



TECHNISCHE UNIVERSITÄT CHEMNITZ

Informatik

Graphische Datenverarbeitung und Visualisierung

Studienarbeit

Rechnergestützte Bestimmung der Merkmale von Rändern
archäologischer Gefäße

Nadine Pollmer

Chemnitz, September 2005

Betreuer: Prof. Dr. Guido Brunnett
Dipl.-Inf. David Brunner

Pollmer, Nadine

Rechnergestützte Bestimmung der Merkmale von Rändern archäologischer Gefäße

Studienarbeit, Informatik

Technische Universität Chemnitz, September 2005

Inhaltsverzeichnis

Abbildungsverzeichnis	v
Tabellenverzeichnis	vii
1 Einleitung	1
1.1 Motivation und Zielstellung	1
1.2 Gliederung dieser Arbeit	2
2 Vorverarbeitung	3
2.1 Randerstellung	3
2.1.1 Profil des Gefäßes	4
2.1.2 Randauswahl und Datenerstellung	6
2.2 Löcher schließen	7
2.3 Kontur drehen	9
2.4 Diskretisierung	11
3 Distanzmessungen im 2D	16
3.1 Distanzmaße	16
3.2 Distance Mapping	16
3.3 Distanzmetriken	18
3.4 Distanztransformationen	22
3.4.1 Zweidimensionale Distanztransformation (Chamfer)	22
3.4.2 Euklidische Distanztransformation	23
Transformation 1	24
Transformation 2	24

4	Berechnung der Mittelachse	26
4.1	Definitionen	26
4.1.1	Mittelachse	26
4.1.2	Topologie	27
4.1.3	Vorder- und Hintergrundkomponente	28
4.2	Methoden	31
4.2.1	Distanztransformation	32
	Bewertung der Methode	33
4.2.2	Voronoi-Diagramm	33
	Bewertung der Methode	34
4.2.3	Ausdünnung	34
	Objektrandbefüllung	35
	Bewertung der Methode	36
	Detaillierter Algorithmus	37
5	Randmerkmale	39
5.1	Randstellung	39
5.1.1	Mittelachsenbestimmung	40
	Mittelachsenpolygonzug	41
5.1.2	Winkelberechnung	42
	Auswertung	43
5.1.3	Entscheidungsbaum	49
5.2	Lippenbildung	50
5.2.1	Analyse	50
5.2.2	Entscheidungsbaum	54
5.3	Randabschluss	55
5.3.1	Analyse	57
5.3.2	Entscheidungsbaum	60
5.4	Beurteilung	61
A	Anhang	63
A.1	Funktionen bei geladenem Gefäßobjekt	63
A.2	Funktionen bei geladenem Randobjekt	65

Literaturverzeichnis

66

Abbildungsverzeichnis

2.1	Dreieck schneidet Ebene	4
2.2	Seitenhalbierenden eines Dreiecks	5
2.3	Mittelpunktberechnung des Schnittdreiecks	5
2.4	Aufbau einer Wavefront-Datei	6
2.5	Gefäßquerschnitt mit Randauswahl	7
2.6	Winkeltest vor dem Schließen einer Lücke	8
2.7	Lückenhafte Randkontur	9
2.8	Randkontur nach Lückenschließung	9
2.9	Randkontur	10
2.10	Gedrehte Randkontur	11
2.11	Bestimmung des besten Nachfolgers	12
2.12	Linienunterscheidung	13
2.13	Randkontur als Polygonzug	15
2.14	Diskretisierte Randkontur	15
3.1	Euklidische Distanz im 3D	19
3.2	City-Block Distanz	19
3.3	Schachbrett Distanz	20
3.4	Chamfer Distanz	20
3.5	Digitale Wege im 2D	21
3.6	Unterschiedliche Distanzmetriken	25
4.1	Mittelachse eines Rechteckes	27
4.2	N_4 - und N_8 -Nachbarschaft	29
4.3	1 Vorder- und 2 Hintergrundkomponenten	29
4.4	2 Vorder- und 1 Hintergrundkomponente	29

4.5	Nachbarschaft eines Pixels mit linearem Feld	30
4.6	Voronoi-Diagramm einer Punktmenge	33
4.7	Floodfill vom Punkt P aus	35
4.8	Scanlineverfahren bei geschlossenem Objekt	36
4.9	Scanlineverfahren bei nicht geschlossenem Objekt	37
4.10	Randobjekt mit Mittelachse	38
5.1	Schema zur Randstellung	40
5.2	Diskretisiertes gefülltes Randobjekt	40
5.3	Fehlerhafte Mittelachse	41
5.4	Korrigierte Mittelachse	42
5.5	Winkel eines Mittelachsenpunktes	43
5.6	Dateiformat der Winkeldaten	43
5.7	Vergleich Prototyp mit gegebenem Rand	44
5.8	Fehlerhafter Prototypvergleich	49
5.9	Entscheidungsbaum Randstellung	49
5.10	Schema zur Lippenbildung	50
5.11	Mittelachsenuntersuchung paralleler Linien	51
5.12	Unterteilung des Randpolygons in 2 Seiten/Teile	52
5.13	Entscheidungsbaum Lippenbildung	54
5.14	Schema zum Randabschluss	55
5.15	Bestimmung der Randabschlusspixel	56
5.16	Randabschluss gekehlt innen	57
5.17	Winkelkurven	58
5.18	Flächeninhaltsbestimmung zwischen Maxima	59
5.19	Entscheidungsbaum Randabschluss	60
5.20	Entscheidungsbaumknoten	61
5.21	fehlerhaft klassifizierter Rand bzgl. Randstellung	61
5.22	fehlerhaft klassifizierter Rand bzgl. Randabschluss	62
A.1	Einteilung eines Gefäßobjekts	64
A.2	Randauswahl	64

Tabellenverzeichnis

4.1	Vergleich der Skelettierungsmethoden	32
A.1	Funktionalitäten der Button	65

1 Einleitung

1.1 Motivation und Zielstellung

Die Professur für Graphische Datenverarbeitung und Visualisierung der Technischen Universität Chemnitz arbeitet bereits seit geraumer Zeit mit dem Landesamt für Archäologie mit Landesmuseum für Vorgeschichte in Sachsen zusammen. Ziel dieser Zusammenarbeit ist u.a. die Entwicklung von Methoden, welche es erlauben, Vorgänge bei Klassifikation und Typisierung archäologischer Funde zu automatisieren. Diese Problematik ist innerhalb der Archäologie ein zentraler Bestandteil bei der Auswertung archäologischer Funde, daher sollen Computeranalysen die Form- und Eigenschaftsbeschreibungen liefern, die den jeweiligen Gefäßfunden zugeordnet werden.

Die Analyse des Gefäßrandes ist für Archäologen von besonderer Bedeutung, da dieser ein außerordentlich signifikantes Kriterium bei der Klassifizierung des Gefäßes darstellt. Der Gefäßrand lässt sich mit vier Merkmalen beschreiben: die Randstellung, die Lippenbildung, den Randabschluss und evtl. vorhandene Details.

Die Studienarbeit soll eine automatische Bestimmung der Gefäßränder durch Differenzierung der ersten drei Merkmale für 3D-Modelle archäologischer Gefäße aus der Bronzezeit ermöglichen. Dies soll mit der Weiterentwicklung der Software „ArchViewer“ umgesetzt werden. Die erweiterte Software soll Tests mit reellen Objekten durchführen können (Randextrahierung, -speicherung und -klassifizierung), so dass die theoretischen Erkenntnisse und die erarbeiteten Algorithmen praktisch fundiert werden.

1.2 Gliederung dieser Arbeit

Zunächst werden im nachfolgenden KAPITEL 2 die Schritte der Vorverarbeitung, die für die Klassifizierungsalgorithmen nötig sind, näher erläutert.

KAPITEL 3 beleuchtet Vorgehen und Möglichkeiten Distanzen im diskretisierten zweidimensionalen Raum zu messen.

Anschließend werden in KAPITEL 4 Verfahren für die Berechnung der Mittelachse vorgestellt, bewertet und Schlussfolgerungen für den Einsatz bei der Randklassifizierung gezogen.

Abschließend wird in KAPITEL 5 die genauere Untersuchung der Klassifizierungsmerkmale vorgenommen und die zugehörigen Untersuchungsschritte näher beschrieben.

2 Vorverarbeitung

Damit der Rand eines Gefäßes analysiert und bestimmt werden kann, müssen einige Vorverarbeitungsschritte durchgeführt werden. Dazu gehört die Gefäßrandextrahierung um ein Randprofil zu erhalten, das „Löcher schließen“, falls der Polygonzug nicht zusammenhängend ist, also aus mehreren Teilen besteht, sowie das Ausrichten der Kontur und schließlich die Diskretisierung (dabei werden die Objektinformationen in eine abzählbare Menge abgebildet).

Bevor jedoch näher auf die einzelnen Schritte eingegangen wird, zunächst einige Namenskonventionen:

- Vertex: Punkt der Triangulierung der Gefäßvorlage
- Randpunkt: Punkt der extrahierten Randkontur (Punkt auf Polygonzug)
- Abschlusspunkt: Punkt, welcher den Randpolygonzug abschließt, also nur durch eine Linie mit einem anderen Punkt verbunden ist

2.1 Randerstellung

Die Randprofilextrahierung erfolgt in zwei Schritten:

1. Profil des Gefäßes berechnen
2. manuelle Randauswahl und Datenerstellung

2.1.1 Profil des Gefäßes

Als erstes wird das gesamte Profil des Gefäßes dargestellt. Dies erreicht man, indem eine Ebene das Gefäß längs der Rotationsachse schneidet. Dann werden die Eckpunkte aller Dreiecke der Gefäßtriangulierung auf ihre Position hin untersucht. Man berechnet den Abstand zwischen Punkt und Ebene. Das Vorzeichen dieser Distanz beschreibt, auf welcher Seite der Ebene der jeweilige Vertex liegt. Findet man nun ein Dreieck bei dem die Distanzen der Eckpunkte unterschiedliche Vorzeichen haben oder alle Abstände gleich null sind, so handelt es sich um ein Dreieck, welches die Ebene schneidet (Abbildung 2.1) bzw. in der Ebene liegt.

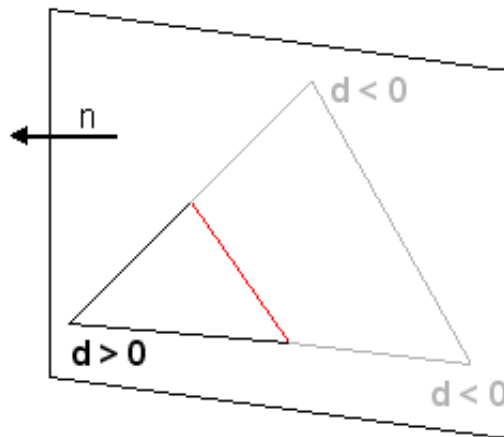


Abbildung 2.1: Dreieck schneidet Ebene

Das Dreieck wird dann für eine spätere Weiterverarbeitung in einer Liste gespeichert und für die Darstellung durch einen Punkt ersetzt. Dabei gelten folgende Regeln:

1. Liegt das Dreieck in der Querschnittsebene, so wird es durch seinen Schwerpunkt ersetzt. Der Schwerpunkt S teilt jede Seitenhalbierende im Verhältnis 2:1, siehe auch Abbildung 2.2.

$$s_a = \frac{1}{2} \sqrt{2(b^2 + c^2) - a^2}$$

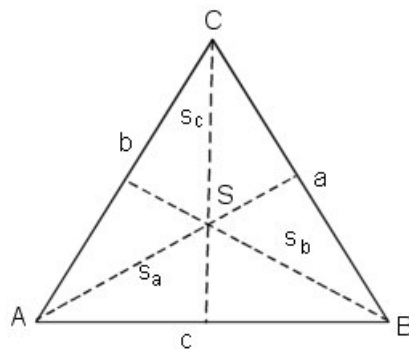


Abbildung 2.2: Seitenhalbierenden eines Dreiecks

2. Schneidet das Dreieck die Ebene, so werden zuerst die zwei Schnittpunkte der Dreieckskanten mit der Ebene bestimmt. Der Mittelpunkt der Strecke Schnittpunkt 1 - Schnittpunkt 2 ist dann der Punkt, der das Dreieck ersetzt (Abbildung 2.3).

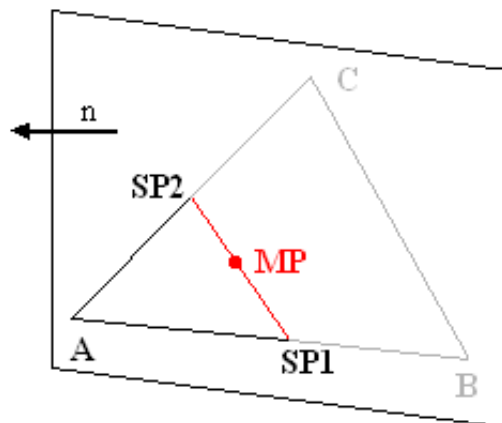


Abbildung 2.3: Mittelpunktberechnung des Schnittdreiecks

2.1.2 Randauswahl und Datenerstellung

Das Gefäßprofil wird nun als Punktemenge dargestellt. Durch Aufziehen eines „Randrahmens“ mit Hilfe der Maus, kann der gewünschte Randbereich ausgewählt werden (Abbildung 2.5). Das Betätigen der rechten Maustaste veranlasst die Erstellung der benötigten Randdaten. Alle Dreiecke die innerhalb des Rahmens liegen, werden dabei für die Polygonzugerstellung genutzt und bilden das gewünschte Randprofil. Der Polygonzug wird wie folgt berechnet:

Die Schnittpunkte der Dreieckskanten mit der Randebene werden bestimmt und in eine Randpunktliste eingefügt, falls sie sich noch nicht in dieser befinden. Weiterhin wird eine Randlinienliste geführt, in welche die Schnittlinie des Dreiecks eingefügt wird. Nachdem alle Dreiecke durchlaufen wurden, können die erstellten Daten als Rand abgespeichert werden. Dabei wird zuerst das „Randobjekt“ in den 1. Quadranten des Koordinatensystems verschoben (alle Werte sind positiv) und anschließend eine Datei auf Wavefront-Basis erstellt, die schließlich als neues Objekt in den ArchViewer geladen werden kann. Der Aufbau innerhalb einer Datei gestaltet sich wie in Abbildung 2.4, dabei steht ein „r“ für das Verhältnis der Randgröße zur Gefäßgröße, ein „v“ für Vertex mit x- und y-Koordinaten und ein „l“ für Linie mit Indizes der Vertices.

```
#ArchViewer - TU Chemnitz
#Randdaten
r 0.213434
v 122.950882 102.314345
v 123.172751 102.329704
v 134.908284 106.389145
v 134.926809 106.339344
[....]
l 0 1
l 2 3
l 0 6
l 6 7
l 9 10
[....]
```

Abbildung 2.4: Aufbau einer Wavefront-Datei

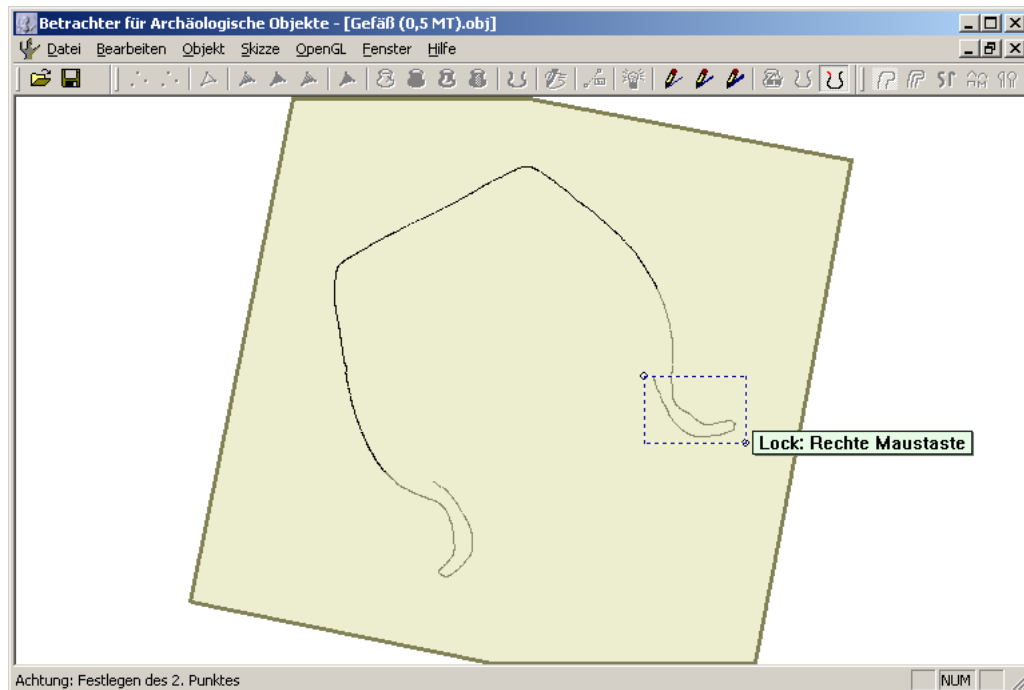


Abbildung 2.5: Gefäßquerschnitt mit Randauswahl

2.2 Löcher schließen

Damit die Klassifizierungsalgorithmen korrekt ablaufen können, darf es im Polygonzug keine „Löcher“ / „Lücken“ geben, das heißt der Polygonzug muss zusammenhängend sein. Da es sonst, zum Beispiel bei der Mittelachsenberechnung zu Ungenauigkeiten und Fehlern kommt.

Die vorhandenen Randdaten werden daher nachgebessert, in dem eine Routine automatisch aufgerufen wird, die versucht, kleinere Lücken im Randprofil zu schließen. Die Routine verwendet einen Winkeltest, der die Winkel α, β der neuen entstehenden Verbindung der Randpunkte P-Q testet (Abbildung 2.6). Sind die Winkel α, β kleiner als 100 Grad wird die Verbindung zwischen P und Q nicht hergestellt. Der Wert 100 Grad ergab sich bei Tests mit der Software, dieser Schwellwert füllte die fehlerhaften Konturen am besten.

Es wird also mit dieser Methode nach einem Verbindungspartner gesucht, durch

den eine „glatte“ Verbindung der Polygonteile entsteht und damit der Polygonzug knickfrei bleibt.

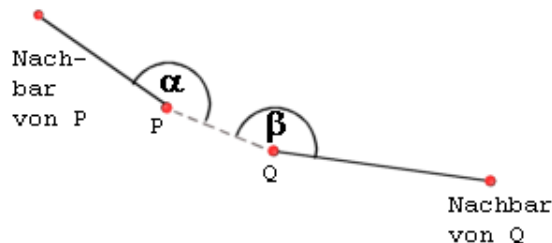


Abbildung 2.6: Winkeltest vor dem Schließen einer Lücke

Algorithmus *Löcher schließen*

1. Bestimme Umkreisradius: 1/8 der größten Bounding-Box-Ausdehnung
2. $MAX \leftarrow$ größte Bounding-Box-Ausdehnung
3. Finde alle Randpunkte, die entweder keinen linken oder keinen rechten Nachbarn haben. Speichere diese Punkte in einer Liste L .
4. **while** Punkte in L **do**
5. Nimm beliebigen Punkt A aus L .
6. Berechne Distanzen zu allen anderen Punkten in L .
7. **while** Punkte mit Distanzen $< MAX$ in L **do**
8. Wähle Punkt B , der geringsten Abstand zu A besitzt und in dessen Umkreis liegt.
9. Berechne Winkel α und β bei entstehender Verbindung AB .
10. **if** $\alpha > 100^\circ$ **and** $\beta > 100^\circ$ **then**
11. Stelle Verbindung zwischen A und B her und lösche die 2 Punkte aus L .
12. **break** /* Schleifenabbruch => weiter bei 3. */
13. **else** Setze Abstand zu B auf MAX .
14. **if** A besitzt keine zwei Nachbarn **then**
15. Es konnte keine Verbindung von A zu einem anderen Punkt im Umkreisradius gefunden werden. A ist evtl. ein Abschlusspunkt und wird aus L gelöscht.

Bei größeren Löchern müssen die Daten per Hand ergänzt werden. Diese Funktionalität wird in ArchViewer nicht ermöglicht, ist aber relativ einfach zu implementieren.

Die Abbildungen 2.7 und 2.8 veranschaulichen die Ergebnisse.

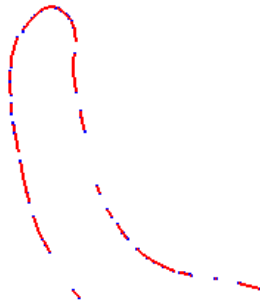


Abbildung 2.7: Lückenhafte Randkontur

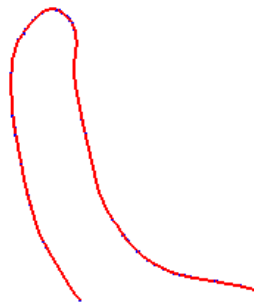


Abbildung 2.8: Randkontur nach Lückenschließung

2.3 Kontur drehen

Um den Polygonzug besser weiterverarbeiten zu können, ist es von Vorteil, wenn die Öffnung der Randkontur parallel zur x-Achse verläuft. Dadurch ist die Einteilung, welcher Bereich der untere bzw. der obere Randabschnitt ist, einfach möglich und

die Mittelachse kann korrekt ausgewertet werden.

Zunächst werden die zwei Randpunkte gesucht, welche nur mit einem Nachbarn verbunden sind (also Abschlusspunkte sind). Da die Routine des Lückenschließens bereits durchlaufen wurde, sollte es nur noch zwei solcher Punkte geben. Sind jedoch mehrere Randpunkte mit nur einem Nachbarpunkt vorhanden, so waren die Lücken im Polygonzug zu groß und der Datensatz ist für eine Randklassifizierung ungeeignet.

Im nächsten Schritt, nach erfolgreicher Bestimmung der zwei Abschlusspunkte, wird der Winkel α zwischen der Geraden, welche die beiden Punkte verbindet, und der x-Achse berechnet. Mit Hilfe dieses Winkels wird nun das gesamte Objekt gedreht, dies geschieht mit folgender Projektionsmatrix,

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

die auf alle Punkte $P = (p_x, p_y)$ angewendet wird, um die Punkte $P' = (p'_x, p'_y)$ zu berechnen.

Die beiden folgenden Abbildungen (2.9 und 2.10) zeigen die Funktionalität des Konturdrehens.

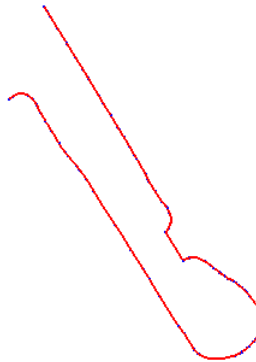


Abbildung 2.9: Randkontur



Abbildung 2.10: Gedrehte Randkontur

2.4 Diskretisierung

Für spätere Verarbeitungsverfahren (u.a. die Mittelachsenberechnung) ist es von Nöten, die gegebenen Daten, also den Polygonzug, zu diskretisieren. Denn dadurch verlaufen die verwendeten Berechnungen schneller, eine Gleichverteilung der Punkte ist gegeben und die Analyse kann unabhängig von der Auflösung erfolgen.

Bei der Diskretisierung des Polygonzuges wird ein Gitter erzeugt, welches in 100×100 Feldern aufgeteilt ist. Dabei ist jedes Feld eingefärbt / gesetzt, indem die Randlinie liegt. Um das Raster mit der Größenordnung des Randobjektes abzustimmen, wird zuerst der maximale Ausdehnungswert max bestimmt, der dann, dividiert durch 100, den „ Δ -Wert“ ergibt.

$$\Delta = \frac{\max(dx, dy)}{100}$$

Mit Hilfe dieses Wertes lassen sich nun die Endpunkte der Linien, die den Polygonzug bilden, in „Rasterkoordinaten“ (Wertebereich jeweils von 0 bis 99) umrechnen:

$$ARaster.x = A.x \text{ div } \Delta$$

$$ARaster.y = A.y \text{ div } \Delta$$

Der Bresenham-Algorithmus, für die Rasterisierung von Linien, verwendet diese neuen Koordinaten und setzt die Rasterfelder. Das Prinzip des Algorithmus ist, aus-

gehend von einem aktuellen Rasterpunkt $P = (x_p, y_p)$ unter den potentiellen Nachfolger, diejenigen zu wählen, welcher der Linie am nächsten liegt.

Laut [Brn03] ist der Algorithmus besonders effizient, da er ausschließlich Integeraddition und Shifts (Multiplikation mit 2) benötigt. Um den Nachfolger von Punkt P zu bestimmen, wird der Mittelpunkt MP zwischen den Punkten E und NE berechnet, liegt MP oberhalb der Linie, so wird E gesetzt, sonst NE (Abbildung 2.11).

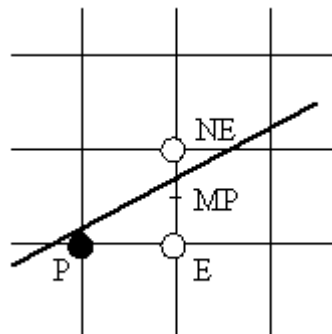


Abbildung 2.11: Bestimmung des besten Nachfolgers

Der Bresenham-Algorithmus ($m = \text{Anstieg}$, Linienanfangspunkt $P_0(x_0, y_0)$, Linienendpunkt $P_1(x_1, y_1)$) für den Fall $0 \leq m \leq 1$, $x_0 \leq x_1$, $E = (x_p + 1, y_p)$, $NE = (x_p + 1, y_p + 1)$ lautet:

Algorithmus *Bresenham*

1. $\Delta x \leftarrow x_1 - x_0$
2. $\Delta y \leftarrow y_1 - y_0$
3. $d \leftarrow 2\Delta y - \Delta x$
4. $\Delta E \leftarrow 2\Delta y$
5. $\Delta NE \leftarrow 2(\Delta y - \Delta x)$
6. $x \leftarrow x_0$
7. $y \leftarrow y_0$
8. Setze Rasterpunkt $P = (x, y)$
9. **while** $x < x_1$ **do**
10. **if** $d < 0$ **then**

11. $d \leftarrow d + \Delta E$
12. $x \leftarrow x + 1$
13. **else**
14. $d \leftarrow d + \Delta NE$
15. $x \leftarrow x + 1$
16. $y \leftarrow y + 1$
17. Setze Rasterpunkt $P = (x, y)$

Da dieser Algorithmus nur für Linien mit oben genannten Eigenschaften zutrifft, gilt es verschiedene Fälle zu unterscheiden (Abbildung 2.12):

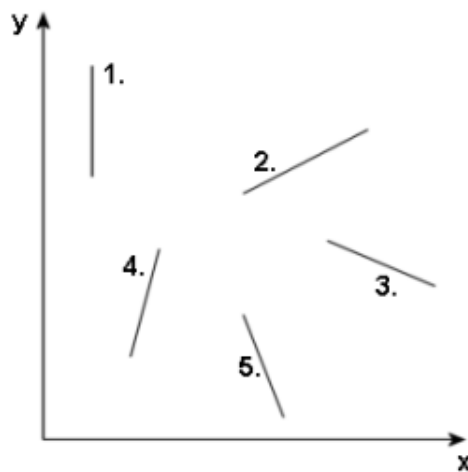


Abbildung 2.12: Linienunterscheidung

1. Linie verläuft parallel zur y-Achse
Der Anstiegswinkel der Linie beträgt 90 Grad.
Beginne bei Punkt $P_0 = (x_p, y_{min})$ und setze jedes Feld bis Punkt $P_1 = (x_p, y_{max})$.

Algorithmus

1. **for each** $i: y_{min} \leq i \leq y_{max}$ **do**
2. Setze Rasterpunkt $P = (x_p, i)$

2. Linie hat positiven Anstieg, $0 \leq m \leq 1$

Die Linie besitzt einen Anstiegswinkel im Intervall $[0, 45]$ Grad. Dies beinhaltet den Sonderfall, dass die Linie parallel zur x-Achse verläuft.

Der Bresenham-Algorithmus (siehe oben) inkrementiert x und setzt in jeder Spalte einen y-Wert.

3. Linie hat negativen Anstieg, $-1 \leq m < 0$

Die Linie besitzt einen Anstiegswinkel im Intervall $[-45, 0)$ Grad.

Der Bresenham-Algorithmus inkrementiert x und setzt in jeder Spalte einen y-Wert. Im obigen Algorithmus müssen folgende Zeilen ausgetauscht werden:

3. $d \leftarrow 2(-\Delta y) - \Delta x$

4. $\Delta E \leftarrow 2(-\Delta y)$

5. $\Delta NE \leftarrow 2((-\Delta y) - \Delta x)$

16. $y \leftarrow y - 1$

4. Linie hat positiven Anstieg, $m > 1$

Die Linie besitzt einen Anstiegswinkel im Intervall von $(45, 90)$ Grad.

Der Bresenham-Algorithmus inkrementiert nun y und setzt in jeder Zeile einen x-Wert. Im obigen Algorithmus müssen folgende Zeilen ausgetauscht werden:

3. $d \leftarrow 2\Delta x - \Delta y$

4. $\Delta E \leftarrow 2\Delta x$

5. $\Delta NE \leftarrow 2(\Delta x - \Delta y)$

9. **while** $y < y_1$ **do**

12. $y \leftarrow y + 1$

5. Linie hat negativen Anstieg, $m < -1$

Die Linie besitzt einen Anstiegswinkel im Intervall von $(-90, -45)$ Grad.

Der Bresenham-Algorithmus inkrementiert y und setzt in jeder Zeile einen x-Wert. Im obigen Algorithmus müssen folgende Zeilen ausgetauscht werden:

3. $d \leftarrow 2\Delta x + \Delta y$

4. $\Delta E \leftarrow 2\Delta x$

5. $\Delta NE \leftarrow 2(\Delta x + \Delta y)$

9. **while** $y > y_1$ **do**
12. $y \leftarrow y - 1$
16. $y \leftarrow y - 1$

Abschließend wird noch die 4-Nachbarschaft der Pixelkontur hergestellt, d.h. Pixel sind stets über Kanten und nicht nur über Ecken miteinander verbunden.

Nun liegt der Polygonzug des Randes als Rasterbild vor und kann zur Weiterbe- und -verarbeitung genutzt werden (Abbildung 2.13 und 2.14).



Abbildung 2.13: Randkontur als Polygonzug

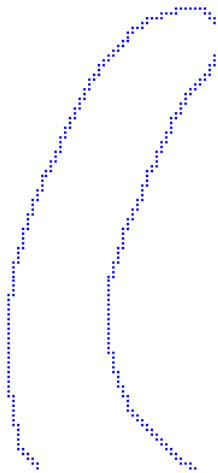


Abbildung 2.14: Diskretisierte Randkontur

3 Distanzmessungen im 2D

In vielen Anwendungen ist es von großer Bedeutung Abstände zwischen Objekten in diskretisierten Bildern messen zu können. So muss beispielsweise die Dicke einer Scherbe berechnet werden, oder man benötigt Informationen darüber, wie weit die Mittelachse von einer Polygonlinie entfernt ist. Im folgenden Kapitel wird diese Problematik aufgegriffen und näher beleuchtet (siehe auch [Loh98]).

3.1 Distanzmaße

Ein Distanzmaß muss alle drei Axiome bezüglich eines metrischen Raumes erfüllen ([Lju68]):

$$p, q, r \in P$$

1. Nichtnegativität: $d(p, q) \geq 0$ und $d(p, q) = 0 \Leftrightarrow p = q$
2. Symmetrie: $d(p, q) = d(q, p)$
3. Dreiecksungleichung: $d(p, r) \leq d(p, q) + d(q, r)$

3.2 Distance Mapping

Der einfachste denkbare Fall ist die Distanzmessung entlang einer eindimensionalen Linie. Angenommen es gibt auf ihr nur einen Merkmalspunkt, so wird dieser mit „0“ kodiert (es existiert keine Distanz) und alle anderen Pixel auf der Linie erhalten die Initialisierung „ ∞ “ (d.h. die Entfernung zum nächsten Merkmal wurde noch nicht berechnet). Algorithmen, welche die Distanzen zu Merkmalspunkten berechnen, werden auch Distanztransformationen genannt. Sie erstellen eine Distance

Map („Distanzkarte“), in der jedes Pixel einen Wert erhält, welcher aussagt, wie groß der Abstand zum nächsten Merkmalspunkt ist. Beispiel:

Initialisierung:	∞	∞	∞	0	∞	∞	∞	∞	0	∞
Distance Map:	3	2	1	0	1	2	2	1	0	1

Es gibt zwei Haupttypen von Algorithmen. Der erste Typ berechnet die exakten euklidischen Abstände, ist jedoch sehr zeit- und rechenintensiv, während der zweite Algorithmustyp die euklidischen Distanzen nur annähert, dabei aber schneller abläuft.

Zuerst soll die Grundidee am eindimensionalen Fall demonstriert werden, um später auf den zweidimensionalen überzugehen.

Distanzen in einem Bild sind globale Merkmale, welche das gesamte Bild betreffen und sich nicht nur auf eine kleine Nachbarschaft beziehen. Nichtsdestotrotz ist es von Bedeutung mit lokalen Operationen zu messen, da es sehr aufwendig ist, für jede Messung das komplette Bild nach dem kürzesten Pfad zu durchsuchen. Und tatsächlich eignen sich lokale Berechnungen, um Distanzen abzuschätzen.

Die Grundidee ist, dass sich lokale Distanzen durch Iterationen im Bild „ausbreiten“. Ein einfaches Beispiel soll die ersten Durchläufe des Verfahrens verdeutlichen. Dabei wird die Maske $1\ 0\ 1$ über das ganze Bild bewegt, die lokale Distanz des „Maskenpixels“ wird jeweils mit dem Wert der benachbarten Pixel addiert und das Minimum dieser Summen ergibt den neuen Wert:

Initialisierung:	∞	∞	∞	0	∞	∞	∞	∞	0	∞
1. Schritt:	∞	∞	1	0	1	∞	∞	1	0	1
2. Schritt:	∞	2	1	0	1	2	2	1	0	1

Die Anzahl der Berechnungsschritte, bis das komplette Bild korrekt vermessen ist, wird durch die größte Distanz von einem Merkmalspixel begrenzt. Im oberen Beispiel sind demnach drei Schritte nötig. Dieses Verfahren bietet sich vor allem bei einer parallelen Verarbeitung an, für eine Ein-Prozessor-Maschine hingegen sind sequentielle Algorithmen sinnvoller, welche in zwei (oder mehr) Durchläufen (engl. Passes) vermessen. Im oberen Beispiel wäre der erste Durchlauf (forward pass), das Ablufen von links nach rechts, und der zweite Durchlauf (backward pass) das

anschließende rückwärtige Ablaufen. Der Wert eines Pixels ist jeweils das Minimum aus forward und backward Pass. Die unteren Abbildungen verdeutlichen wiederum das Verfahren.

forward pass:

∞	∞	∞	0	∞	∞	∞	∞	0	∞
∞	∞	∞	0	1	∞	∞	∞	0	1
∞	∞	∞	0	1	2	∞	∞	0	1
∞	∞	∞	0	1	2	3	∞	0	1
∞	∞	∞	0	1	2	3	4	0	1

backward pass:

∞	∞	∞	0	1	2	3	4	0	1
∞	∞	1	0	1	2	3	1	0	1
∞	2	1	0	1	2	2	1	0	1
3	2	1	0	1	2	2	1	0	1

3.3 Distanzmetriken

Distanzen zwischen Punkten werden immer mittels einer vordefinierten Metrik berechnet oder angenähert.

Häufige Distanzmetriken sind zum Beispiel ([Pal00]):

1. Euklidische Distanz (Abb. 3.1)

Ist die lineare Verbindung zweier Punkte im Raum.

$$d_{euclidian}(p, q) = \sqrt{(p_x - q_x)^2 + (p_y - q_y)^2 + (p_z - q_z)^2}$$

2. City-Block Distanz (Abb. 3.2)

Die Distanz zwischen zwei Punkten ist definiert als die Summe der horizontalen und vertikalen Schritte, um die Punkte zu verbinden (4-Nachbarschaft im Rasterbild).

$$d_{cityblock}(p, q) = |p_x - q_x| + |p_y - q_y| + |p_z - q_z|$$

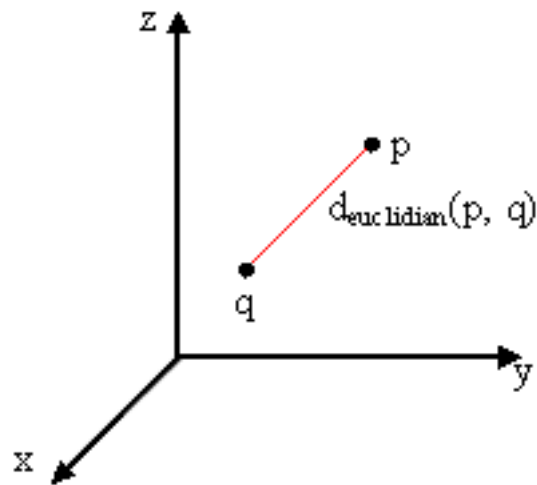


Abbildung 3.1: Euklidische Distanz im 3D

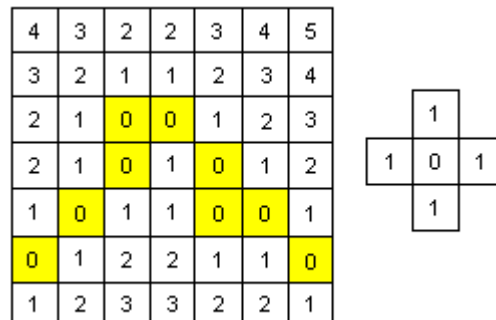


Abbildung 3.2: City-Block Distanz

3. Schachbrett Distanz (Abb. 3.3)

Ist bezüglich der City-Block-Distanz die 8-Nachbarschaft im Rasterbild.

$$d_{chessboard}(p, q) = \max\{|p_x - q_x|, |p_y - q_y|, |p_z - q_z|\}$$

2	2	2	2	2	2	3
2	1	1	1	1	2	2
2	1	0	0	1	1	2
1	1	0	1	0	1	1
1	0	1	1	0	0	1
0	1	1	1	1	1	0
1	1	2	2	2	1	1

1	1	1
1	0	1
1	1	1

Abbildung 3.3: Schachbrett Distanz

4. Chamfer Distanz

Kompromiss zwischen den ersten drei Metriken.

$$d_{chamfer}(p, q) = \sum_p^{q-1} d_{euclidian}(i, i+1)$$

Abbildung 3.4 zeigt die (3,4)-Chamfer Distanz:

8	7	6	6	7	8	11
7	4	3	3	4	7	8
6	3	0	0	3	4	7
4	3	0	3	0	3	4
3	0	3	3	0	0	3
0	3	4	4	3	3	0
3	4	7	7	6	4	3

4	3	4
3	0	3
4	3	4

Abbildung 3.4: Chamfer Distanz

Am Besten wäre es, wenn die Distanztransformation eine Distance Map erzeugen würde, welche die Euklidischen Distanzen enthält. Dies ist jedoch in einem digitalen Bild nicht trivial zu berechnen, da die Berücksichtigung des Pixelrasters den Euklidischen Abstand nicht zulässt und deshalb lediglich die Länge eines zusammen-

hängenden Weges (mit n-Nachbarschaft) berechnet wird. Abbildung 3.5 verdeutlicht das Problem.



Abbildung 3.5: Digitale Wege im 2D

Die Euklidische Distanz zwischen den zwei Punkten beträgt $\sqrt{12^2 + 9^2} = 15$. Der kürzeste digitale Pfad (mit 8-Nachbarschaft) kann die Distanz zwischen den zwei Punkten nur approximieren. Die einfachste Methode ist, die Summe der horizontalen, vertikalen und diagonalen Schritte entlang des Pfades zu bestimmen. Die Länge der horizontalen und vertikalen Schritte beträgt dabei beispielsweise 1 und die Länge der diagonalen $\sqrt{2}$. Damit ergibt sich folgende Gleichung

$$D = m_1 + \sqrt{2} * m_2$$

wobei m_1 die Anzahl der horizontalen und vertikalen, und m_2 die Anzahl der diagonalen Schritte darstellt. In Bezug auf das obige Beispiel ist die Distanz somit $3 + 9 * \sqrt{2} \approx 15.73$ und damit länger als der tatsächliche Euklidische Abstand.

Allgemein ausgedrückt, d.h. die Distanzen für die einzelnen Schritte stellen die Faktoren a und b dar, ergibt sich folgende lokale Distance Map:

$$D = a * m_1 + b * m_2$$

$$\begin{array}{ccc} b & a & b \\ a & 0 & a \\ b & a & b \end{array}$$

Die Schwierigkeit besteht nun darin, die Werte für a und b so zu wählen, dass die Differenz zwischen dem wahren Euklidischen Abstand und der angenäherten Metrik minimiert wird. Borgefors ([Bor86]) hat sich unter anderem mit der Optimierung dieses Problems befasst und folgende Werte ermittelt:

$$\begin{aligned} a &= 0.95509 \\ b &= 1.36930 \end{aligned}$$

Diese Metrik wird auch „Chamfer Distanz“ genannt (oben bereits erwähnt) und nähert sich gut an die Euklidische Distanz an.

3.4 Distanztransformationen

3.4.1 Zweidimensionale Distanztransformation (Chamfer)

Wie bereits in Kapitel 3.3 erwähnt, benötigen sequentielle Algorithmen zwei Durchläufe um eine Distance Map zu erzeugen. Ein Pass beginnt in der linken oberen Ecke und arbeitet sich zur rechten unteren Ecke des 2D-Bildes, während der Rückwärtsthroughlauf von rechts unten nach links oben läuft.

Bei jedem Durchlauf wird eine lokale Distance Map über das Bild bewegt, so dass sich lokale Distanzen entlang des Weges im ganzen Bild „ausbreiten“. Dabei werden für Vorwärts- und Rückwärtspass verschiedene Distanzmasken verwendet:

forward mask

$$\begin{array}{ccc} \infty & \infty & \infty \\ \infty & 0 & a \\ \infty & a & b \end{array}$$

backward mask

$$\begin{array}{ccc} b & a & \infty \\ a & 0 & \infty \\ \infty & \infty & \infty \end{array}$$

Die Werte für a und b können gewählt werden, wie unter 3.3 beschrieben.

In jedem einzelnen Schritt - z.B. bei Pixel i_0, j_0 - wird folgende Berechnung durchgeführt:

$$pixel_{i_0, j_0} = \min \{ mask_{i_0+i, j_0+j} + pixel_{i_0+i, j_0+j} \mid i, j = -1, 0, 1 \}$$

Mit anderen Worten, der aktuelle Wert jedes Pixels wird mit dem lokalen Distanzwert addiert, welcher aus der „überliegenden“ Maske entnommen wird. Das zentrale Pixel wird dann mit dem Minimum all dieser Summen ersetzt. Ein kleines Beispiel soll dies verdeutlichen.

	∞	∞	∞	∞	∞
<i>Ausgangsbild</i>	∞	0	∞	∞	∞
	∞	∞	∞	∞	∞
	∞	∞	∞	∞	∞

	∞	∞	∞	∞	∞
<i>forward pass: Erreichen des ersten Merkmalspixel</i>	∞	0	a	∞	∞
	∞	a	b	∞	∞
	∞	∞	∞	∞	∞

	∞	∞	∞	∞	∞
<i>Endergebnis des forward passes</i>	∞	0	a	2a	3a
	∞	a	b	(b+a)	(b+2a)
	∞	2a	(b+a)	2b	(2b+a)

3.4.2 Euklidische Distanztransformation

Saito und Toriwaki ([Sai94]) haben eine Methode entwickelt, die eine exakte Euklidische Distanztransformation berechnet. Sie ist natürlich langsamer als die approximierenden Distanztransformationen, da sie nicht nur lokale Berechnungen benutzt. Jedoch für Fälle, welche exakte Distanzen benötigen, ist diese Variante eine gute Alternative.

Saito und Toriwaki benutzen drei Verarbeitungsschritte, um die Distanzen im Dreidimensionalen zu berechnen. Für unsere Belange im 2D reichen daher die ersten zwei Schritte aus. In jedem Schritt werden die Distanzen entlang einer Richtungsachse berechnet.

Transformation 1

Zuerst werden die Abstände entlang der Zeilen berechnet. Das gegebene zweidimensionale digitale Bild ist wie folgt definiert (n_i und n_j stehen für die Anzahl der Zeilen bzw. Spalten):

$$\{v_{i,j} \in \{0,1\} \mid 0 \leq i < n_i, 0 \leq j < n_j\}$$

Im ersten Verarbeitungsschritt wird das Ausgangsbild Pixel für Pixel in ein Zwischenbild f_{ij} transformiert. Dabei wird jedes Hintergrundpixel folgendermaßen ersetzt:

$$f_{ij} = \min \left\{ (i-d)^2 \mid v_{dj} = 0, 0 \leq d < n_i \right\}$$

Vordergrundpixel haben natürlich keine Distanz (Wert gleich 0). Verdeutlicht werden soll dies am Beispiel einer Zeile:

Ausgangsbild:	1	0	0	1	0	0	0	0	0	1	1
Transformation 1:	0	1	1	0	1	4	9	4	1	0	0

Transformation 2

Für die zweite Transformation wird das Bild, welches in Transformation 1 erzeugt wurde, verwendet und jedes Pixel durch folgenden Wert ersetzt:

$$g_{ij} = \min \left\{ f_{id} + (j-d)^2 \mid 0 \leq d < n_j \right\}$$

Anhand eines weiteren Beispiels sollen die einzelnen Schritte und ihre Ergebnisse erneut gezeigt werden:

Ausgangsbild	Transformation 1	Transformation 2
0 0 0 0 0	1 4 9 4 1	1 2 5 4 1
1 0 0 0 0	0 1 4 4 1	0 1 4 4 1
0 0 0 0 0	1 4 9 4 1	1 2 5 2 1
0 0 0 0 1	1 4 4 1 0	1 4 4 1 0
0 0 0 0 0	1 4 9 4 1	1 4 5 2 1

Auffallend ist, dass die Pixel, welche den Rand einer Zeile berühren, den Distanzwert 1 besitzen. Dies ist der Fall, da der Algorithmus die Pixel, welche außerhalb der Bildgrenzen liegen, als Merkmalspixel definiert. Der Algorithmus berechnet also die Distanz zum nächsten Merkmalspixel und zum Bildrand und gibt den kleineren der zwei Werte zurück. Ist dies unerwünscht, so ist eine geringfügige Modifikation ausreichend um das Problem zu beheben.

Die nun vorliegende Distance Map enthält nach dem zweiten Verarbeitungsschritt die quadratischen euklidischen Abstände zu den nächsten Merkmalspixeln.

In Abbildung 3.6 sollen abschließend noch einmal die Unterschiede zwischen der Chamfer und der Euklidischen Distanztransformation gezeigt werden. Dabei stellen die Punkte in den zwei rechten Abbildungen diejenigen dar, welche die gleiche Distanz (anhand der jeweiligen Distance Map) zum Mittelpunkt (linkes Bild) besitzen.

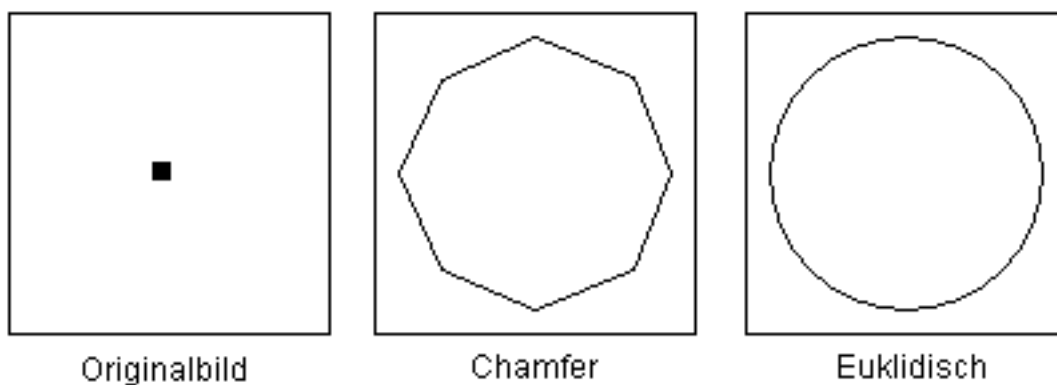


Abbildung 3.6: Unterschiedliche Distanzmetriken

4 Berechnung der Mittelachse

Es existieren verschiedene Berechnungsmethoden, um die Mittelachse eines Polygonzuges, auch Skelett genannt, zu berechnen. Dabei beziehen sich die Methoden jedoch auf unterschiedliche Ausgangskörper, so dass verschiedene Mittelachsen entstehen können. Außerdem ist es häufig auch eine Frage der Berechnungsgeschwindigkeit, welche Methode Verwendung findet.

Nach einigen grundsätzlichen Definitionen sollen die Berechnungsmethoden näher beleuchtet werden.

4.1 Definitionen

4.1.1 Mittelachse

Die Mittelachse bzw. die mediale Achse wurde 1967 von Harry Blum ([Blu67]) zur Darstellung und Analyse ebener und abgeschlossener Gebiete eingeführt. Die Definition lautet wie folgt (nachzulesen z.B. bei [Wip97] oder [Arm]):

Die Mediale Achse M eines 2D-Gebietes U besteht aus dem Abschluss der Menge aller Mittelpunkte maximaler Kreisscheiben (MP). Dabei heißt eine Kreisscheibe K dann maximal, wenn sie ganz in dem 2D-Gebiet U enthalten und nicht echte Teilmenge einer ebenfalls in dem zu betrachtenden Gebiet liegenden Kreisscheibe K' ist.

Mathematisch ausgedrückt:

$$M = \overline{MP}$$

$$K \subseteq U \text{ ist maximal} \Leftrightarrow \forall K' \subseteq U : K \cap K' \neq K$$

Die Mittelachse ist also die Menge aller Punkte, die mindestens zwei nächste Punkte zur Konturlinie besitzen.

Die „örtliche Merkmalsgröße“ (local feature size) eines Punktes P auf der Kontur ist die Distanz zwischen P und dem am nächsten gelegenen Punkt auf der Mittelachse.

In Abbildung 4.1 wird die Mittelachse eines Rechtecks verdeutlicht. Die Kreismittelpunkte A und B gehören dabei zur medialen Achse, während der Kreismittelpunkt C nicht dazu gehört.

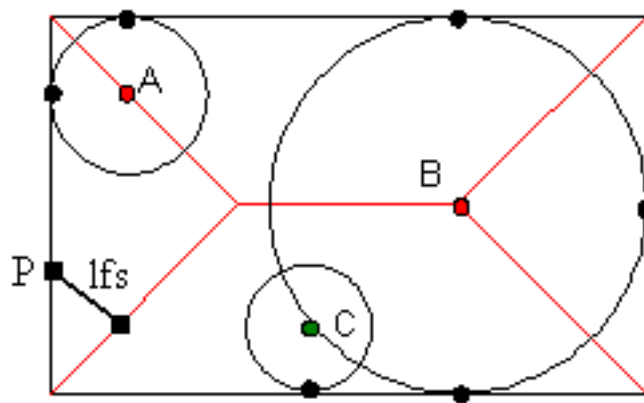


Abbildung 4.1: Mittelachse eines Rechtecks

Laut [Rom] gilt die obige Definition nur im kontinuierlichen Raum, da im diskreten Raum die Mediale Achse nur approximiert werden kann.

4.1.2 Topologie

Topologische Charakteristika eines Körpers sind die Eigenschaften, die durch „flexible“ Transformationen (z.B. Streckung, Drehung, Verbiegung) nicht verändert werden ([Loh98]). Dazu gehört u.a. das Geschlecht, welches für die Anzahl der Löcher im Objekt steht. Die angewendeten Transformationen müssen die Topologie des Körpers erhalten.

Die Topologie gilt als nicht erhalten ([Pal00]), wenn

- ein Objekt in mehrere aufgetrennt wird
- ein Objekt komplett gelöscht wird
- ein Loch (Hintergrundkomponente, die von Objekt umgeben ist) entsteht
- ein Loch mit dem Hintergrund vermischt wird
- zwei Löcher zu einem verschmelzen
- ein Loch verschwindet

Skelettierungstechniken sollen die Einhaltung dieser Regeln durch Einsatz konkreter Kriterien umsetzen, da die berechnete Mittelachse den gegebenen Körper repräsentieren soll. Das heißt aus der Mittelachse muss der Körper wieder herstellbar sein und alle geometrisch erforderlichen Informationen sollten aus der Mittelachse abgeleitet werden können.

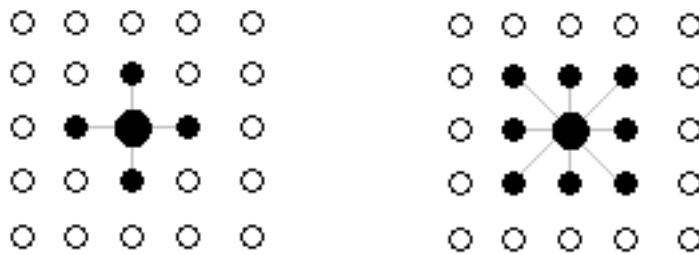
Obwohl die Topologie eine globale Eigenschaft eines Körpers ist, kann sie mit Hilfe lokaler Betrachtungen überprüft werden. Es reicht aus, die Umgebung, also die direkte Nachbarschaft, um die Stelle zu betrachten, die einer Änderung unterliegt (siehe 4.1.3).

4.1.3 Vorder- und Hintergrundkomponente

Für die lokalen topologieerhaltenden Kriterien werden sogenannte Vorder- und Hintergrundkomponenten ausgewertet. Um diese erklären zu können, muss jedoch erst einmal der Begriff der Nachbarschaft verdeutlicht werden.

Im 2D besitzt jedes Pixel P vier Nachbarn, deren Rasteradressen sich nur in einer Koordinate von P um eins unterscheiden. Weiterhin gibt es vier diagonale Nachbarn, bei denen beide Koordinaten von P um eins verschieden sind ([Loh98]). Die Menge der vier direkten Nachbarn wird die N_4 -Nachbarschaft eines Pixels genannt. Kommen noch die diagonalen Nachbarn hinzu, so handelt es sich um die N_8 -Nachbarschaft (Abbildung 4.2).

Unter einer Vordergrundkomponente versteht man demnach gefüllte Pixel, die untereinander mit einer N_8 -Nachbarschaft verbunden sind. Hintergrundkomponenten sind hingegen nicht gefüllte Pixel, die untereinander nur mit einer N_4 -Nachbarschaft

Abbildung 4.2: N_4 - und N_8 -Nachbarschaft

verbunden sind.

Am deutlichsten wird der Sachverhalt mit Hilfe eines Beispiels (Abb. 4.3):

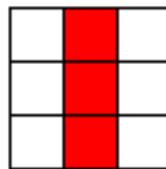


Abbildung 4.3: 1 Vorder- und 2 Hintergrundkomponenten

Wird nun das mittlere Pixel gelöscht, so verändert sich der Fall folgendermaßen:

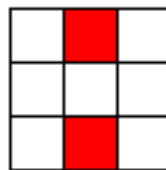


Abbildung 4.4: 2 Vorder- und 1 Hintergrundkomponente

Der Algorithmus zur Berechnung der Vordergrundkomponenten in der direkten Nachbarschaft eines Pixels nutzt ein lineares Feld $State[8]$. Dabei wird der Eintrag an Position k , dann auf $true$ gesetzt, wenn das entsprechende Nachbarpixel k gefüllt ist,

andernfalls ist der Eintrag *false*.



Abbildung 4.5: Nachbarschaft eines Pixels mit linearem Feld

Ist das Pixel $P = (i, j)$ gefüllt, dann ist die Anzahl der Vordergrundkomponenten $VK = 1$, da P mit allen Nachbarpixel durch die N_8 -Nachbarschaft verbunden ist. Ist P jedoch nicht gefüllt, so ergibt die Auswertung des linearen Feldes die Vordergrundkomponenten:

Algorithmus *Vordergrundkomponentenberechnung*

Input: *State*[8]

Output: Anzahl der Vordergrundkomponenten

/ Vordergrundkomponenten = VK */*

1. $VK \leftarrow 0$
2. $VorgaengerVK \leftarrow State[7]$
3. **for each** k : ($0 \leq k < 8$) **do**
4. **if** $VorgaengerVK = \text{false}$ **then**
5. **if** $State[k] = \text{true}$ **then**
6. $VK \leftarrow VK + 1$
7. $VorgaengerVK \leftarrow \text{true}$
8. **else**
9. **if** $State[k] = \text{false}$ **then**
10. $VorgaengerVK \leftarrow \text{false}$
- 11.

```

12. /* Eckverbindungen 1-3, 3-5, 5-7, 7-1 nur testen, wenn dazwischen Lücken
    (Eintrag = false) sind. */
13. if State[2] = false then
14.   if (State[1] = true) and (State[3] = true) then
15.      $VK \leftarrow VK - 1$ 
16. if State[4] = false then
17.   if (State[3] = true) and (State[5] = true) then
18.      $VK \leftarrow VK - 1$ 
19. if State[6] = false then
20.   if (State[5] = true) and (State[7] = true) then
21.      $VK \leftarrow VK - 1$ 
22. if State[0] = false then
23.   if (State[7] = true) and (State[1] = true) then
24.      $VK \leftarrow VK - 1$ 
25. return VK

```

Die Berechnung der Hintergrundkomponenten erfolgt ähnlich, nur dass der Eintrag k in Feld *State*[8] dann *true* ist, wenn das Nachbarpixel nicht gefüllt ist, andernfalls ist er *false*. Ist das Pixel $P = (i, j)$ gefüllt, so werden die Tests ab Zeile 12 nicht durchlaufen. Diese Tests sind nur bei leerem P notwendig, außerdem muss zusätzlich getestet werden, ob die berechneten Hintergrundkomponenten über Pixel P mit einer Kante verbunden sind. Ist dies der Fall, so wird die Anzahl der Hintergrundkomponenten dekrementiert.

4.2 Methoden

Da im diskreten Raum die Mittelachse nur an das „wahre Skelett“ angenähert werden kann, gelten laut [Pal00] folgende Anforderungen:

- topologische Anforderung (die Topologie bleibt erhalten, das Originalobjekts kann aus der Mittelachse rekonstruiert werden)
- geometrische Anforderung (die Mittelachse liegt im Inneren des Objektes, ist mittig gelagert und unabhängig von Translation, Rotation und Skalierung)

Es existieren drei Hauptskelettierungstechniken:

1. Distanztransformation
2. Voronoi-Skelettierung
3. Ausdünnung

In folgender Tabelle werden die Erfüllungen der Anforderungen der einzelnen Methoden gegenüber gestellt.

Methode	geometrisch	topologisch
Distanztransformation	ja	nein
Voronoi-Skelettierung	ja	ja
Ausdünnung	nein	ja

Tabelle 4.1: Vergleich der Skelettierungsmethoden

4.2.1 Distanztransformation

Mit Hilfe einer bestimmten Metrik wird zu jedem Punkt des Objektes ein Abstand zum Rand berechnet. Anschließend wird der Grat (lokale Extrema), auch „Ridges“ genannt, gesucht, welcher Teilmenge der gesuchten Mittelachse ist.

Das verwendete metrische Distanzmaß (siehe 3.3) entscheidet maßgeblich über das Ergebnis und die Komplexität der Berechnung.

Die Distanztransformation weist eine gute Berechnungsgeschwindigkeit auf ([Bru03]), denn der Algorithmus besitzt, da nicht mehrfach iteriert werden muss, eine lineare Laufzeit $O(n)$ in der Anzahl der Bildelemente (Pixel im 2D). Da die Distanztransformation jedoch nur Ridges liefert, müssen anschließend Algorithmen verwendet werden, die diese verbinden. Solche Algorithmen erhöhen natürlich die Laufzeit.

Die Methode erfüllt zwar die geometrische Anforderung (wenn z.B. ein fehlerfreies euklidisches Distanzfeld berechnet wurde), aber eine topologische Korrektheit ist nicht garantiert. Die Distanztransformation ist abhängig von der Auflösung und der Metrik, mit der das Objekt vermessen wurde, dadurch kann das rekonstruierte Objekt vom ursprünglichen abweichen.

Bewertung der Methode

Obwohl diese Methode die Topologie des Objektes nicht bewahrt, könnte sie für den Einsatz im 2D brauchbar sein. Denn mit einer „guten“ Metrik und einer ausreichenden Auflösung lassen sich die Fehlerquellen verringern, und ein schneller Algorithmus ist gefunden.

Für unseren Fall ist der Topologieerhalt des Objektes jedoch von großer Bedeutung, so dass der Einsatz der Distanztransformation nicht in Frage kommt.

4.2.2 Voronoi-Diagramm

Bei einem Voronoi-Diagramm im 2D wird die Ebene mit den Punkten P_i , $i = 0, \dots, N$ in sogenannte Voronoi-Regionen aufgeteilt ([Van04]).

Die Voronoi-Region R_i ist die Menge der Punkte x , die näher zum Punkt P_i liegen, als zu einem anderen Punkt von M ($\|x, P_i\| < \|x, P_j\|$, $j = 0, \dots, N - 1$; $j \neq i$). Das In-

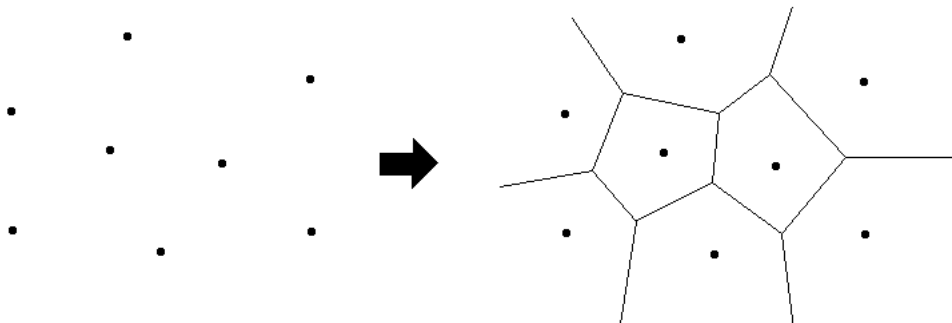


Abbildung 4.6: Voronoi-Diagramm einer Punktmenge

terne Voronoi Skelett (IVS) ist ein Teilgraph des Voronoi-Diagramms eines Objektes. Es liegt im Inneren des Objektes und ist der Ort, der zwei minimale Entfernungen zum Objektrand besitzt, womit die Bedingungen für ein Skelett erfüllt wären. Wenn die Anzahl der Punkte der Objektkontur gegen Unendlich geht, dann fällt das Interne Voronoi Skelett mit der Mittleren Achse der Kontur zusammen. Bei einer diskreten Kontur wird die mittlere Achse vom IVS lediglich approximiert.

Das Interne Voronoi Skelett zu berechnen, ist laut [Bru03] kein großer Aufwand,

wenn man das zum Objekt gehörige Voronoi-Diagramm bereits berechnet hat. Die Berechnung dieses Voronoi-Diagramms ist allerdings ein sehr aufwendiger und zeitintensiver Prozess.

Daher wird die Voronoi-Skelettierung aufgrund ihrer hohen Zeitkomplexität selten umgesetzt, obwohl sie die topologische und geometrische Anforderung erfüllt.

Bewertung der Methode

Bei großen Datensätzen ist die Erstellung des Voronoi-Diagramms zeitintensiv und besitzt eine Laufzeit von $O(n \log n)$.

Durch die exakte Berechnung der Mittelachse entstehen starke Verästelungen. Deshalb müssten in der Nachbereitung noch Algorithmen verwendet werden, die die Mittelachse vereinfacht, um ein Vergleichen mit anderen Medialen Achsen zu ermöglichen.

Für unseren Fall ist diese exakte Berechnung nicht nötig und die Voronoi-Skelettierung daher eher ungeeignet.

4.2.3 Ausdünnung

Die Ausdünnung, auch Thinning genannt, ist eine iterative Objektreduktionstechnik, das heißt es wird wiederholt der Randbereich des Objektvolumens, im 2D des Flächeninhaltes, abgeschält. Diese Löschoption der Randelemente wird auch als Erosion bezeichnet.

Der allgemeine Ausdünnungsalgorithmus ist in drei Schritte unterteilt, die stets wiederholt werden, bis kein Randelement mehr gelöscht werden kann ([Rom]):

```
while löschrare Randelemente vorhanden do  
    Identifiziere und markiere Randelemente  
    Untersuche markierte Elemente auf Löschrbarkeit  
    Löschr entsprechende Elemente
```

Die Technik des Ausdünnens besitzt einige günstige Eigenschaften ([Pal00]):

- sie bewahrt die Topologie des Objektes
- sie bewahrt die Form (signifikante Merkmale bleiben erhalten)
- die Mittelachse liegt zwangsweise im Inneren des Objektes
- die Mittelachse besitzt eine Stärke von einem Pixel (2D)

Um das Thinning im Zweidimensionalen auf Polygonzüge anwenden zu können, muss jedoch vorher ein „Flächeninhalt“ erzeugt werden, die Objektberandung muss sozusagen gefüllt werden.

Objektrandbefüllung

Um ein Objekt zu „füllen“, existieren im 2D folgende Ansätze ([Bru03]):

1. Floodfill:

Hier benötigt man einen Punkt im Inneren des Objektes. Ist dieser gefunden, so wird von ihm aus solange alles „ausgefüllt“, bis der Rand erreicht wird.

Dies kann rekursiv implementiert werden. Es ergeben sich jedoch auch einige Probleme: Zum Beispiel beim Finden des inneren Punktes für nicht geschlossene Objekte oder bei Verengungen, die die Füllung nicht „hindurchlassen“.

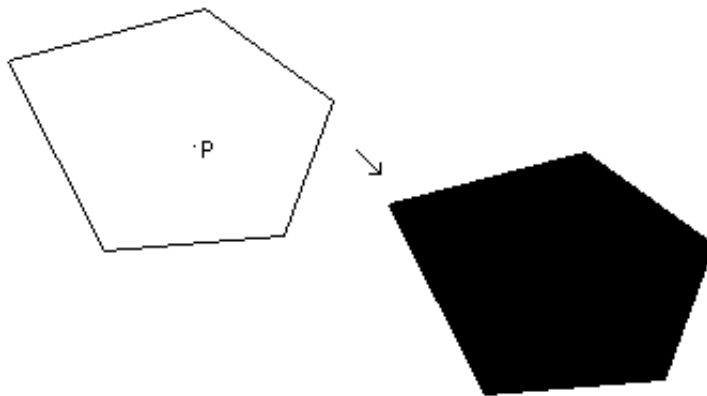


Abbildung 4.7: Floodfill vom Punkt P aus

2. Scanline- / Abtastmethode:

Hierbei wird eine Scanline von oben nach unten verschoben und die Schnittpunkte mit dem Rand des Objektes werden berechnet.

Bei geschlossenen Kurven ist die Anzahl der Schnittpunkte gerade, es wird also jeder ungerade Abschnitt zwischen den Schnittpunkten gefüllt (Abbildung 4.8).

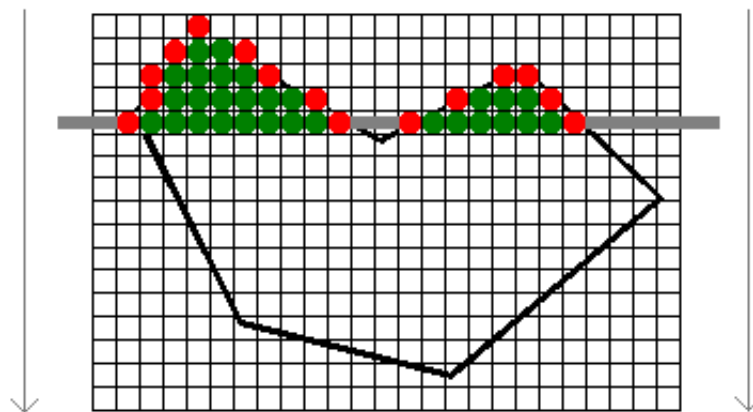


Abbildung 4.8: Scanlineverfahren bei geschlossenem Objekt

Bei nicht geschlossenen Kurven wird das Verfahren erweitert, denn nun wird nicht nur ein Scan durchgeführt, sondern mehrere aus verschiedenen Richtungen. Eine Schnittmenge aus diesen verschiedenen Scans beschreibt schließlich die Füllung des Objektes (Abbildung 4.9).

Bewertung der Methode

Ausdünnung erfüllt die Anforderungen in folgender Prioritätsreihenfolge:

1. Verbundenheit
2. Dünnhheit
3. Rekonstruktion

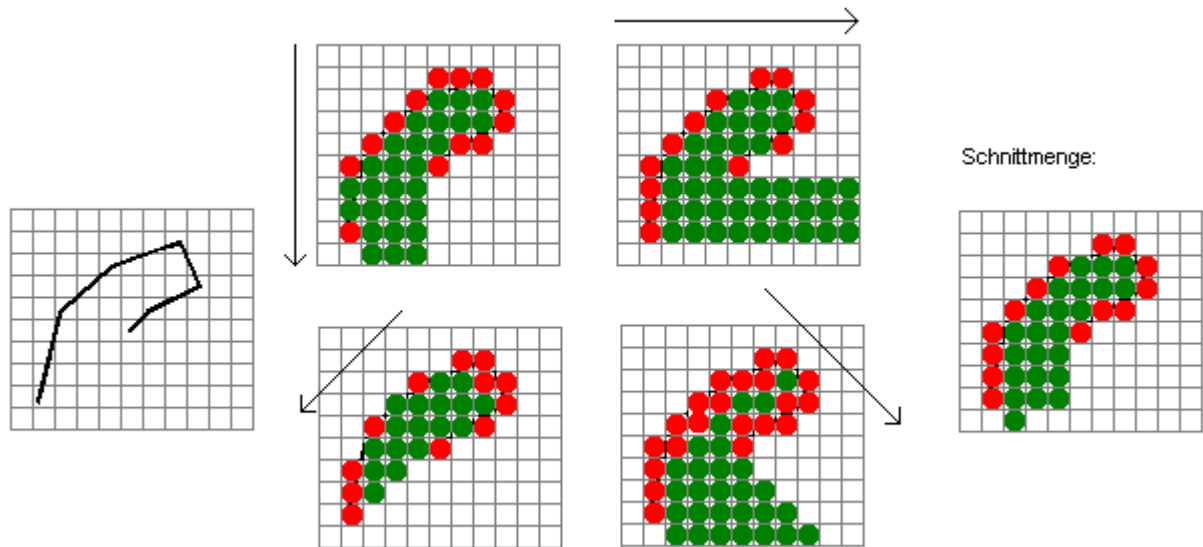


Abbildung 4.9: Scanlineverfahren bei nicht geschlossenem Objekt

Die Laufzeit des Algorithmus beträgt $O(n^3)$, dabei ist n der Wert der kleineren Objektausdehnung bezüglich der Koordinaten. Auf den ersten Blick scheint dies zwar eine hohe Laufzeit zu sein, jedoch ist in unserem Fall die maximale Objektausdehnung 100×100 , so dass der Algorithmus nach maximal einer Million Iterationen beendet ist und somit nahezu eine konstante Laufzeit besitzt.

Aufgrund dieser guten Eigenschaften lässt sich der Thinningalgorithmus im 2D gut verwenden.

Detaillierter Algorithmus

Der Ablauf des Ausdünnungsalgorithmus gestaltet sich wie folgt:

Algorithmus *Thinning*

1. $noDelete \leftarrow 0$
2. **while** $noDelete \neq 4$ **do**
3. $noDelete \leftarrow 0$
4. **for each** Richtung (+x, -y, -x, +y) **do**

5. Randvoxel identifizieren und markieren
6. Lösbarkeit der Pixel testen
7. Pixel löschen
8. **if** keine Pixel gelöscht **then**
9. $noDelete \leftarrow noDelete + 1$

Die Lösbarkeit eines Pixels wird mit folgenden Bedingungen geprüft:

1. Besitzt ein Pixel weniger als zwei Nachbarn, so darf es nicht gelöscht werden. Dadurch ist gesichert, dass wichtige Pixeläste erhalten bleiben.
2. Ist die Nachbarpixelanzahl jedoch größer gleich zwei, so werden als weitere topologieerhaltende Kriterien die Vorder- und Hintergrundkomponenten (siehe 4.1.3) des zu betrachtenden Pixels verwendet. Ändert sich deren Anzahl bei Entfernen des Pixels, so darf es nicht gelöscht werden und eine Teilung der Mittelachse wird verhindert.

Das Ergebnis des Ausdünnungsalgorithmus bei einem Randobjekt ist in Abbildung 4.10 zu betrachten.



Abbildung 4.10: Randobjekt mit Mittelachse

5 Randmerkmale

In diesem Kapitel wird genauer auf die einzelnen Randmerkmale eingegangen, das heißt es werden die Verfahren zur Klassifizierung der Merkmale detailliert erläutert. Dabei wurden bekannte Daten von mir analysiert, um manuell Muster in Daten zu erkennen und somit Wissen abzuleiten. Entscheidungsbäume fanden hierbei ihren Einsatz. Die Attribute, nach denen die Entscheidungen innerhalb des Baumes getroffen werden, wurden also nach meinen Untersuchungen gewählt. Um die Einordnungen der Ränder nachvollziehen zu können, findet sich zu jedem Randmerkmal ein Entscheidungsbaum zum besseren Verständnis.

5.1 Randstellung

Die Randstellung ist das charakteristischste Merkmal der Randform.

Eine Auflistung möglicher Randstellungen findet sich bei [Paa02] (Abbildung 5.1). Um einen gegebenen Rand bezüglich der Randstellung zu klassifizieren, ist es hilfreich die Mittelachse des Objektes zu berechnen und auszuwerten, da eine Mittelachse eine lokale Symmetrieachse darstellt, die dimensionsreduzierend ist und dadurch eine Analyse ebener Gebiete ermöglicht ([Wip97]). Die Mittelachse beinhaltet also die Informationen, wie sich der Randverlauf verhält, und sie ist ideal auswertbar.

Um die Mediale Achse eines Randobjektes zu bestimmen, wird ein Ausdünnungsalgorithmus verwendet. Anschließend wird der Verlauf der Winkel der Mittelachse untersucht und hinsichtlich der Randstellung ausgewertet.

Randform: Randstellung

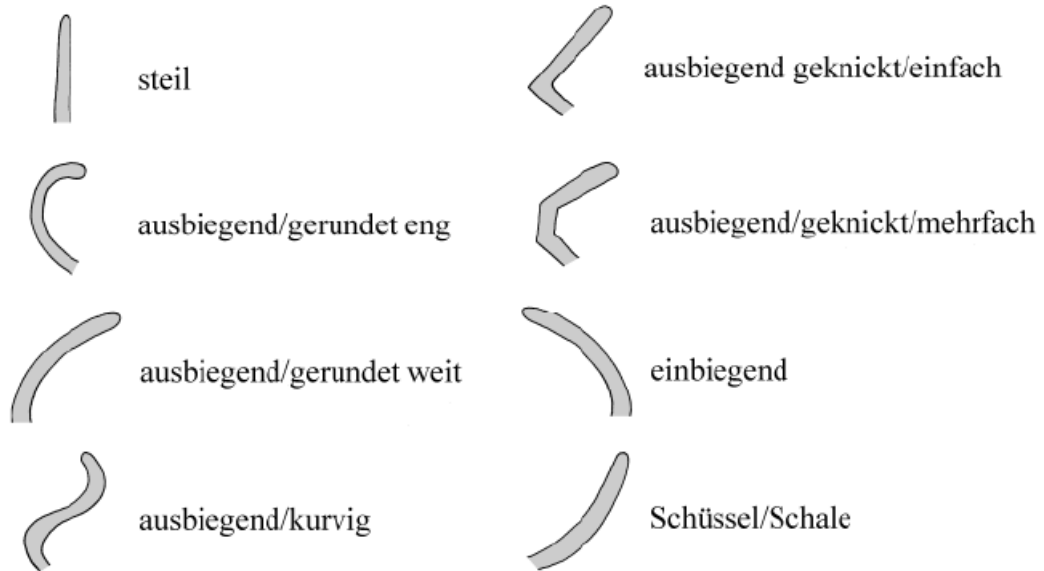


Abbildung 5.1: Schema zur Randstellung

5.1.1 Mittelachsenbestimmung

Nachdem die Randkontur gedreht und diskretisiert wurde, wird sie mit Hilfe des Scanlineverfahrens (siehe unter 4.2.3 „Objektrandbefüllung“) gefüllt. Schließlich liegt ein Objekt wie in Abbildung 5.2 vor.

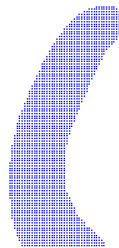


Abbildung 5.2: Diskretisiertes gefülltes Randobjekt

Danach wird der Thinning-Algorithmus (siehe 4.2.3) auf das Objekt angewendet und die Mittelachse berechnet.

Da es sich bei den Randkonturen um zusammenhängende, aber nicht geschlossene Polygonzüge handelt, entsteht beim Füllen dort, wo die Randkontur geöffnet ist, ein Objektrand, der nicht dem echten Objektrand entspricht. Dadurch kommt es beim Ausdünnungsalgorithmus zu einer fehlerhaften Berechnung der Mittelachse. In Abbildung 5.3 stellen die roten Pixel den echten und die dunkelgrünen den unechten Objektrand dar. Die echten Objektrandpixel besitzen jedoch einen höheren Wert und sind dadurch von den unechten unterscheidbar.



Abbildung 5.3: Fehlerhafte Mittelachse

Die fehlerhafte Berechnung ist ein Sachverhalt, der Ungenauigkeiten in die Auswertung der Mittelachse bringt, daher bedarf es einer Nachbesserung. Eigenschaft der Mittelachse ist es ja, dass sie in der Mitte des Objektes liegt und den gleichen Abstand zu beiden Seiten zum Objektrand besitzt. Mit diesem Wissen ist es ein Leichtes die Mittelachse zu bereinigen, denn man muss sie lediglich so lang „von unten nach oben“ traversieren, bis sich ein Pixel findet, bei dem die beiden nächsten zugehörigen Objektrandpixel echte Randpixel sind. Ist dies der Fall, so gehören alle folgenden Mittelachsenpixel zur Medialen Achse.

Nach diesem Schritt erhält man eine korrekte Mittelachse, wie in Abbildung 5.4.

Mittelachsenpolygonzug

Um die Mittelachse bezüglich ihrer Krümmungseigenschaften auswerten zu können, muss sie nun aus dem diskretisierten Raum wieder in einen Polygonzug überführt werden.



Abbildung 5.4: Korrigierte Mittelachse

Dabei stellt jedes Pixel einen Punkt im Zweidimensionalen dar und benachbarte Pixel respektive Punkte werden mit einer Linie verbunden. Weist die Mittelachse Verästelungen auf, so wird jeweils der längste Ast zum Polygonzug hinzugefügt (dies wird auch „Entbartung“ oder „Pruning“ genannt).

Da dieses simple Verfahren jedoch einen sehr stufigen Polygonzug erzeugt, ist eine Glättung notwendig:

- Alle $n+1$ Punkte werden traversiert und für jeden Punkt P_i ($0 < i < n$) wird eine neue Position P'_i berechnet.

$$P'_i = \frac{1}{3} \sum_{j=i-1}^{i+1} P_j$$

Dieses Glättungsverfahren wird zehnmal auf die Mittelachse angewendet. Dadurch glättet sich nicht nur der Verlauf des Polygonzugs, sondern auch die Abstände zwischen den einzelnen Punkten werden in etwa gleich groß.

5.1.2 Winkelberechnung

Für jeden Punkt P_i ($0 < i < n$) der Mittelachse wird nun der Winkel α_i berechnet, den er bezüglich seiner Nachbarn besitzt (Abbildung 5.5).



Abbildung 5.5: Winkel eines Mittelachsenpunktes

Da die Punkte P_0 und P_n jedoch nur einen Nachbarn besitzen, wird ihnen der Wert 180 zugeteilt.

Auswertung

Um die Randstellung mit Hilfe der berechneten Winkel zu ermitteln, gibt es zwei Möglichkeiten:

1. Auswertung mittels Prototypen
2. Auswertung mittels Schwellwerten

Bei der ersten Variante werden Mittelachsen von Randkonturen ermittelt, die der Klassifikation entsprechen (Abbildung 5.1.), anschließend wird ein normiertes Winkeldiagramm erstellt ($\forall i, i = 0 \dots 99 : \in Winkel[i]$) und abgespeichert. Das Dateiformat gestaltet sich folgendermaßen:

```
#ArchViewer - TU Chemnitz
#Winkeldaten
180.000000
178.096359
177.251274
[....]
180.000000
```

Abbildung 5.6: Dateiformat der Winkeldaten

Für den zu klassifizierenden Rand wird nun ebenfalls ein normiertes Winkeldiagramm berechnet, um es mit den Prototypen zu vergleichen. Dabei werden die Differenzen zwischen dem Prototyp und dem zu klassifizierenden Rand bestimmt und aufsummiert (Abbildung 5.7).

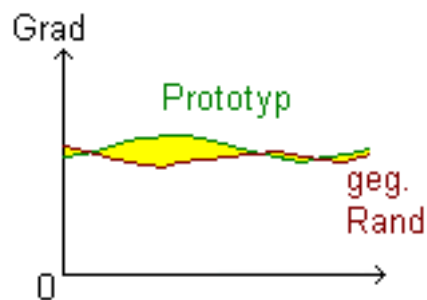


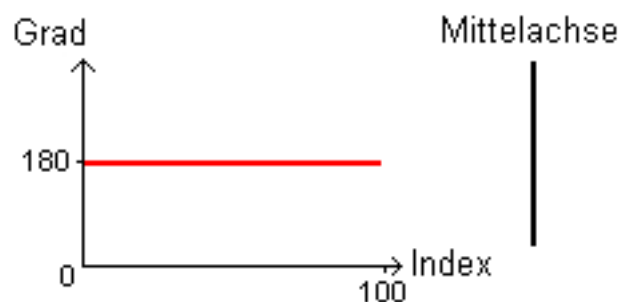
Abbildung 5.7: Vergleich Prototyp mit gegebenem Rand

Die gesuchte Randstellung ist schließlich die, deren Prototyp am ähnlichsten ist (d.h. deren Abweichung am geringsten ist).

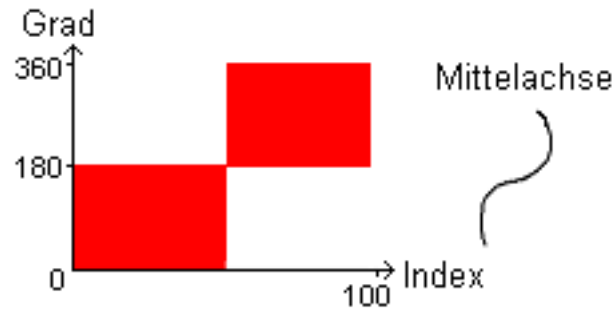
Die Auswertung mittels Schwellwerten erfolgt fast analog, nur dass die Klassifizierung mit Hilfe spezifischer Werte durchgeführt wird. Folgende Untersuchungsstufen werden durchlaufen (die einzelnen Schwellwerte ergaben sich bei Untersuchungstests):

1. Untersuchungsstufe

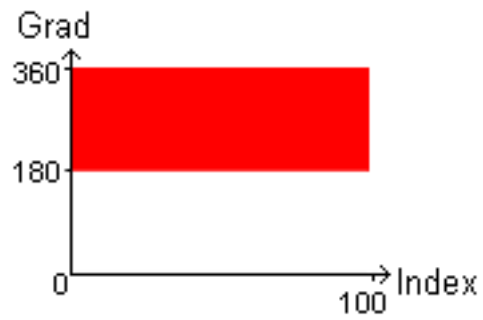
- Steil



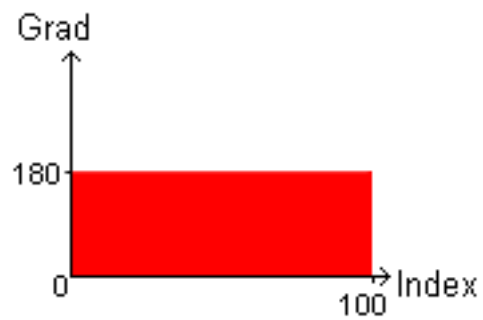
- Kurvig



- Einbiegend/Schale (weiter bei Untersuchungsstufe 2a)



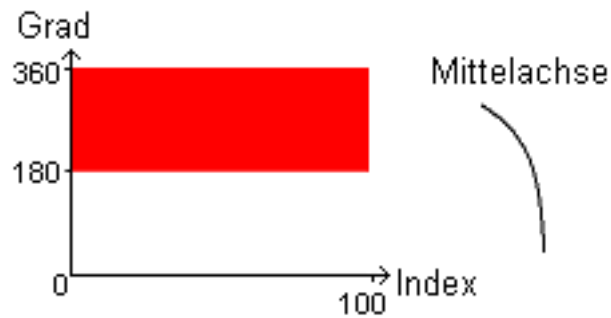
- Ausbiegend (weiter bei Untersuchungsstufe 2b)



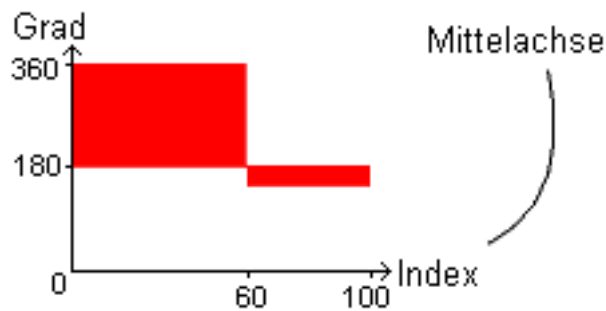
2. Untersuchungsstufe

a) Einbiegend oder Schale

- Einbiegend



- Schüssel/Schale



Bei der Entscheidung, ob der Rand „Einbiegend“ oder „Schüssel/Schale“ ist, spielt zusätzlich das Verhältnis Randgröße-Gefäßgröße eine Rolle. Macht der Rand mind. ein Drittel der Gefäßgröße aus, so handelt es sich um den Typ „Schüssel/Schale“.

b) Gerundet oder Geknickt

Es wird die Abweichung der einzelnen Winkel zu 180 Grad berechnet ($Abweichung = \sum_{i=0}^n 180 - Winkel[i]$), sowie die Anzahl möglicher Knickstellen (lokale Minima kleiner als 176 Grad).

- Geknickt (weiter bei Untersuchungsstufe 3a)

Wenn

$$(Abweichung < 45) \text{ AND } (0 < \text{Knickstellen} < 3)$$

gilt, so ist die Mittelachse geknickt.

- Gerundet (weiter bei Untersuchungsstufe 3b)

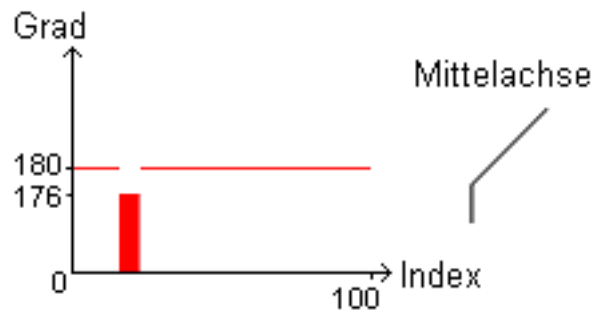
Gilt obige Bedingung nicht, so ist die Mittelachse gerundet.

3. Untersuchungsstufe

a) Einfach oder Mehrfach geknickt

- Einfach

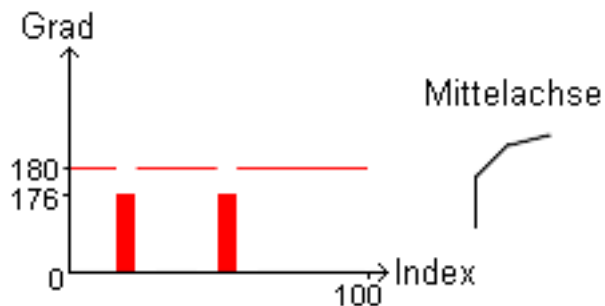
Wenn die Anzahl der Knickstellen gleich eins ist, so handelt es sich um eine einfach geknickte ausbiegende Mittelachse.



Die Knickstelle kann sich überall auf der Mittelachse befinden.

- Mehrfach

Ist die Anzahl der Knickstellen größer eins, so handelt es sich um eine mehrfach geknickte ausbiegende Mittelachse.



Die Knickstellen können überall auf der Mittelachse liegen.

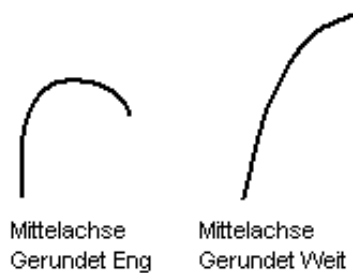
b) Eng oder Weit Gerundet

- Eng
Wenn

$$[(\text{Abweichung} > 80) \text{ AND } (\text{Knickstellen!} = 0)] \text{ OR } [\text{Knickstellen} > 2]$$

gilt, so ist die Mittelachse ausbiegend gerundet eng.

- Weit
Gilt obige Bedingung nicht, so ist die Mittelachse ausbiegend gerundet weit.



Da bei der Prototypvariante bereits in den Klassifizierungsdiagrammen Fehler entstehen können (Rundungsfehler etc.), ist die Schwellwertauswertung vorzuziehen, da sie die besseren Ergebnisse liefert. Außerdem kann ein Prototyp nicht alle Fälle seiner Variationen abdecken. Am deutlichsten wird dies bei der Randstellung „geknickt einfach“. Liegt der Knick beim Prototyp im ersten Drittel, so wird ein Rand,

bei dem sich der Knick im letzten Drittel befindet, nicht als solcher erkannt (siehe Abbildung 5.8). Daher beinhalten die Prototypen bereits Fehler in sich.

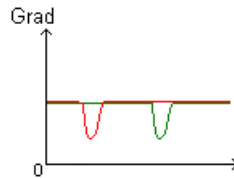


Abbildung 5.8: Fehlerhafter Prototypvergleich

5.1.3 Entscheidungsbaum

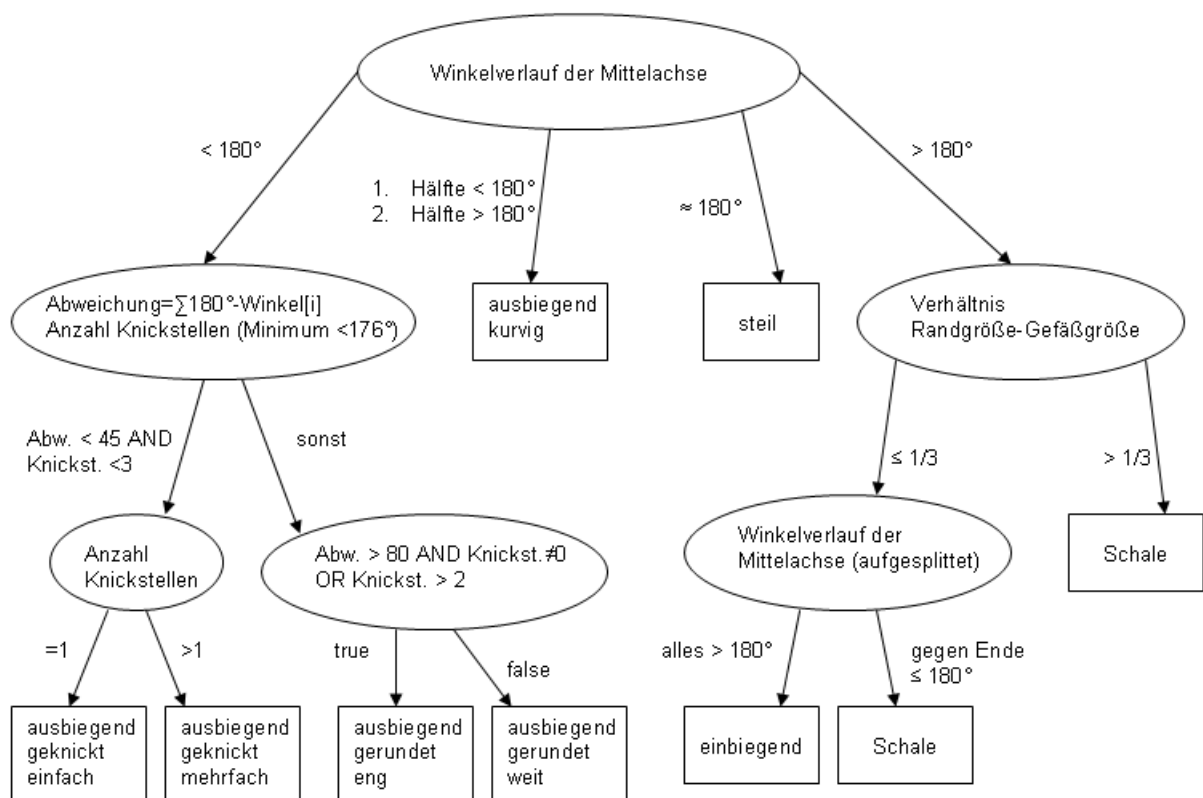


Abbildung 5.9: Entscheidungsbaum Randstellung

5.2 Lippenbildung

Als ein weiteres Merkmal der Randform ist die Lippenbildung zu nennen. Mögliche Formen sind bei [Paa02] veranschaulicht:

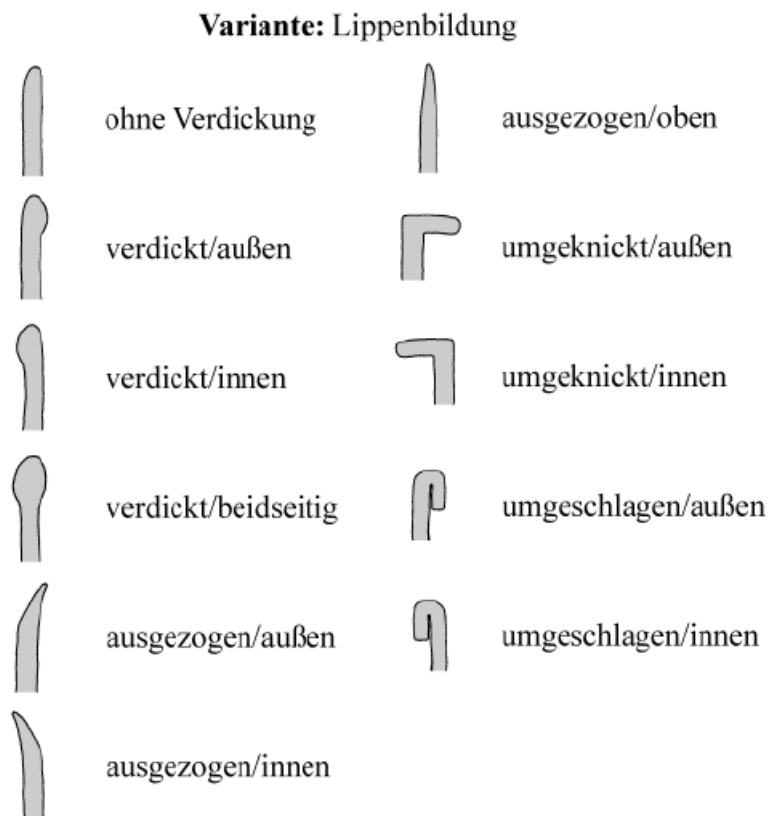


Abbildung 5.10: Schema zur Lippenbildung

5.2.1 Analyse

In Abbildung 5.10 ist deutlich zu erkennen, dass es zwei unterschiedliche Kategorien der Lippenbildung bei Rändern gibt. Einerseits die, deren Mittelachse gerade verläuft und die, bei der die Mittelachse einen Knick besitzt. Außerdem lassen sich verschiedene Arten und Stellungen den Kategorien zuordnen:

- Kategorie „Mittelachse geknickt“
 - Art „Umgeschlagen“ (Stellung innen, außen)
 - Art „Umgeknickt“ (Stellung innen, außen)
- Kategorie „Mittelachse gerade“
 - Art „Ausgezogen“ (Stellung innen, außen, oben)
 - Art „Verdickt“ (Stellung innen, außen, beidseitig)
 - Art „Ohne Verdickung“

Im ersten Schritt wird daher die Kategorie bestimmt, also die Mittelachse bezüglich ihrer Stellung untersucht. Besitzt diese einen auffälligen Knick, so muss lediglich getestet werden, ob es sich um eine umgeschlagene oder eine geknickte Lippenbildung handelt. Für die erste Variante kann ein einfacher Test verwendet werden, der die Datenstruktur der Mittelachse (Vertices und Linien) nutzt. Dabei werden alle Linien vor dem Knick durchlaufen und auf ihre Lage zur letzten Linie untersucht. Gilt für das Skalarprodukt der normierten Richtungsvektoren (u und v) der Geraden, die die zwei zu betrachtenden Linien beschreiben, folgende Bedingung

$$(u \cdot v \leq -0.8) \vee (u \cdot v \geq 0.8),$$

so sind die Linien annähernd parallel zueinander und die Lippenbildung ist umgeschlagen. Gilt die Bedingung jedoch nicht, so handelt es sich um eine umgeknickte Lippenbildung.

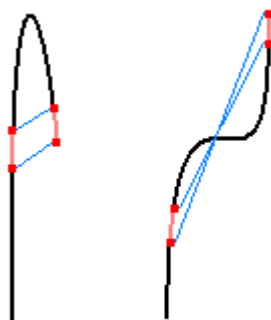


Abbildung 5.11: Mittelachsenuntersuchung paralleler Linien

Abbildung 5.11 zeigt zwei Beispiele für parallele Linien, die die Bedingung erfüllen. Da jedoch nur das linke Beispiel eine umgeschlagene Lippenbildung besitzt, ist noch eine Überprüfung notwendig. Dafür wird getestet, ob die Verbindungsstrecken (in Abbildung 5.11 blau dargestellt) sich schneiden. Liegt kein Schnitt der zwei Strecken vor, so ist die Lippenbildung umgeschlagen, andernfalls ist sie lediglich geknickt.

Ränder, deren Mittelachsen keinen auffälligen Knick besitzen, werden mit Hilfe von Distanztransformationen (siehe Kapitel 3) untersucht. Dabei wird eine Distance Map erstellt, bei der die Merkmalselemente die Mittelachsenvertices und -linien sind. Jedes Pixel der diskretisierten Randdarstellung besitzt nun, nach Erstellung der Distance Map, einen Wert, der aussagt, wie groß der Abstand des Pixels zum nächsten Mittelachsenpixel ist. Mittels dieser Werte lassen sich zwei Funktionen erstellen, jeweils für eine Seite des Randes (siehe Abbildung 5.12), die den Verlauf der Entfernung des Randpolygonzugs zur Mittelachse beschreiben. Durch Auswertung dieser Funktionen lässt sich die Lippenbildung bestimmen.

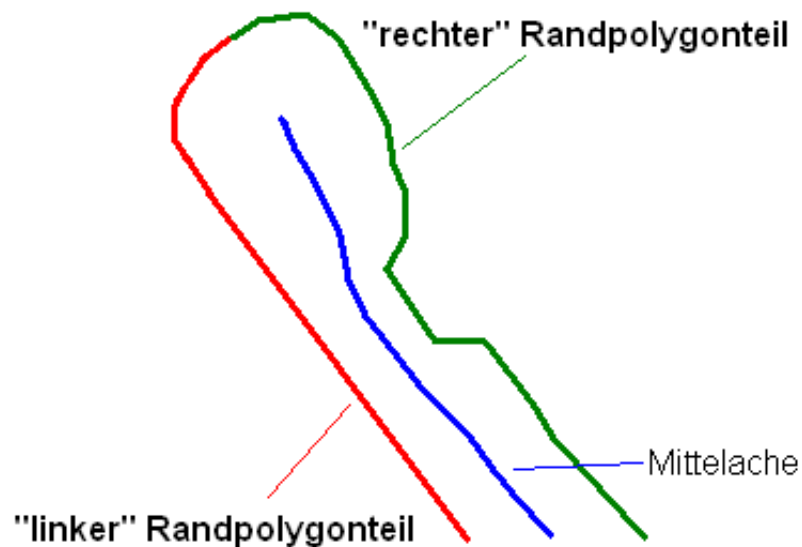
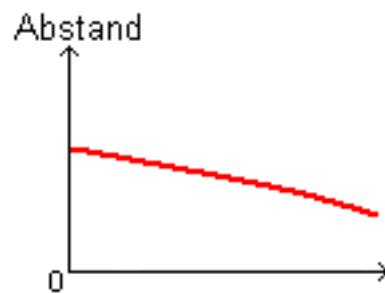


Abbildung 5.12: Unterteilung des Randpolygons in 2 Seiten/Teile

Folgende Fälle sind zu unterscheiden:

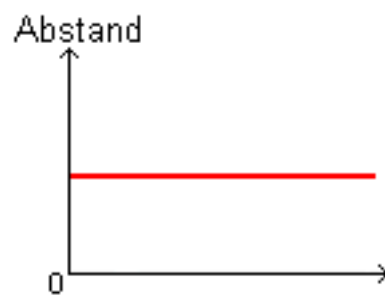
- Ausgezogen

Die Kurven der beiden Funktionen verlaufen fallend.



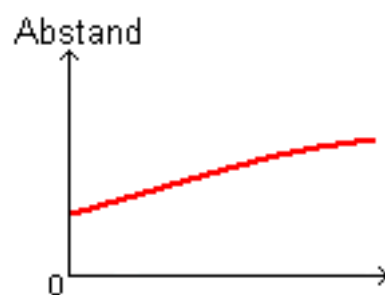
- Ohne Verdickung

Die Kurven der beiden Funktionen verlaufen konstant.



- Verdickt

Die Kurven der beiden Funktionen verlaufen steigend.



Die jeweilige Stellung (innen, außen, oben/beidseitig) wird schließlich noch mit Hilfe der Mittelachse bestimmt. Ist das letzte Drittel der Mittelachse

- einbiegend, so ist die Stellung „innen“.
- ausbiegend, so ist die Stellung „außen“.
- steil/gerade, so ist die Stellung „oben/beidseitig“.

5.2.2 Entscheidungsbaum

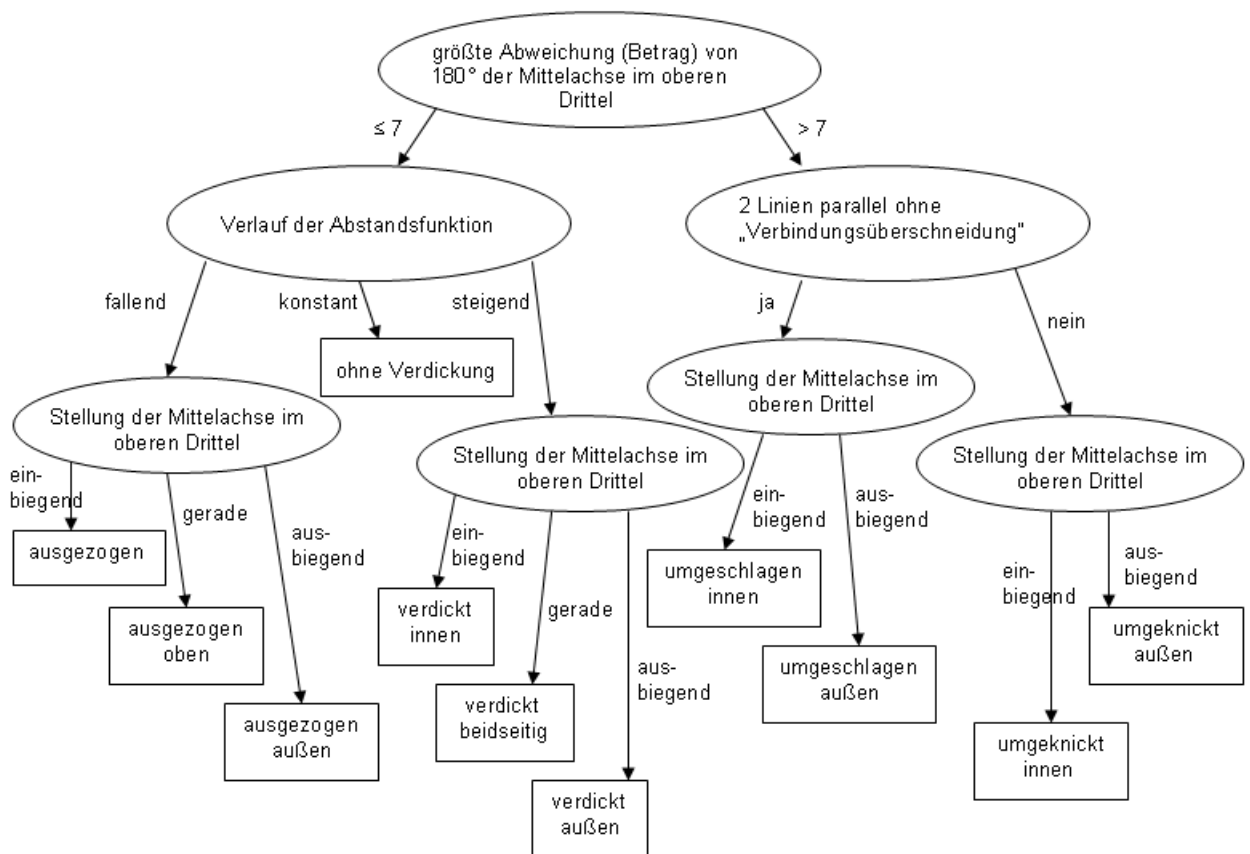


Abbildung 5.13: Entscheidungsbaum Lippenbildung

5.3 Randabschluss

Drittes Merkmal eines Gefäßrandes ist der Randabschluss, der bei [Paa02] aufgeschlüsselt zu finden ist:

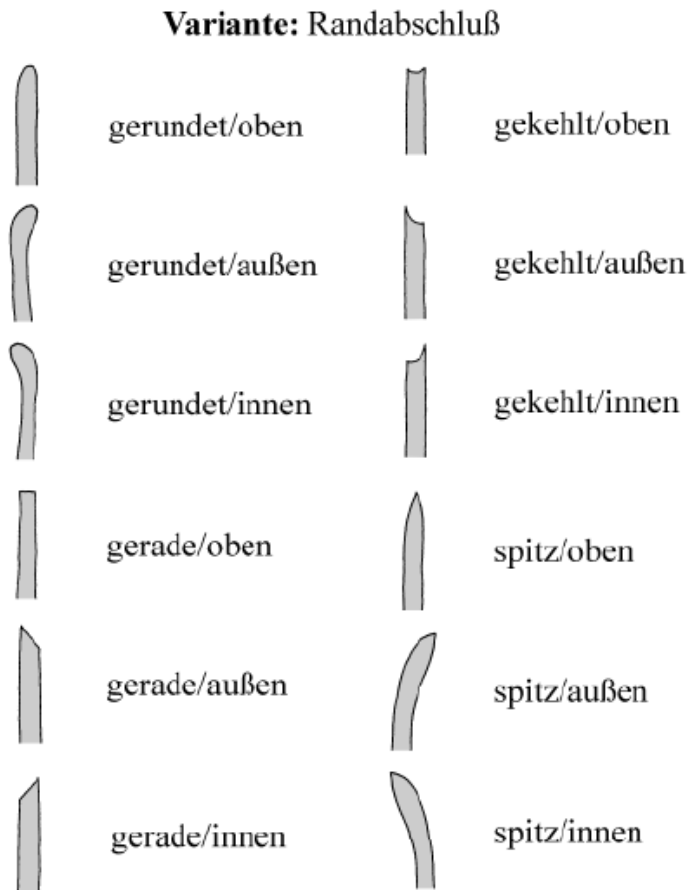


Abbildung 5.14: Schema zum Randabschluss

Da hier vor allem der oberste Teil des Randes entscheidend ist, muss das entsprechende Stück des Polygonzuges extrahiert und untersucht werden. Dazu ist die Mittelachse hilfreich, die ja bereits bei der Randstellung (siehe 5.1) berechnet wurde, da sich mit ihr die obersten Randpunkte finden lassen.

Damit sich die berechnete Mittelachse und die Randkontur im selben Koordinatenraum befinden, wird die Diskretisierung herangezogen, um die zwei nächsten Randpixel zum letzten Mittelachsenpixel zu bestimmen:

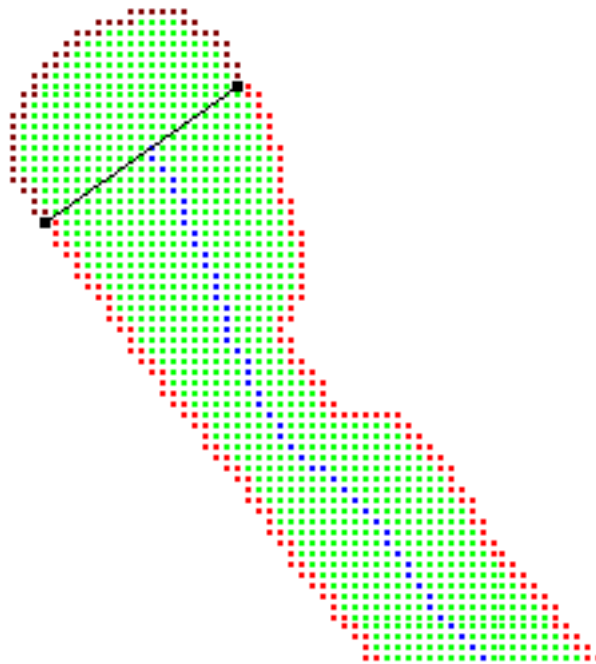


Abbildung 5.15: Bestimmung der Randabschlusspixel

Mit Hilfe einer Geraden, die senkrecht zur Mittelachse durch das letzte oberste Pixel verläuft, werden die Schnittpixel der Randkontur ermittelt. Alle Randpixel, die zwischen diesen beiden Schnittpixeln liegen, werden verwendet, um einen Polygonzug zu berechnen, der zur Analyse des Randabschlusses nötig ist. Dieser, bei dem jedes Pixel einen Punkt darstellt und durch Linien mit seinen Nachbarn verbunden ist, wird durch zehnmalige Anwendung gewichtet geglättet:

$$\forall i : P'_i = \frac{3P_i + P_{i-1} + P_{i+1}}{5}$$

5.3.1 Analyse

Um den Randabschluss anhand des ermittelten Polygonzuges und der Mittelachse zu analysieren, müssen mehrere Tests durchlaufen werden. Dabei sind zwei Merkmale zu identifizieren: Erstens die Art der Rundung (gerundet, gerade, gekehlt, spitz) und zweitens die Stellung (oben, außen, innen).

Der erste Test betrifft eine mögliche Gabelung der Mittelachse im oberen Drittel. Existiert diese, so handelt es sich um einen gekehlten Randabschluss. Welche Ausrichtung (Stellung) die Kehlung besitzt, wird mit Hilfe der Länge der Gabeläste bestimmt: Sind beide gleich lang, so ist der Rand gekehlt oben; ist jedoch der äußere (rechte) Ast länger, so befindet sich die Kehlung innen, andernfalls außen.

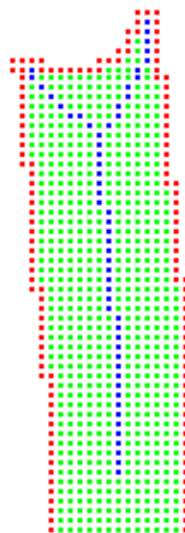


Abbildung 5.16: Randabschluss gekehlt innen

Ist die Mittelachse nicht gegabelt, so wird als weiteres Unterscheidungskriterium der Krümmungsverlauf des ermittelten Polygonzuges herangezogen. Um diesen auswerten zu können, müssen die Winkel an den einzelnen Polygonpunkten berechnet werden; die Berechnung erfolgt analog zu 5.1.2.

Das resultierende Winkeldiagramm lässt sich nun hinsichtlich der Art der Rundung analysieren. Dazu wird zunächst die Anzahl der Maxima über 185 Grad ermittelt, es zählen jedoch nur die Höchstwerte, bei denen die Verlaufskurve wieder unter den Wert 185 Grad sinkt:

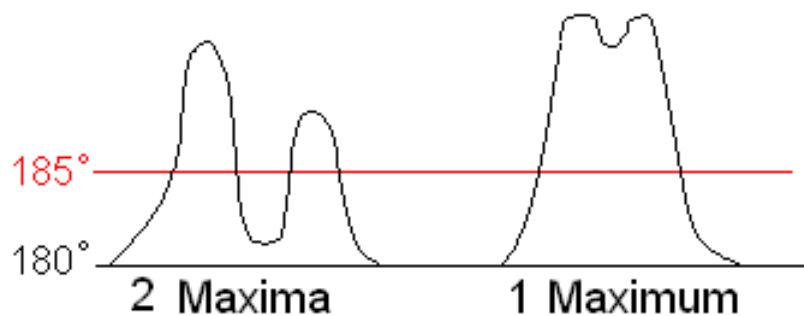


Abbildung 5.17: Winkelkurven

Da die Winkel immer am Außenrand der Kontur gemessen werden, sagt der Wert 185 Grad aus, dass an dieser Stelle ein starker Knick im Polygonzug vorhanden ist. Die Anzahl der Maxima lässt damit Rückschlüsse auf die Rundung zu:

1. Anzahl = 1: Ist die Anzahl der tatsächlichen Maxima über 185 Grad (in Abbildung 5.17 jeweils zwei) ebenfalls eins, so handelt es sich um einen spitzen Randabschluss, andernfalls um einen runden.
2. Anzahl = 2: Der Randabschluss kann gekehlt, gerade oder gerundet sein, es sind also weitere Tests erforderlich:
 - a) Der Flächeninhalt zwischen den zwei Maxima wird vorzeichenbehaftet bestimmt (siehe Abbildung 5.18).
Ist dieser Flächeninhalt negativ, so ist der Randabschluss gekehlt. Handelt es sich jedoch um einen positiven Wert, so muss noch getestet werden, ob zwischen den zwei Maxima Minima existieren, deren Wert unter 180 Grad liegt. Ist auch dies der Fall, so wurde ein gerader Randabschluss identifiziert, andernfalls ein gerundeter.
3. Anzahl > 2: Randabschluss ist gerundet.

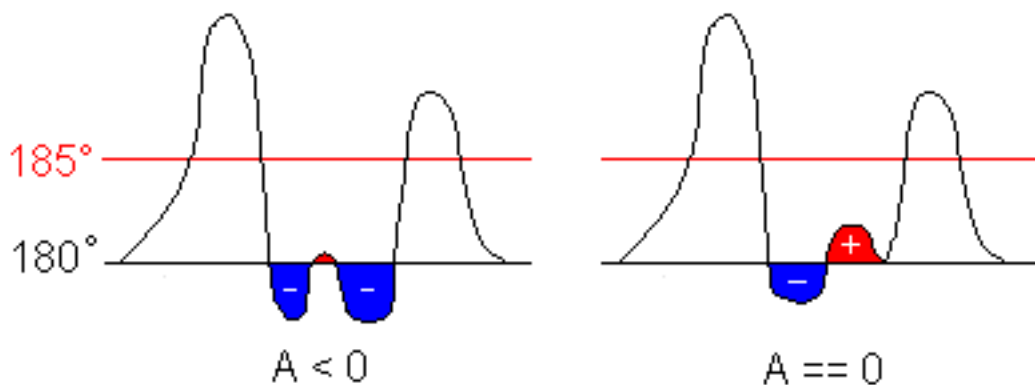


Abbildung 5.18: Flächeninhaltsbestimmung zwischen Maxima

Die Stellung des Randabschlusses ist schließlich von der Art der Rundung abhängig und muss daher in dieser Abhängigkeit bestimmt werden.

Art der Rundung:

- gerundet oder spitz: Das obere Drittel der Mittelachse wird hinsichtlich seines Verlaufes untersucht. Verläuft es gerade, so ist die Stellung des Randabschlusses oben; einbiegend bedeutet innen und ausbiegend außen.
- gekehlt: Auch hier wird das obere Drittel der Mittelachse untersucht, nun jedoch hinsichtlich eines Knickes. Existiert ein Knick nach rechts (Winkel < 180 Grad), so befindet sich die Kehlung innen; knickt die Mittelachse hingegen nach links (Winkel > 180 Grad), so liegt die Kehlung außen; ist beides nicht der Fall, so liegt sie oben.
- gerade: Hier reicht es aus, den Winkelverlauf des Polygonzuges noch einmal genauer zu betrachten. Da die Rundung nur dann gerade ist, wenn zwei Maxima über 185 Grad existieren, kann man diese beiden Höchstwerte untersuchen. Sind beide gleich, so ist die Randabschlussstellung oben, ist jedoch das erste Maximum höher, so ist der Randabschluss gerade außen, andernfalls gerade innen. Voraussetzung für diese Art der Auswertung ist, dass der Polygonzug (und somit der Winkelverlauf) stets von links nach rechts abgearbeitet wird.

5.3.2 Entscheidungsbaum

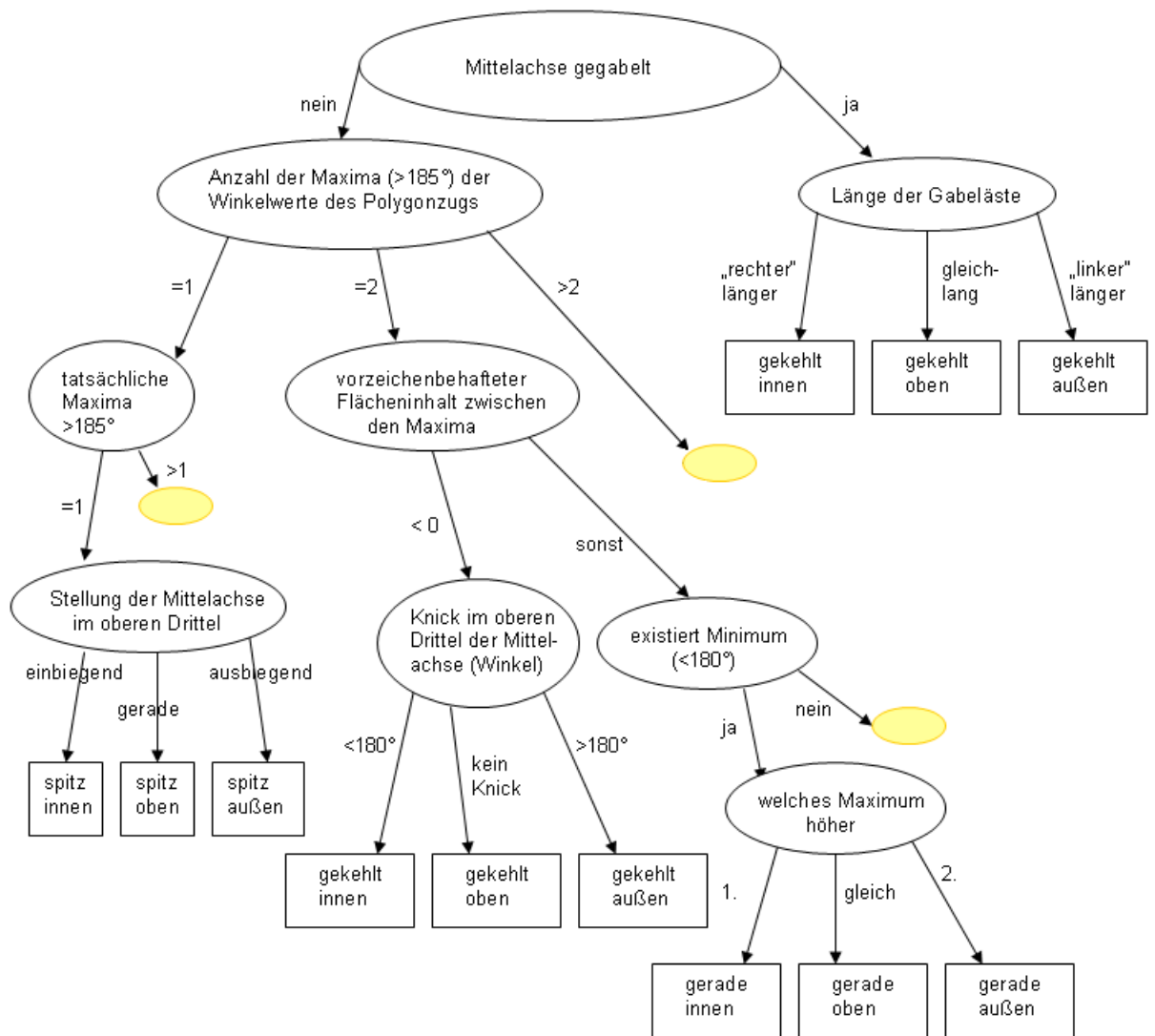


Abbildung 5.19: Entscheidungsbaum Randabschluss

Aus Gründen der Übersichtlichkeit wurde der gelbe Knoten in Abbildung 5.19 einzeln aufgeschlüsselt:

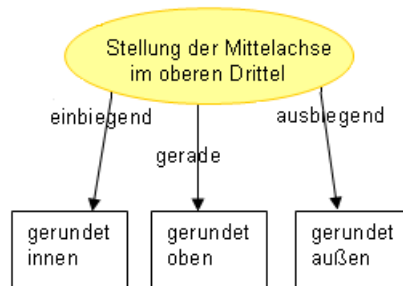


Abbildung 5.20: Entscheidungsbaumknoten

5.4 Beurteilung

Probleme bei der automatischen Randklassifizierung sind meist darauf gegründet, dass die Einteilung der einzelnen Typen und Varianten oft nur mittels Schwellwerten möglich ist. Häufig ist eine bestimmte Eigenschaft auch nur durch mehrere Merkmale einortbar. Diese Merkmale zu finden und Trennbereiche zu definieren ist nicht trivial. Deshalb liefern die entwickelten Methoden nicht für jeden Randdatensatz eine korrekte Klassifizierung. Zwei Beispiele sollen die Klassifizierungsschwächen verdeutlichen:

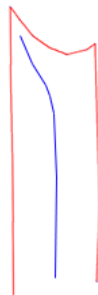


Abbildung 5.21: fehlerhaft klassifizierter Rand bzgl. Randstellung

Abbildung 5.21 zeigt einen Rand, der eindeutig die Randstellung „steil“ besitzt. Da jedoch die Mittelachse (blau dargestellt) in den engen linken Bereich hineingeht, klassifiziert der Algorithmus sie als einbiegend.

Der Randabschluss in Abbildung 5.22 liegt eigentlich im „spitzen“ Bereich, doch aufgrund der Vertex-Verteilung auf dem Randpolygonzug kommt der Klassifizierungsalgorithmus zum Ergebnis „gerundet aussen“.

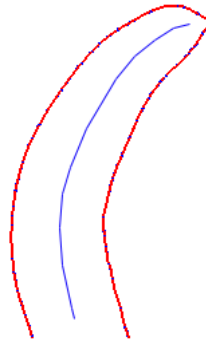


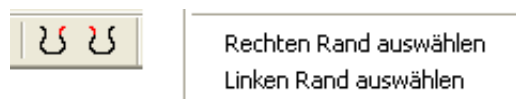
Abbildung 5.22: fehlerhaft klassifizierter Rand bzgl. Randabschluss

A Anhang

Die neuen Funktionen und Bedienelemente der Software ArchViewer werden hier noch einmal zusammengefasst.

A.1 Funktionen bei geladenem Gefäßobjekt

Um ein Randobjekt aus einem Gefäß zu erzeugen gibt es in der Toolbar zwei neue Buttons (bzw. zwei neue Einträge im Menü „Objekt“):



Beide Buttons liefern die Funktionalität, mit Hilfe der Maus einen Rahmen um den gewünschten Bereich zu ziehen. Der erste Button ist jedoch für die Extrahierung des rechten Randes (bzgl. Profilansicht), der zweite für die Extrahierung des linken Randes zu verwenden (siehe Abbildung A.1). Dabei ist darauf zu achten, dass der Rand nahezu gleichmäßig und gesamt gewählt wird, wie in Abbildung A.2 verdeutlicht. Durch das Betätigen der rechten Maustaste werden die Randdaten erstellt, nachdem mit dem aufgezogenem Rahmen ein Bereich ausgewählt wurde. Im Menü „Datei“ ist nun der Eintrag „Rand speichern unter...“ aktiv und das Randobjekt kann abgespeichert werden.

Damit der Rand klassifiziert werden kann, muss das Randobjekt über „Datei > Öffnen“ in den ArchViewer geladen werden.

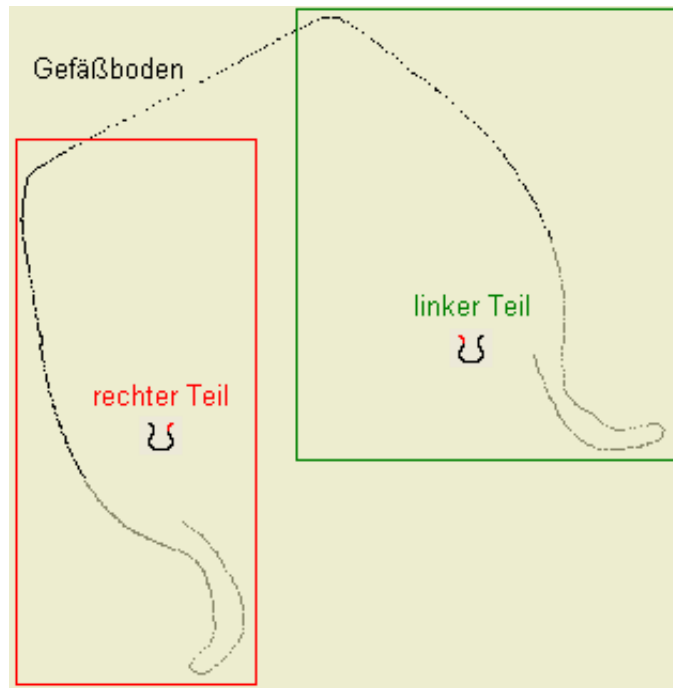


Abbildung A.1: Einteilung eines Gefäßobjekts

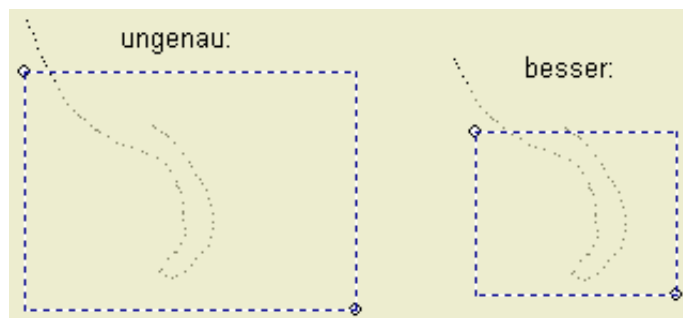


Abbildung A.2: Randauswahl

A.2 Funktionen bei geladenem Randobjekt

Wurde ein Randobjekt in den ArchViewer geladen, so ist folgende Toolbar aktiv:



Ebenso sind die Einträge im Menüpunkt „Randobjekt“ aktiv, welche die selben Funktionalitäten, wie ihre zugehörigen Button besitzen.






Button	Funktionalität
	Stellt den Randpolygonzug dar.
	Berechnet die Mittelachse des Randobjektes und stellt sie dar.
	Bestimmt die Randstellung, wenn die Mittelachse berechnet werden konnte. Angezeigt wird das Winkeldiagramm der Mittelachse.
	Bestimmt den Randabschluss, wenn die Mittelachse berechnet werden konnte. Angezeigt wird das Winkeldiagramm des oberen Randpolygons.
	Bestimmt die Lippenbildung, wenn die Mittelachse berechnet werden konnte. Angezeigt werden die Distanzkurven des oberen Randpolygons.

Tabelle A.1: Funktionalitäten der Button

Literaturverzeichnis

- [Arm] C. G. Armstrong et al.: „Medials for meshing and more”, The Queen’s University of Belfast.
- [Blu67] H. Blum: „A transformation for extracting new descriptors of shape. Models for the Perception of Speech and Visual Form”, Cambridge MA, MIT Press, 1967.
- [Bor86] G. Borgefors: „Distance transformations in digital images”, Source Computer Vision, Graphics, and Image Processing archive Volume 34 , Issue 3, 1986.
- [Brn03] G. Brunnert: „Computergrafik 1”, Chemnitz, 2003.
- [Bru03] D. Brunner: „Ein Ansatz für die interaktive Manipulierung der zusammenhängenden Körpersegmente eines triangulierten 3D-Objektes mittels der Objektmittelachse”, Diplomarbeit, Technische Universität Chemnitz, Fakultät für Informatik, 2003.
- [Lju68] L.A. Ljusternik, W.I. Sobolew: „Elemente der Funktionalanalysis”, Akademie-Verlag Berlin, 1968.
- [Loh98] G. Lohmann: „Volumetric Image Analysis”, Wiley-Teubner, Chichester, 1998.
- [Noo03] F. S. Nooruddin et al.: „Simplification and Repair of Polygonal Models Using Volumetric Techniques”, Georgia, 2003.
- [Paa02] I. Paap: „Die Keramik von Khyinga: Mustang District, Nepal”, http://hss.ulb.uni-bonn.de/diss_online/phil_fak/2002/paap_iken, Dissertation, Universität Bonn, Philosophische Fakultät, 2002.

- [Pal00] K. Palágyi: <http://www.inf.u-szeged.hu/~palagyi/skel/skel.html>, Szeged, 2000.
- [Rom] F. Romero et al.: „Fast Skeletonization of Spatially Encoded Objects”, Catalonia, Spanien.
- [Sai94] T. Saito and J. Toriwaki: „New algorithms for n-dimensional Euclidean distance transformation”, Pattern Recognition, 1994.
- [Ste05] J. Steinmüller: „Bildverarbeitung”, Chemnitz, 2005.
- [Van04] M. Vanco: „Grundlagen der Objektrekonstruktion”, Chemnitz, 2004.
- [Wip97] J. Wipper: „Mediale Achsen und Voronoj-Diagramme in der euklidischen Ebene”, Diplomarbeit, Universität Stuttgart, Mathematisches Institut, 1997.

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig angefertigt, nicht anderweitig zu Prüfungszwecken vorgelegt und keine anderen als die angegebenen Hilfsmittel verwendet habe. Sämtliche wissentlich verwendete Textausschnitte, Zitate oder Inhalte anderer Verfasser wurden ausdrücklich als solche gekennzeichnet.

Chemnitz, den 30. September 2005

Pollmer, Nadine