Implementierung Geometrischer Algorithmen

Berechnung von Voronoi Diagrammen für Liniensegmente

Jörg Reiberg

04.07.1995

1 Einleitung

Ein Voronoi-Diagramm für eine Menge S von Objekten, auch Sites genannt, unterteilt eine Ebene in eine Menge von Regionen, für jede Site genau eine. Alle Punkte dieser Regionen haben zur zugehörigen Site einen geringeren Abstand, als zu allen anderen Sites aus S. In Abb.1 sind die Bisektoren (s.3.1) und das resultierende Voronoi-Diagramm für einen Punkt und zwei Liniensegmente dargestellt.

Mit Punkten und Liniensegmenten als Sites ergeben sich komplexere Bisektoren als bei Point-Sites (Abb.7). Daher sind auch aufwendigere Abstandsberechnungen notwendig. Zur Berechnung von Voronoi-Diagrammen für Liniensegmente gibt es Algorithmen, die auf Methoden wie Devide & Conquer,

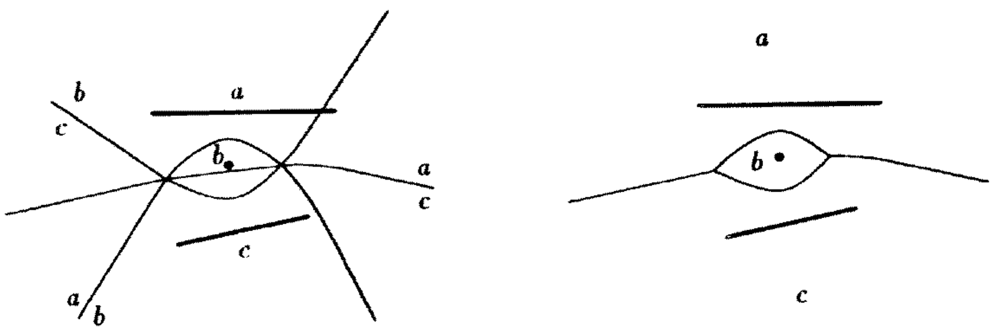

Abbildung 1: Bisektoren und Voronoi Diagramm für 2 Segmente und 1 Punkt.

Plane Sweep und Randomized Incremental Construction basieren. Die Laufzeit dieser Algorithmen, die in vielen Arbeiten ausführlich diskutiert wird, ist $O(n \log n)$. Ich möchte darauf in diesem Paper nicht genauer eingehen, sondern die Robustheit der Implementierung untersuchen, die grundsätzlich ein Problem darstellt, da Rechner nicht über exakte reelle Arithmetik, wie sie die meisten Algorithmen voraussetzen, verfügen.

Die Grundlage für die Berechnung von Voronoi-Diagrammen ist der *Incircle Test*, der im Abschnitt 2.3 besprochen wird. Für dessen Implementierung, gibt es drei Ansätze:

- Floating Point Arithmetik mit ϵ-Umgebungstest
- Floating Point Arithmetik mit Sonderbehandlung bei Nulltests
- Exakte Arithmetik mit beliebiger Genauigkeit

Erstere Ansätze liefern bei vielen Beispielen richtige Ergebnisse. Es gibt aber Konfigurationen, bei denen die Algorithmen fehlschlagen. Wir wollen daher den letzten Ansatz besprechen, der für den randomisierten Algorithmus von Klein et al. [KMM90] genauer studiert wird. Es soll untersucht werden, wie man bei notwendigen Abstandsbestimmungen die Präzision beliebig genauer Arithmetik einschränken kann, ohne daß das Ergebnis falsch wird, dabei wird gezeigt, daß statt einer üblichen Präzision von 9000k Stellen für die exakte Berechnung des Incircle-Tests nur 48k Stellen notwendig sind, was zu folgendem Vorgehen führt:

Es werden mit BIGNUMS[1] und wiederholtem Quadrieren (s.3) gerechnet, oder mit BIGFLOATS[2] die ersten 48k Stellen zweier zu vergleichender Zahlen berechnet. Falls sich die zu vergleichenden Zahlen unterscheiden, so tun sie dies in der Regel nicht in der letzten Stelle und deswegen sollte im Normalfall eine viel geringere Genauigkeit ausreichen. Falls die Zahlen jedoch gleich sind, so muß natürlich mit höchster Genauigkeit gerechnet werden. Dies ist besonders oft der Fall, wenn die Eingabe hochgradig degeneriert ist.

In einer detailierteren Version des Reports von Burnikel et al., der diesem Paper zugrundeliegt, wird gezeigt, daß ein Null-Test für Summen von Radikalen statt einer Präzision von 100k Stellen, wie sie der Algorithmus von Blömer benötigt, nur 12k Stellen erfordert. Dies führt zu folgender Alternative:

Es wird zunächst mit 12k Integers getestet, ob eine vierte Site den Inkreis

[1] Bibliotheken zur Implementierung großer Ganzzahlen

[2] Bibliothek zur implementierung sehr genauer Fließkommazahlen

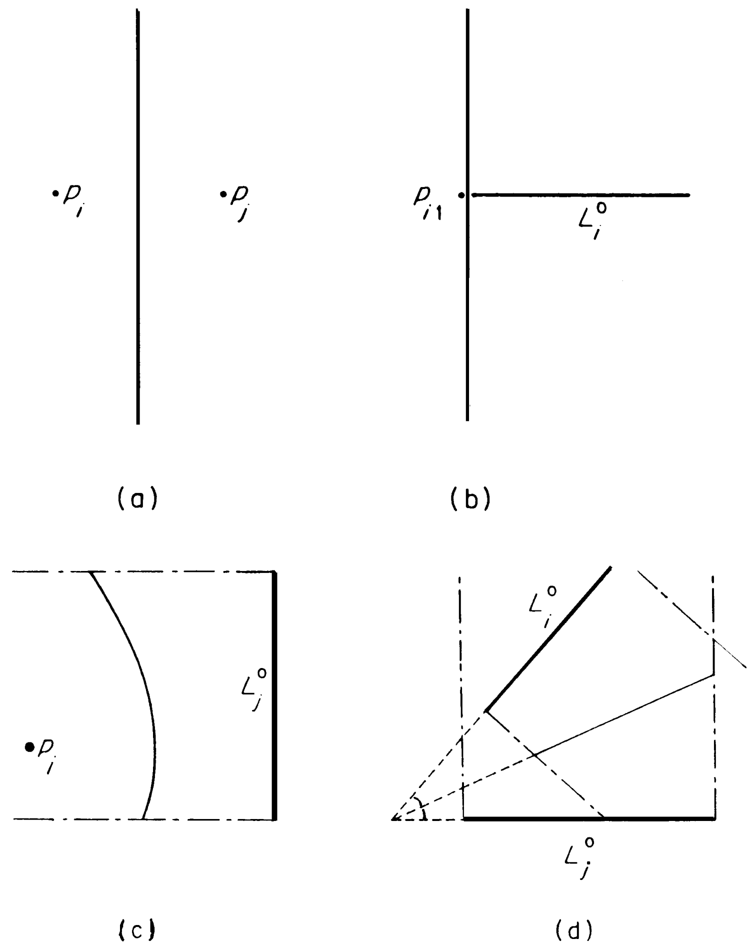

Abbildung 2: Voronoi-Kanten: a) Typ E1 b) Typ E2 c) Typ E3 d) Typ E4

dreier Sites berührt (Nulltest). Ist dies nicht der Fall, so weiß man, das sich die zu vergleichenden Werte unterscheiden und rechnet mit k, 2k, 4k, ... 48k weiter bis sie verschieden sind.

Diese Alternative wird jedoch nur in den experimentellen Ergebnissen aufgeführt, die abschließend im letzten Teil vorgestellt und analysiert werden.

2 Berechnung von Voronoi Diagrammen

2.1 Voronoi-Ecken, -Kanten und -Regionen

Kanten in einem Voronoi-Diagramm für Liniensegmente sind Linien oder parabolische Kurven. Die Kanten können in folgende vier Typen (s.Abb.2)

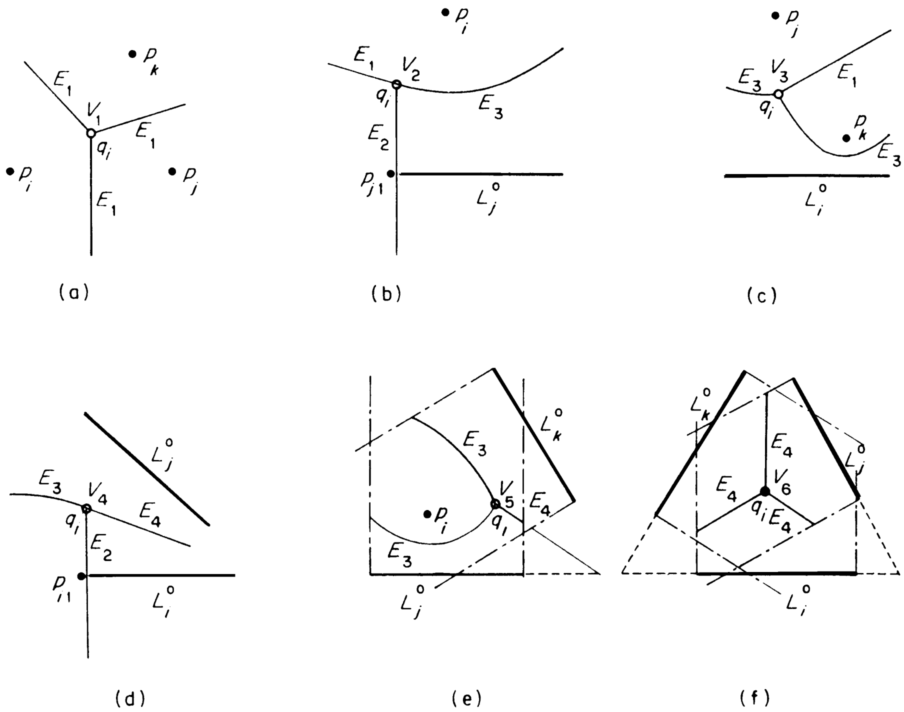

Abbildung 3: Voronoi-Ecken: a-f $\simeq$ Typen V1-V6

eingeteilt werden [BOS92]:

- Typ E1
  Eine Kante, die durch zwei Punkte p_i, p_j definiert wird. Die Kante ist eine Gerade, die $\overline{p_i p_j}$ senkrecht in zwei gleichlange Segmente teilt.

- Typ E2
  Eine Kante, die durch ein Liniensegment L_i^0 und einem Punkt $p_i^1 \in \partial L_i$ definiert wird. Die Kante ist eine Gerade senkrecht zu L_i^0, die durch p_{i1} geht.

- Typ E3 Eine Kante, die durch ein Liniensegment L_j^0 und einem Punkt $p_i \notin \partial L_j$ definiert wird. Die Kante ist eine parabolische Kurve, deren Brennpunkt p_i ist, und deren Leitlinie das Segment L_j^0 enthält.

- Typ E4
  Eine Kante, die durch zwei Liniensegmente L_i^0 und L_j^0 definiert wird. Die Kante ist ein Liniensegment, das den Winkel zwischen den gegebenen Segmenten halbiert.

Voronoi-Ecken in nicht degenerierten Voronoi-Diagrammen für Liniensegmente können in sechs Typen (s.Abb.3) eingeteilt werden [BOS92]:

- Typ V1
  Eine Ecke q_i, die durch drei Punkte p_i, p_j, p_k definiert wird. Alle drei Voronoi-Kanten sind dabei vom Typ E1.

- Typ V2
  Eine Ecke q_i, die durch ein Liniensegment L_i^0 und zwei Punkte $p_i \in \partial L_i$ und $p_j \notin \partial L_i$ defieniert wird. Die Voronoi-Kanten sind vom Typ E1, E2 und E3.

- Typ V3
  Eine Ecke q_i, die durch ein Liniensegment L_i^0 und zwei Punkte $p_j, p_k \notin \partial L_i$ defieniert wird. Die Voronoi-Kanten sind Typ E1 und E3. Die Leitlinie der Kanten vom Typ E3 enthält L_i^0.

- Typ V4
  Eine Ecke q_i, die durch zwei Liniensegmente L_i^0 und L_j^0 und einen derer Endpunkte $p_{i1} \in \partial L_i$ definiert wird. Die Voronoi-Kanten sind vom Typ E2, E3 und E4.

- Typ V5
  Eine Ecke q_i, die durch zwei Liniensegmente L_i^0 und L_j^0 und einem Punkt p_i $p_i \notin \partial L_i^0, p_i \notin \partial L_j^0)$ definiert wird. Die Voronoi-Kanten sind vom Typ E3 und E4.

- Typ V6
  Eine Ecke q_i, die durch drei Liniensegmente L_i^0, L_j^0 und L_k^0 definiert wird. Die Voronoi-Kanten sind vom Typ E4.

Bemerkungen:
Eine Voronoi-Region VR hat folgende Eigenschaften:

- $VR \neq \emptyset$

- $VR(p_i) = p_i \Leftrightarrow p_i$ ist Endpunkt dreier oder mehr Liniensegmente.

- VR ist nicht notwendigerweise konvex.

- $VR(L_i)$ ist unbeschränkt $\Leftrightarrow L_i \subset \partial(conv(L^{(d)}))$

2.2 Randomisierter inkrementeller Algorithmus

Die Konstruktion von abstrakten Voronoi-Diagrammen in der Ebene kann effizient und rein kombinatorisch in die Konstruktion von Voronoi-Diagrammen von fünf Generatoren transformiert werden.
Der inkrementelle Algorithmus fügt nacheinander einen weiteren Generator in ein Voronoi-Diagramm ein. Die Laufzeit des Algorithmus liegt in $O(n^2 - n)$.
Der randomisierte inkrementelle Algorithmus (RIA) hat eine Laufzeit von $O(n \log n)$. Dabei werden alle Generatoren in zufälliger Reihenfolge eingefügt.
Der RIA berechnet dabei rein kombinatorisch neue Voronoi-Ecken, -Kanten und -Regionen (Voronoi-Objekte). Diese werden in einem Graphen gespeichert. Entscheident für den Typ neuentstehender Voronoi-Objekte ist die Lage zu den bereits im Voronoi-Diagramm eingefügten Generatoren. Den dazu notwendigen Algorithmus nennt man Incircle-Test. Dieser Test entscheidet, ob ein Generator im Inkreis dreier anderer Generatoren liegt. Dies ist der wesentliche Schritt, da eine Voronoi-Ecke durch drei Generatoren eindeutig festgelegt wird.
Auf die Analyse der Korrektheit und der Laufzeit des randomisierten inkrementellen Algorithmus möchte ich hier nicht näher eingehen. Dies würde leider den Rahmen dieses Vortrags weit überschreiten. Ich möchte jedoch auf zwei detaillierte Reporte zu diesem Thema hinweisen: [KMM90] und [KMM93]

2.3 Der Incircle-Test

Der Incircle-Test bestimmt, ob ein Generator im Inkreis von drei weiteren Generatoren liegt, ihn berührt oder außerhalb liegt. Eine Methode das VD einer Punkt-Generatorenmenge zu bestimmen ist die Bestimmung der sogenannten Delaunay-Triangulierung.

2.3.1 Die Delaunay-Triangulierung

Die Delaunay-Triangulierung einer Punktmenge S bildet das Dual zum Voronoi-Diagramm $V(S)$ von S falls $V(S)$ einfach. D.h. v ist Voronoi-Ecke der Punkte p_1, p_2, p genau dann wenn p_1, p_2, p_3 Ecken eines Dreiecks der Delaunay-Triangulierung von S sind (Abb.5).

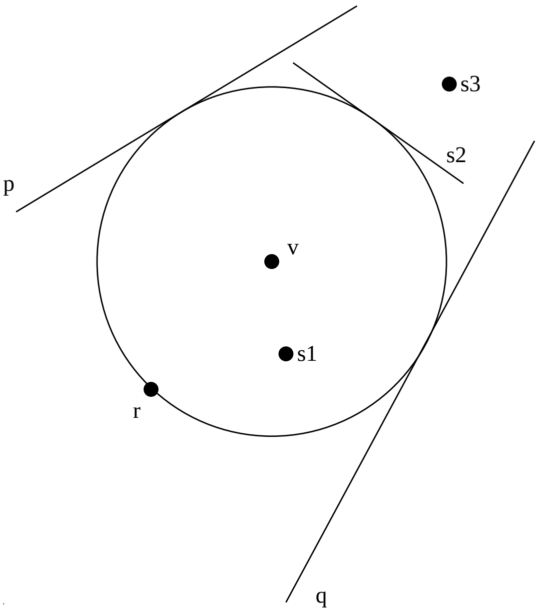

Abbildung 4: Der Incircle Test (Voronoi-Ecke v durch p,q,r bestimmt)

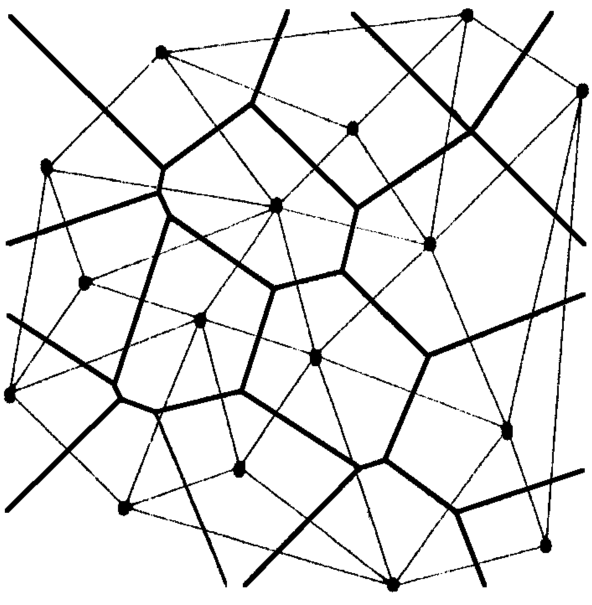

Abbildung 5: Delaunay-Triangulierung und Voronoi Diagramm (fett)

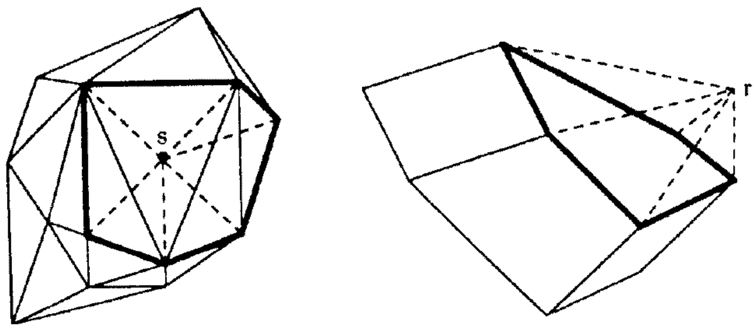

Abbildung 6: Die Triangulierung (gestrichelt) von P (fett) durch s bzw. r

Beweis. (Skizze)
"$\Leftarrow$" Sei $\triangle(p_1, p_2, p_3) \in DT(S)$. Sei U Umkreis von $\triangle(p_1, p_2, p_3)$. Dann gilt für den Mittelpunkt M von U: $d(M, p_i) = d(M, p_j) \quad \forall\, 1 \leq i, j \leq 3$. $\Rightarrow M$ ist Voronoi-Ecke des Voronoi-Diagramms $V(S)$ von S.

"$\Rightarrow$" (Skizze) v Voronoi-Ecke zu $p_1, p_2, p_3 \in S$. $\Rightarrow B_v(d(v, p_1)) \cap S = \emptyset$ falls $V(S)$ einfach. $\Rightarrow \triangle(p_1, p_2, p_3)$ ist lokal delaunay. $\Rightarrow \triangle(p_1, p_2, p_3) \in DT(S)$. □

Typisch für eine Delaunay-Triangulierung ist, daß sie den kleinsten in allen Dreiecken auftretenden Winkel maximiert [E87]. Seien $d+1$ Punkte $p_1, \ldots, p_{d+1}$ gegeben, mit $p_i = (p_{i1}, \ldots, p_{id})$. Eine bekannte Methode, die Orientierung der Punkte zu bestimmen, ist die Bestimmung des Vorzeichens der Determinante

$$\begin{vmatrix} p_{11} & p_{12} & \cdots & p_{1d} & 1 \\ p_{21} & p_{22} & \cdots & p_{2d} & 1 \\ \vdots & \vdots & & \vdots & \vdots \\ p_{d+1,1} & p_{d+1,2} & \cdots & p_{d+1,d} & 1 \end{vmatrix}$$

Betrachtet man nun eine Menge von $d+1$ Punkten $(p_2, \ldots, p_{d+2})$, die positiv orientiert sind, dann ist p_1 außerhalb der $p_2, \ldots, p_{d+2}$ umgebenden Späre, wenn sich $\lambda(p_1)$ oberhalb der Hyperebene befindet, die von $\lambda(p_2), \ldots, \lambda(p_{d+2})$ aufgespannt wird. Das ist natürlich dann der Fall, wenn die Orientierung von $\lambda(p_1), \ldots, \lambda(p_{d+2})$ positiv ist also

$$\begin{vmatrix} p_{11} & \cdots & p_{1d} & p_{11}^2 + & \cdots & +p_{1d}^2 & 1 \\ p_{21} & \cdots & p_{2d} & p_{21}^2 + & \cdots & +p_{2d}^2 & 1 \\ \vdots & & \vdots & & & \vdots & \vdots \\ p_{d+2,1} & \cdots & p_{d+2,d} & p_{d+2,1}^2 + & \cdots & +p_{d+2,d}^2 & 1 \end{vmatrix}$$

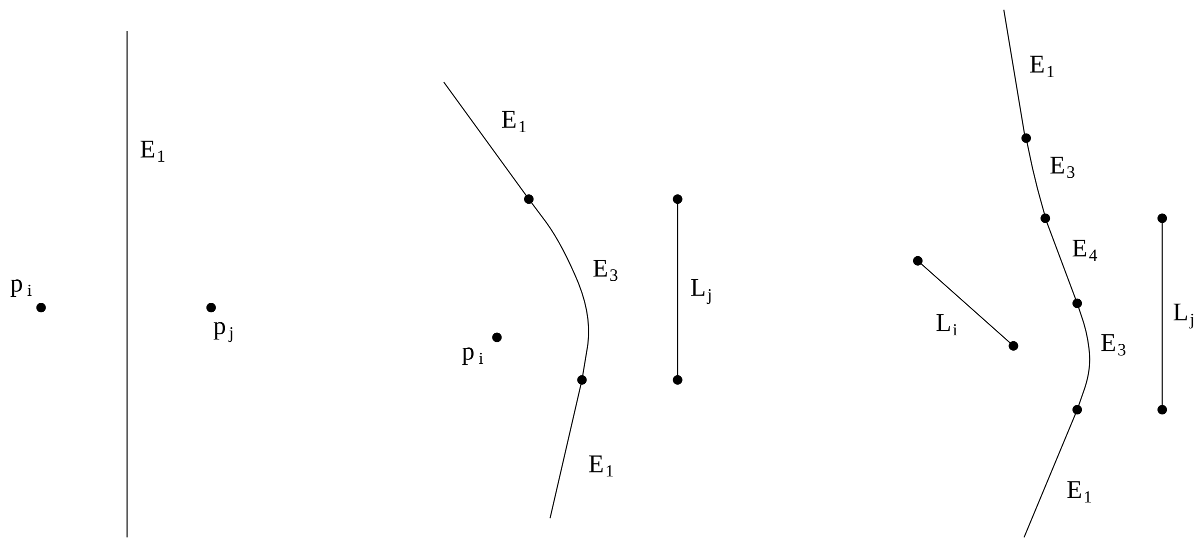

Abbildung 7: Bisektoren zwischen Punkt und Liniensegment

positives Vorzeichen besitzt. Der Incircle-Test kann also auf die Bestimmung des Vorzeichens o.a. Determinante zurückgeführt werden.

Dieses Determinanten-Verfahren läßt sich natürlich nur bei Punkt-Generatoren verwenden. Bei Liniensegmenten muß überlegt werden, wie man den Incircle-Test anders implementiert. Dazu betrachtet man zunächst die Bisektoren von Punkten und Liniensegmenten.

3 Präzisionsverringerung

Es seien im folgenden ein Punkt $p = (x_p, y_p, z_p)$ und zwei Linien $l_i = a_i x + b_i y + c_i = 0 \quad ; i = 1, 2$ als Sites gegeben (bei anderen Konfigurationen verlaufen die Berechnungen ähnlich). Die gegebenen Größen x_p, y_p, z_p und a_i, b_i, c_i seien alle k-bit Integer.

3.1 Bisektoren

Der Bisektor zweier Sites ist die Zerlegung einer Ebene in zwei Halbebenen. Hierbei hat jede Halbebene die Eigenschaft, daß alle in ihr enthaltenen Punkte einen geringeren Abstand zur eigenen Site haben, als zur jeweils anderen Site. Der Bisektor zwischen p und einer der Geraden l_i ist durch eine Parabel gegeben:

$$(a_i x + b_i y + c_i)^2 = (a_i^2 + b_i^2)\left(\left(x - \frac{x_p}{z_p}\right)^2 + \left(y - \frac{y_p}{z_p})^2\right)\right)$$

Der Bisektor zwischen den zwei Geraden l_i ist durch zwei Geraden gegeben:

$$\frac{a_1 x + b_1 y + c_1}{D_1} = \pm \frac{a_2 x + b_2 y + c_2}{D_2}$$

wobei $D_i = \sqrt{N_i}$ und $N_i = a_i^2 + b_i^2$.

3.2 Analyse der Rechengenauigkeit

Zunächst transformieren wir die Koordinaten der Sites so, daß der Punkt p in $(0,0,1)$ liegt. Dabei werden die Gleichungen der Linien l_i transformiert in

$$\underbrace{(a_i z_p)}_{\hat{a}_i} x + \underbrace{(b_i z_p)}_{\hat{b}_i} y + \underbrace{(a_i x_p + b_i y_p + c_i z_p)}_{\hat{c}_i} = 0.$$

Die Koeffizienten $\hat{a}_i$, $\hat{b}_i$, $\hat{c}_i$ müssen daher für eine exakte Berechnung $(2k+2)$-bit Integers sein. Die zwei Voronoi-Ecken, die durch die Sites l_1, l_2, p definiert sind, ergeben sich aus dem Schnitt der Bisektoren, also jeweils aus dem Schnittpunkt einer Parabel und einer Linie. Durch folgende Zuweisungen

$$\begin{array}{lcllcl}
I & = & b_1 c_2 + b_2 c_1 & J & = & a_1 c_2 + a_2 c_1 \\
F & = & 2c_1 c_2 (D_1 D_2 + b_1 b_2 - a_1 a_2) & G & = & 2c_1 c_2 (D_1 D_2 + a_1 a_2 - b_1 b_2 \\
r & = & a_1 D_2 - a_2 D_1 & s & = & b_1 D_2 - b_2 D_1 \\
X & = & D_1 D_2 - a_1 a_2 - b_1 b_2 & \alpha & \in & 0, 1
\end{array}$$

ergeben sich für die Voronoi-Ecken folgende homogene Koordinaten:

$$v = \left(\underbrace{J + \alpha\sqrt{G}}_{x_s}, \underbrace{I - \alpha\, sign(rs)\sqrt{F}}_{y_s}, \underbrace{X}_{z_s}\right) \tag{1}$$

Angenommen wir haben als nächste Site nun eine Linie $l_3 :\ a_3 x + b_3 y + c_3 = 0$ (Koeffizienten wiederum k-bit Integer), so vergleichen wir $(d(v,p))^2$ mit $(d(v,l_3))^2$. Wir müssen also entscheiden ob

$$\left(\frac{x_s}{z_s}\right)^2 + \left(\frac{y_s}{z_s}\right)^2 < \left(\frac{a_3 \frac{x_s}{z_s} + b_3 \frac{x_s}{z_s} + c_3}{D_3}\right)^2 \tag{2}$$

ist. Dies kann, verwendet man obige Bezeichnungen, umformuliert werden in:

$$D_3^2\left((J+\alpha\sqrt{G})^2+(I-\alpha\, sign(rs)\sqrt{F})^2\right) <$$
$$\left(a_3(J+\alpha\sqrt{G})+b_3(I+\alpha\, sign(rs)\sqrt{F})+c_3X\right)^2$$

Nach Umformung der Terme unter Berücksichtigung ihrer Genauigkeit erhält man einen Ausdruck der Form:

$$A_1+A_2\sqrt{N} < A_3\sqrt{F}+A_4\sqrt{G}+\sqrt{N}\left(A_5\sqrt{F}+A_6\sqrt{G}\right) \tag{3}$$

Dabei ist A_1 ein $12k$, A_2, A_3, A_4 sind $8k$, A_5, A_6 sind $4k$ und $N = N_1N_2$ ist ein $8k$-bit Integer. Beachtet man, daß $\sqrt{FG}$ ein Integer ist, so erhält man durch Quadrieren einen Ausdruck der Form:

$$A_7+A_8\sqrt{N} < A_9+A_10\sqrt{N}+\sqrt{N} \tag{4}$$

wobei man ausnutzt, daß

$$\begin{aligned}\left(A_3\sqrt{F}+A_4\sqrt{G}\right)^2 &= A_3^2F+A_4^2G+2A_3A_4\sqrt{FG}\\ \left(\sqrt{N}\left(A_5\sqrt{F}+A_6\sqrt{G}\right)\right)^2 &= N\left(A_5^2F+A_6^2G+2A_5A_6\sqrt{FG}\right)\end{aligned}$$

$$\sqrt{N}\left(A_3\sqrt{F}+A_4\sqrt{G}\right)\left(A_5\sqrt{F}+A_6\sqrt{G}\right)$$
$$=\sqrt{N}\left(A_3A_5F+(A_3A_6+A_4A_5)\sqrt{FG}+A_4A_6G\right)$$

Durch Quadrieren ist der Faktor mit höchster Präzision der Faktor A_7, mit 24k Stellen. Erneutes Umformen, Zusammenfassen und Quadrieren führt zur gewünschten Integer Relation mit maximal benötigter Präzision von $48k$ Stellen.

Will man statt mit Integer-Arithmetik mit Fließkomma-Arithmetik rechnen, so ergibt sich eine Obergrenze der benötigten Genauigkeit von $48k + O(1)$ auf beiden Seiten von Ungleichung (3). Dies resultiert direkt aus folgendem Lemma.

Lemma 1. *Bezeichne l die linke, r die rechte Seite der Ungleichung (3). Dann ist entweder $l = r$ oder $\exists u, v \in \mathbb{Z} : 2^u \geq |l|, |r|$ und $|l-r| \geq 2^{u-v}$ wobei $v = 48k + O(1)$.*

Beweis. Zunächst stellt man fest, daß quadrieren die Anzahl relevanter Bits nur um eine additive Konstante verändert, d.h., wenn sich a^2 und b^2 im t-ten Bit unterscheiden, dann unterscheiden sich a und b im $(t+1)$-ten Bit.

Lemma 2. *Seien* $0 \leq a < b$ *mit* $a^2, b^2 \leq 2^s$ *und* $b^2 - a^2 \geq 2^{s-t}$. *Dann gilt* $a < b \leq 2^{s/2}$ *und* $b - a \geq 2^{s/2-t-1}$.

Beweis. Offensichtlich sind $a, b \leq 2^{s/2} \Rightarrow b + a \leq 2^{s/2+1}$. Daher ist
$2^{s-t} \leq (b^2 - a^2) = (b-a)(b+a) \leq (b-a)2^{s/2+1}$. □

In Fall der Ungleichung 3 erhält man eine Integerrelation durch zweimaliges Quadrieren. Also sind $t = s = 48k + O(1)$ nach dem zweiten Quadrieren. Die Behauptung folgt durch zweimaliges Anwenden von Lemma 2. □

4 Experimentelle Ergebnisse

Burnikel et al. haben in C++ zwei Algorithmen RS und FF für den Incircle-Test in Gleichung 3 implementiert.

Algorithmus RS benutzt exakte Integer-Arithmetik, die durch eine BIGNUM Bibliothek bereitgestellt wird. Um unnötigen Overhead der BIGNUMS zu vermeiden, versucht man im RS Algorithmus die Relation 3 zunächst mit double Werten zu bestimmen. Für jede auftretende Art von algebraischen Zahlen sind geeignete Klassen definiert, die die Rechenoperatoren $+, -, *$ überladen. Zur weiteren Berechnung stehen Routinen für Umordnung, Vorzeichentest und Quadrieren der Ausdrücke zur Verfügung.

Algorithmus FF verwendet eine BIGFLOAT Bibliothek, die Fließkommazahlen bereitstellt, dessen Genauigkeit der Benutzer voreinstellen kann. Der bei der Auswertung eines BIGFLOAT Ausdrucks E entstehende Fehler kann durch statische Fehleranalyse abgeschätzt werden durch:

$$\epsilon(E) * M(E) * 2^{-p}$$

wobei $M(E)$ eine vorgegebene obere Schranke des Wertes von E, p die aktuelle Genauigkeit und $\epsilon(E)$ eine kleine Konstante ist. Die Fehleranalyse ergibt, daß eine Genauigkeit von $2k + 3$ in jedem Fall benötigt wird. $2k + 3$ wird als Anfangsgenauigkeit in FF verwendet und, falls nötig, auf $3k$, $4k$, ... erhöht, bis das Vorzeichen zuverlässig bestimmt werden kann.

Beide Algorithmen, FF und RS wurden auf drei verschiedenen Arten von Eingabedaten getestet:

4.1 Test mit einfachen Daten

Punkte und Endpunkte der Liniensegmente wurden mit $k = 10, 20, 30$ Stellen Genauigkeit zufällig und gleichverteilt gewählt.

k	Σ Tests	$2k+3$	$3k$	$4k$	$5k$	RS	FF
10	10000	2781	6801	398	20	26	80
20	10000	9962	38			25	38
30	10000	10000				26	70

Tabelle 1: Anzahl Test in Algorithmus FF mit angegebener Präzision, sowie Laufzeit der Algortihmen FF und RS bei einfachen Daten, gemessen in Sekunden auf einer SUN SPARCstation 10.

File	Σ Tests	$\leq 3k$	$4k$	$5k$	$6k$	$7k$	$8k$	$9k$	RS	FF
B1	387	8	122	136	75	9			3.9	2.5
B2	398	45	204	134	12	3			2.3	1.9
B3	918	10	224	620	64				6.3	7.8
B4	1126	35	449	628	14				6.6	6.8
B5	4149		125	2310	1697	17			47	44
B6	13021		23	1266	10367	1310	42	4	141	138
B7	140	25	80	35					0.73	0.95
B8	276	3	63	197	13				2.1	2.4
B9	376		17	191	143	13	12		4.0	3.8
B10	414		46	235	119	14			4.3	3.8
B11	912	11	21	217	641	18	4		11.5	8.9
B12	385		2	7	341	30	5		4.9	15.2

Tabelle 2: Anzahl Test in Algorithmus FF mit angegebener Präzision, sowie Laufzeit der Algortihmen FF und RS bei realistische Daten, gemessen in Sekunden auf einer SUN SPARCstation 10.

Obwohl der Algorithmus FF nur sehr geringe Präzision benötigt (Tab.1) ist RS schneller, da alle Entscheidungen durch Berechnung mit doubles gemacht werden können. Die BIGNUM Bibliothek wird somit gar nicht verwendet.

4.2 Test mit realistischen Daten

Es werden die polygonalen Eingabedatensätze B1-B12 (Abb.8) verwendet. Es wurde eine bereits vorhandene Implementierung des randomisierten Algorithmus von Klein et al. [KMM90] benutzt. Die Incircle-Test wurden allerdings nicht durchgeführt sondern aufgezeichnet und als Eingabe für die Algorithmen RS und FF verwendet.

In Tabelle 2 sind die statistischen Ergebnisse für die Tests mit realistischen

k	Σ Tests	RS	FF
10	1000	11	40
20	1000	12	60
30	1000	14	97

Tabelle 3: Laufzeit der Algortihmen FF und RS bei schwierigen Daten, gemessen in Sekunden auf einer SUN SPARCstation 10.

Daten aufgeführt. Die Datensätze B7-B12 enthalten beinahe degenerierte Fälle. Die durchschnittlich benötigte Genauigkeit ist dabei höher als bei den Datensätzen B1-B6. Allerdings ist sie im Vergleich zu der theoretisch hergeleiteten Präzisionsanforderung sehr gering. Die Laufzeiten beider Algorithmen weichen nur leicht voneinander ab.

4.3 Test mit schwierigen Daten

Es wurde wiederum eine Genauigkeit der Eingabedaten von $k = 10, 20, 30$ verwendet. Allerdings waren die Eingabedaten so beschaffen, daß in der Relation 3 immer Gleichheit bestand und somit stets mit größter Genauigkeit gerechnet werden mußte.

Tabelle 3 zeigt die Laufzeit beider Algorithmen. FF ist hier langsamer als RS, da er zunächst viele Berechnungen mit geringerer Genauigkeit durchführt, bevor er schließlich mit größter Genauigkeit rechnet.

4.4 Vergleich der Algorithmen FF und RS

Bei Operation auf nicht-degenerierten Eingabedaten benötigen beide Algorithmen eine erfreulich geringe Genauigkeit. Dabei ist die BIGFLOAT Variante etwas flexibler, was sie bei degeneriertem Input allerdings mehr Zeit kostet. Hier ist der Ansatz mit BIGNUMS besser geeignet.

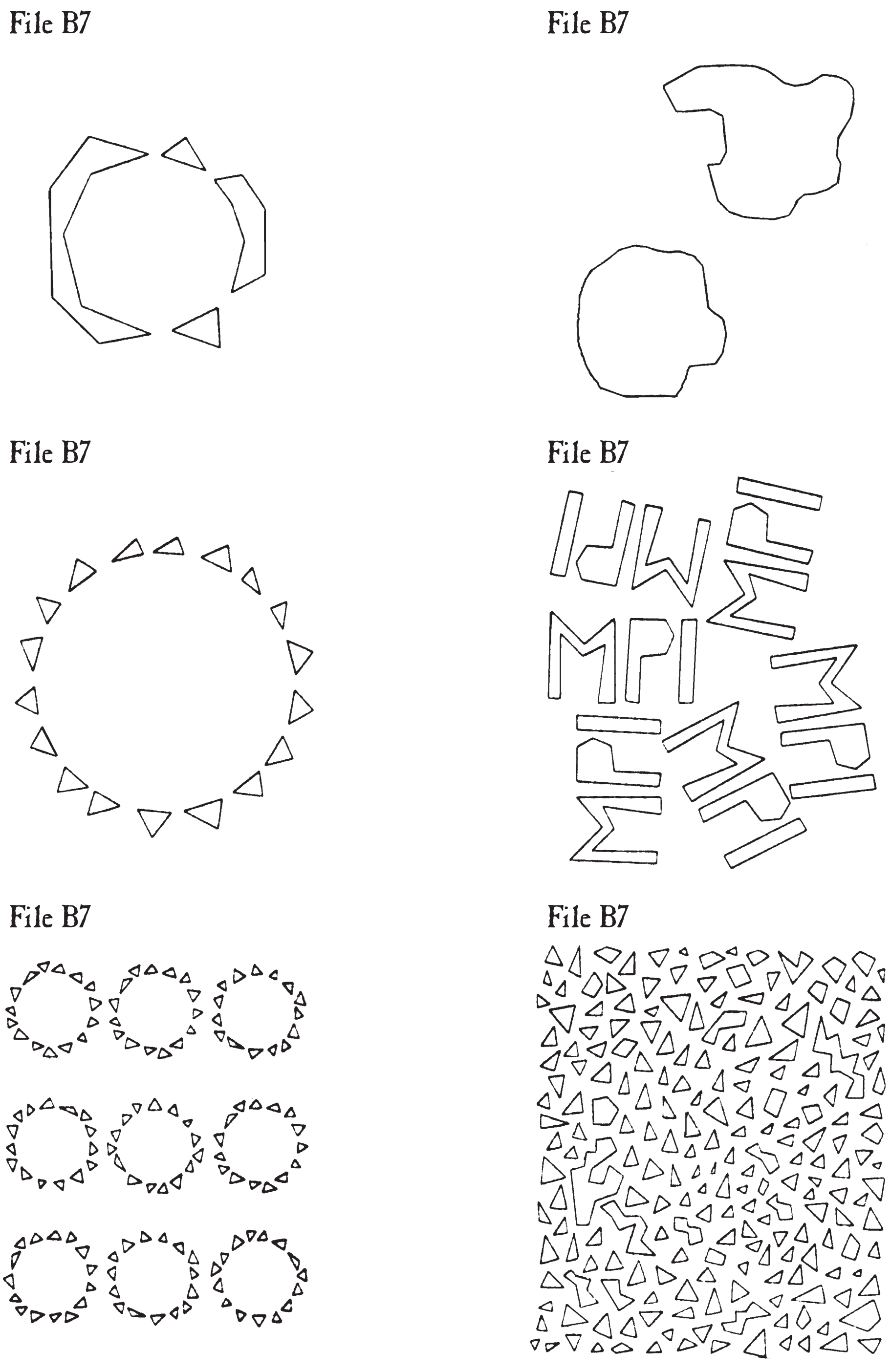

Abbildung 8: Degenerierte Daten (links). Polygonale Daten (rechts).

Literatur

[BMS94] Ch.Burnikel, K.Mehlhorn, St.Schirra, *How to Compute the Voronoi Diagram of Line Segments: Theoretical and Experimental Results*, ESA94 Conference, LNCS, pp. 227-239, Springer 1994.

[F92] S.Fortune, *Voronoi Diagrams and Delaunay Triangulations*, In Computing in euclidean Geometry, Lecture Notes Series on Computation, Vol.1, World Scientific, 1992.

[BOS92] B.N.Boots, A.Okabe, K.Sugihara, *Spatial Tesselations - Concepts & Applications of Voronoi Diagrams*, Wiley, New York, 1992.

[E87] H.Edelsbrunner, *Algorithms in Combinatorial Geometry* EATCS Monographs on theoretical Computer Siecnce, Springer, 1987.

[KMM90] R.Klein, K.Mehlhordn, S.Meiser, *On the construction of abstract Voronoi diagrams ii.* In Proc. SIGAL Symp. on Algorithms, Tokyo, 1990. Springer Verlag. LNCS 450.

[KMM93] R.Klein, K.Mehlhorn, S.Meiser, *Randomized incremental construction of abstract Voronoi diagrams*, In Computational Geometry: Theory and Applications (3), pp157-184, Elsevier, 1993.

[S92] R.Sedgewick, *Algorithms in C++*, Addison Wesley, 1992.