Technische Universität
München

Fakultät für Informatik

Forschungs- und Lehreinheit Informatik IX

Arrangements

Proseminar

Steffen Manthey

Betreuer: Dipl.-Inform. Robert Hanne

Abgabetermin: 28. 06. 2000

Inhaltsverzeichnis

1 Einleitung **2**

2 Eigenschaften von Arrangements **4**
2.1 Kombinatorisches 4
2.1.1 Das *Zone Theorem* 5
2.2 Inkrementelle Konstruktion 6
2.3 Drei und mehr Dimensionen 7
2.4 Dualität 8
2.4.1 Duale Abbildungen 8
2.4.2 Dualitäts-Eigenschaften 9

3 Anwendungen **12**
3.1 Hidden Surface Removal 12
3.2 Aspect Graphs 14
3.3 Kleinster Polytop-Schatten 16
3.4 Ham-Sandwich Cuts 17
3.4.1 Red-Blue Matching 20
3.4.2 Höhere Dimensionen 21

Literaturverzeichnis **22**

Stichwortverzeichnis **23**

Kapitel 1

Einleitung

Arrangements von Geraden (und Ebenen) bilden, neben Konvexen Hüllen und Voronoi Diagrammen, die dritte bedeutende Struktur, die in der Computational Geometry genutzt wird. Bevor wir uns eingehender mit ihnen beschäftigen, will ich zunächst definieren, was ein Arrangement überhaupt ist und eine kurze Motivation geben.

Definition 1.1 *Gegeben sei eine Menge von (unendlich vielen) Geraden, die in einer Ebene angeordnet (engl: arranged) sind. Diese Geraden bewirken eine Partition der Ebene in:*

- *konvexe Polygone, die als* Zellen *(engl: cells) oder* Facetten *(engl: faces) bezeichnet werden (Abkürzung F).*
- Segmente *oder* Kanten *(engl: edges) zwischen den Schnittpunkten der Geraden (Abkürzung E).*
- Knoten *(engl: vertices), wo sich die Geraden schneiden (Abkürzung V).*

Es ist diese Partition, die als Arrangement *bezeichnet wird. Es ist günstig die Zellen als offene Mengen (ohne ihre Kanten) und die Kanten als offene Segmente (ohne die sie begrenzenden Knoten) zu betrachten, so dass die Untergliederung eine echte Partition ist. Ihre Teile bedecken die Ebene vollständig, aber sie sind paarweise disjunkt.*

Ein Arrangement von Geraden ist in Abbildung 1.1 dargestellt. Es verfügt über $V = 45$ Knoten, $E = 100$ Kanten und $F = 56$ Facetten. Nicht alle sind jedoch in dem beschränkten Abbildungsbereich sichtbar.

Arrangements erscheinen viel zu abstrakt, um viele Anwendungen zu haben, aber in Wirklichkeit gibt es eine ganze Reihe davon:

1. *Hidden Surface Removal*
   Als Hidden Surface Removal wird die Ermittlung der verdeckten Objekte in einer dreidimensionalen Szene bezeichnet. Der erste Worst-Case optimale $\Theta(n^2)$ Algorithmus basiert auf Arrangements.

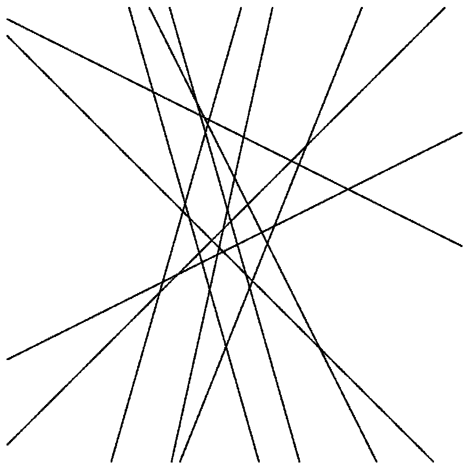

Abbildung 1.1. Ein Arrangement von 10 Geraden.

2. *Empty Convex Polygons*
   Gegeben sei mit S eine Menge von n Punkten in einer Ebene. Unsere Aufgabe besteht nun darin, das größte leere konvexe Polygon, dessen Knoten aus S stammen, zu finden. Hier bedeutet "größtes" Polygon das Polygon mit den meisten Knoten. Dieses Problem geht auf eine ungelöste Frage zurück, die von Erdős aufgestellt wurde: Es ist unbekannt ob jede hinlänglich große Menge von Punkten ein leeres Hexagon enthält. Wenn man Arrangements verwendet, kann das größte leere Polygon in $O(n^3)$ gefunden werden.

3. *Ham-Sandwich Cuts*
   Es gibt einen bemerkenswerten Satz, der besagt, dass jedes Schinken- oder Käse-Sandwich so von einer Ebene geschnitten werden kann, dass beide Hälften die gleiche Menge an Schinken bzw. Käse und Brot enthalten. Die zweidimensionale Version dieses Satzes besagt, dass immer eine Gerade existiert, die zwei Punktmengen genau halbiert. Arrangements ermöglichen es, diese Halbierung in linearer Zeit zu finden, also $O(n)$, wobei n die Anzahl der Punkte ist.

In meinem Vortrag will ich die Grundlagen von Arrangements von Geraden darstellen, aber ich werde nicht tief genug in die Materie eindringen, um alle drei Anwendungen zu erklären. Vielmehr ist es mein Ziel, die Grundlagen darzustellen und überlasse den Rest anderen Quellen.

Kapitel 2

Allgemeine Eigenschaften von Arrangements

2.1 Kombinatorisches

Definition 2.1 *Ein Arrangement von Geraden wird als* einfach *(engl: simple) bezeichnet, wenn jedes Paar von Geraden exakt einen Schnittpunkt hat und sich nie drei (oder mehr) Geraden in einem Punkt schneiden.*

Dies bedingt unter anderem, dass es in einem einfachen Arrangement keine parallelen Geraden gibt. Arrangements, die nicht zur Gruppe der einfachen Arrangements gehören, sind in mancher Beziehung "degeneriert" und oft sind Algorithmen und Sätze mit einfachen Arrangements auch am einfachsten.

Es ist bemerkenswert, dass alle einfachen Arrangements mit genau n Geraden exakt die gleiche Anzahl von Kanten, Knoten und Facetten haben.

Satz 2.1 *In einem einfachen Arrangement von Geraden gilt für Anzahl der Knoten V, Kanten E und Facetten F: $V = \binom{n}{2}$, $E = n^2$ und $F = \binom{n}{2} + n + 1$. Kein Arrangement, das nicht einfach ist, übertrifft diese Werte.*

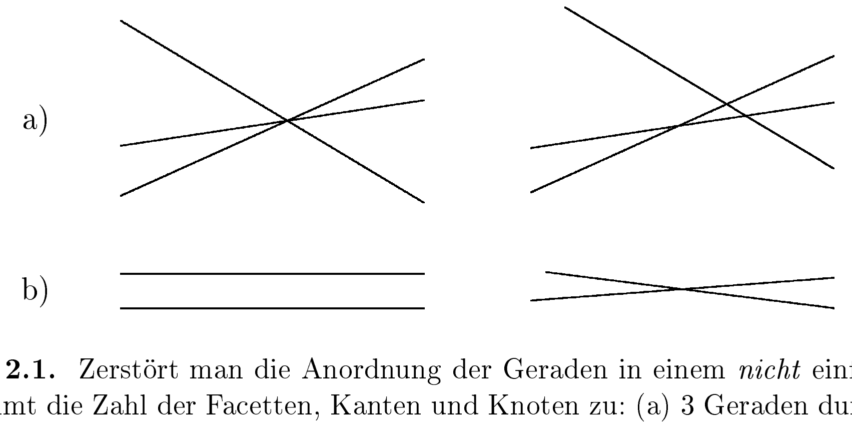

Abbildung 2.1. Zerstört man die Anordnung der Geraden in einem *nicht* einfachen Arrangement, so nimmt die Zahl der Facetten, Kanten und Knoten zu: (a) 3 Geraden durch einen Punkt (b) parallele Geraden.

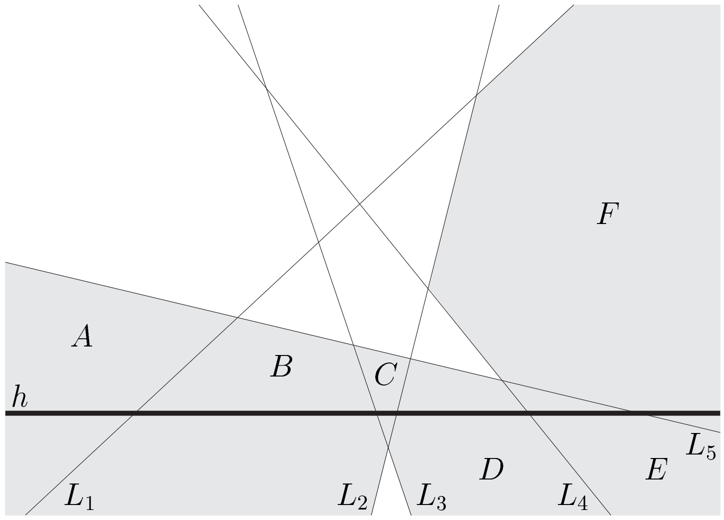

Abbildung 2.2. Die Zone von h ist $Z(h) = \{A, B, C, D, E, F\}$; $z(h) = 2+4+3+4+2+4 = 19$. Die Geraden von $\mathcal{A}$ sind nummeriert $L_1, \ldots L_5$.

Eine wichtige Erkenntnis aus diesem Satz für das Design von Algorithmen ist, dass Arrangements in der Ebene im wesentlichen quadratisch sind: V, E und F sind alle $\Theta(n^2)$.

Die kombinatorische Schlüsseleigenschaft von Arrangements, die es erlaubt, diese effizient zu konstruieren, ist, dass keine Gerade eine Zelle mit zu vielen Ecken durchdringt. Der Grund dafür, dass dies eine Schlüsseleigenschaft ist, wird (hoffentlich) klar sein, wenn wir die Notation einer "Zone" präzisiert haben. Daher werden wir das Zone Theorem einführen.

2.1.1 Das *Zone Theorem*

Sei $\mathcal{A}$ ein Arrangement von n Geraden und sei L eine andere Gerade (die für gewöhnlich nicht in $\mathcal{A}$ liegt). Wir nehmen der Klarheit halber an, dass das Arrangement $\mathcal{A} \cup \{L\}$ einfach ist. Die *Zone* von L in $\mathcal{A}$ wird mit $Z_{\mathcal{A}}(L)$ bezeichnet oder einfach mit $Z(L)$, wenn aus dem Kontext klar hervorgeht, welches Arrangement gemeint ist. Die Zone $Z_{\mathcal{A}}(L)$ ist die Menge der Zellen (Facetten), die von L durchlaufen wird. Zum Beispiel ist in Abbildung 2.2 $Z(h) = \{A, B, C, D, E, F\}$. Das Zone Theorem begrenzt die Anzahl der Kanten dieser Zellen. In der Abbildung ist die Anzahl der Kanten in den einzelnen Zellen wie folgt: $|A| = 2$, $|B| = 4$, $|C| = 3$, $|D| = 4$, $|E| = 2$ und $|F| = 4$. Gesamtzahl der Kanten in den Zellen einer Zone $Z(L)$ wird mit $z(L)$ bezeichnet. Dementsprechend ist in unserem Beispiel $z(h) = 19$. Zu bemerken ist, dass Kanten, die an zwei Zellen aus $Z(h)$, grenzen auch doppelt gezählt werden. Schließlich sei z_n der größtmögliche Wert von $Z(L)$ von allen möglichen Geraden L in einem Arrangement von n Geraden: Das größte $z(L)$ kann immer auch als Funktion von n aufgefasst werden.

Um einen kurzen Vorgriff auf Abschnitt 2.2 zu machen: man erzeugt ein Arrangement von Geraden inkrementell, indem man eine Grade nach der anderen in das Arrangement einfügt. Die Komplexität dieser Einfügeoperation wird nach oben durch z_n begrenzt, da

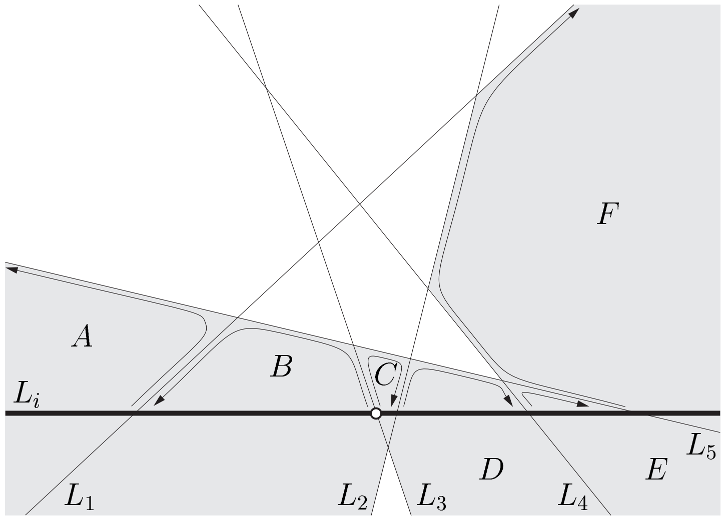

Abbildung 2.3. Das einfügen einer Gerade L_i in ein Arrangement. Die Kurven zeigen den Weg, der beim Einfügen abgelaufen wird, um die Knoten von L_i zu bestimmen.

die Kanten der Zone der eingefügten Gerade von dem Algorithmus abgelaufen werden.

Nun konzentrieren wir uns auf das Zone Theorem, das besagt, dass $z_n = O(n)$. Dies wurde zuerst von Chazelle, Guibas & Lee (1985) [2] und Edelsbrunner, O'Rourke & Seidel (1986) [3] bewiesen und seit dem sind eine ganze Reihe von alternativen Beweisen gefunden worden.

Satz 2.2 *Die Gesamtzahl der Kanten in all den Zellen, die von einer Geraden geschnitten werden, ist in einem Arrangement von n Geraden $O(n)$. Insbesondere gilt: $z_n \leq 6n$.*

2.2 Inkrementelle Konstruktion eines Arrangements

Nunmehr haben wir genug Wissen, um einen Algorithmus zur Konstruktion von Arrangements von Geraden zu diskutieren. Zuerst müssen wir uns über die Ein- und Ausgabe klar werden: Jede mögliche Repräsentation einer Geraden, wie zum Beispiel durch Stützstelle und Steigung, ist hier geeignet. Bei der Ausgabe ist dies weniger klar, hier können wir die bereits besprochenen Datenstrukturen für Polytope verwenden, wenn man leichte Modifikationen vornimmt, um den unendlichen Kanten Rechnung zu tragen. Wir wollen uns aber nicht eingehender mit Repräsentationsfragen beschäftigen, sondern lediglich annehmen, dass die Repräsentation es auf einfache Art gestattet, die Kanten einer Facette abzulaufen und sich zwischen benachbarten Facetten zu bewegen.

Der inkrementelle Algorithmus zur Konstruktion eines Arrangements (Algorithmus 1) ist angenehm einfach. In jeder gegebenen Phase haben wir ein Arrangement $\mathcal{A}_{i-1}$, das aus den ersten $i-1$ Geraden konstruiert wurde. Die Aufgabe ist es nun, alle Schnittpunkte zwischen $\mathcal{A}_{i-1}$ und L_i, der i. Eingabe-Gerade, zu finden. Ein einzelner Schnittpunkt x zwischen L_i und jeder Geraden aus $\mathcal{A}_{i-1}$ kann in konstanter Zeit gefunden werden. In unserem Beispielbild (Abbildung 2.3) ist $x = L_i \cap L_2$. Dann gehen wir entlang der Zone von L_i ($Z(L_i)$) vorwärts, indem wir die Kanten der Facetten im Uhrzeigersinn durchlaufen, bis wir wieder auf eine Kante stoßen, die L_i schneidet, um dann mit der nächsten Facette fortzufahren usw. ... . In unserem Beispiel durchlaufen wir drei Kanten von C, bevor wir auf den Schnittpunkt von L_i mit L_3 treffen, dann durchlaufen wir drei Kanten von D und so weiter wie dargestellt. Das Vorwärtsschreiten endet, wenn man auf die Facette trifft, die in Richtung des Vorwärtsschreitenes nicht mehr durch eine Kante begrenzt ist. Der Prozess wird dann von x aus in der entgegengesetzten Richtung wiederholt, dabei werden die Kanten der einzelnen Facetten entgegen dem Uhrzeigersinn durchlaufen (die Facetten A und B in der Abbildung). Jeder dieser Schritte in diesem Durchlauf bewegt sich zwischen zusammenhängenden oder benachbarten Objekten, so dass jeder Schritt eine konstante Zeit benötigt. Die Gesamtkosten der Einfügeoperation hängen von der Komplexität der Zone ab, die, wie wir im Satz 2.2 gesagt haben, $O(n)$ ist. Beachten Sie, wie die Struktur des Arrangements benutzt wird, um eine Sortierung zu erreichen. Nachdem alle Schnittpunkte mit L_i gefunden wurden oder (besser) während des Durchlaufes selbst, wird die Datenstruktur $\mathcal{A}_{i-1}$ zu $\mathcal{A}_i$ erweitert.

Es ist klar, dass die gesamte Konstruktion einen Zeitaufwand von $O(n^2)$ benötigt - ein Ergebnis, das zuerst von Chazelle und Edelsbrunner erreicht wurde.

Satz 2.3 *Ein Arrangement von n Geraden in einer Ebene kann man mit einem Zeit- und Speicheraufwand von $\Theta(n^2)$ konstruieren.*

Erzeuge $\mathcal{A}_0$, die Datenstruktur für ein leeres Arrangement;
for $i = 0$ **to** n **do**;
  Füge die Gerade L_i wie folgt ein
    Finde einen Schnittpunkt x zwischen L_i und $\mathcal{A}_{i-1}$
    Gehe entlang der Facetten in $Z(L_i)$ von x aus vorwärts
    Gehe entlang der Facetten in $Z(L_i)$ von x aus rückwärts
  Erweitere $\mathcal{A}_{i-1}$ zu $\mathcal{A}_i$

2.3 Drei und mehr Dimensionen

Einer der schönsten Aspekte der Theorie über Arrangements ist es, dass sich fast jede Eigenschaft problemlos auf höhere Dimensionen übertragen lässt. Obwohl wir dieses Thema nicht im Detail diskutieren wollen, sind die Analogien zu den Sätzen 2.1, 2.2 und 2.3 erwähnenswert.

Satz 2.4 *Die Anzahl der Facetten jeder Dimension in einem Arrangement von n Hyperebenen in der d. Dimension ist $O(n^d)$, die Zone von jeder Hyperebene hat die Komplexität $O(n^{d-1})$ und ein solches Arrangement kann man mit einem Zeit- und Speicheraufwand von $O(n^d)$ konstruieren.*

Zum Beispiel hat ein Arrangement von Ebenen im dreidimensionalen Raum die Komplexität $O(n^3)$ und diese Tatsache werden wir im Abschnitt 3 benutzen.

2.4 Dualität

Es mag sonderbar erscheinen, dass Arrangements so nützlich für Probleme mit Mengen von Punkten in der Ebene sind, wie im Beispiel 3 auf Seite 3. Der Schlüssel dazu und zu vielen anderen Anwendungen liegt in einem wichtigen Konzept, das als *Dualität* bezeichnet wird. Die Grundidee ist, dass Geraden, da sie sich durch zwei Zahlen eindeutig bestimmen lassen, mit einem Punkt assoziiert werden können, dessen Koordinaten eben diese beiden Nummern sind. Zum Beispiel kann die Gerade die durch $y = mx + b$ spezifiziert ist mit dem Punkt (m, b) assoziiert werden. Oft wird der Raum dieser Punkte als *Parameter-Raum* bezeichnet, da die Koordinaten des Punktes die Parameter der Geraden sind. Da beide Räume, sowohl der erste als auch der Parameter-Raum, äquivalente zweidimensionale Räume sind, ist es üblich (wenn auch verwirrend) sie als einen Raum zu betrachten, dessen Koordinaten zwei verschiedene Bedeutungen haben. Da die Abbildung zwischen Geraden und Punkten eindeutig ist, kann sie auch umgekehrt werden: Jeder Punkt in der Ebene kann als die Gerade, die er spezifiziert, dargestellt werden, wenn die Koordinaten interpretiert werden (z.B. als Steigung und Stützstelle). Zusammen bestimmen diese Abbildungen eine Dualität zwischen Punkten und Geraden: Jede[1] Gerade ist eindeutig einem Punkt und jeder Punkt ist eindeutig einer Gerade zugeordnet.

Basierend auf den unterschiedlichen Darstellungen einer Geraden existieren eine ganze Reihe unterschiedlicher Abbildungen für die Punkt-Geraden-Dualität. Je nach Fall hat jede ihre Vor- und Nachteile. Einige Abbildungen seien im folgenden kurz vorgestellt:

1. $L : y = mx + b \Leftrightarrow p : (m, b)$
   Diese Abbildung ist uns bereits bekannt, sie hat den Vorteil, dass sie gut mit der Definition einer Geraden durch Anstieg und Stützstelle harmoniert.

2. $L : 1 = ax + by \Leftrightarrow p : (a, b)$
   Diese Abbildung wird als sogenannte polare Dualität bezeichnet.

3. $L : y = 2ax - b \Leftrightarrow p : (a, b)$
   Dies ist die Abbildung, die wir im weiteren Verlauf des Kapitels benutzen werden. Wir benutzen das Symbol $\mathbb{D}$, um diese Abbildung zu bezeichnen: $\mathbb{D}(L) = p$ und $\mathbb{D}(p) = L$. Obwohl die gewählte Abbildung sehr ungewöhnlich aussieht, ist sie eine der geeignetsten in der Computational Geometry, hauptsächlich wegen ihrer engen Verbindung mit der parabolischen Transformation. Im folgenden untersuchen wir diese Beziehung genauer, zunächst informell und dann formal.

2.4.1 Duale Abbildungen

Die Beziehung zwischen dem Punkt $p = (a, b)$ und der Gerade $L : y = 2ax - b$ ist nicht direkt ersichtlich. Wie auch immer, $y = 2ax + a^2$ ist die Tangente der Parabel $y = x^2$ im Punkt (a, a^2). Damit ist $\mathbb{D}(p)$ mit $p = (a, b)$ und $b = a^2$ äquivalent mit dieser Tangente.

[1]Dieses "Jeder" wird im Satz 2.6 auf Seite 9 genauer definiert.

Ist $b < a^2$, so ist $\mathbb{D}(p)$ eine Gerade parallel zu dieser Tangente, die um $a^2 - b$ nach oben verschoben wurde. Ist hingegen $b > a^2$, so ist $\mathbb{D}(p)$ eine Gerade parallel zu dieser Tangente, die um $b - a^2$ nach unten verschoben wurde. Dies ist in der Abbildung 2.4 für drei Punkte mit $a = 2$ und $b \in \{0, 4, 8\}$ illustriert. Dabei sind die Punkte (wie auch im weiteren) im selben Raum wie die Geraden dargestellt.

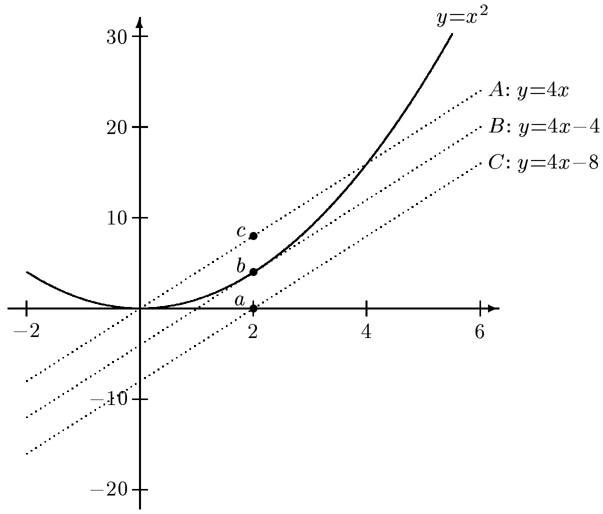

Abbildung 2.4. $\mathbb{D}(a) = A$, $\mathbb{D}(b) = B$ und $\mathbb{D}(c) = C$

Mit dieser dualen Transformation kann man jede Menge von Punkten in ein Arrangement überführen und anders herum. Ein Grund dafür, dass dies oft nützlich ist, ist dass die Beziehungen zwischen den Punkten in dem entsprechenden dualen Arrangement deutlicher hervortreten. Abbildung 2.4 zeigt die Konstruktion der zu den Geraden aus Abbildung 1.1 dualen Punkte und Abbildung 2.6 zeigt diese Punkte alleine. Dieses Beispiel wird im Abschnitt 3.4 genauer bearbeitet werden.

2.4.2 Dualitäts-Eigenschaften

In diesem Abschnitt werden einige grundlegende Eigenschaften der dualen Transformation dargelegt, die dann in den folgenden Abschnitten angewendet werden.

Satz 2.5 $\mathbb{D}$ *ist sein eigenes Inverses.* $\mathbb{D}(\mathbb{D}(x)) = x$*, wobei* x *entweder ein Punkt oder aber eine Gerade ist.*

Satz 2.6 $\mathbb{D}$ *ist eine bijektive Abbildung für alle nicht-vertikalen Geraden und alle Punkte in der Ebene.*

In den Spezialfällen, in denen vertikale Geraden notwendig sind können die Geraden ein Stück gedreht werden, so dass keine von ihnen vertikal ist. Wir schließen vertikale Geraden einfach von unseren Betrachtungen aus.

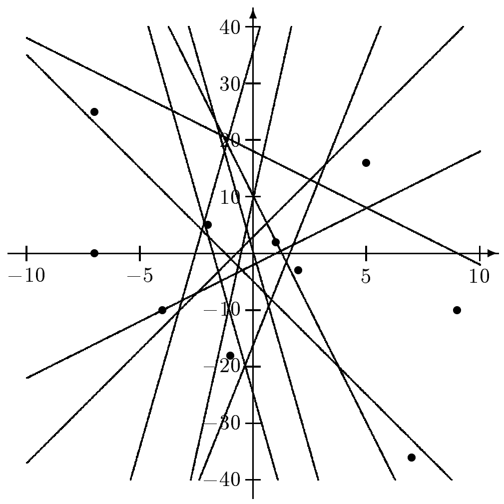

Abbildung 2.5. Die dualen Punkte/Geraden aus Abbildung 1.1

Satz 2.7 *Der Punkt p liegt genau dann auf der Geraden L, wenn der Punkt $\mathbb{D}(L)$ auf der Geraden $\mathbb{D}(p)$ liegt.*

Die Tatsache, dass zwei Punkte eine Gerade bestimmen entspricht zwei Geraden die einen gemeinsamen Schnittpunkt bestimmen.

Satz 2.8 *Die Geraden L_1 und L_2 scheiden sich genau dann im Punkt p, falls die Gerade $\mathbb{D}(p)$ durch die zwei Punkte $\mathbb{D}(L_1)$ und $\mathbb{D}(L_2)$ verläuft.*

Wenn wir vertikale Geraden von der Betrachtung ausschließen, können zwei Punkte eindeutig mit oberhalb und unterhalb in Beziehung gesetzt werden. Die duale Abbildung

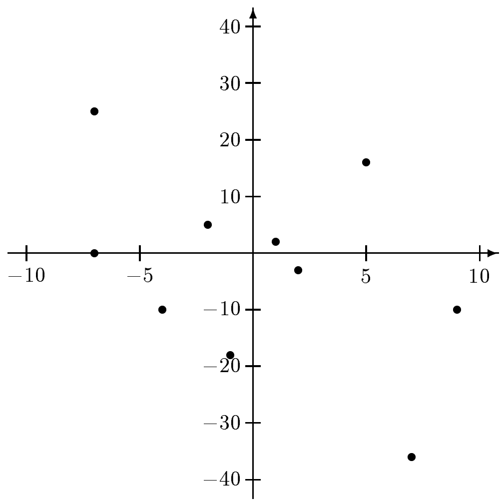

Abbildung 2.6. Die Punkte aus Abbildung 2.5: $\{(-7,25),\ (-7,0),\ (-4,-10),\ (-2,5),\ (-1,-18),\ (1,2),\ (2,-3),\ (5,16),\ (7,-36),\ (9,-10)\}$

kann genutzt werden, um eine vertikale Ordnung wie folgt zu definieren:

Satz 2.9 *Wenn der Punkt p oberhalb der Gerade L liegt, dann liegt die Gerade $\mathbb{D}(p)$ unterhalb des Punktes $\mathbb{D}(L)$. Entsprechendes gilt auch, wenn p unterhalb von L liegt, dann liegt $\mathbb{D}(p)$ oberhalb von $\mathbb{D}(L)$.*

Dies kann man deutlich in Abbildung 2.4 erkennen. Zum Beispiel liegt der Punkt c oberhalb der Gerade B und die Gerade C unterhalb des Punktes b.

Kapitel 3

Anwendungen

3.1 Hidden Surface Removal

Sicherlich gibt es heute kaum eine geometrische Berechnung, die öfter als das *Hidden Surface Removal* angewendet wird, da es die Grundlage für die 3D-Grafik am Computer darstellt. Wir gehen bei den weiteren Betrachtungen dieses Themas davon aus, dass wir von einem bestimmten Punkt aus auf eine Menge flacher, undurchsichtiger und farbiger Polygone schauen. Aus dem jeweiligen Erscheinungsbild der Polygone wird dann das Bild oder die "Szene" berechnet. Oft sind die Polygone miteinander verbunden und bilden eine Oberfläche, deren abgewendete Teile verdeckt sind und vor der Berechnung entfernt werden müssen (um Rechenzeit zu sparen).

Sei n die Anzahl der Knoten des Eingabepolygons. Man kann zeigen, dass die Komplexität der Ausgabeszene in etwa bei $\Omega(n)$ liegen kann. Ein Gitter, gebildet aus vertikalen Rechtecken, die durch horizontale Rechtecke teilweise verdeckt werden, führt zu $> \frac{1}{16}n^2$ Knoten in der Szene, wie in Abbildung 3.1 dargestellt. Wenn wir als Ausgabe eine Liste von Polygonen verlangen, gibt es keinen Algorithmus, der quadratische Zeit im Worst-Case unterbieten kann. Viele Algorithmen erreichen $O(n^2 \log n)$, nur $O(\log n)$ langsamer als optimal, indem zu bestimmten Zeitpunkten eine Sortierung mit einem Aufwand von $O(n \log n)$ vorgenommen wird . Die Theorie der Arrangements führte schließlich zu einem Worst-Case $O(n^2)$ Algorithmus (McKenna (1987) [5]), den ich nun vorstellen will.

Zunächst nehmen wir an, dass sich die Polygone nicht gegenseitig durchdringen: Ihr Inneres ist disjunkt, obwohl sie über gemeinsame Randpunkte verfügen können. Des weiteren nehmen wir an, dass der Standpunkt des Betrachters (Viewpoint) unendlich weit von der Szene entfernt ist, so dass alle Geraden des Sichtbereiches parallel verlaufen und wir uns nicht mit den Komplikationen der Perspektive beschäftigen müssen. Obwohl es nicht offensichtlich ist, kann jede Szene mit einem endlichen Viewpoint in eine Szene mit einem unendlichen überführt werden, so das dies kein Verlust an Allgemeingültigkeit ist. Der Standpunkt des Betrachters sei bei $(0, 0, \infty)$; damit ist die Bildebene die xy-Ebene, $z = 0$. Es ist im weiteren günstig, ein großes Hintergrundpolygon unter all den anderen zu platzieren, so dass alle Sichtlinien auf dieses Polygon treffen.

Der erste Schritt ist, alle Kanten der Eingabepolygone in die xy-Ebene zu projizieren

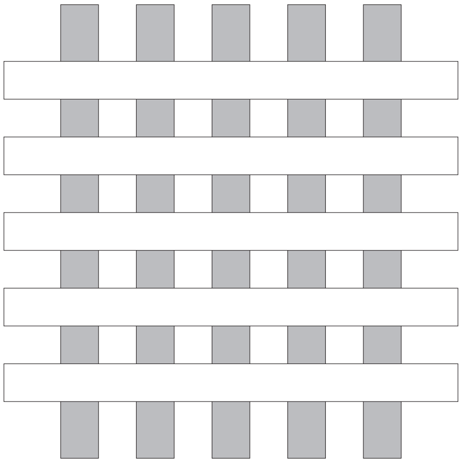

Abbildung 3.1. Ein Gitter aus sich überkreuzenden Rechtecken erzeugt eine $\Omega(n^2)$ Untergrenze für die Komplexität der Ausgabe.

(indem die z-Koordinaten ihrer Endpunkte ignoriert werden). Dies wird als *orthographische Projektion* bezeichnet (im Gegensatz zur perspektivischen Projektion). Als nächstes wird jede Kante zu einer Geraden erweitert, die diese Kante enthält. Das Ergebnis ist ein Arrangement $\mathcal{A}$ von n Geraden in der xy-Ebene, das in $O(n^2)$ Zeit konstruiert werden kann (siehe Satz 2.3). Nunmehr ist es die Aufgabe, für jede Zelle von $\mathcal{A}$ zu entscheiden, welches Polygon, aus der Projektion, das höchste im Raum ist und damit am nächsten am Betrachter ist. Ist dies bekannt, so können die Zellen entsprechend der Färbung des Polygons (und seiner Position falls Shading erwünscht ist) gezeichnet werden.

Ein einfacher Algorithmus würde ungefähr $O(n^3)$ Zeit benötigen: Für jede der $O(n^2)$ Zellen muss die Höhe jedes der $O(n)$ Polygone berechnet werden. Die Herausforderung ist, nun nur noch konstante Zeit für jede Zelle zu verbrauchen.

McKenna's Algorithmus verwendet ein modifiziertes Plane-Sweep Verfahren. Statt eine senkrechte Linie über das Arrangement zu bewegen, verwenden wir eine vertikale "Pseudolinie" L, eine Kurve, die jede Gerade von $\mathcal{A}$ genau einmal schneidet. Der Vorteil, die Linie "biegsam" zu machen, liegt darin, dass es dann unnötig ist, $O(\log n)$ Zeit in einer Priority-Queue zu verbringen, um zu bestimmen, welcher Knoten der nächste ist. Vielmehr kann man eine unsortierte Liste von Knoten, die "sweepable" sind, verwenden. Diese Liste enthält alle Knoten, die mit mindestens zwei Kanten verbunden sind, die von L geschnitten werden. In der Abbildung 3.2 ist der Knoten v "sweepable", da die beiden Kanten der Zelle C, die von L geschnitten wird, mit v benachbart sind.

Die von dem Algorithmus verwendeten Datenstrukturen sind:

- Das feste Arrangement, das eine Liste der *aktiven* Zellen und Kanten enthält. Das sind die Zellen und Kanten, die von L geschnitten werden (so wie die Zelle C in der Abbildung 3.2).

- Für jede aktive Zelle C eine Liste aller Polygone deren Projektionen C enthält, sortiert nach der z-Tiefe. Diese Listen müssen nur für die aktiven Zellen angelegt

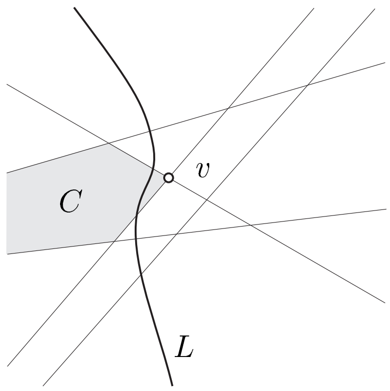

Abbildung 3.2. v ist “sweepable”

werden, und da L n Geraden schneidet, gibt es immer genau $n - 1$ Listen.

Es dürfte klar sein, dass diese Listen genug Informationen bereitstellen, um das vorderste Polygon für jede Zelle von $\mathcal{A}$ zu ermitteln. Ich möchte nicht den Ablauf des gesamten Algorithmus beschreiben, sondern nur auf ein Feature des Algorithmus eingehen. Wird ein Knoten “gesweept”, “sterben” die alten Zellen und neue werden aktiv, aber ihre Listen, die die Polygone enthalten bleiben, gleich oder ändern sich nur geringfügig. Diese Übereinstimmung kann ausgenutzt werden, um Kosten für das Update der Listen in etwa bei konstanter Zeit zu halten, die sich über alle Zellen von $\mathcal{A}$ gesehen amorphisiert. Das Ergebnis ist ein Hidden-Surface-Algorithmus mit einer Laufzeit von $O(n^2)$ im Worst-Case.

In der Praxis ist dies nicht unbedingt der beste Algorithmus, da er immer einen Zeit- und Speicheraufwand von $\Omega(n^2)$ hat. In der Praxis werden im Allgemeinen Szenen mit einer eher geringeren Komplexität verwendet. Da es aber bei Szenen von hoher Qualität nicht ungewöhnlich ist, dass n etwa bei 10^6 liegt, ist es hier wichtig, quadratischen Zeitaufwand, wenn immer möglich, zu vermeiden. Algorithmen, deren Geschwindigkeit von der Komplexität der Ausgabeszene abhängt werden als *Output Size Sensitiv* Hidden Surface Algorithmen bezeichnet und sind Gegenstand bedeutender aktueller Forschungen.

3.2 Aspect Graphs

In den späten 70-ern führten Forscher auf dem Gebiet der Computergrafik einen sogenannten *Aspect Graph* ein, um die Bilderkennung zu unterstützen. Die Idee ist es, alle charakteristischen Ansichten (*engl: characteristic views*), die ein Objekt einem Betrachter bietet, zu speichern und es dann mit dem zu vergleichen, was man gerade sieht. Für ein polyedrisches Objekt wird diese charakteristische Ansicht durch kombinatorische Äquivalenz bestimmt: Zwei Betrachter sehen die gleiche Ansicht eines Polyeders, wenn das Bild die gleiche kombinatorische Struktur hat, das heißt, dass der (nummerierte) ebene Graph, der durch die Projektion des Polyeders auf die Sichtebene entsteht, der gleiche ist. Die sogenannte *Visual Space Partition* (VSP) ist eine Partition des gesamten Raumes außerhalb eines Objektes in zusammenhängende Regionen oder Zellen, von denen aus das

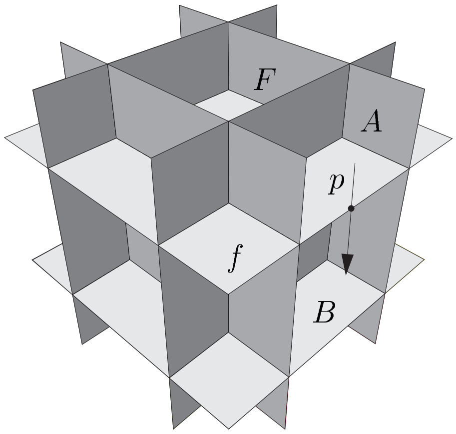

Abbildung 3.3. Ein Arrangement von Ebenen, das die Flächen eines Würfels enthält. p kann F von A aus sehen, aber nicht von B aus.

zu betrachtende Objekt die gleiche Ansicht bietet.

Arrangements bieten ein sauberes Gerüst, um VSP's (und damit auch Aspect Graphs) für konvexe Polytope, einem wichtigen Sonderfall, zu verstehen. Für ein Polytop P ist das VSP genau das Arrangement, das von den Ebenen gebildet wird, die die Facetten von P enthalten (Plantinga & Dyer 1990) [?]. Zum Beispiel partitioniert ein Würfel den Raum in 26 unbegrenzte Regionen wie in Abbildung 3.3 dargestellt, das sind:

- sechs rechteckige Zylinder die auf den Flächen des Würfels basieren
- acht Oktanten, an jedem Knoten des Würfels einer
- zwölf "Keile", an jeder Kante des Würfels eines

Angenommen, Sie nehmen ein Bild des Würfels von einem Punkt p aus auf. Der Punkt p liegt dabei in einer Zelle A und Sie schauen in Richtung einer benachbarten Zelle B durch eine Facette f des Arrangements, wie es in Abbildung 3.3 dargestellt ist. Die Fläche F des Würfels, in dessen Ebene die Facette f des Arrangements liegt, ist von der Zelle A aus sichtbar. Bewegen wir nun p in Richtung B. Liegt p in f, so sieht man von der Fläche F nur noch die Kante. Liegt p schließlich in B, so ist F nicht länger sichtbar. Damit repräsentiert f eine Transition in der Ansicht(ein Übergang von einer Ansicht in eine andere).

Aus Satz 2.4 ist bekannt, dass das VSP eines konvexen Polytops die Größe $O(n^3)$ hat und in $O(n^3)$ Zeit konstruiert werden kann. Der Aspect Graph kann dann durch eine Überführung der Repräsentation des VSP erzeugt werden. Der Aspect Graph kann auch für nicht konvexe Polyeder definiert werden, allerdings erhöht sich die Komplexität dann auf $O(n^9)$.

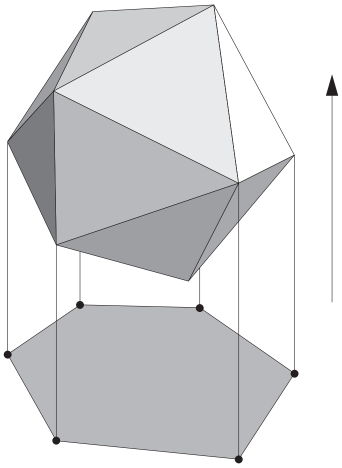

Abbildung 3.4. Der Schatten eines Polytops von einer unendlich weit entfernten Lichtquelle ist ein konvexes Polygon.

3.3 Kleinster Polytop-Schatten

Befassen wir uns nun mit dem Problem, die kleinste Schattenfläche zu finden, die von einem gegebenen Polytop P geworfen wird, wenn die Lichtquelle unendlich weit entfernt ist und die Ebene, auf die der Schatten geworfen wird, senkrecht zu der Linie Lichtquelle-Polytop steht. Dieses Problem wurde zuerst von McKenna & Seidel (1985) [?] und McKenna (1989) [?] untersucht, die eine Lösung angaben, die auf Arrangements basiert. Ich will auf die Verwendung der Arrangements eingehen, ohne diese Lösung vollständig zu erklären.

Der grundlegende Gedanke ist der gleiche wie bei den Aspect Graphs: Die kombinatorische Struktur des Schattens ändert sich, wenn Standpunkt des Betrachters eine Ebene durchquert,AuthentizierungenAuthentizierungen die eine Fläche von P enthält. Was dieses Problem von der VSP-Konstruktion unterscheidet ist, dass sich die Lichtquelle/Viewpoint unendlich weit weg befindet, so dass die Projektion unabhängig von der Perspektive immer orthographisch ist. Von einem Viewpoint, der unendlich weit weg liegt, schrumpft P letztendlich zu einem Punkt und alle Ebenen der Flächen des Polyeders enthalten diesen Punkt.

Sei nun π_f die Ebene, die parallel zu der Facette f des Polytops P durch den Koordinatenursprung verläuft. Sei $\mathcal{A}$ das Arrangement von Ebenen, das gebildet wird, wenn man für jedes f von P π_f bestimmt. $\mathcal{A}$ unterteilt den Raum in unbegrenzte Kegel, deren Spitzen im Ursprung liegen. Ein Vector u repräsentiert die Richtung der Lichtstrahlen, die in die Kegel fallen. Der Kegel bestimmt die kombinatorische Äquivalenzklasse der Ansicht von dem unendlich weit entfernten Punkt in Richtung u und damit auch die kombinatorische Struktur des Schattens auf der Ebene orthogonal zu u (siehe Abbildung 3.4).

Obwohl die kombinatorische Struktur des Schattens für jeden Richtungsvektor im Inneren eines Kegels konstant ist, ist die Fläche des Schattens nicht konstant. McKenna

& Seidel bewiesen das die kleinste Fläche für einen Kegel jeweils dann erreicht wird, wenn der Richtungsvektor entlang der Kanten von $\mathcal{A}$ verläuft, dass heißt entlang einer Schnittgeraden von zwei Ebenen, die den Kegel begrenzen.

Jedoch ist $\mathcal{A}$ ein Arrangement von Ebenen und hat damit eine Größe von $O(n^3)$. Das ist natürlich ziemlich abartig, da alle Ebenen den Koordinatenursprung enthalten. In Wirklichkeit liegt die Größe des Arrangements aber nur bei $O(n^2)$, wie die nun folgende Ausführung zeigt.

Man schneidet $\mathcal{A}$ mit einer Ebene π, die parallel zur xy-Ebene verläuft, zum Beispiel $\pi : \quad z = 1$. Es sollte klar sein, dass $\mathcal{A} \cap \pi = \mathcal{A}'$ ein Arrangement von Geraden ist. Jeder Richtungsvektor u entspricht dem Punkt in π, der durch den Schnitt von π mit der Geraden, in der u liegt, entsteht. Damit gibt es eine eindeutige Abbildung zwischen den Viewpoints im unendlichen und den Punkten in dem zweidimensionalen Arrangement $\mathcal{A}'$, das die Komplexität $O(n^2)$ hat. Und letztendlich korrespondiert ein Viewpoint, der einen minimalen Schatten erreicht, mit einem Knoten von $\mathcal{A}'$.

Nunmehr verfügen wir über einen Algorithmus.

1. Konstruiere $\mathcal{A}'$ in $O(n^2)$ Zeit (Es besteht ja keine Veranlassung, $\mathcal{A}$ zu konstruieren).
2. Berechne für jede der $O(n^2)$ Kanten die Fläche des Schattens auf der Ebene senkrecht zu der Richtung, die durch den Knoten vorgegeben wird.
3. Gib die kleinste Schattenfläche aus.

Was übrig bleibt, ist die Berechnung der Schattenfläche für jeden Knoten von $\mathcal{A}$, und das ist der Teil, auf den ich nicht eingehen möchte. Es existiert eine clevere Methode, die es vermeidet die Schattenfläche jedes Mal neu zu berechnen und damit in etwa konstante Zeit pro Knoten verbraucht. Damit kann der gesamte Algorithmus etwa in $O(n^2)$ Zeit abgearbeitet werden.

3.4 Ham-Sandwich Cuts

Wir wollen uns nunmehr damit beschäftigen, wie sich Arrangements nutzen lassen, um Ham-Sandwich Cuts für voneinander getrennte Punktmengen zu finden. Doch zunächst eine Definition:

Definition 3.1 *Ein* Bisektor *einer Punktmenge ist eine Gerade, die diese Punkmenge so teilt, dass auf jeder Seite genau die Hälfte der Punkte liegt.*

Der Einfachheit halber wollen wir noch ein paar Einschränkungen treffen. So gehen wir davon aus, dass keine drei Punkte auf einer Geraden liegen. Das weiteren sollen unsere Mengen immer eine ungerade Anzahl von Punkten enthalten, so dass letztendlich der Bisektor einer solchen Menge immer einen ihrer Punkte enthält.

Behandeln wir zunächst die Bisektion einer einfachen Menge von n Punkten. Durch unsere oben getroffenen Einschränkungen wird eine Menge niemals nur über senkrechte

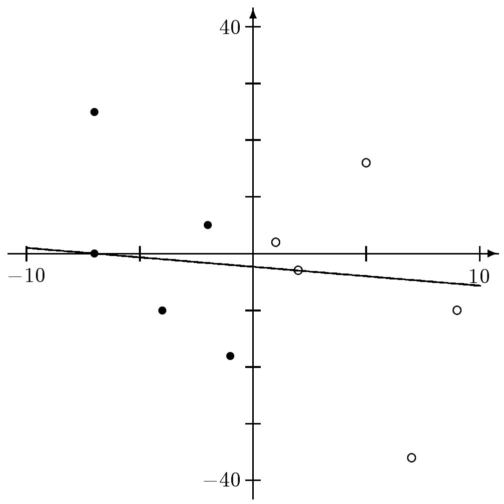

Abbildung 3.5. Zwei Mengen von je fünf Punkten (aus Abbildung 2.6): A rechts von der y-Achse und B links davon. Die eingezeichnete Gerade ist ein Ham-Sandwich Cut, der A und B teilt.

Bisektoren verfügen, so dass wir sie ignorieren können. Aus den Punkten der Menge A können wir durch die Abbildung $\mathbb{D}$ ein Arrangement $\mathcal{A}$ mit n Geraden erzeugen. Wir gehen nun darauf ein, dass die Bisektoren von A genau die dualen Geraden zu den Punkten aus dem *Median Level* $M_{\mathcal{A}}$ von A entsprechen. Das Median Level ist die Menge der Kanten von $\mathcal{A}$ (und der sie verbindenden Knoten), deren Punkte genau $(n-1)/2$ Geraden vertikal über sich haben (und genau die gleiche Anzahl unter sich). Mit Satz 2.9 gilt, dass jeder Punkt aus $p \in M_{\mathcal{A}}$ eine duale Gerade $\mathbb{D}(p)$ besitzt, die die gleiche Anzahl von Punkten unter sich hat wie p Geraden über sich. Da laut Definition genau $(n-1)/2$ Geraden über p liegen, hat $\mathbb{D}(p)$ genau $(n-1)/2$ Punkte unter sich: Das heißt $\mathbb{D}(p)$ teilt A in zwei gleich große Hälften. Damit ist $\mathbb{D}(p)$ genau dann ein Bisektor, wenn $p \in M_{\mathcal{A}}$.

Satz 3.1 *Für die Bisektoren $\mathbb{D}(p)$ einer Menge P von Punkten gilt, dass p ein Element des Medial Levels des zu den Punkten dualen Arrangements $\mathcal{A}$ ist ($p \in M_{\mathcal{A}}$).*

Mit diesem Satz gilt, dass eine Gerade, die ein Ham-Sandwich Cut für zwei Punktmengen A und B ist, die duale Abbildung eines Punktes sein muss, der sowohl in $M_{\mathcal{A}}$ als auch in $M_{\mathcal{B}}$ enthalten ist (wobei $\mathcal{B}$ das duale Arrangement zu B ist). Daher können alle Ham-Sandwich Cuts gefunden werden in dem man $M_{\mathcal{A}}$ und $M_{\mathcal{B}}$ miteinander schneidet.

Die beiden Median Levels können natürlich auf komplizierte Art und Weise mathematisch geschnitten werden, aber die Situation ist wesentlich einfacher, wenn sich die beiden Punktmengen durch eine Gerade voneinander trennen lassen (wie es in den meisten Anwendungen der Fall ist). Seien nun A' und B' zwei durch eine Linie separierbare Punktmengen. Diese Punktmengen können durch eine einfache Translation und Rotation in zwei Punktmengen A und B überführt werden, die durch die y-Achse voneinander getrennt sind (A rechts und B links), siehe Abbildung 3.5 für ein Beispiel. Nun wenden wir die duale Abbildung $\mathbb{D}$ auf beide Mengen an. Die Geraden im Arrangement $\mathcal{A}$ haben alle

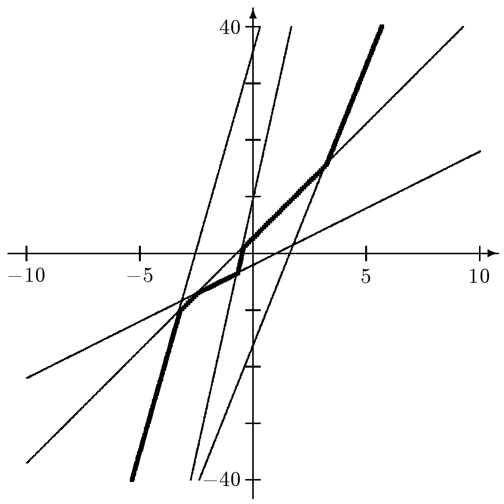

Abbildung 3.6. Die zu den Punkten mit einer x-Koordinate größer Null dualen Geraden (Menge A in Abbildung 3.5) haben alle einen positiven Anstieg.

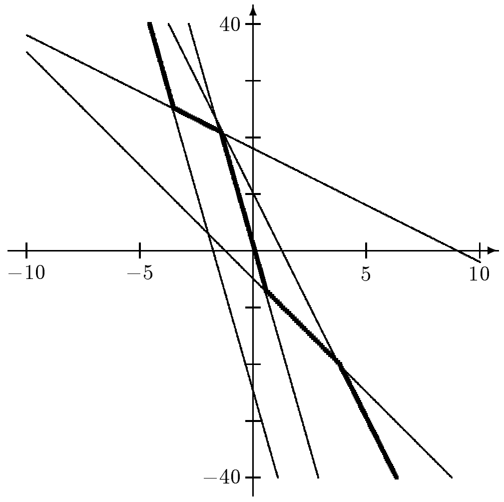

Abbildung 3.7. Die zu den Punkten mit einer x-Koordinate kleiner Null dualen Geraden (Menge B in Abbildung 3.5) haben alle einen negativen Anstieg.

einen positiven Anstieg (siehe Abbildung 3.6), während die Geraden im Arrangement $\mathcal{B}$ alle einen negativen Anstieg haben (siehe Abbildung 3.6).

Da $M_{\mathcal{A}}$ aus Segmenten von Geraden zusammengesetzt ist, die alle einen positiven Anstieg haben ist $M_{\mathcal{A}}$ streng monoton wachsend und $M_{\mathcal{B}}$ dementsprechend streng monoton fallend (beide sind in den Abbildungen 3.6 und 3.7 schattiert eingezeichnet). Sie treffen sich daher sie sich in einem einzelnen Punkt: Der Ham-Sandwich Cut ist eindeutig. Wie Abbildung 3.8 zeigt, liegt der Schnittpunkt in unserem Beispiel bei $(-\frac{1}{6}, \frac{7}{3})$ und demzufolge ist die Gerade $y = -\frac{2}{6}x - \frac{7}{3}$ der entsprechende Ham-Sandwich Cut.

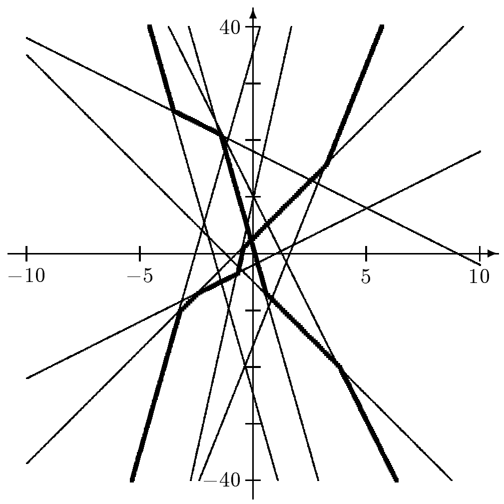

Abbildung 3.8. $\mathcal{A}$ und $\mathcal{B}$ zusammen. Der Schnittpunkt ihrer Median Levels liegt bei $(-\frac{1}{6}, \frac{7}{3})$.

Dieser Schnittpunkt kann gefunden werden, ohne dass ein Arrangement aufgebaut werden muss. Der Zeitaufwand für diesen Algorithmus beträgt $O(n+m)$, wenn n und m die Anzahl der Punkte in den beiden Mengen ist. Allerdings ist dieser Algorithmus ziemlich kompliziert und ich will ihn hier nicht wiedergeben (Edelsbrunner 1987) [4]. Außerdem kann der selbe lineare Zeitaufwand auch für Punktmengen erreicht werden die nicht durch Geraden voneinander getrennt sind (Lo, Mantǒusek & Steiger 1994). Dies führt zu einem Algorithmus für ein interessantes Matching-Problem, das zuerst von Atallah (1985) [1] untersucht wurde und auf das ich nun eingehen will.

3.4.1 Red-Blue Matching

Es seien n rote und n blaue Punkte in der Ebene gegeben. Die Aufgabe ist es, sie in Paare von je einem roten und einem blauen Knoten einzuteilen, ohne dass sich die Kanten zwischen den Paaren schneiden. Die Punkte könnten zum Beispiel die Knoten eines Objektes sein, das zwischen zwei aufeinander folgenden ("roten" und "blauen") Frames verschoben wurde: Das Matching repräsentiert dann die Translation. Dieser Spezialfall des Problems verlangt nicht nur, dass die Kanten sich nicht überkreuzen, sondern auch dass ihre Längen übereinstimmen. Andere Anwendungen haben nicht solche starken Einschränkungen.

Ein allgemeines Red-Blue-Matching-Problem kann durch einen Divide-and-Conquer Algorithmus gelöst werden, der Ham-Sandwich Cuts verwendet. Gehen wir von einer Menge von $n = 6$ roten und blauen Knoten aus, wie in Abbildung 3.9(a). Zunächst wird die Menge durch einen Ham-Sandwich Cut in zwei Mengen von je drei roten und blauen Punkten geteilt (durch die Gerade 1 in Abbildung 3.9(b)). Jede dieser Mengen wird nun erneut geteilt, so dass jede der Mengen einen roten und blauen Punkt enthält und je ein roter und ein blauer Punkt auf dem Schnitt liegt. Das je ein roter und blauer Punkt auf den Schnitt liegt ist klar, da die geteilte Menge eine ungerade Anzahl von roten bzw.

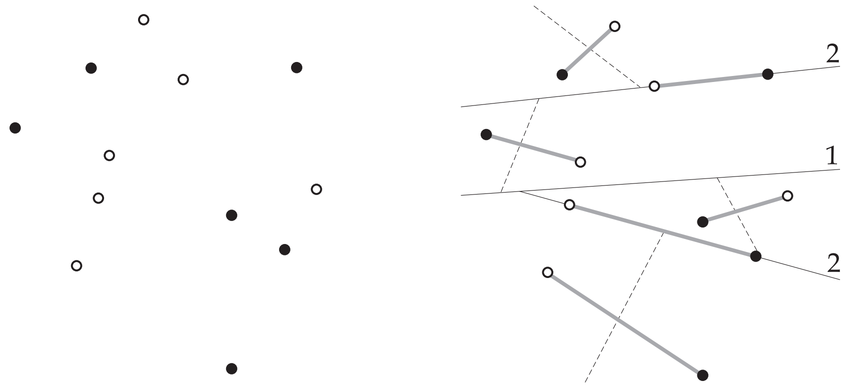

Abbildung 3.9. (a) $n = 6$ rote (gefüllte) und 6 blaue (leere) Punkte (b) Die Aufteilung in sich nicht überkreuzende Matchings nach der Ham-Sandwich-Aufteilung.

blauen Knoten enthielt. Schließlich werden die verbliebenen Mengen noch einmal geteilt. Nunmehr ist es trivial, die Knoten durch sich nicht überschneidende Kanten zu verbinden. Die letzte Aufteilung ergibt direkt das Matching der durch sie getrennten Punkte und die anderen Matchings liegen auf den Schnitten, die Punkte enthalten. Es dürfte klar sein, dass sich keine Matchings überschneiden können.

Der Zeitaufwand dieses Algorithmus liegt bei $O(n \log n)$. Es wird lineare Zeit für jeden der $O(\log n)$ Schritte benötigt.

3.4.2 Höhere Dimensionen

Das Ham-Sandwich Problem kann analog auf höhere Dimensionen übertragen werden.

Satz 3.2 *In einem d-dimensionalen Raum existiert eine Hyperebene, die d paarweise disjunkte Mengen P_1, … P_n von Punkten so teilt, dass jedes P_i in zwei exakt gleich große Mengen geteilt wird.*

Literaturverzeichnis

[1] Atallah, M. J. (1985), *A matching problem in the pane*, J. Comput. Syst. Sci. **31**, 63-70.

[2] Chazelle, B., Guibas, L. J. & Lee, D. T. (1985), *The power of geometric duality*, BIT **25**, 70-90.

[3] Edelsbrunner, H., O'Rourke, J. & Seidel, R. (1986), *Constructing arrangments of lines and hyperpanes with applications*, SIAM Journal on Computing **15**, 351-63.

[4] Edelsbrunner, H. (1987), *Algorithms in Combinatorial Geometry*, volume 10 of EATCS Monographs on Theoretical Computer Science, SpringerVerlag, Heidelberg.

[5] McKenna, M. (1987), *Worst-Case optimal hidden-surface removal*, ACM Trans. Graph. **6**, 19-28.

[6] J. O'Rourke (1998), *Combinatorial Geometry in C (Second Edition)*, Cambridge University Press.

Index

A

Arrangements 2
  einfache 4
Aspect Graph 14

B

Bisektor 17

D

Dualität 8

E

Empty Convex Polygons 3

F

Facetten 2

H

Ham-Sandwich Cuts 3
Hidden Surface Removal 2, 12

K

Kanten 2
Knoten 2

M

Median Level 18

O

orthographische Projektion 13

P

Parameter-Raum 8

S

Segmente 2

V

Visual Space Partition 14

Z

Zellen 2
Zone 5
Zone Theorem 5