu für  $(0 \leq j \leq \min\{a, b\} - 1)$  Polynome  $U_j(X)$  mit Grad  $b - j - 1$ , und  $V_j(X)$  mit Grad  $a - j - 1$ . Diese Polynome sollen die Gleichung

$$A(X)U_j(X) - B(X)V_j(X) = 0$$

erfüllen. Das Polynom  $U_j$  hat  $b-j$  Koeffizienten  $u_i$  und das Polynom  $V_j$  hat  $a-j$  Koeffizienten  $v_i$ . Das Polynom auf der linken Seite obiger Gleichung hat somit Grad  $a + b - j - 1$ .

**Lemma 3.11** *Genau dann wenn  $A(X)U_j(X) - B(X)V_j(X) = 0$  eine nicht-triviale Lösung hat, haben die Polynome  $A(X)$  und  $B(X)$   $j + 1$  gemeinsame Nullstellen.*

**Beweis.** O.B.d.A. betrachten wir  $A, B, U$  und  $V$  als normierte Polynome. Also sei  $A(X) = (X - \alpha_1) \cdots (X - \alpha_a)$ ,  $B(X) = (X - \beta_1) \cdots (X - \beta_b)$ ,  $U_j(X) = (X - \mu_1) \cdots (X - \mu_{b-j-1})$  und  $V_j(X) = (X - \nu_1) \cdots (X - \nu_{a-j-1})$ . Sei  $j$  die kleinste Zahl, so dass  $A(X)U_j(X) - B(X)V_j(X) = 0$  eine nicht-triviale Lösung hat. Dann gilt  $\mu_r \neq \nu_s$ ,  $r \in [1, b-j-1]$ ,  $s \in [1, a-j-1]$ . Sonst liesse sich ein Faktor entfernen und die Gleichung gelte immer noch. Ein Widerspruch zum kleinsten  $j$ ! Es existiert eine eindeutige Faktorisierung von  $A(X)U_j(X)$  und  $B(X)V_j(X)$ . Jede Nullstelle von  $U_j$  muss nun eine Nullstelle von  $B$  und jede Nullstelle von  $V_j$  eine Nullstelle von  $A$  sein. Dann haben  $A$  und  $B$   $j + 1$  gemeinsame Nullstellen!

Umgekehrt, falls  $A(X)$  und  $B(X)$   $j + 1$  gemeinsame Nullstellen haben, dann können wir die restlichen Nullstellen von  $A$  auf  $V_j$  und die restlichen Nullstellen von  $B$  auf  $U_j$  verteilen und  $A(X)U_j(X) - B(X)V_j(X) = 0$  hat eine nicht-triviale Lösung!  $\square$

Mit diesen Aussagen können wir eine Bedingung für die Anzahl der gemeinsamen Nullstellen aufstellen. Das lineare Gleichungssystem der Koeffizienten von  $A(X)U_j(X) - B(X)V_j(X) = 0$  besteht aus  $a + b - j$  Gleichungen aber nur  $a + b - 2j$  Unbekannten (Koeffizienten aus  $U_j(X)$  und  $V_j(X)$ ). Wir betrachten lediglich die ersten  $a + b - 2j$  Koeffizienten des Polynoms  $A(X)U_j(X) - B(X)V_j(X)$ . Dann ergibt sich ein lineares Gleichungssystem mit  $a + b - 2j$  Unbekannten und  $a + b - 2j$  Gleichungen.

Sei  $\psi_j(A, B)$  die Determinante der Matrix dieses Gleichungssystems. Wir werden zeigen, dass es ausreicht, das kleinste  $j$  zu bestimmen, so dass  $\psi(A, B) \neq 0$  ist.

**Theorem 3.12** *Die Anzahl der gemeinsamen Nullstellen von  $A$  und  $B$  ist die kleinste Zahl  $j$ , so dass  $\psi_j(A, B) \neq 0$  gilt.*

**Beweis.** Sei  $j = 0$  und  $\psi_j(A, B)$  die Determinante der ersten  $a + b - 2j$  Gleichungen für die Koeffizienten mit höchstem Grad in  $X$  aus  $A(X)U_j(X) - B(X)V_j(X) = 0$ . Also haben wir hier ein lineares Gleichungssystem mit  $a + b$  Gleichungen und  $a + b$  Unbekannten. Falls  $\psi_j(A, B) = 0$  ist, dann hat  $A(X)U_j(X) - B(X)V_j(X) = 0$  gemäß Lemma 3.11 eine nicht-triviale Lösung und somit eine gemeinsame Nullstelle. Ansonsten gibt es keine gemeinsame Nullstelle.

Nehmen wir nun an es gäbe nicht-triviale Lösungen für  $A(X)U_i(X) - B(X)V_i(X) = 0$  für  $i = 0, \dots, j - 1$ , so dass  $A$  und  $B$  mindestens  $j$  gemeinsame Nullstellen haben. Wir zeigen: Das Vorzeichen von  $\psi_j(A, B)$  entscheidet, ob  $j + 1$  oder nur  $j$  gemeinsame Nullstellen existieren.

Betrachte dazu  $A(X)U_j(X) - B(X)V_j(X) = C_j(X)$ , wobei  $C_j$  ein Polynom vom Grad  $\leq j - 1$  sei. Sei  $\Sigma$  das lineare Gleichungssystem mit Koeffizienten aus  $U_j, V_j$  und  $C_j$ . Zu den  $a + b - 2j$  Unbekannten kommen also  $j$  Unbekannte dazu und  $\Sigma$  ist ein quadratisches System. Das Subsystem  $\Sigma'$  mit den ersten  $a + b - 2j$  Unbekannten und  $a + b - 2j$  Gleichungen für  $X^i$  für  $i \in [j, a + b - j - 1]$  benutzt keine Koeffizienten aus  $C_j$ .

Es ist leicht zu sehen, dass  $\Sigma'$  genau dann nicht-triviale Lösung ( $\psi_j(A, B) = 0$ ) hat, wenn  $\Sigma$  eine nicht-triviale Lösung besitzt. Die Matrix von  $\Sigma'$  ist eine echte Submatrix von  $\Sigma$ , die Determinante von  $\Sigma'$  ist genau dann gleich null, wenn die Determinante von  $\Sigma$  null ist.

Falls nun  $\psi_j(A, B) \neq 0$  gilt, dann hat  $A(X)U_j(X) - B(X)V_j(X) = C_j(X)$  nur die triviale Lösung. Insbesondere gilt das dann auch für  $A(X)U_j(X) - B(X)V_j(X) = 0$ , nach Lemma 3.11 haben  $A$  und  $B$  genau  $j$  gemeinsame Nullstellen.

Falls aber  $\psi_j(A, B) = 0$  gilt, dann existieren  $U_j, V_j$  und  $C_j$  die die Gleichung  $A(X)U_j(X) - B(X)V_j(X) = C_j(X)$  erfüllen. Nach Annahme haben  $A$  und  $B$  mindestens  $j$  gemeinsame Nullstellen. Dann teilt das Produkt der Nullstellen das Polygon  $C_j(X)$ . Da der Grad von  $C_j$  aber  $j - 1$  ist, kann nur  $C_j \equiv 0$  gelten und die Gleichung  $A(X)U_j(X) - B(X)V_j(X) = 0$  hat ebenfalls eine nicht-triviale Lösung. Nach Lemma 3.11 haben  $A$  und  $B$  dann mind  $j + 1$  gemeinsame Nullstellen.  $\square$

Wir betrachten nun zwei Beispiele. Zunächst ein Beispiel, dass zwei Polynome mit festen Koeffizienten verwendet. Danach ein Beispiel, dass gemäß der Aufgabenstellung Koeffizienten verwendet, die selbst wieder Polynome sind.

**Beispiel I:** Betrachte  $A(X) = X^3 - 6X^2 + 11X - 6$  und  $B(X) = 2X^2 - 6X + 4$ , dann ist  $(0 \leq j \leq \min\{a, b\} - 1 = 1)$ . Als Übungsaufgabe für den Leser ist zu zeigen, dass  $\psi_0(A, B) = 0$  gilt.

Sei nun  $j = 1$ , dann wähle  $U_1(X) = u_0$  mit Grad  $b - j - 1 = 0$ , und  $V_1(X) = v_1X + v_0$  mit Grad  $a - j - 1 = 1$ . Aus  $A(X)U_j(X) - B(X)V_j(X) = 0$  folgt  $u_0(X^3 - 6X^2 + 11X - 6) = (2X^2 - 6X + 4)(v_1X + v_0)$ . Wir betrachten die ersten  $a + b - 2j = 3$  Gleichungen von  $a + b - j = 4$  Gleichungen mit  $a + b - 2j = 3$  Unbekannten. Aus  $u_0(X^3 - 6X^2 + 11X - 6) = 2v_1X^3 + (-6v_1 + 2v_0)X^2 + (4v_1 - 6v_0)X + 4v_0$  erhalten wir folgende Rechnung:

$$\psi_1(A, B) = \begin{vmatrix} 2 & 0 & -1 \\ -6 & 2 & 6 \\ 4 & -6 & -11 \end{vmatrix} = \begin{vmatrix} 2 & 0 & -1 \\ 6 & 2 & 0 \\ 18 & -6 & 0 \end{vmatrix} = \begin{vmatrix} 2 & 0 & 1 \\ 6 & -2 & 0 \\ 0 & 0 & 0 \end{vmatrix} = 0.$$

Daraus folgt, dass  $A(X) = X^3 - 6X^2 + 11X - 6$  und  $B(X) = 2X^2 - 6X + 4$  mindestens zwei gemeinsame Nullstellen haben. Das stimmt auch, denn  $(X - 3)(X - 2)(X - 1) = A(X)$  und  $2(X - 2)(X - 1) = B(X)$  läßt sich leicht verifizieren.

**Beispiel II:** Sei  $P(X, Y, Z) = X^2 + Y^2 + Z^2 - 1$ . Gemäß der rekursiven Konstruktion einer zylindrischen Zerlegung wollen wir den Grad von  $\hat{P}(X_n) := P_{x_1, \dots, x_{n-1}}(X_n)$  und die Anzahl der gemeinsamen Nullstellen von  $\hat{P}(X_n)$  und  $\hat{P}'(X_n)$  konstant halten! Nun sei  $P_{X,Y}(Z) = Z^2 + (X^2 + Y^2 - 1) =: A(Z)$  und  $P'_{X,Y}(Z) = 2Z =: B(Z)$ . Offensichtlich hat  $P_{X,Y}(Z)$  konstant Grad 2 unabhängig von  $X$  und  $Y$ .

Für die Festlegung der Anzahl gemeinsamer Nullstellen betrachten wir  $(0 \leq j \leq \min\{a, b\} - 1 = 0)$ . Für  $j = 0$  haben wir  $U_0(Z) = u_0$  mit Grad  $b - j - 1 = 0$  und  $V_0(Z) = v_1Z + v_0$  mit Grad  $a - j - 1 = 1$ . Dann ist  $A(Z)U_j(Z) - B(Z)V_j(Z) = u_0(Z^2 + (X^2 + Y^2 - 1)) - (v_1Z + v_0)2Z = (u_0 - 2v_1)Z^2 - 2v_0Z + u_0(X^2 + Y^2 - 1) = 0$ . Die ersten  $a + b - 2j = 3$  Gleichungen mit  $a + b - j = 3$  Unbekannten ergeben:

$$\begin{vmatrix} -2 & 0 & 1 \\ 0 & -2 & 0 \\ 0 & 0 & (X^2 + Y^2 - 1) \end{vmatrix} = 4(X^2 + Y^2 - 1)$$

und das Vorzeichen von  $Q(X, Y) = 4(X^2 + Y^2 - 1)$  bestimmt die Anzahl der Nullstellen von  $P$ .

Aus den bisherigen Überlegungen läßt sich der folgende Vorschlag für die allgemeine Aufstellung der polynomiellen Bedingungen extrahieren:

**Theorem 3.13** Sei  $P_{(X_1, \dots, X_{n-1})}(X_n)$  gegeben und der Grad  $X_n$  in  $P$  sei  $k$ . Für  $j = 0, \dots, k$  sei  $Q_j(X_1, \dots, X_{n-1})$  der Koeffizient von  $P_{(X_1, \dots, X_{n-1})}(X_n)$  mit Grad  $j$ . Sei  $P_{(j, (X_1, \dots, X_{n-1}))}(X_n)$  die Summe aller Terme des Polynoms  $P_{(X_1, \dots, X_{n-1})}(X_n)$  mit Grad  $\leq j$  in  $X_n$ . Wir definieren weiterhin  $R_{(j,l)} := \psi_l(P_{(j, (X_1, \dots, X_{n-1}))}(X_n), P'_{(j, (X_1, \dots, X_{n-1}))}(X_n))$  für  $l = 0, \dots, j - 2$ . Sei  $\mathcal{Q}$  die Menge der Polynome aus  $Q_j$  und  $R_{(j,l)}$ . Dann gilt: Die Anzahl der verschiedener Nullstellen von  $P_{(X_1, \dots, X_{n-1})}(X_n)$  ist konstant falls alle Polynome aus  $\mathcal{Q}$  ein konstantes Vorzeichen haben.

Die Menge der betrachteten Polynome erscheint auf den ersten Blick etwas hochgegriffen. Es kann aber tatsächlich sein, dass einige Koeffizienten entfallen und die Betrachtungen dann für

die entstehenden Polynome mit kleinerem Grad ebenfalls durchgeführt werden müssen. Insofern ist dieser konservative Ansatz gerechtfertigt.

## Kapitel 4

# Roboter in der Fertigungstechnik

Abschließend wollen wir ein Thema behandeln, das im Zusammenhang mit Robotern in der industriellen Fertigung interessant ist.

### 4.1 Orientierung polygonaler Werkstücke ohne Sensoren

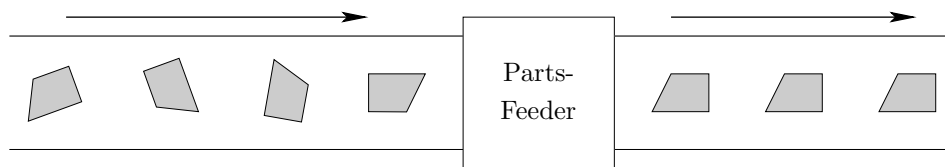


Abbildung 4.1: Parts-Feeder zur Orientierung von Werkstücken.

Stellen wir uns folgendes Problem vor: In einer Fertigungsanlage wird ein Fließband mit planaren Werkstücken — z. B. Rohlingen für Schlüssel — beschickt. Diese werden einer Maschine zur Verarbeitung zugeführt, die die Werkstücke nur verarbeiten kann, wenn diese in einer bestimmten Lage auf dem Fließband liegen. Die Lage der Werkstücke auf dem Fließband ist jedoch zunächst einmal zufällig; wenn etwa die Werkstücke aus einem Behälter auf das Band fallen. Zu konstruieren ist also eine Einheit, die die Werkstücke in die richtige Lage bringt, ein sogenannter Parts-Feeder. Die Eingabe in den Parts-Feeder ist ein Strom von Werkstücken in beliebiger Orientierung, die Ausgabe ein Strom von Werkstücken in eindeutiger Orientierung, siehe Abbildung 4.1.

Häufig sind Parts-Feeder direkt auf die Rohlinge abgestimmt, was zur Folge hat, daß die Hardware des Parts-Feeders auszutauschen ist, wenn sich die Werkstücke ändern. Interessanter wären reprogrammierbare Parts-Feeder, die aus Kostengründen ohne Sensoren auskommen sollten.

Wir betrachten hier den Parallel-Jaw Gripper, der aus einem Greifer mit zwei parallelen Backen besteht und ein Werkstück greifen und drehen kann, siehe Abbildung 4.2. Dieser Greifer ermöglicht die Ausführung der folgenden Aktion (Greifaktion)  $A$ :

- Drehe den Greifer auf einen Winkel  $\alpha$  bzgl. einer Normalstellung.
- Schließe beide Backen gleichzeitig, bis das Werkstück eingeklemmt ist.
- Öffne beide Backen.

Der einzige Parameter einer Greifaktion ist der Winkel  $\alpha$ ; wir können also eine Folge von Aktionen  $A_i$  durch eine Folge von Winkeln  $\alpha_i$  beschreiben. Das Problem läßt sich damit folgendermaßen spezifizieren: Gegeben ist die Beschreibung eines polygonalen Werkstücks<sup>1</sup> mit  $n$  Kanten und eine Zielorientierung des Werkstücks. Finde einen Plan  $\mathcal{A} = (A_1, \dots, A_k) = (\alpha_1, \dots, \alpha_k)$  zur Ansteuerung des Greifers, der ein Werkstück in beliebiger Ausgangslage in die Zielorientierung

<sup>1</sup>D.h. die konvexe Hülle des Werkstücks ist ein Polygon.

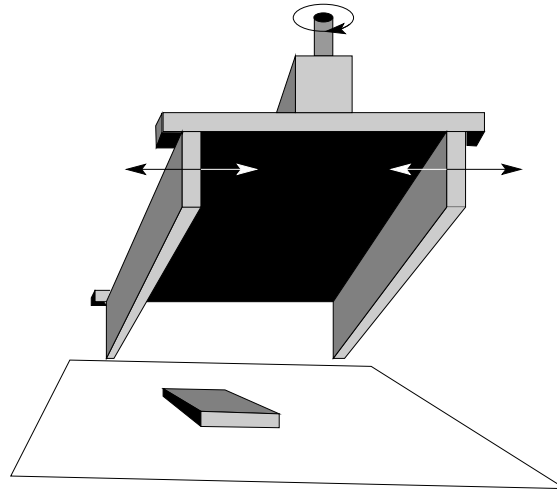
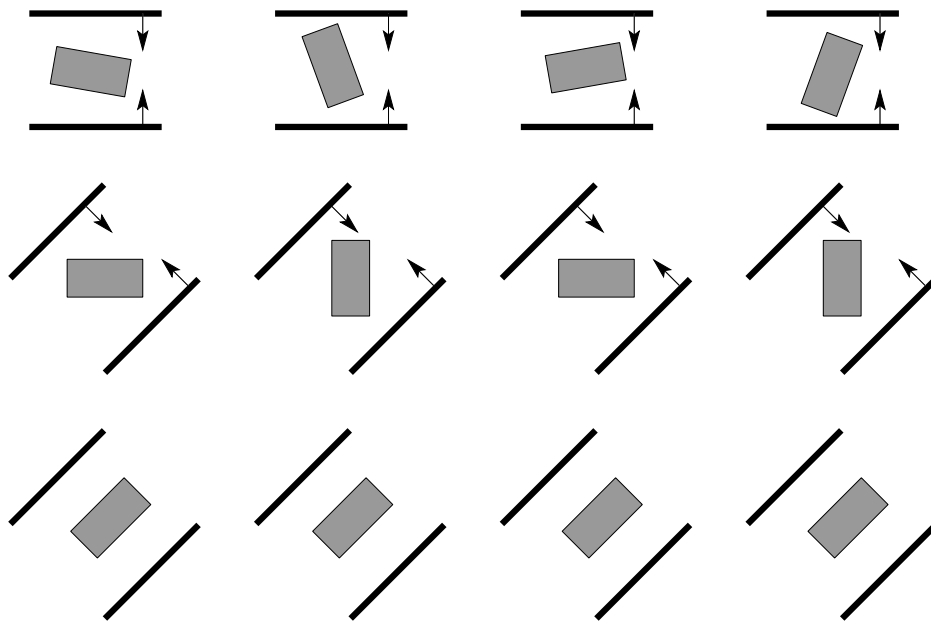
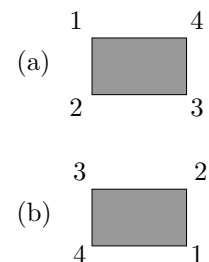


Abbildung 4.2: Parallel-Jaw Gripper.

Abbildung 4.3: Vier Beispiele für die Ausführung eines Plans  $\mathcal{A} = (0^\circ, 45^\circ)$ .

bringt. Abbildung 4.3 zeigt vier Beispiele für die Ausführung eines Plans  $\mathcal{A} = (0^\circ, 45^\circ)$  mit unterschiedlichen Ausgangslagen.

Die Zielorientierung kann dabei nur bis auf Symmetrien in der konvexen Hülle eindeutig sein, in nebenstehender Abbildung hat der Greifer keine Möglichkeit, die Orientierungen (a) und (b) voneinander zu unterscheiden.



Fassen wir die Annahmen zusammen:

1. Der Greifer besteht aus zwei parallelen Backen.
2. Die Richtung des Greifers ist orthogonal zu den Backen.



3. Die konvexe Hülle des Werkstücks wird als festes, planares Polygon angesehen, alle Aktionen beziehen sich auf die konvexe Hülle des Werkstücks.
4. Das Werkstück liegt isoliert vor, d. h. es muß nicht aus einem Haufen anderer Werkstücke gezogen werden und der Abstand zwischen zwei Werkstücken auf dem Band ist groß genug.
5. Die Ausgangsposition des Werkstücks ist unbekannt; bekannt ist nur, daß es zwischen den Backen des Greifers liegt.
6. Beide Backen treffen gleichzeitig auf das Werkstück (Pure Squeezing). Wir werden später betrachten, wie man mit einer kombinierten Schiebe-/Greifaktion ohne diese Annahme auskommt.
7. Es gibt keine Reibung zwischen den Backen und dem Werkstück.

Die letzte Annahme soll verhindern, daß sich das Werkstück zwischen den Backen einklemmt, wir nennen so eine Lage ein **instabiles Gleichgewicht**. Man kann diese Annahme durch mechanische Abhilfen wie Rollbänder am Greifer oder Vibrationen simulieren und algorithmisch verhindern, daß das Werkstück nach dem ersten Schritt in eine Lage gebracht wird, die zu einem instabilen Gleichgewicht führen könnte.

Eine Lage, in der mindestens eine Kante der konvexen Hülle des Werkstücks mit einer Backe Kontakt hat, nennen wir ein **stabiles Gleichgewicht**.

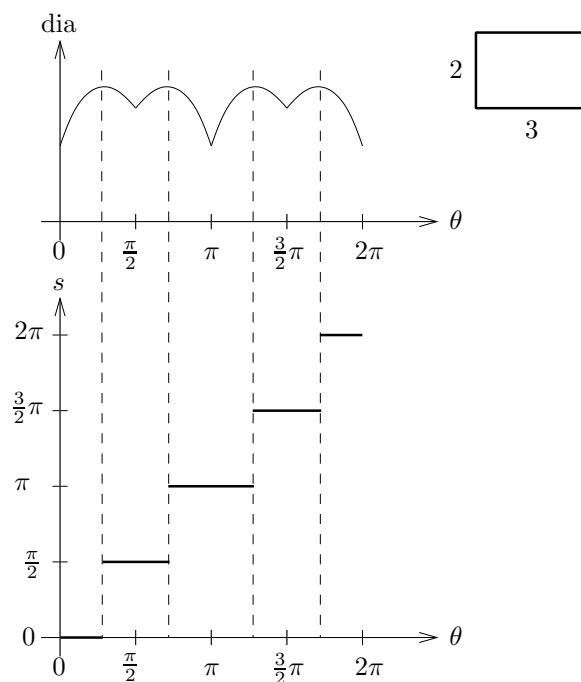
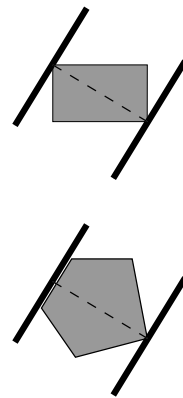


Abbildung 4.4: Durchmesser- und Greiffunktion.

Der Algorithmus zur Berechnung eines Plans für den Greifer stützt sich auf die beiden im folgenden definierten Funktionen:

**Definition 4.1**

- (i) Die **Durchmesserfunktion**  $\text{dia} : [0, 2\pi) \rightarrow \mathbb{R}$ , gibt für eine feste Orientierung des Werkstücks den Abstand  $\text{dia}(\theta)$  zweier paralleler Linien mit Winkel  $\theta$  an, die das Werkstück berühren.

- (ii) Die **Greiffunktion**  $s : [0, 2\pi) \rightarrow [0, 2\pi)$  bildet eine Orientierung  $\theta$  des Werkstücks *bezüglich des Greifers* auf eine Orientierung  $s(\theta)$  bezüglich des Greifers nach der Greifaktion ab. Sie ist stückweise konstant und berechnet sich aus der Durchmesserfunktion, indem alle Orientierungen, die zwischen zwei lokalen Maxima von  $\text{dia}(\theta)$  liegen, auf die Orientierung  $\theta$  abgebildet werden, an der  $\text{dia}(\theta)$  das lokale Minimum zwischen den beiden Maxima annimmt, vgl. Abbildung 4.4. Wir legen fest, daß die Intervalle nach rechts offen und nach links abgeschlossen sind.
- (iii) Für eine Sequenz von Aktionen  $\mathcal{A} = (\alpha_1, \alpha_2, \dots, \alpha_{k-1}, \alpha_k)$  und eine Ausgangsposition  $\theta$  bezeichne

$$S(\mathcal{A}, \theta) := s(s(\dots s(s(\theta - \alpha_1) - \alpha_2) \dots) - \alpha_{k-1}) - \alpha_k)$$

die Position des Werkstücks bzgl. des Greifers nach Ausführung der Sequenz  $\mathcal{A}$ .

Bei der Greiffunktion ist zu beachten, daß sie bezüglich des Greifers definiert ist. Befindet sich das Werkstück in einer Orientierung  $\theta$  bzgl. des Greifers und der Greifer führt eine Greifaktion bei Winkel  $\alpha$  aus, so ist die Orientierung des Werkstücks bzgl. des Greifers nach der Aktion  $s(\theta - \alpha)$ .

Die Durchmesserfunktion ist uns schon bei der Berechnung von kürzesten Pfaden für Liniensegmenten, in Theorem 1.46 auf Seite 57 begegnet. Sie beschreibt, wie sich der Abstand zweier paralleler Linien — in unserem Fall die Backen des Greifers — ändert, wenn die Linien um das Werkstück rotieren. Ein stabiles Gleichgewicht wird offensichtlich in den lokalen Minima der Durchmesserfunktion erreicht.

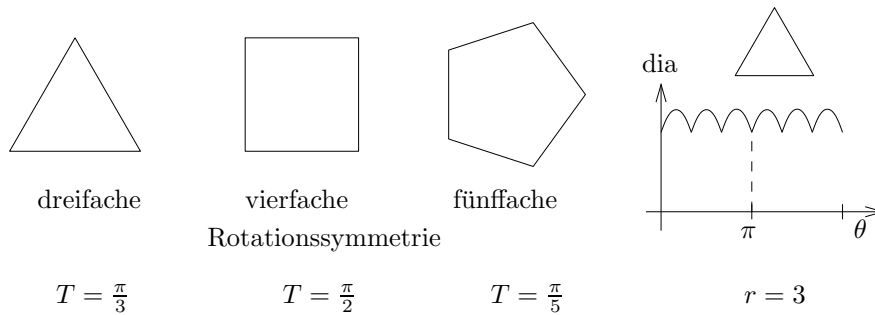


Abbildung 4.5: Rotationssymmetrie und Periodizität von  $\text{dia}$ ;  $\text{dia}$  für  $r = 3$ .

Die Durchmesserfunktion hat aufgrund ihrer Definition stets eine Periode von  $\pi$ : eine Rotation der parallelen Linien (bzw. des Greifers) um  $180^\circ$  liefert denselben Durchmesser. Weitere Perioden entstehen durch Rotationssymmetrie in der konvexen Hülle des Werkstücks, siehe Abbildung 4.5. Perioden in der Durchmesserfunktion induzieren eine Struktur in die Greiffunktion:

**Definition 4.2** Eine Greiffunktion  $s$  ist **T-periodisch**, falls für alle  $\theta \in [0, 2\pi)$  gilt:

$$s(\theta + T) = s(\theta) + T.$$

**Lemma 4.3**

(i) Für Polygone mit  $r$ -facher Rotationssymmetrie ist die Greiffunktion  $T$ -periodisch mit

$$T = \frac{2\pi}{r(1 + (r \bmod 2))}.$$

(ii) Für eine  $T$ -periodische Greiffunktion  $s$  gilt:

$$S(\mathcal{A}, \theta + T) = S(\mathcal{A}, \theta) + T.$$

**Beweis.** Übungsaufgabe. □

Aus Lemma 4.3(ii) folgt, daß es keinen Plan gibt, der die Orientierungen  $\theta$  und  $\theta + T$  in eine gemeinsame Zielorientierung überführen kann.

**Definition 4.4** Gegeben sei ein Werkstück  $W$ . Sei  $T$  die kleinste Periode der zu  $W$  gehörenden Greiffunktion  $s$ . Ein Plan  $\mathcal{A}$  orientiert die konvexe Hülle von  $W$  **bis auf Symmetrie**, falls in der Menge der möglichen letzten Orientierungen  $S(\mathcal{A}, \theta)$  genau  $\frac{2\pi}{T}$  Orientierungen sind, die gleichverteilt auf  $[0, 2\pi)$  liegen.

Für das Werkstück in Abbildung 4.3 z. B. gilt  $r = 2, T = \pi$ . Für den Plan  $\mathcal{A} = (0^\circ, 45^\circ)$  gibt es die möglichen Zielorientierungen  $\frac{\pi}{4}$  und  $\frac{5}{4}\pi$ .

Zur Beschreibung des Algorithmus müssen wir noch einige Begriffe festlegen:

- (i) Ein **Intervall**  $\Theta$  ist eine zusammenhängende Teilmenge von  $[0, 2\pi)$ ,  $|\Theta|$  bezeichne die Länge des Intervalls.
- (ii) Ein **s-Intervall** ist ein halboffenes Intervall  $[\xi_i, \nu_i)$ , wobei  $\xi_i$  und  $\nu_i$  Unstetigkeitsstellen der Greiffunktion sind. Da die Greiffunktion  $O(n)$  Unstetigkeitsstellen hat, gibt es  $O(n^2)$  eindeutige s-Intervalle.
- (iii) Zu einem s-Intervall  $\Theta$  sei das **s-Image**  $s(\Theta)$  das kleinste Intervall, das die Menge

$$\{s(\theta) \mid \theta \in \Theta\}$$

enthält. Ein s-Image ist stets ein abgeschlossenes Intervall.

---

**Algorithmus 4.1** Orientierung eines Werkstücks (Pure Squeezing)

---

- Berechne die Durchmesserfunktion und die Greiffunktion.
- Bestimme das längste s-Intervall  $\Theta_1$ , über dem die Greiffunktion stetig ist. Wir legen fest, daß dieses Intervall zur Periode gehört. Setze  $i := 1$ .
- Solange ein s-Intervall  $\Theta$  mit  $|s(\Theta)| < |\Theta_i|$  existiert:
  - Setze  $\Theta_{i+1}$  auf das Größte dieser s-Intervalle.
  - Inkrementiere  $i$ .

$\Rightarrow$  Liste  $\mathcal{L} = (\Theta_1, \Theta_2, \dots, \Theta_i)$  mit  $|\Theta_i| = T$ .

- Berechne aus  $\mathcal{L}$  einen Plan  $\mathcal{A} = (\alpha_i, \alpha_{i-1}, \dots, \alpha_1)$ :

–  $\alpha_i := 0$ .

– Für  $j = i - 1, i - 2, \dots, 1$ :

$$\alpha_j := s(\xi_{j+1}) - \xi_j - \varepsilon_j + \alpha_{j+1}.$$

Dabei ist  $\varepsilon_j = \frac{1}{2} (|\Theta_j| - |s(\Theta_{j+1})|)$  eine Fehlertoleranz, u.a. zur Vermeidung instabiler Gleichgewichte.

---

Algorithmus 4.1 findet einen Plan zur Orientierung des Werkstücks. Zur Illustration betrachten wir ein Rechteck mit den Kantenlängen 2 und 3 (Abbildung 4.4). Da das Seitenverhältnis  $\frac{3}{2}$  beträgt, sei  $a = \arctan \frac{3}{2}$ . In Schritt 2 wird der größte Schritt in der Greiffunktion gefunden, in unserem Fall  $\Theta_1 = [\pi - a, \pi + a)$  mit  $|\Theta_1| = 2a$ . Man beachte, daß  $s(\Theta_1)$  eine einzige Orientierung mit Winkel  $\pi$  ist (Abbildung 4.6(i)). Im dritten Schritt suchen wir das größte s-Intervall mit einem s-Image kleiner als  $\Theta_1$ . Wie in Abbildung 4.6(ii) gezeigt, kann dies durch Anlegen eines Quadrats mit Seitenlängen  $|\Theta_i|$  linksbündig an alle Stufen der Greiffunktion gesucht werden.

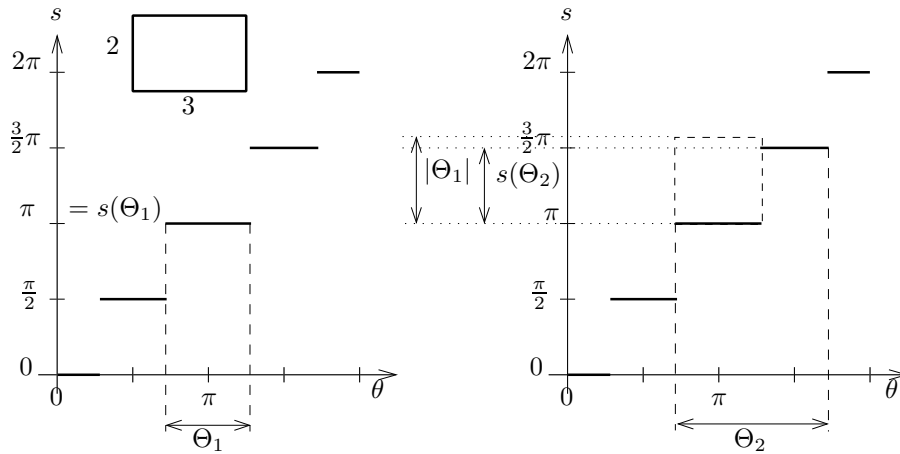


Abbildung 4.6: Finden von s-Intervallen.

Berührt das nächste Intervall der Greiffunktion das Quadrat, hat man ein solches  $s$ -Intervall gefunden, hier bei  $\Theta_2 = [\pi - a, 2\pi - a]$  mit  $s(\Theta_2) = [\pi, \frac{3}{2}\pi]$  und  $|s(\Theta_2)| = \frac{\pi}{2} < |\Theta_1|$ . So fahren wir fort, bis der Algorithmus mit  $|\Theta_i| = T$  terminiert, was in unserem Beispiel wegen  $|\Theta_2| = \pi$  bereits der Fall ist.

Unser Werkstück liegt zu Beginn in einer Orientierung  $\theta \in \Theta_2$ . Nach der ersten Aktion bei Winkel  $0^\circ$  liegt die Orientierung des Werkstücks in  $s(\Theta_2)$ . Wegen  $|s(\Theta_2)| < |\Theta_1|$  können alle Orientierungen aus  $s(\Theta_2)$  mit einer Aktion auf einen Punkt abgebildet werden, wenn  $s(\Theta_2)$  in  $\Theta_1$  ausgerichtet werden kann. Dies kann durch Drehung des Greifers auf den Winkel  $s(\xi_2) - \xi_1$  erreicht werden. Da die Orientierung des Werkstücks dann in  $\Theta_1$  bzgl. der Greiferstellung liegt, kann mit einer zweiten Aktion das Werkstück in die Orientierung  $s(\Theta_1)$  gebracht werden.

Wenn die Orientierung des Werkstücks zu Beginn im Bereich  $\Theta_i$  liegt, wird es auf die Orientierung  $s(\Theta_i) + \alpha_i$  gedreht. Wegen  $|\Theta_i| = T$  orientiert der Plan  $\mathcal{A}$  das Werkstück bis auf Symmetrie. Unser Rechteck wird also mit dem Plan  $\mathcal{A} = (0, \frac{\pi}{4})$  auf den Winkel  $\frac{\pi}{4}$  gedreht, wenn seine Orientierung zu Beginn im Bereich  $[-a, \pi - a]$  lag, und auf den Winkel  $\frac{5}{4}\pi$ , wenn es im Bereich  $[\pi - a, -a]$  lag.

**Theorem 4.5** (Goldberg, 1993) Gegeben sei eine Liste von  $n$  Kanten, die die konvexe Hülle eines gegebenen Werkstücks repräsentieren. Dann läßt sich in Zeit  $O(n^2 \log n)$  die kürzeste Sequenz von Greifaktionen finden, die eine Orientierung des Werkstücks bis auf Symmetrie garantiert. Der gefundene Plan hat eine Länge von  $O(n^2)$ . [Gol93]

### Beweis.

Wir zeigen die Korrektheit und die Vollständigkeit von Algorithmus 4.1 und schätzen seine Laufzeit ab:

### Korrektheit:

Zu zeigen ist: (i) der berechnete Plan orientiert das Werkstück eindeutig bis auf Symmetrie, und (ii) es gibt keinen kürzeren.

- (i) Der Algorithmus findet einen Plan, der ein  $s$ -Intervall  $\Theta$  der Länge  $T$  auf einen Punkt  $\theta'$  abbildet, wobei  $T$  die kleinste Periode der Greiffunktion des Werkstücks ist. Wegen  $s(\theta + T) = s(\theta) + T$  gilt für jeden Plan  $\mathcal{A}$ :  $\mathcal{A}(\theta + T) = \mathcal{A}(\theta) + T$ . Z.B. gilt für einen zweistufigen Plan  $\mathcal{A} = (\alpha_1, \alpha_2)$ :

$$\begin{aligned} S(\mathcal{A}, \theta + T) &= s(s(\theta + T - \alpha_1) - \alpha_2) \\ &= s(s(\theta - \alpha_1) + T - \alpha_2) \\ &= s(s(\theta - \alpha_1) - \alpha_2) + T \end{aligned}$$

$$\begin{aligned}
&= S(\mathcal{A}, \theta) + T \\
&= \theta' + T
\end{aligned}$$

Wenn der Plan alle Orientierungen eines s-Intervalls  $\Theta$  auf  $\theta'$  abbildet, bildet er alle Orientierungen eines s-Intervalls  $\Theta'$  mit gleicher Länge wie  $\Theta$  und einem Offset von  $kT$  auf  $\theta' + kT$  ab. Die Zielorientierung wird also eine der  $\frac{2\pi}{T}$  Orientierungen sein, die gleichverteilt auf  $[0, 2\pi)$  liegen. Der Plan wird also das Werkstück bis auf Symmetrie orientieren.

- (ii) Zum Beweis, daß es keinen kürzeren Plan gibt, benutzen wir folgende

**Behauptung:** Jeder Plan, der eine Menge  $\Theta \subseteq [0, \pi)$  auf einen Punkt  $\theta$  abbildet, bildet auch das kleinste zusammenhängende Intervall, das  $\Theta$  enthält, auf  $\theta$  ab.

**Beweis:** Sei  $\Theta'$  das kleinste zusammenhängende Intervall, das  $\Theta$  enthält. Wegen der Monotonie der Greiffunktion gilt  $s(\Theta') = s(\Theta)$ . Die erste Greiffaktion wird also in beiden Fällen das gleiche s-Image erzeugen und mit dem Rest des Plans kann  $\Theta'$  auf einen Punkt abgebildet werden.  $\diamond$

Nehmen wir nun an, es existiert ein Plan  $\mathcal{A}'$  mit weniger Schritten als  $\mathcal{A}$ . Sei nun  $(\Theta_1, \Theta_2, \dots, \Theta_i)$  die Liste der s-Intervalle, die der Algorithmus gefunden hat, und seien  $(\Theta'_1, \Theta'_2, \dots, \Theta'_j)$  die zum Plan  $\mathcal{A}'$  gehörenden Intervalle erweitert auf zusammenhängende Intervalle. Aufgrund des Algorithmus gilt  $|\Theta_1| \geq |\Theta'_1|$ . Da  $\mathcal{A}'$  nach  $j$  Schritten terminiert,  $\mathcal{A}$  jedoch nicht, muß  $|\Theta_j| < |\Theta'_j|$  gelten. Also existiert ein  $k$  mit  $|\Theta_k| \geq |\Theta'_k|$  und  $|\Theta_{k+1}| < |\Theta'_{k+1}|$ .

✗, der Algorithmus hätte das größere Intervall  $\Theta'_{k+1}$  gewählt.

### Vollständigkeit:

Für jedes Werkstück finden wir einen Plan. Aus technischen Gründen erweitern wir hier die Greiffunktion mit Periode  $T$  auf  $s : \mathbb{R} \rightarrow \mathbb{R}$  — wegen  $s(\theta + T) = s(\theta) + T$  ist dies möglich. Wir beweisen die Vollständigkeit, indem wir zeigen, daß wir für jede Greiffunktion stets eine Sequenz von s-Intervallen  $(\Theta_1, \Theta_2, \dots, \Theta_n)$  finden, bei der  $|\Theta_j|$  größer ist als  $|s(\Theta_{j+1})|$ . Wir zeigen, daß wir für jede stückweise konstante, monotone Greiffunktion und jedes s-Intervall stets ein größeres s-Intervall finden, bis wir auf die Periode  $T$  in der Funktion stoßen.

Sei  $h$  die Länge des bereits gefundenen s-Intervalls, dann ist zu zeigen:

- (i) Entweder finden wir ein s-Intervall, dessen s-Image kleiner ist:

$$\exists \theta : s(\theta + h) - s(\theta) < h,$$

- (ii) oder  $h$  ist die Periode der Greiffunktion:

$$\forall \theta : s(\theta + h) = s(\theta) + h.$$

(i) sagt aus, daß zum aktuellen s-Intervall  $\Theta_j = [\theta_j, \theta_j + h)$  ein Intervall  $\Theta = [\theta, \theta + h]$  existiert, dessen s-Image kleiner ist als  $\Theta_j$ . Wir können  $\Theta$  nach rechts bis zur nächsten Unstetigkeitsstelle erweitern, ohne das s-Image zu vergrößern. Dies liefert ein s-Intervall  $\Theta'$  mit kleinerem s-Image als  $\Theta_j$ , dessen Differenz zu  $\Theta$  ein offenes Intervall ist, dessen Länge somit ungleich Null ist. Also gilt  $|\Theta'| > |\Theta_j|$ .

Um zu zeigen, daß entweder (i) oder (ii) gilt, betrachte das Integral von  $s(\theta + h) - s(\theta) - h$  über  $[0, T)$ :

$$\begin{aligned}
\int_0^T s(\theta + h) - s(\theta) - h \, d\theta &= \int_h^{T+h} s(\theta) \, d\theta - \int_0^T s(\theta) \, d\theta - hT \\
&=^2 - \int_0^h s(\theta) \, d\theta + \int_T^{T+h} s(\theta) \, d\theta - hT
\end{aligned}$$

$$\begin{aligned}
&= -\int_0^h s(\theta) d\theta + \int_0^h s(\theta) + T d\theta - hT \\
&= -\int_0^h s(\theta) d\theta + \int_0^h s(\theta) d\theta + hT - hT \\
&= 0
\end{aligned}$$

Also existiert ein Punkt, an dem die Funktion einen negativen Wert annimmt (i), oder die Funktion ist konstant Null (ii).

#### Laufzeit:

Durchmesser- und Greiffunktion lassen sich in Zeit  $O(n)$  berechnen. Auch der größte Teilschritt in der Greiffunktion (Schritt 2) läßt sich in Zeit  $O(n)$  berechnen, da die Funktion maximal  $O(n)$  Unstetigkeitsstellen hat. In der Schleife (Schritt 3) werden  $O(n^2)$  s-Intervalle durchlaufen, die in einer nach  $|s(\Theta)|$  sortierten Liste gespeichert sind; Schritt 3 benötigt also Zeit  $O(n^2)$ . Die Konstruktion des Plans aus der Liste der Intervalle  $(\Theta_1, \Theta_2, \dots, \Theta_i)$  ist in Zeit  $O(i)$  möglich. Es dominiert die Zeit für das Sortieren der Intervalle; insgesamt erhalten wir eine Laufzeit von  $O(n^2 \log n)$ . □

Bisher sind wir von der unrealistischen Annahme ausgegangen, daß beide Backen des Greifers das Werkstück gleichzeitig berühren. Von dieser Annahme wollen wir uns im folgenden lösen und Aktionen betrachten, bei denen das Werkstück erst ein Stück geschoben und dann gegriffen wird (Push-and-Grasp). Eine Aktion des Greifers besteht jetzt aus folgenden Schritten:

- Drehe den Greifer auf einen Winkel  $\alpha$  bzgl. einer Normalstellung.
- Verschiebe den Greifer in Richtung  $\alpha + \frac{\pi}{2}$  um eine Distanz  $\delta$ .
- Schließe beide Backen gleichzeitig, bis das Werkstück eingeklemmt ist.
- Verschiebe den Greifer in Richtung  $-(\alpha + \frac{\pi}{2})$  um  $-\delta$ .
- Öffne beide Backen.

Wir nehmen jetzt an, daß der Schwerpunkt des Werkstücks bekannt ist und die Distanz  $\delta$  so groß ist, daß das Werkstück durch Verschieben in eine stabile Orientierung gebracht werden kann.

Analog zur Durchmesser- und Greiffunktion benutzen wir hier folgende Funktionen:

#### Definition 4.6

- (i) Die **Radiusfunktion**  $\text{rad} : [0, 2\pi) \rightarrow \mathbb{R}$  gibt für eine feste Orientierung des Werkstücks den Abstand  $\text{rad}(\theta)$  zwischen dem Schwerpunkt und einer Linie mit Winkel  $\theta$  an, die das Werkstück berührt.
- (ii) Die **Schiebefunktion**  $p : [0, 2\pi) \rightarrow [0, 2\pi)$  bildet eine Orientierung  $\theta$  des Werkstücks *bezüglich des Greifers* auf eine Orientierung  $p(\theta)$  bezüglich des Greifers nach der Schiebeaktion ab. Sie ist stückweise konstant und berechnet sich aus der Radiusfunktion, indem alle Orientierungen, die zwischen zwei lokalen Maxima von  $\text{rad}(\theta)$  liegen, auf die Orientierung  $\theta$  abgebildet werden, an der  $\text{rad}(\theta)$  das lokale Minimum zwischen den beiden Maxima annimmt.
- (iii) Die **Transferfunktion**  $g : [0, 2\pi) \rightarrow [0, 2\pi)$  beschreibt eine Aktion, bei der zunächst geschoben, danach gegriffen wird:

$$g := s \circ p.$$

Damit können wir mit Algorithmus 4.1 einen Plan berechnen; im ersten Schritt ist natürlich die Transferfunktion statt der Greiffunktion zu berechnen.

---

<sup>2</sup>Wegen der Periodizität gilt:  $\int_h^{T+h} \dots = \int_0^T \dots - \int_0^h \dots + \int_T^{T+h} \dots$







# Literaturverzeichnis

- [AAAS97] Pankaj K. Agarwal, Nina Amenta, Boris Aronov, and Micha Sharir. Largest placements and motion planning of a convex polygon. In Jean-Paul Laumond and Mark Overmars, editors, *Algorithms for Robotic Motion and Manipulation*, pages 143–154, Wellesley, MA, 1997. A. K. Peters.
- [Ack28] W. Ackermann. Zum Hilbertschen Aufbau der reellen Zahlen. *Mathematical Annals*, 99:118–133, 1928.
- [AK00] Franz Aurenhammer and Rolf Klein. Voronoi diagrams. In Jörg-Rüdiger Sack and Jorge Urrutia, editors, *Handbook of Computational Geometry*, pages 201–290. Elsevier Science Publishers B.V. North-Holland, Amsterdam, 2000.
- [AKM<sup>+</sup>87a] A. Aggarwal, M. M. Klawe, S. Moran, P. W. Shor, and R. Wilber. Geometric applications of a matrix-searching algorithm. *Algorithmica*, 2:195–208, 1987.
- [Akm87b] V. Akman. *Unobstructed Shortest Paths in Polyhedral Environments*, volume 251 of *Lecture Notes Comput. Sci.* Springer-Verlag, 1987.
- [ALMS98] L. Aleksandrov, M. Lanthier, A. Maheshwari, and J.-R. Sack. An  $\varepsilon$ -approximation algorithm for weighted shortest paths on polyhedral surfaces. In *Proc. 6th Scand. Workshop Algorithm Theory*, volume 1432 of *Lecture Notes Comput. Sci.*, pages 11–22. Springer-Verlag, 1998.
- [AS00] Pankaj K. Agarwal and Micha Sharir. Davenport-Schinzel sequences and their geometric applications. In Jörg-Rüdiger Sack and Jorge Urrutia, editors, *Handbook of Computational Geometry*, pages 1–47. Elsevier Science Publishers B.V. North-Holland, Amsterdam, 2000.
- [Baj85] C. Bajaj. The algebraic complexity of shortest paths in polyhedral spaces. In *Proc. 23rd Allerton Conf. Commun. Control Comput.*, pages 510–517, 1985.
- [BGG96] E. Börger, E. Grädel, and Y. Gurevic. *The Classical Decision Problem. Perspectives in Mathematical Logic*. Springer Verlag, Berlin, 1996.
- [ČČ61] Josef Čapek and Karel Čapek. *R.U.R. and the Insect Play*. Oxford Paperbacks, 1961.
- [Cha82] Bernard Chazelle. A theorem on polygon cutting with applications. In *Proc. 23rd Annu. IEEE Sympos. Found. Comput. Sci.*, pages 339–349, 1982.
- [Cha91] Bernard Chazelle. Triangulating a simple polygon in linear time. *Discrete Comput. Geom.*, 6(5):485–524, 1991.
- [CJ95] Svante Carlsson and Hakan Jonsson. Computing a shortest watchman path in a simple polygon in polynomial time. In *Proc. 4th Workshop Algorithms Data Struct.*, volume 955 of *Lecture Notes Comput. Sci.*, pages 122–134. Springer-Verlag, 1995.

- [CJN99] S. Carlsson, H. Jonsson, and B. J. Nilsson. Finding the shortest watchman route in a simple polygon. *Discrete Comput. Geom.*, 22(3):377–402, 1999.
- [Cla87] K. L. Clarkson. Approximation algorithms for shortest path motion planning. In *Proc. 19th Annu. ACM Sympos. Theory Comput.*, pages 56–65, 1987.
- [CN86] W. Chin and S. Ntafos. Optimum watchman routes. In *Proc. 2nd Annu. ACM Sympos. Comput. Geom.*, pages 24–33, 1986.
- [CN88] W. Chin and S. Ntafos. Optimum watchman routes. *Inform. Process. Lett.*, 28:39–44, 1988.
- [CN91] W.-P. Chin and S. Ntafos. Shortest watchman routes in simple polygons. *Discrete Comput. Geom.*, 6(1):9–31, 1991.
- [Col75] G. E. Collins. Quantifier elimination for real closed fields by cylindrical algebraic decomposition. In *Proc. 2nd GI Conference on Automata Theory and Formal Languages*, volume 33 of *Lecture Notes Comput. Sci.*, pages 134–183. Springer-Verlag, 1975.
- [Coo71] S. A. Cook. The complexity of theorem-proving procedures. In *Proc. 3rd Ann. ACM Symp. on Theory of Computing*, pages 151–158. Shaker Heights, 1971.
- [CR87] J. Canny and J. H. Reif. New lower bound techniques for robot motion planning problems. In *Proc. 28th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 49–60, 1987.
- [dBvKOS97] Mark de Berg, Marc van Kreveld, Mark Overmars, and Otfried Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, Berlin, 1997.
- [DELM03] Moshe Dror, Alon Efrat, Anna Lubiw, and Joseph S. B. Mitchell. Touring a sequence of polygons. In *Proc. 35th Annu. ACM Sympos. Theory Comput.*, pages 473–482, 2003.
- [DGST88] J. R. Driscoll, H. N. Gabow, R. Shrairaman, and R. E. Tarjan. Relaxed heaps: An alternative to Fibonacci heaps with applications to parallel computation. *Commun. ACM*, 31:1343–1354, 1988.
- [Dij59] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.
- [DS65] H. Davenport and A. Schinzel. A combinatorial problem connected with differential equations. *Amer. J. Math.*, 87:684–689, 1965.
- [Dub85] V. A. Dubovitskij. The Ulam problem of optimal motion of line segments. In *Optimization Software*. ??, New York, NY, 1985.
- [EG86] H. Edelsbrunner and Leonidas J. Guibas. Topologically sweeping an arrangement. In *Proc. 18th Annu. ACM Sympos. Theory Comput.*, pages 389–403, 1986.
- [EG89] H. Edelsbrunner and Leonidas J. Guibas. Topologically sweeping an arrangement. *J. Comput. Syst. Sci.*, 38:165–194, 1989. Corrigendum in 42 (1991), 249–251.
- [EGS90] H. Edelsbrunner, Leonidas J. Guibas, and Micha Sharir. The complexity and construction of many faces in arrangements of lines and of segments. *Discrete Comput. Geom.*, 5:161–196, 1990.

- [EOS86] H. Edelsbrunner, J. O'Rourke, and R. Seidel. Constructing arrangements of lines and hyperplanes with applications. *SIAM J. Comput.*, 15:341–363, 1986.
- [FT87] M. L. Fredman and R. E. Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *J. ACM*, 34:596–615, 1987.
- [Gar83] Martin Gardner. The game of life, parts I–III. In *Wheels, Life, and other Mathematical Amusements*. W. H. Freeman, New York, 1983.
- [GH87] Leonidas J. Guibas and J. Hershberger. Optimal shortest path queries in a simple polygon. In *Proc. 3rd Annu. ACM Sympos. Comput. Geom.*, pages 50–63, 1987.
- [GH89] Leonidas J. Guibas and J. Hershberger. Optimal shortest path queries in a simple polygon. *J. Comput. Syst. Sci.*, 39(2):126–152, October 1989.
- [GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, New York, NY, 1979.
- [GM87] S. K. Ghosh and D. M. Mount. An output sensitive algorithm for computing visibility graphs. In *Proc. 28th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 11–19, 1987.
- [GM91] S. K. Ghosh and D. M. Mount. An output-sensitive algorithm for computing visibility graphs. *SIAM J. Comput.*, 20:888–910, 1991.
- [Gol93] Kenneth Y. Goldberg. Orienting polygonal parts without sensors. *Algorithmica*, 10:201–225, 1993.
- [GSS89] Leonidas J. Guibas, Micha Sharir, and S. Sifrony. On the general motion planning problem with two degrees of freedom. *Discrete Comput. Geom.*, 4:491–521, 1989.
- [Gur75] A. B. Gurevich. The 'most economical' displacement of a segment. *Differential Equations*, 11:1583–1589, 1975.
- [Her91] J. Hershberger. A new data structure for shortest path queries in a simple polygon. *Inform. Process. Lett.*, 38:231–235, 1991.
- [HN97] M. Hammar and Bengt J. Nilsson. Concerning the time bounds of existing shortest watchman route algorithms. In *Proc. 11th International Symposium on Fundamentals of Computation Theory*, volume 1279 of *Lecture Notes Comput. Sci.*, pages 210–221. Springer-Verlag, September 1997.
- [HS86] S. Hart and Micha Sharir. Nonlinearity of Davenport-Schinzel sequences and of generalized path compression schemes. *Combinatorica*, 6:151–177, 1986.
- [HS93] J. Hershberger and Subhash Suri. Matrix searching with the shortest path metric. In *Proc. 25th Annu. ACM Sympos. Theory Comput.*, pages 485–494, 1993.
- [HS96] D. Halperin and Micha Sharir. A near-quadratic algorithm for planning the motion of a polygon in a polygonal environment. *Discrete Comput. Geom.*, 16:121–134, 1996.
- [HS97] J. Hershberger and Subhash Suri. Matrix searching with the shortest path metric. *SIAM J. Comput.*, 26(6):1612–1634, December 1997.
- [HS99] John Hershberger and Subhash Suri. An optimal algorithm for Euclidean shortest paths in the plane. *SIAM J. Comput.*, 28(6):2215–2256, 1999.

- [HSS84] J. E. Hopcroft, J. T. Schwartz, and Micha Sharir. On the complexity of motion planning for multiple independent objects: P-space hardness of the “Warehouseman’s Problem”. *Internat. J. Robot. Res.*, 3(4):76–88, 1984.
- [IKKM97] Christian Icking, Rolf Klein, Peter Köllner, and Lihong Ma. A Java applet for the dynamic visualization of Voronoi diagrams. In *Abstracts 13th European Workshop Comput. Geom.*, pages 46–47. Universität Würzburg, 1997.
- [IRWY93] Christian Icking, Günter Rote, Emo Welzl, and Chee Yap. Shortest paths for line segments. *Algorithmica*, 10:182–200, 1993.
- [Kle89] Rolf Klein. *Concrete and Abstract Voronoi Diagrams*, volume 400 of *Lecture Notes Comput. Sci.* Springer-Verlag, 1989.
- [Kle97] Rolf Klein. *Algorithmische Geometrie*. Addison-Wesley, Bonn, 1997.
- [Kle05] Rolf Klein. *Algorithmische Geometrie - Grundlagen, Methoden, Anwendungen*. Springer, Heidelberg, 2nd edition, 2005.
- [KLPS86] K. Kedem, R. Livne, J. Pach, and Micha Sharir. On the union of Jordan regions and collision-free translational motion amidst polygonal obstacles. *Discrete Comput. Geom.*, 1:59–71, 1986.
- [KO88] Y. Ke and J. O’Rourke. Lower bounds on moving a ladder in two and three dimensions. *Discrete Comput. Geom.*, 3:197–217, 1988.
- [KS90] K. Kedem and Micha Sharir. An efficient motion planning algorithm for a convex rigid polygonal object in 2-dimensional polygonal space. *Discrete Comput. Geom.*, 5:43–75, 1990.
- [KST97] K. Kedem, Micha Sharir, and S. Toledo. On critical orientations in the Kedem-Sharir motion planning algorithm for a convex polygon in the plane. *Discrete Comput. Geom.*, 17:227–240, 1997.
- [Lat91] J.-C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Boston, 1991.
- [LP83] T. Lozano-Pérez. Spatial planning: A configuration space approach. *IEEE Trans. Comput.*, C-32:108–120, 1983.
- [LP84] D. T. Lee and F. P. Preparata. Euclidean shortest paths in the presence of rectilinear barriers. *Networks*, 14:393–410, 1984.
- [LS87] D. Leven and Micha Sharir. On the number of critical free contacts of a convex polygonal object moving in two-dimensional polygonal space. *Discrete Comput. Geom.*, 2:255–270, 1987.
- [Ma00] Lihong Ma. Bisectors and voronoi diagrams for convex distance functions, 2000.
- [Mit00] Joseph S. B. Mitchell. Geometric shortest paths and network optimization. In Jörg-Rüdiger Sack and Jorge Urrutia, editors, *Handbook of Computational Geometry*, pages 633–701. Elsevier Science Publishers B.V. North-Holland, Amsterdam, 2000.
- [MMP87] Joseph S. B. Mitchell, D. M. Mount, and C. H. Papadimitriou. The discrete geodesic problem. *SIAM J. Comput.*, 16:647–668, 1987.
- [OW88] M. H. Overmars and Emo Welzl. New methods for computing visibility graphs. In *Proc. 4th Annu. ACM Sympos. Comput. Geom.*, pages 164–171, 1988.

- [OW93] Th. Ottmann and P. Widmayer. *Algorithmen und Datenstrukturen*, volume 70 of *Reihe Informatik*. BI Wissenschaftsverlag, Mannheim, 1993.
- [PV95] M. Pocchiola and G. Vegter. Computing the visibility graph via pseudo-triangulations. In *Proc. 11th Annu. ACM Sympos. Comput. Geom.*, pages 248–257, 1995.
- [RS94] J. H. Reif and J. A. Storer. A single-exponential upper bound for finding shortest paths in three dimensions. *J. ACM*, 41(5):1013–1019, 1994.
- [SA95] Micha Sharir and P. K. Agarwal. *Davenport-Schinzel Sequences and Their Geometric Applications*. Cambridge University Press, New York, 1995.
- [Sei91] R. Seidel. A simple and fast incremental randomized algorithm for computing trapezoidal decompositions and for triangulating polygons. *Comput. Geom. Theory Appl.*, 1(1):51–64, 1991.
- [Sha97] Micha Sharir. Algorithmic motion planning. In Jacob E. Goodman and Joseph O’Rourke, editors, *Handbook of Discrete and Computational Geometry*, chapter 40, pages 733–754. CRC Press LLC, Boca Raton, FL, 1997.
- [SS83a] J. T. Schwartz and Micha Sharir. On the “piano movers” problem I: The case of a two-dimensional rigid polygonal body moving amidst polygonal barriers. *Commun. Pure Appl. Math.*, 36:345–398, 1983.
- [SS83b] J. T. Schwartz and Micha Sharir. On the “piano movers” problem II: General techniques for computing topological properties of real algebraic manifolds. *Adv. Appl. Math.*, 4:298–351, 1983.
- [SS84] Micha Sharir and A. Schorr. On shortest paths in polyhedral spaces. In *Proc. 16th Annu. ACM Sympos. Theory Comput.*, pages 144–153, 1984.
- [SS86] Micha Sharir and A. Schorr. On shortest paths in polyhedral spaces. *SIAM J. Comput.*, 15:193–215, 1986.
- [Stö99] H. Stöcker. *Taschenbuch mathematischer Formeln und moderner Verfahren*. Verlag Harri Deutsch, Thun und Frankfurt am Main, 1999.
- [Tar51] A. Tarski. *A decision method for elementary algebra and geometry*. Univ. of California Press, Berkeley, CA, 1951.
- [TH93] Xuehou Tan and Tomio Hirata. Constructing shortest watchman routes by divide-and-conquer. In *Proc. 4th Annu. Internat. Sympos. Algorithms Comput.*, volume 762 of *Lecture Notes Comput. Sci.*, pages 68–77. Springer-Verlag, 1993.
- [THI93] X. H. Tan, T. Hirata, and Y. Inagaki. An incremental algorithm for constructing shortest watchman routes. *Internat. J. Comput. Geom. Appl.*, 3(4):351–365, 1993.
- [THI99] X. Tan, T. Hirata, and Y. Inagaki. Corrigendum to “an incremental algorithm for constructing shortest watchman routes”. *Internat. J. Comput. Geom. Appl.*, 9(3):319–323, 1999.
- [vdS94] A. F. van der Stappen. *Motion Planning amidst Fat Obstacles*. Ph.D. dissertation, Dept. Comput. Sci., Utrecht Univ., Utrecht, Netherlands, 1994.
- [Wel85] Emo Welzl. Constructing the visibility graph for  $n$  line segments in  $O(n^2)$  time. *Inform. Process. Lett.*, 20:167–171, 1985.

- [Yap87] C.-K. Yap. Algorithmic motion planning. In J. T. Schwartz and C.-K. Yap, editors, *Advances in Robotics 1: Algorithmic and Geometric Aspects of Robotics*, pages 95–143. Lawrence Erlbaum Associates, Hillsdale, NJ, 1987.

# Inhaltsverzeichnis

<b>1</b>	<b>Kürzeste Pfade</b>	<b>3</b>
1.1	Kürzeste Pfade in der Ebene mit polygonalen Hindernissen . . . . .	3
1.2	Kürzeste Pfade im Inneren eines einfachen Polygons . . . . .	18
1.2.1	Der Algorithmus von Lee und Preparata . . . . .	18
1.2.2	Der Algorithmus von Guibas und Hershberger . . . . .	20
1.2.3	Anwendung: Berechnung des geodätischen Durchmessers . . . . .	28
1.2.4	Shortest Watchman Routes . . . . .	33
1.3	Kürzeste Pfade im Raum . . . . .	45
1.4	Kürzeste Pfade auf der Oberfläche eines Polyeders . . . . .	50
1.5	Kürzeste Pfade für Liniensegmente . . . . .	57
<b>2</b>	<b>Kollisionsfreie Bahnen für polygonale Roboter</b>	<b>61</b>
2.1	Reine Translationsbewegungen für konvexe Roboter . . . . .	63
2.1.1	Voronoi-Diagramme, Translationsbewegungen mit Sicherheitsabstand . .	63
2.1.2	Berechnung von $\mathcal{C}_{\text{frei}}$ . . . . .	64
2.2	Reine Translationsbewegungen für beliebige Roboter . . . . .	73
2.2.1	Komplexität und Berechnung einzelner Zellen . . . . .	73
2.2.2	Anwendung: Translation eines beliebigen Polygons . . . . .	84
2.2.3	Anwendung: Allgemeine Systeme mit zwei Freiheitsgraden . . . . .	84
2.3	Rotations- und Translationsbewegungen für konvexe Roboter . . . . .	87
2.3.1	Kritische Plazierungen . . . . .	90
2.3.2	Knotengraph und Kantengraph . . . . .	98
2.3.3	Berechnung der kritischen Orientierungen . . . . .	104
2.3.4	Bahnplanung mit dem Kantengraph $E$ . . . . .	108
<b>3</b>	<b>Bewegungsplanung für allgemeine Systeme</b>	<b>115</b>
3.1	Eine untere Schranke für das allgemeine Bahnplanungsproblem . . . . .	115
3.2	Der Konfigurationsraum als semialgebraische Menge . . . . .	118
3.3	Collins' zylindrische algebraische Zerlegung . . . . .	120
3.4	Anwendung auf die Bewegungsplanung . . . . .	128
<b>4</b>	<b>Roboter in der Fertigungstechnik</b>	<b>131</b>
4.1	Orientierung polygonaler Werkstücke ohne Sensoren . . . . .	131
<b>A</b>	<b>Grundlagen</b>	<b>141</b>
A.1	Komplexitätsklassen . . . . .	141
A.2	Entscheidbarkeit . . . . .	143
A.3	Voronoi-Diagramme . . . . .	145
A.3.1	Definition und Eigenschaften . . . . .	145
A.3.2	Voronoi-Diagramme bzgl. konvexer Distanzfunktionen . . . . .	147
A.3.3	Voronoi-Diagramme von Liniensegmenten . . . . .	148
A.3.4	Voronoi-Diagramme mit additiven Gewichten . . . . .	149

A.4	Davenport–Schinzel–Sequenzen . . . . .	152
A.4.1	Definition und Eigenschaften . . . . .	152
A.4.2	Anwendung: die untere Kontur von Funktionen . . . . .	153
<b>B</b>	<b>Roboter</b>	<b>155</b>



# Abbildungsverzeichnis

1.1	Kürzeste von $a$ nach $b$ , Geodätische von $a'$ nach $b'$ . . . . .	3
1.2	Umgebung mit polygonalen Hindernissen. . . . .	4
1.3	Es kann exponentiell viele kürzeste Wege geben. . . . .	4
1.4	Der Sichtbarkeitsgraph kann (i) $\Omega(n^2)$ viele Kanten enthalten, aber auch (ii) nur $O(n)$ viele. . . . .	5
1.5	VisG und Kanten von $s$ und zu $t$ . . . . .	5
1.6	(i) Sichtbarkeitsinformationen können von $a$ an $f$ weitergereicht werden, (ii) Typen von Bedingungen an die Bearbeitungsreihenfolge. . . . .	6
1.7	Es gibt drei mögliche Ausgangssituationen eines Punktes $p$ mit zuletzt bearbeitetem Partner $q$ und sichtbarem Segment $\sigma$ . . . . .	7
1.8	Drei Fälle bei der Bearbeitung eines Paares $(p, r)$ . . . . .	8
1.9	Dualisierung $p = (p_x, p_y) \mapsto p^* = \{ Y = p_x X - p_y \}$ . . . . .	9
1.10	(i) Topologischer Sweep, (ii) Elementarschritt, (iii) Die Sweepkurve kann eine Zelle nur von oben betreten und nach unten verlassen. . . . .	10
1.11	Oberer und unterer Horizontbaum. . . . .	11
1.12	Aktualisierung der Horizontbäume am Beispiel von $T^O$ . . . . .	11
1.13	Zone einer Geraden $\ell$ in einem Arrangement $\mathcal{A}$ . . . . .	12
1.14	Im rechtesten Schnittpunkt schneiden sich zwei oder mehrere Geraden. Die gestrichelten Kanten werden neu eingefügt. . . . .	13
1.15	Welle, Ausläufer und Knotenmarkierungen während des Algorithmus von Dijkstra. . . . .	14
1.16	Reduktion: Konstruktion der konvexen Hülle auf Bestimmung des kürzesten Pfades. . . . .	16
1.17	Berechnung der Shortest Path Map. . . . .	16
1.18	Kürzester Pfad im Inneren eines einfachen Polygons. . . . .	18
1.19	Triangulation $T$ und dualer Graph $T^*$ . . . . .	18
1.20	Kette von Dreiecken. . . . .	19
1.21	Trichter. . . . .	19
1.22	Balancierte hierarchische Triangulation. . . . .	20
1.23	Baum der balancierten hierarchischen Triangulation. . . . .	21
1.24	Subpolygon $P(d_{4,3})$ und Abkürzungen in $\hat{G}$ . . . . .	21
1.25	Diagonalen $j, k, l, d_1, d_{first}$ im dualen Baum $T^*$ mit Wurzel $w$ . . . . .	23
1.26	Prä- und Postorderdurchlauf. . . . .	23
1.27	Offene und geschlossene Sanduhr zwischen $d_1 = \overline{ab}$ und $d_2 = \overline{cd}$ . . . . .	25
1.28	Kürzester Pfad in offener Sanduhr. . . . .	26
1.29	Speicherung einer Sanduhr in einem binären Baum. . . . .	26
1.30	Ein Polygon und sein geodätischer Durchmesser. . . . .	28
1.31	Fallunterscheidung nach Lage von $A$ in $T$ . . . . .	29
1.32	Spaltenreduktion. . . . .	32
1.33	Die Berechnung einer SWR liefert eine Besuchsreihenfolge der Punkte. . . . .	33
1.34	(i) X-monotones Polygon, (ii) nicht-monotones Polygon, (iii) rechtwinkliges Polygon. . . . .	34
1.35	Polygon mit notwendigen Cuts (gepunktet), wesentlichen Cuts (gestrichelt) und Shortest Watchman Route. . . . .	34

1.36	Berechnung der SWR in einem rechtwinkligen Polygon. . . . .	36
1.37	Eine “Corner”: die Cuts $c_1, c_2, c_3$ und $c_4$ schneiden sich gegenseitig. . . . .	36
1.38	Ein Beispiel für das einfache Touring Polygon Problem. . . . .	37
1.39	Ein Beispiel für das allgemeine Touring Polygon Problem. . . . .	38
1.40	Lokale Optimalität: (i) bei Reflektion an einer Kante gilt $\alpha = \beta$ , (ii) bei Reflektion an einer Ecke liegt die Ausgangskante im Winkelbereich $\gamma$ . . . . .	38
1.41	Vollständige kombinatorische Shortest Path Map. . . . .	39
1.42	Last Step Shortest Path Map zu $P_1$ . . . . .	39
1.43	Last Step Shortest Path Map zu $P_2$ . . . . .	40
1.44	Last Step Shortest Path Map zu $P$ . . . . .	40
1.45	Queries beim einfachen TPP: $q_1$ und $q_2$ liegen in der Durchgangsregion, $q_3$ und $q_4$ in einer Zelle zu einer Ecke, $q_5$ und $q_6$ in der Zelle einer Kante. . . . .	41
1.46	(i) Kürzeste Wege können an reflexen Ecken der Zäune abknicken, (ii) mögliche Kombinationen bei überlappenden Polygonen. . . . .	43
1.47	Kürzester Pfad im 3D. . . . .	45
1.48	Beispiel: Filter für Klausel $(X_1 \vee X_3) \wedge \overline{X_2}$ . . . . .	46
1.49	Die drei Ebenen eines Verdopplers. . . . .	46
1.50	Die vier Ebenen eines Mischers. . . . .	47
1.51	Literalfilter. . . . .	48
1.52	Klauselfilter. . . . .	49
1.53	Eigenschaften von Kürzesten auf Polyedern. . . . .	50
1.54	(i) Kürzeste über nichtkonvexe Ecke, (ii) Geodätische von $s$ nach $t$ , aber keine Kürzeste. . . . .	51
1.55	Pfad $\pi$ von $a$ nach $b$ über $x \in e$ . . . . .	52
1.56	$I(v, \mathcal{E})$ muß zusammenhängend sein. . . . .	53
1.57	Der Bisektor von $\bar{v}_{k-1}$ und $\bar{w}_{\ell-1}$ ist eine Hyperbel. . . . .	53
1.58	Dreiecksfläche $f$ mit Intervallen $I(v, \mathcal{E})$ . . . . .	55
1.59	Ein Dreieck $g$ mit Ecke $w$ fungiert höchstens zweimal als Trenner für zwei Intervalle. . . . .	55
1.60	Bewegung eines Liniensegmentes. . . . .	57
1.61	(i) Freiheitsgrade eines Liniensegmentes, (ii) geradlinige Bewegung (iii) Rotation. . . . .	57
1.62	(i) $\text{dia}_C(\alpha)$ , (ii) Bewegung $B$ , (iii) konvexe Hülle von $C$ . . . . .	58
1.63	$\text{dia}_D$ im <i>kritischen</i> Winkelbereich. . . . .	58
2.1	Roboter $R$ in einem Arbeitsraum. . . . .	61
2.2	(i) Verbotene Konfiguration, (ii) freie Platzierung (erlaubt), (iii) halbfreie Platzierung (erlaubt). . . . .	61
2.3	(i) Bahnplanung im Voronoi-Diagramm für Liniensegmente (ii) Konstruktion von $s'$ . . . . .	63
2.4	Hindernis, Roboter und Konfigurationsraum-Hindernis. . . . .	64
2.5	Konfigurationsraum-Hindernisse zu Abbildung 2.1. . . . .	65
2.6	(i) Pseudokreise, (ii) keine Pseudokreise, (iii) die Konvexität ist eine notwendige Forderung. . . . .	67
2.7	(i)–(iii) Beispiele für Pseudokreise, (iv) Rechtecke sind im allgemeinen keine Pseudokreise. . . . .	67
2.8	(i) $p$ muß Element von $Z$ sein, (ii) $A \cup B$ kann nicht aus zwei Zusammenhangskomponenten bestehen. . . . .	67
2.9	Um $A \cup B$ rotierende Tangenten wechseln zweimal zwischen $A$ und $B$ , weil $P_1$ und $P_2$ disjunkt sind. . . . .	68
2.10	(i) $P_1 \oplus P_2$ hat $\Theta(mn)$ Ecken, wenn nur eines der Polygone konvex ist, (iii) $P_1 \oplus P_2$ hat $\Theta(m^2n^2)$ Ecken, wenn keines der Polygone konvex ist. . . . .	71
2.11	Konfigurationsraum-Hindernisse und Roadmap. . . . .	72
2.12	Paare von Ecken und Kanten von $P_i$ und $R$ erzeugen im Konfigurationsraum ein Liniensegment. . . . .	73

2.13	Zelle in Arrangement von Kurvenstücken. . . . .	74
2.14	Die orientierten Segmente von $C$ . . . . .	74
2.15	Die Zelle $Z$ läge hier auf beiden Seiten von $C$ . . . . .	75
2.16	Kein Stück von $\gamma_i$ taucht zwischen $\zeta$ und $\eta$ entlang $C$ auf. . . . .	75
2.17	Das Segment $\gamma_3^-$ wird in der nicht-zyklischen Sequenz nicht in der Reihenfolge gemäß der Orientierung besucht und wird deshalb am Startpunkt $p$ verdoppelt. . . . .	75
2.18	Wird der Startpunkt $p$ gewählt, so müssen alle Segmente geteilt werden. . . . .	76
2.19	Die paarweise disjunkten Bögen $\beta_x^z, \beta_z^y, \beta_y^w, \beta_w^x, \beta_x^y$ und $\beta_z^w$ . . . . .	77
2.20	Eine kreuzungsfreie Einbettung, wie sie von der angenommenen Lage der Bögen induziert wird, existiert nicht. . . . .	77
2.21	$Z_R \cap Z_B$ kann sehr komplex sein, obwohl $Z$ recht einfach ist. . . . .	78
2.22	Arrangement von roten und blauen Zellen und Überlagerung. Komplexität der linken blauen (linken roten) Zelle: 6 (5) Segmente. . . . .	79
2.23	Der Bogen $\ell$ trägt zu vielen violetten Zellen bei. . . . .	79
2.24	(i) Zusätzliche vertikale Segmente, (ii) bei $q$ beginnt keine violette Region. . . . .	81
2.25	(i) Sweep von links nach rechts, (ii) Sweep von rechts nach links. . . . .	81
2.26	In $q$ starten (i) eine oder (ii) zwei violette Teilregionen mit Scouts. . . . .	82
2.27	Beispiele für die Aufgaben der Scouts. . . . .	82
2.28	Trapezzerlegung und Zusammenhangsgraph einer Zelle. . . . .	84
2.29	Roboter mit zwei Armen und Freiheitsgraden $\alpha$ und $\gamma$ . . . . .	85
2.30	Bahn eines Roboters mit Rotationen und Translationen. . . . .	87
2.31	Kann die Leiter von $s$ nach $t$ bewegt werden? . . . . .	87
2.32	Die Bewegung eines Liniensegments kann $\Omega(n^2)$ viele Schritte erfordern. . . . .	88
2.33	Konfigurationsraum-Hindernis bei Rotation und Translation. . . . .	89
2.34	Kritische Plazierungen. . . . .	90
2.35	Beweis: In jeder Zusammenhangskomponente gibt es mindestens eine kritische Platzierung. . . . .	91
2.36	Ein nichtkonvexer Roboter mit $\Theta(m^3n^3)$ kritischen Plazierungen. . . . .	92
2.37	$O_2$ beschränkt $O_1$ . . . . .	93
2.38	$\mathcal{C}_{\text{frei}}^\theta, \mathcal{C}_{\text{verb}}^\theta$ und Hilfsknoten von $V^\theta$ . . . . .	98
2.39	Konvexe und konkave Ecken in $\partial\mathcal{C}_{\text{verb}}^\theta$ . . . . .	98
2.40	(i) eine Kante verschwindet, (ii) eine Komponente von $\mathcal{C}_{\text{frei}}$ verschwindet, (iii) ein lokaler Zusammenhang verschwindet. . . . .	101
2.41	Zwei benachbarte Kanten werden kollinear. . . . .	101
2.42	Eine Kante wird waagerecht, das lokale Y-Maximum ändert sich. . . . .	102
2.43	Ein Hilfsknoten wandert auf eine Ecke. . . . .	102
2.44	Kantengraph $E$ . . . . .	103
2.45	Vereinigung zweier Komponenten $N$ und $Z_s$ . . . . .	105
2.46	Nicht-konvexer Roboter mit $\Omega(n^3)$ kritischen Platzierungen. . . . .	114
3.1	Umgebung zur Reduktion von Partition auf das allgemeine Bahnplanungsproblem. . . . .	116
3.2	Beispiele für semialgebraische Mengen. . . . .	118
3.3	Beispiel für eine zylindrische algebraische Zerlegungen. . . . .	121
3.4	Zylindrische algebraische Zerlegung für $n = 1$ . . . . .	122
3.5	Zerlegung des Zylinders $C \times \mathbb{R}$ . . . . .	124
3.6	Induktive Konstruktion von $\pi$ : Projektion und Liftung. . . . .	129
4.1	Parts-Feeder zur Orientierung von Werkstücken. . . . .	131
4.2	Parallel-Jaw Gripper. . . . .	132
4.3	Vier Beispiele für die Ausführung eines Plans $\mathcal{A} = (0^\circ, 45^\circ)$ . . . . .	132
4.4	Durchmesser- und Greiffunktion. . . . .	133
4.5	Rotationssymmetrie und Periodizität von dia; dia für $r = 3$ . . . . .	134

4.6	Finden von s-Intervallen. . . . .	136
A.1	(i) Bisektor zwischen $p$ und $q$ , (ii) Regionen von $p, q, r$ , (iii) Regionen von $p, q, r, s$ .	145
A.2	(i) Delaunay-Triangulation und konvexe Hülle, $x$ ist (ii) in Region von $p$ , (iii) auf Voronoi-Kante zwischen $p$ und $q$ , (iv) ein Voronoi-Knoten. . . . .	146
A.3	(i) Der neue Punkt $p_{i+1}$ liegt außerhalb der konvexen Hülle, (ii) $p_{i+1}$ liegt innerhalb des Umkreises des Dreiecks $(p_j, p_k, p_m)$ , die Kante $\overline{p_k p_m}$ wird umgeklappt. . . . .	147
A.4	(i) Einheitskreis der $L_1$ -Metrik, (ii) Bisektor zwischen $p$ und $q$ , (iii) der Bisektor zwischen $p$ und $q$ kann zwei Viertelebenen enthalten. . . . .	148
A.5	(i) Abstand von $p$ zum Ursprung bzgl. konvexer Distanzfunktion, (ii) Abstand von $p$ und $q$ . . . . .	148
A.6	Der zu $p$ nächstgelegene Punkt auf einem Liniensegment. . . . .	149
A.7	Der Bisektor eines Punktes und eines Liniensegmentes. . . . .	150
A.8	Der Bisektor zwischen zwei Liniensegmenten. . . . .	150
A.9	Voronoi-Diagramm von sieben Liniensegmenten. . . . .	151
A.10	Die untere Kontur von Funktionen (i) über einem gemeinsamen Intervall $I$ und (ii) über Intervallen $I_j$ . . . . .	153
A.11	Die untere Kontur von Liniensegmenten. . . . .	153
B.1	Roboter zum Ausschneiden . . . . .	155

# Verzeichnis der Algorithmen

1.1	Berechnung des kürzesten Weges mit Hilfe des Sichtbarkeitsgraphen . . . . .	5
1.2	Sweep zur Berechnung des Sichtbarkeitsgraphen . . . . .	6
1.3	Kürzeste Pfade in gewichtetem Graph (Dijkstra, 1959) . . . . .	14
1.4	Kürzester Pfad in einem Polygon (Lee, Preparata) . . . . .	20
1.5	Schneller Test auf Vorgängerrelation im Baum . . . . .	24
1.6	Kürzeste Pfade in einem Polygon (Guibas, Hershberger) . . . . .	27
1.7	Spaltenreduktion . . . . .	31
1.8	Zeilenmaxima . . . . .	31
1.9	Shortest Watchman Route in einfachen Polygonen . . . . .	35
1.10	Einfaches TPP: Query . . . . .	42
1.11	Einfaches TPP: Konstruktion der $\mathcal{S}_i$ . . . . .	42
1.12	Continuous Dijkstra zur Berechnung der $I(v, \mathcal{E})$ . . . . .	54
1.13	Berechnung kürzester Wege auf Polyedern . . . . .	56
2.1	Bahnplanung mit Sicherheitsabstand für kreisförmige Roboter . . . . .	64
2.2	Berechnung von $\mathcal{C}_{\text{verb}}$ . . . . .	70
2.3	Roadmap-Verfahren . . . . .	70
2.4	Berechnung einer Zelle im Arrangement von $n$ Bögen . . . . .	78
2.5	Red-Blue-Merge . . . . .	80
2.6	Sweep zur Berechnung der Teilzellen . . . . .	83
2.7	Translation eines beliebigen Roboters . . . . .	84
2.8	Berechnung von $T^+$ . . . . .	97
2.9	Berechnung von $T'$ . . . . .	104
2.10	Berechnung von $T''$ . . . . .	106
2.11	Berechnung von $T$ . . . . .	107
2.12	Berechnung des Kantengraphen . . . . .	112
2.13	Bewegungsplanung mit Kantengraph . . . . .	113
4.1	Orientierung eines Werkstücks (Pure Squeezing) . . . . .	135
A.1	Inkrementelle Berechnung der Delaunay-Triangulation . . . . .	147

