RHEINISCH-WESTFÄLISCHE TECHNISCHE HOCHSCHULE AACHEN
MEDIZINISCHE FAKULTÄT
INSTITUT FÜR MEDIZINISCHE INFORMATIK
UNIV.-PROF DR. DR. KLAUS SPITZER

RHEINISCH-WESTFÄLISCHE TECHNISCHE HOCHSCHULE AACHEN
MATHEMATISCH-NATURWISSENSCHAFTLICHE FAKULTÄT
LEHRSTUHL FÜR INFORMATIK VIII
UNIV.-PROF DR. LEIF KOBBELT

Hauptseminar im Sommersemester 2002

Medizinische Bildverarbeitung

Volume 1, Band 6 Oktober 2002

ISSN 1860-8906

ISBN 978-3-9811848-0-8

Aachener Schriften zur Medizinischen Informatik

Aachener Schriften zur Medizinischen Informatik
ISSN 1860-8906
ISBN 978-3-9811848-0-8

Herausgeber: Institut für Medizinische Informatik der RWTH Aachen
Pauwelsstr. 30
D-52074 Aachen

Geschäftsführender Direktor: Universitätsprofessor Dr. Dr. Klaus Spitzer

Vorwort

Jedes Jahr bietet das Institut für Medizinische Informatik der RWTH Aachen das Seminar *Medizinische Bildverarbeitung* für Studierende der Informatik im Hauptstudium an. Wie in der Vergangenheit wurde das Seminar in Zusammenarbeit mit den Lehrstühlen für Informatik VI und VII sowie der Klinik für Radiologische Diagnostik veranstaltet. In diesem Jahr übernahm erstmals auch Professor Kobbelt, Lehrstuhl für Informatik VIII, die Betreuung praxisorientierter Beiträge. Der theoretische Teil dieses Seminars wurde traditionell von Professor Oberschelp betreut.

Das Seminar bietet eine Vertiefung der ebenfalls jährlich angebotenen, interdisziplinären Ringvorlesung, die den gleichen Namen trägt. Wie die Personen, denen viele Beiträge das interessante, aber leider oft pathologische Bildmaterial "verdanken", lag auch das Seminar selbst noch vor kurzem "auf dem Krankenbett". Gesunkene Teilnehmerzahlen ließen es um die traditionsreiche Veranstaltung mit seinem einstmaligen, dem Aachender Unialltag fernen Veranstaltungsort in der Jugendherberge Hellenthal ein wenig stiller werden. Umso erfreulicher ist es, daß nach der Verlegung des Seminars ins Sommersemester (im Tausch mit der Ringvorlesung) wieder eine gestiegene Teilnehmerzahl zu vermelden ist. In diesem Jahr bilden acht Beiträge wieder einen respektablen Seminarband und unterstreichen die wiedergewonnene Sichtbarkeit der Veranstaltung.

Leider erst Monate nach den Präsentationen liegt nun der Seminarband vor, der die Ausarbeitungen aller Vorträge enthält. Dafür sind in diese auch die Ergebnisse aus den umfangreichen Sachdiskussionen der Vorträge eingeflossen. Allen Beteiligten ein herzliches Dankeschön für ihren Beitrag und zum Gelingen der Veranstaltung.

Dies leitet dann auch in den Abschnitt der *Danksagungen* über, in dem neben den Seminaristen natürlich auch die Professoren Oberschelp und Kobbelt, der Organiator Dr. Lehmann, die Betreuer der einzelnen Beiträge, Keysers, Kohnen, Lehmann, Thies, Güld und die an der "Post-Produktion" beteiligten studentischen Hilfskräfte am Institut für Medizinische Informatik, Andrea Wenning und Martin Eul, zu nennen sind.

Aachen, im Januar 2003
Mark Oliver Güld

Inhaltsverzeichnis

Seminar Medizinische Bildverarbeitung 2002

Programm der Blockveranstaltung vom 3.7.2002 - 4.7.2002 in Aachen

Donnerstag, 3. Juli 2002

09:00 - 09:55	Probabilistisches Tracking mit dem "Condensation"-Algorithmus	Referent: *Axel Janßen* Betreuer: *Daniel Keysers*
09:55 - 10:50	Segmentierung medizinischer Bilddaten mittels Analyse von Isolable-Kontur-Karten	Referent: *Paul Herr* Betreuer: *Mark O. Güld*
11:10 - 12:05	Konnektivität in vollständigen Verbänden: Ein algebraisches Konzept zur formbasierten Bildverarbeitung	Referent: *Lutz Ißler* Betreuer: *Christian Thies*
12:05 - 13:00	Erscheinungsbasierte 3-D Objekterkennung	Referent: *Andrea Wenning* Betreuer: *Daniel Keysers*

Freitag, 4. Juli 2002

09:00 - 09:55	Kantenerkennung in medizinischen Bildern	Referent: *Markus Jogmin* Betreuer: *Thomas M. Lehmann*
09:55 - 10:50	Morphologische Multiskalen-Bildanalyse auf der Basis von Levelings	Referent: *Jörg Freudenstein* Betreuer: *Christian Thies*
11:10 - 12:05	Automatische Detektion und Verfolgung von Zungenkonturen	Referent: *Florian Hasibether* Betreuer: *Mark O. Güld*
12:05 - 13:00	"Live-Wire" - Ein semiautomatisches Segmentierungsverfahren für medizinische Bilder	Referent: *Michael Sauren* Betreuer: *Michael Kohnen*

Probabilistisches Tracking mit dem Condensation Algorithmus

von Axel Janßen

Betreuer: Daniel Keysers

Inhaltsverzeichnis

Zusammenfassung

Der Condensation Algorithmus wurde von Michael Isard und Andrew Blake in der Veröffentlichung „Condensation - Conditional Density Propagation for Visual Tracking“ [1] vorgestellt. Der Algorithmus verfolgt mit Hilfe eines probabilistischen Ansatzes, der die konkurrierende Betrachtung verschiedener Handlungshypothesen erlaubt, ein Objekt in einer Videosequenz. Das Ziel ist, auch bei komplexer Objektumgebung eine möglichst sichere Erkennung in Echtzeitverarbeitung zu erreichen. An ausgewählten Samplestellen wird eine Wahrscheinlichkeitsverteilung repräsentiert. Durch einen Vergleich der Bilddaten mit diesen Stellen wird eine fortwährende Anpassung der Wahrscheinlichkeitsverteilung erreicht.

Keywords: Condensation, Visual Tracking, Video-Sequenz, Bayes-Theorem

1 Einleitung

Sehr viele Artikel der letzten Jahre verweisen auf die Ausarbeitung [1] der Autoren Michael Isard und Andrew Blake aus dem Jahre 1998. Der in der Ausarbeitung beschriebene Condensation Algorithmus ist ein populäres Verfahren der Bildverarbeitung der letzten Jahre.
Ziel dieses Algorithmus ist die Verfolgung von Objekten *(visual tracking)* innerhalb einer Videosequenz. Gefordert wird eine hohe Verarbeitungsgeschwindigkeit, die gelieferte Videobilder bei hoher Fehlertoleranz in Echtzeit verarbeiten kann.
Mögliche Anwendungen eines solchen Verfahrens in der Medizin könnte zum Beispiel die Ultraschallanalyse des menschlichen Herzens sein. Ultraschall bietet sich hierfür aufgrund der gelieferten Echtzeitvideosequenzen an, allerdings ist der Nachteil dieser Technologie nicht von der Hand zu weisen: Die von den Ultraschallgeräten gelieferten Sequenzen sind oft mit starkem Rauschen überlagert, so daß dem untersuchenden Arzt eine sofortige Analyse der gelieferten Bilder nicht immer leicht fällt. Hier könnte der Condensation Algorithmus gute Dienste leisten: Hat der Arzt das Objekt (das menschliche Herz) erst einmal detektiert, kann der Algorithmus während der folgenden Untersuchung das Herz permanent verfolgen und das Ergebnis seiner Analyse mit einer Umrandung sichtbar machen.
Grundsätzlich ist die erfolgreiche Verfolgung eines Objektes innerhalb eines Videos von verschiedenen Parametern abhängig. Zwei Arten von Fehlerquellen können das Ergebnis verfälschen oder den Algorithmus sogar überfordern: Veränderungen des Objektes und Veränderungen der Umgebung bzw. des Hintergrundes. Im ersten Fall kann sich das Objekt in Position und Lage und in Beschaffenheit und Form von Bild zu Bild auf teilweise nur sehr schwer vorhersagbare Art ändern; man denke beispielsweise an die Flugbewegung einer Vogelfeder bei starkem Wind. Auch könnte das gesuchte Objekt in manchen Bildern durch andere Objekte teilweise überlagert sein oder in einer Teilsequenz komplett verschwinden. Im zweiten Fall stellen ein wechselnder Hintergrund (man denke an die Personenverfolgung, bei der die Erkenntnisse des Algorithmus an den Steuermotor einer beobachtenden Kamera weitergegeben werden) oder eine dem Objekt sehr ähnliche Umgebung (beispielsweise die Verfolgung eines Blattes im Wind) Schwierigkeiten dar, die den Algorithmus ebenfalls überfordern können.
Der Condensation Algorithmus verfolgt im Gegensatz zu anderen populären Verfahren einen probabilistischen Ansatz, der auf einer diskreten Menge von Zuständen operiert: Um das Objekt im aktuellen Bild detektieren zu können, werden die in den letzten Bildern über das Objekt gewonnene Erkenntnisse genutzt.
Wie dies vonstatten geht, wird in den folgenden Abschnitten näher vorgestellt. Zunächst werden in Abschnitt 2 die die relevanten stochastischen Grundlagen erläutert und wieso dem Theorem von Bayes eine zentrale Rolle bezüglich der Idee des Condensation Algorithmus zukommt. Anschließend wird in Abschnitt 3 der Algorithmus selbst vorgestellt. In einem kurzem Exkurs (Abschnitt 4) wird auf den Kalman Filter, einen anderen Ansatz aus dem Gebiet der Bildverarbeitung, eingegangen. Hier werden auch Unterschiede der beiden Verfahren deutlich gemacht.

2 Stochastische Grundlagen

Dieser Abschnitt behandelt die stochastischen Grundlagen, die zum Verständnis des Condensation Algorithmus notwendig sind. Der erste Teil (Abschnitt 2.1) beschäftigt sich mit den relevanten Begriffen der Wahrscheinlichkeitsrechnung und führt anhand von Beispielen in diesen Bereich ein. Der zweite Teil

Abb. 2.1: Mit *Splines* umrandetes Herz (Ultraschallaufnahme).

(Abschnitt 2.2) erklärt das Theorem von Bayes und stellt einen Zusammenhang des Theorems mit der dem Condensation Algorithmus zugrunde liegenden Idee her.

2.1 Grundlegende Begriffe

Für das Rechnen mit Wahrscheinlichkeiten wird im Allgemeinen das Kolmogorovsche Axiomensystem verwendet. Im Folgenden seien hier Eigenschaften von Wahrscheinlichkeiten für zwei Ereignisse a und b aufgeführt:

- Wahrscheinlichkeiten liegen immer zwischen 0 und 1:
  $0 \leq P(a) \leq 1$
- Die Summe der Wahrscheinlichkeiten der Ereignisse ist 1:
  $\sum_k P(a_k) = 1$
- Das komplementäre Ereignis zu a ist genau dann realisiert, wenn a nicht realisiert ist:
  $P(\bar{a}) = 1 - P(a)$

Eine *Zufallsvariable* repräsentiert ein Ereignis, deren Eintrittswahrscheinlichkeit bestimmt werden soll oder deren Wahrscheinlichkeit schon gegeben ist. Sie kann dabei vorgegebene Werte aus einem vorgegebenen Definitionsbereich annehmen.
Bisher wurden ausschließlich einfache Wahrscheinlichkeiten betrachtet, also Aussagen „wie groß ist die Wahrscheinlichkeit, daß ein Ereignis x auftritt?“ Der Begriff *a-priori Wahrscheinlichkeit* meint diese

Abb. 2.2: Mit *Splines* umrandetes Blatt eines Busches.

unbedingte Wahrscheinlichkeit eines Ereignisses a: $P(a)$. Hier sei beispielhaft das Würfeln mit einem Würfel aufgeführt: $P(\text{Sechser beim Würfeln}) = 1/6$.
Von zentraler Bedeutung dagegen ist in diesem Fall der Begriff der *a-posteriori Wahrscheinlichkeit*, der die *bedingte* Wahrscheinlichkeit eines Ereignisses a unter Beobachtung von b (also wenn b gilt): $P(a|b)$: $P(\text{Sechser beim Würfeln} \mid \text{Gerade Augenzahl}) = 1/3$ meint, also Aussagen der Form: „Wie groß ist die Wahrscheinlichkeit, daß Ereignis x auftritt, unter Bedingung von y?“

2.2 Das Theorem von Bayes

Die Gleichung für bedingte Wahrscheinlichkeiten besagt:

$$P(x|y) = \frac{P(x \wedge y)}{P(y)} = \frac{P(y|x) \cdot P(x)}{P(y)} \tag{2.1}$$

Das folgende Beispiel erläutert diese Gleichung näher:

- x: Hypothese (es ist sonnig)
- y: Daten, Ursache (es sind 25 Grad Celsius)
- $P(y)$: Wahrscheinlichkeit, daß die Temperatur mindestens 25 Grad beträgt
- $P(x)$: Wahrscheinlichkeit, daß es sonnig ist.
- $P(x|y)$: Wahrscheinlichkeit, daß es sonnig ist, unter der Bedingung, daß die Temperatur mindestens 25 Grad beträgt.
- $P(y|x)$: Wahrscheinlichkeit, daß es an einem sonnigen Tag mindestens 25 Grad warm ist.
- $P(x \wedge y) = P(y|x) \cdot P(x)$
- $P(y|x) \cdot P(x) = P(x|y) \cdot P(y)$

Für die Berechnungen bezüglich des Condensation Algorithmus gelten im folgenden diese Variablen:

- x sei eine mögliche Position des Objektes (Hyphothese, *Sample*)
- y sei die gelieferte Bildinformation

Das Theorem von Bayes wird nun benötigt, um die schwierige direkte Bestimmung der Objektposition $P(x|y)$ („befindet sich das Objekt an dieser Stelle im Bild“) durch die einfachere Bestimmung, ob die Bildinformation der vermuteten Position entspricht $P(y|x)$ („entspricht die Bildinformation der Hypothese?“) auszudrücken.

$$P(x|y) = \frac{P(y|x) \cdot P(x)}{P(y)} = k \cdot P(y|x) \cdot P(x) \tag{2.2}$$

Der Quotient $\frac{1}{P(y)}$ ist unabhängig von x und wird daher durch die Konstante k ausgedrückt. Die Berechnung von $P(x|y)$ durch $P(y|x)$ stellt die zentrale Idee des Condensation Algorithmus dar.
Im folgenden Abschnitt 3 soll nun erklärt werden, wie aus den Hypothesen über die Position eines Objektes in einem bestimmten Bild der Videosequenz $P(x_{t-1})$ Aussagen über die Wahrscheinlichkeit, die Position des Objektes im nächsten Bild der Videosequenz $P(x_t)$ betreffend, gewonnen werden können.

3 Der Condensation Algorithmus

Das Ziel des Condensation Algorithmus ist die möglichst korrekte Identifizierung eines Objektes in jedem Zeitschritt einer Videosequenz und dies möglichst in Echtzeit. Der Condensation Algorithmus basiert, wie schon erwähnt, auf einem probabilistischen Ansatz, der auf einer diskreten Menge von Zuständen operiert, um anhand von gewichteten Hyphothesen (im folgenden als *Samples* bezeichnet) Vorhersagen über das Verhalten des beobachteten Objektes im nächsten Zeitschritt machen zu können. Im folgenden soll nun dargestellt werden, was dies genau bedeutet, und wie der Algorithmus abläuft.

3.1 Vorbereitungen

Ein Objekt in einem Bild wird im einfachsten Fall durch seine Position beschrieben. Aber auch andere Informationen können betrachtet werden, wie die Lage im Raum, Form des Objektes etc. Im folgenden werden diese Variablen wie angegeben verwendet:

- x_t ist der Zustand des modellierten Objektes im Zeitschritt t
- y_{t-1} beschreibt den vorangehenden Zeitschritt (das vorherige Bild der Videosequenz)
- y_t beschreibt den aktuellen Zeitschritt (das aktuelle Bild der Videosequenz).

Nach jedem Zeitschritt gibt der Algorithmus eine gewichtete Menge von Hypothesen (im folgenden als *Sample-Set* bezeichnet) $\{x_t^{(1)}, \pi_t^{(1)}, \ldots, x_t^{(N)}, \pi_t^{(N)}\}$ aus, die die bedingte Zustandsdichte $p(x_t|y_t)$ in Zeitschritt t repräsentieren. Ein Sample n zum Zeitschritt t ist allgemein mit $x_t^{(n)}$ bezeichnet und $\pi_t^{(n)}$ meint die Gewichtung eines solchen Samples. Der folgende Abschnitt erklärt, wie man an dieses Sample-Set gelangt.

3.2 Ein Schritt des Condensation Algorithmus

Vor dem ersten Schritt

Für jeden Zeitschritt t der Videosequenz wird eine Iteration gestartet, um an das vorgenannte Sample-Set zu gelangen. Die Iteration jedes Zeitschrittes benötigt dazu das Sample-Set aus dem vorherigen Zeitschritt. Also ist klar, daß vor der ersten Iteration bereits eine Anfangsverteilung für Samples erzeugt werden und ein Sample-Set $\{x_0^{(n)}\}$ vorliegen muß. Eine Menge N von Samples, also Hypothesen möglicher Zustände des Objektes, wird im Bild verteilt (Schritt 0 in Abbildung 2.3). Dies kann durch eine Gleichverteilung geschehen. Hat man allerdings schon eine Vermutung, wo sich das Objekt befinden könnte, kann man die Samples in dieser Bildregion mit einer geeigneten Verteilung $P(x)$ enger streuen. Weil man von vornherein die Menge von Samples auf N beschränkt, kann ein Ablauf innheralb gegebener Rechenressourcen garantiert werden.

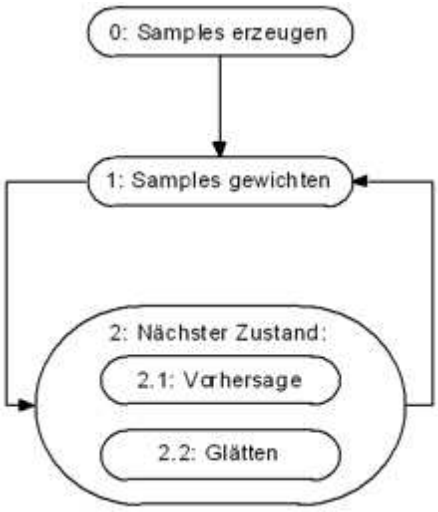


Abb. 2.3: Schematischer Ablauf des Condensation Algorithmus.

Die eigentliche Iteration

An dieser Stelle beginnt nun der erste Schritt der Abbildung 2.4. Er ist mit $P(x_{t-1})$ bezeichnet. Abbildung 2.4 beschreibt die Iteration des Condensation Algorithmus zu einem beliebigen Zeitpunkt t. Es soll nun aus dem Sample-Set des letzten Zeitschrittes $t-1$ ein neues Sample-Set für den aktuellen Zeitschritt t gewonnen werden.

Die schraffierten Kreise stellen die Samples zum Zeitpunkschritt $t-1$ dar. Der Mittelpunkt eines Kreises meint das jeweilige Sample und die Größe der schraffierten Fläche um diesen Mittelpunkt gibt die Gewichtung des Samples an.
Der erste Schritt in Abbildung 2.4, das *Sampling*, geht folgendermaßen vonstatten: Aus der bisherigen Samplemenge $\{x_{t-1}^{(1)}, \pi_{t-1}^{(1)}, \ldots, x_{t-1}^{(N)}, \pi_{t-1}^{(N)}\}$ des letzten Zeitschrittes werden N Samples $x_{t-1}^{(n)}$ entnommen. Man kann dies durch wiederholtes Ziehen von Samples (mit Zurücklegen) erreichen. Einige Samples, vor allem diejeinigen mit höherer Gewichtung $\pi_{t-1}^{(n)}$, werden häufiger gezogen, wohingegen manche Samples vielleicht gar nicht gezogen werden. Dies läuft schematisch folgendermaßen ab:

1. Ziehe ein Sample $x_{t-1}^{(n)}$.
2. Übernehme Sample $x_{t-1}^{(n)}$ mit der assoziierten Wahrscheinlichkeit $\pi_{t-1}^{(n)}$ in das neue Sample-Set.
3. Gehe solange zu Schritt 1 zurück, bis die Anzahl der Ziehungen gleich der Anzahl N Elemente der alten Samplemenge ist.

Die Sampleverteilung entspricht nun wieder der gesuchten Wahrscheinlichkeitsverteilung $P(x_{t-1})$. Jedes Element des neuen Sample-Sets wird nun dem nächsten Schritt unterzogen: Das Ziel ist ja die Bewegung eines Objektes nachzuvollziehen. Dazu muß auch die Bewegung probabilistisch erfaßt werden, was mit dem Modell $P(x_t|x_{t-1})$ in Schritt 2.1 geschieht. Zunächst wird jedes Element der neuen Samplemenge anhand eines vorhergesagten Richtungsvektors bewegt.

$$P(x_t) \leftarrow P(x_t|x_{t-1}) \tag{2.3}$$

Identische Elemente des neuen Sample-Sets werden identisch bewegt. Dies wird durch die Pfeile in Abbildung 2.4, die von der ersten Samplemenge ausgehen, angezeigt. Man erhält die neue Verteilung $P(x_t)$. Im nächsten Schritt (Schritt 2.2) wird Rauschen hinzugefügt, um identische Elemente zu trennen.
An dieser Stelle im Algorithmus ist die neue Samplemenge komplett, allerdings sind die Samples nicht gewichtet, was nun anhand der gelieferten Bildinformation geschieht. Ein Bildanalysealgorithmus trifft also Aussagen dazu, wie hoch die Wahrscheinlichkeit ist, an einer gegebenen Stelle x das Objekt auch wirklich in dem beschriebenen Zustand zu finden. Dies wird für alle N Samples durchgeführt.

$$\pi_t \leftarrow P(x_t) \leftarrow P(x_t|y_t) = k \cdot P(y_t|x_t) \cdot P(x_t) \tag{2.4}$$

Nach Normalisierung der Gewichte $\pi_t^{(n)}$ (es gilt also $\sum_n \pi_t^{(n)} = 1$) erhält man ein N-elementiges, gewichtetes Sample-Set des Zeitschrittes y_t. Dieses Sample-Set $\{x_t^{(n)}, \pi_t^{(n)}, n = 1, \ldots, N\}$ repräsentiert die a-posteriori Wahrscheinlichkeit $P(x_t|y_t)$.
Am Ende dieses Schrittes befindet man sich wieder am oberen Ende der Abbildung 2.4, allerdings im nächsten Zeitschritt; der Algorithmus startet die nächste Iteration.

Auffinden des Objektes

Es kann zu jedem Zeitschritt, nachdem das aktuelle Sample-Set gebildet wurde, die wahrscheinlichste Position des Objektes gefunden werden: In der Nähe dieser Position werden die meisten Samples verteilt sein. Man kann nun anhand von verschiedenen Methoden, wobei sich eine globale Durchschnittsbildung oder „feiner“, eine Durchschnittsbildung über lokalen Zentren anbietet, aus den verschiedenen Samples, die wahrscheinlichste Objektposition schätzen mit

$$\varepsilon[f(x_t)] = \sum_{n=1}^{N} \pi_t^{(n)} \cdot f(x_t^{(n)}), \tag{2.5}$$

wobei man eine geeignete $f(x)$ verwendet, beispielsweise $f(x) = x$. Dieser Vorgang ist in Abbildung 2.5 anhand des Kopfes eines Kindes graphisch dargestellt.

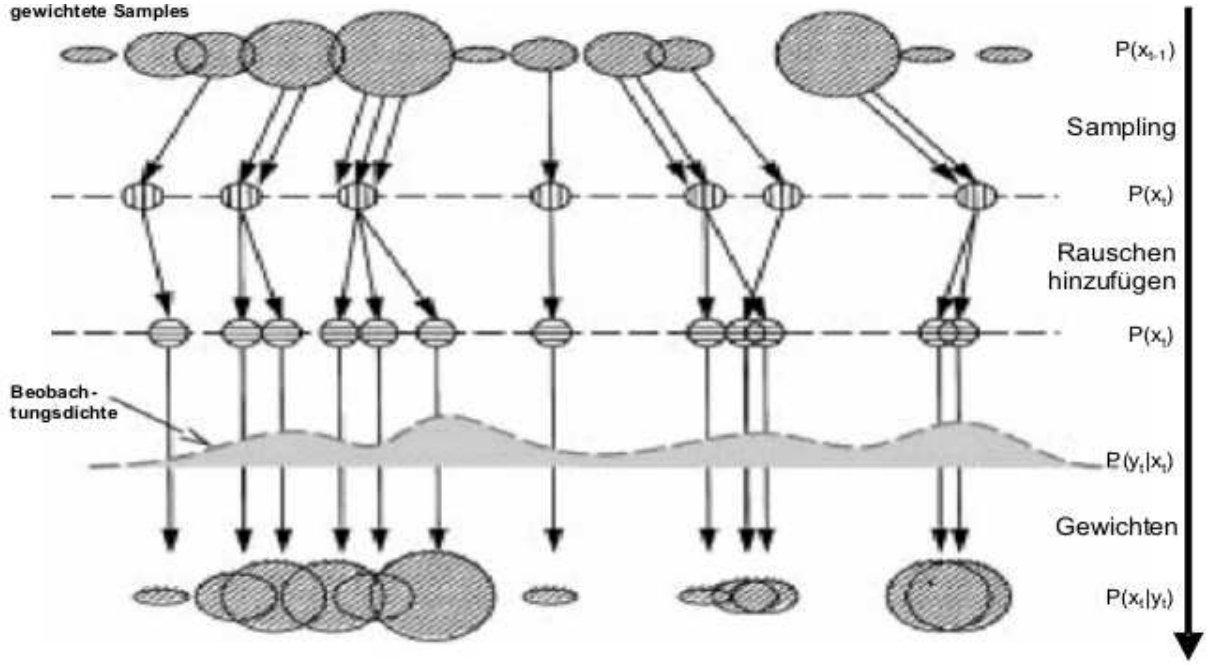


Abb. 2.4: Bildhafter Ablauf des Condensation Algorithmus. Jeder der drei Schritte *Sampling* - *Rauschen hinzufügen* - *Gewichten* findet sich in der Iteration des Algorithmus wieder.

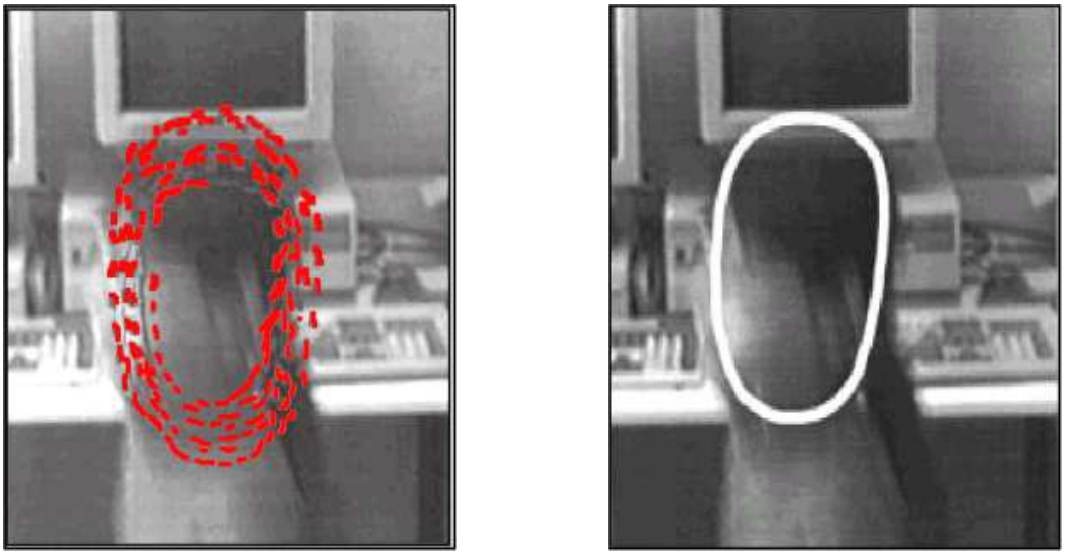


Abb. 2.5: Eine Menge an Hypothesen (Sample-Set, links). Wahrscheinlichste Position des Objektes (rechts).

3.3 Modellierung der Objektdynamik

In diesem Abschnitt wird kurz erläutert, wie die Objektdynamik modelliert wird. Die Entwickler des Condensation Algorithmus führen folgende lineare Differentialgleichung zweiter Ordnung an:

$$x_t - \bar{x} = A \cdot (x_{t-1} - \bar{x}) + B \cdot w_t. \tag{2.6}$$

$\bar{x}$ ist der Durchschnittswert des Zustands, A und B sind Matrizen, die die deterministischen und stochastischen Komponenten des dynamischen Modells repräsentieren und w_t ist eine Variable, die das Rauschen

ausdrückt. Gleichung 2.7 beschreibt folgende Wahrscheinlichkeit $P(x_t|x_{t-1})$:

$$P(x_t|x_{t-1}) \propto \exp -\frac{1}{2} \parallel B^{-1} \cdot ((x_t - \bar{x}) - A \cdot (x_{t-1} - \bar{x})) \parallel^2 \tag{2.7}$$

Hier sind $A \cdot (x_{t-1} - \bar{x_t})$ als die Vorhersage des neuen Zustands und $x_t - \bar{x}$ als der neue Zustand zu verstehen, wobei jeweils die Differenz mit dem Mittelwert $\bar{x}$ gebildet wird. Ist die Differenz der beiden Werte Null, also $(x_t - \bar{x}) - A \cdot (x_{t-1} - \bar{x_t}) = 0$, stimmt die Vorhersage genau. B^{-1} gewichtet den Term in Bezug auf das Rauschen. Je größer nun der Abstand zwischen Vorhersage und neuem Zustand ist, desto kleiner ist die Wahrscheinlichkeit $P(x_t|x_{t-1})$. Es empfiehlt sich, die Parameter A, $\bar{x}$ und B des dynamischen Modells je nach Anwendung anzupassen, beispielsweise während das Objekt typische Bewegungen in einer Trainingsphase für den Algorithmus ausführt.

4 Exkurs: Der Kalman Filter

Die Entwickler des Condensation Algorithmus vergleichen eine Anwendung ihres Algorithmus mit dem sehr populären und etablierten Kalman Filter, der 1960 von Rudolph E. Kalman entwickelt wurde. Sie wollen die Robustheit des Condensation Algorithmus bezüglich komplexer Bildhintergründe zeigen. Der erste Teil (4.1) dieses Abschnitts stellt den Kalman Filter kurz vor, während der zweite Teil (4.2) des Abschnitts näher auf die Versuchsparameter, die Versuchsdurchführung und die Ergebnisse eingeht.

4.1 Der Kalman Filter

Der Kalman Filter löst das allgemeine Problem der Schätzung und Vorhersage des zukünftigen Verhaltens eines Prozesses, der als lineares, dynamisches System modelliert ist [5],[6],[7]. Die Untersuchung und Vorhersage des Prozesses erfolgt diskret über die Zeit und rekursiv. Bei jedem Iterationsschritt wird der Zustand des Prozesses neu berechnet, wobei die Matrizen, die die Historie des Prozesses, die Varianzen und die Kovarianzen der Prozeßvariablen beschreiben, aktualisiert werden. Man nimmt eine dem Prozeßrauschen zugrundeliegende Gaußverteilung an. Der Kalman Filter bietet eine geschlossene Lösung. Die folgenden beiden Gleichungen werden als Grundlage zur Beschreibung eines Prozesses angenommen:

$$x_t = A_t \cdot x_{t-1} + w_t \tag{2.8}$$
$$z_t = H_t \cdot x_t + v_t \tag{2.9}$$

x_t ist der Zustandsvektor, also beispielsweise Position oder Geschwindigkeit des Objektes. Er setzt sich zusammen aus der Prozeßmatrix A, die den vorherigen Zustandsvektor x_{t-1} gewichtet und dem Rauschen des Prozesses w_t. Der Meßvektor z_t besteht aus der Meßmatrix H, die den Meßvektor gewichtet und dem Meßrauschen v_t. Die erste Gleichung bestimmt die Entwicklung des Zustands über die Zeit, die zweite Gleichung bringt Zustand und Abmessung in Bezug zueinander.
Um einen so modellierten Prozeß nun mathematisch beschreiben bzw. Aussagen über den Prozeß oder sein zukünftiges Verhalten machen zu können, muß man den Zustand x_t des Systems bestimmen können, was aber nicht nicht direkt möglich ist, da die (unbekannten) Rauschen w_t und v_t vorhanden sind. Diese werden als Zufallsprozesse angenommen, die gaußverteilt sind.
Der Algorithmus besteht aus zwei Schritten, dem *time update* Schritt mit den Gleichungen

$$x := A \cdot x \tag{2.10}$$
$$P := A \cdot P \cdot A^T + Q \tag{2.11}$$

und dem *measurement update* Schritt mit den Gleichungen

$$K_1 := H \cdot P \cdot H^T + R \tag{2.12}$$
$$K := P \cdot H^T \cdot K_1^{-1} \tag{2.13}$$
$$x := x + K \cdot (z - H \cdot x) \tag{2.14}$$
$$P := (I - K \cdot H) \cdot P. \tag{2.15}$$

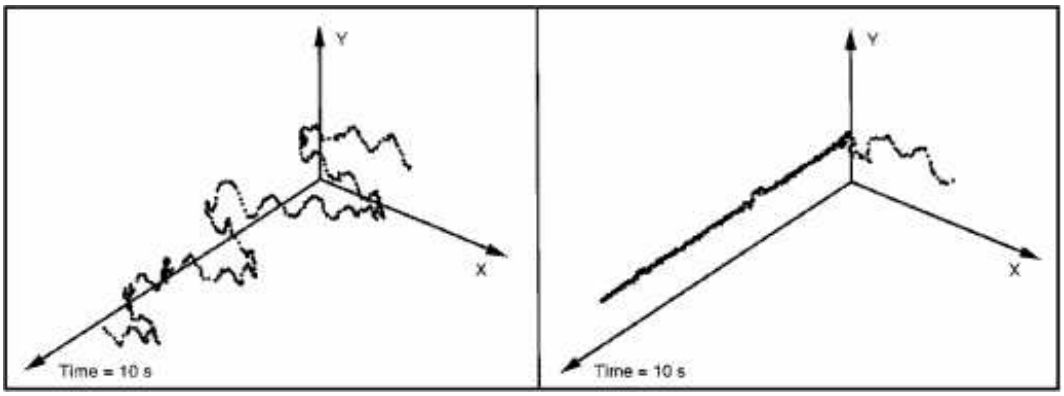


Abb. 2.6: Darstellung der Verfolgung eines Objektes als Koordinatensystemplot (Position des Objektes (x- und y-Achse) im Bild zu einem bestimmten Zeitschritt (z-Achse)). Der Condensation Algorithmus (links). Der Kalman Filter (rechts).

Im *time update* Schritt werden der a priori Schätzwert x (Gleichung 2.10) für den Zustand des nächsten Zeitschrittes und die Fehlerkovarianzmatrix P (Gleichung 2.11) berechnet, im *measurement update* Schritt wird der neue Meß wert x in die Berechnung einbezogen, die Kalman-Gain berechnet (Gleichungen 2.12 und 2.13), der a posteriori Schätzwert (Gleichung 2.14) und die Fehlerkovarianzmatrix P (Gleichung 2.15) bestimmt. Der endgültige Algorithmus beschreibt dann die geschlossene Lösung des Prozesses.

4.2 Vergleich des Condensation Algorithmus mit dem Kalman Filter

Die Entwickler vergleichen eine Anwedung des Condensation Algorithmus mit dem Kalman Filter. Sie machen dies anhand eines zehn Sekunden dauernden Videobeispiels (500 Einzelbilder), in dem ein Kind unruhig vor einem komplexen Hintergrund hin- und hertänzelt. Die Aufgabe ist die Verfolgung des Kopfes des Kindes, der durch einen *Shape-Space* beschrieben wird, der auf einer handgemalten Umrandung des Kopfes basiert.
Dem Bewegungsmodell zugrunde liegen zwei aufeinander aufbauende Trainingssequenzen, in denen sich das Kind vor einem einfarbigen Hintergrund bewegt. Dieses Modell wird durch einen Kalman Filter erstellt, der den Kopf des Kindes in der ersten Trainingssequenz zunächst 2,8 Sekunden (140 Einzelbilder) lang korrekt verfolgt. Anhand der aus dieser ersten Sequenz gewonnen Informationen gelingt es dem Kalman Filter, in einer weiteren Trainingssequenz, den Kopf des Kindes 16 Sekunden (800 Einzelbilder) lang korrekt zu verfolgen.
Der Vergleichstest findet nun mit dem tanzenden Kind vor dem schon erwähnten komplexen Hintergrund statt, wobei sowohl dem Condensation Algorithmus als auch dem Kalman Filter das aus der 16 Sekunden dauernden Trainingsequenz gewonne Bewegungsmodell zugrunde liegt.
Hier zeigt sich - so die Autoren - die Robustheit des Condensation Algortihmus bezüglich komplexer Bilder. Obwohl der Algorithmus nur auf einer Sampledichte von $N = 100$ operiert, gelingt eine korrekte Verfolgung des Kopfes über die gesamten zehn Sekunden (Abbildung 5). Der Kalman Filter verliert den Kopf des Kindes nach 0,8 Sekunden (40 Bildern) und ist bis zum Ende der Videosequenz nicht zu einer Wiedererkennung des gesuchten Objekts im Stande.
Offenbar ist in diesem Fall die Eigenschaft des Condensation Algorithmus, verschiedene Hypothesen gleichzeitig verfolgen zu können, dem Kalman Filter (der ja auf der Annahme einer Gauß-Verteilung operiert und eine geschlossene Lösung bietet) überlegen.

5 Zusammenfassung

Der Condensation Algorithmus zielt auf die Verfolgung eines Objektes in einer Videosequenz ab. Dazu wird ein probabilistischer Ansatz benutzt, der sich des Theorems von Bayes bedient und auf einer diskreten Menge an Hypothesen operiert. Samplestellen innerhalb eines Bildes repräsentieren eine Wahrscheinlichkeitsverteilung $P(x)$. Durch den Vergleich mit den aktuellen Bildinformationen wird diese Verteilung gewichtet und fortwährend angepaßt. Das Theorem von Bayes dient dazu, die schwierige Aufgabe, die

direkte Bestimmung der Objektposition im Bild, durch die einfachere Aufgabe, ob sich das Objekt in einem vermuteten Zustand befindet, auszudrücken. Je nach Anwendung können sehr verschiedene Ansätze verwendet werden, um den Zustand des gesuchten Objekts zu beschreiben. Isard und Blake bedienen sich in ihrer Arbeit der Shape-Spaces, aber auch andere Ansätze sind denkbar.
Man muß bei der Implementierung des Algorithmus einen Kompromiß zwischen Echtzeitanalyse und Genauigkeit eingehen: Je genauer die Wahrscheinlichkeitsverteilung ist (je mehr Samples man im Bild verteilen läßt), um so rechenintensiver ist die Analyse. Die Autoren führen in einigen Videobeispielen auf, daß oft schon eine Zahl von 100 Samples genügt, um ein für Echtzeitanalysen hinreichend genaues Ergebnis zu erzielen. Leider läßt sich dies nicht immer ohne weiteres an den Videobeispielen erkennen. Die Vermutung, daß die Autoren aufsehenerregende Videosequenzen (beispielsweise die Bewegung eines Strauchblattes im Wind) so lange mit unterschiedlichen Parametern getestet haben, bis sie eine für diese Anwendung genügende Genauigkeit erreicht haben, läßt sich oft nicht vollständig von der Hand weisen, da eine unabhängige Testquelle nicht genannt wird.
Da der Condensation Algorithmus aber in seinen Komponenten leicht verändert (beispielsweise das zugrunde liegende Bewegungsmodell) und so den Erfordernissen bestimmter Anwendungen angepaßt werden kann, scheint er ein guter Ansatz für die verschiedenen Problemstellungen der Objektverfolgung zu sein.

Literaturverzeichnis

[1] Isard M, Blake A: Condensation - Conditional Density Propagation for Visual Tracking. International Journal of Computer Vision, 1998; 29(1):5-28.

[2] Isard M: The Condensation Algorithm Home Page. University of Edinburgh, Division of Informatics, Sept. 1998; http://www.dai.ed.ac.uk/CVonline/LOCAL_COPIES/ISARD1/condensation.html

[3] Fisher RB: CVonline - Compendium of Computer Vision. Division of Informatics University of Edinburgh, Juli 2002; http://www.dai.ed.ac.uk/CVonline

[4] Denzler J: Probabilistische Folgenmodellierung. Universität Erlangen - Nürnberg, Juli 2001; http://www5.informatik.uni-erlangen.de/lectures/SS01/LectureDescription/profo/archive.htm

[5] Guéziec A: Tracking Pitches for Broadcast Television; Computer; 35(3):38-43, März 2002

[6] Welch G, Bishop G: An Introduction to the Kalman Filter. University of North Carolina - Chapel Hill, Department of Computer Science, 1998; http://www.informatik.uni-bonn.de/III/lehre/seminare/Mustererkennung/WS99/kalman.pdf

[7] Welch G: The Kalman Filter Home Page. University of North Carolina - Chapel Hill, Department of Computer Science, 2002; http://www.cs.unc.edu/∼welch/kalman/

Seminar Medizinische Bildverarbeitung: Segmentierung medizinischer Bilddaten mittels Analyse von Isolabel-Kontur Karten

von Paul Herr

Betreuer: Mark Oliver Güld

Inhaltsverzeichnis

Zusammenfassung

Automatische Segmentierungsverfahren haben oft Schwierigkeiten bei der Erkennung von Objektgrenzen, wenn Objekte mit ähnlicher Intensität nah beieinander liegen oder teilweise wegen des Partialvolumen–Effekts verwaschene Grenzen haben. In diesem Beitrag wird ein Verfahren vorgestellt, welches dieser Herausforderung gerecht werden soll. Das von den Entwicklern "intrinsic shape for segmentation" (ISeg) genannte Verfahren analysiert Isolabel-Kontur Karten, um Bildregionen zu erkennen, die wichtige Objekte repräsentieren. Dazu generiert ISeg mit Hilfe von Multilevel-Intensitätsschwellenwertverfahren eine Isolabel-Kontur Karte des vorliegenden Bildes und ermittelt die Objektgrenzen, indem es die benachbarten Konturen schrittweise von innen nach außen miteinander vergleicht. ISeg erfordert keine manuelle Vorverarbeitung der Bilddaten und auch keine Vorgabe der Objektkonturen. Anschließende Validierung soll zeigen, daß ISeg robuster ist als die konventionellen Schwellenwertverfahren und Ergebnisse liefert, die mit den Ergebnissen manueller Segmentierung vergleichbar sind.

Keywords: Segmentierung, Angiographie, Konturenvergleich, Schwellenwertverfahren

1 Einleitung

Segmentierung (Unterteilung der Bildelemente in zusammenhängende Regionen, die verschiedene Objekte darstellen; Trennung der Objekte vom Hintergrund) ist ein wichtiger Schritt in vielen Verfahren, die auf medizinischen Bilddaten operieren. So ist Segmentierung beispielsweise notwendig bei der Erkennung und Beurteilung von Ausprägungen morphologischer Krankheiten (morphologisch: formverändernd), bei Bestrahlungsplanungen oder auch bei der Konstruktion anatomischer Modelle. Das in dieser Arbeit vorgestellte Verfahren entstand aus der Notwendigkeit, die Gefäße und die Gefäßstruktur, die auf den CT-Angiograpie-Aufnahmen (CTA) (Angiographie: Gefäßuntersuchung) sichtbar sind, zu visualisieren.

ISeg [1] wurde an der Stanford University von Smadar Schiffman, Geoffrey G. Rubin und Sandy Napel entwickelt. Ein Ziel bei der Entwicklung von ISeg war es, die Interaktion mit dem Benutzer zu reduzieren. Die Entwickler haben zwei Eigenschaften ausgemacht, die alle CTA-Aufnahmen gemeinsam haben. Zum einen ist es die Ähnlichkeit und räumliche Nähe der Gefäße und der Knochen auf den CTA-Aufnahmen: beide erscheinen als helle Regionen. Wenn sich die Knochen und die Gefäße berühren, so treten Partialvolumen-Effekte (PVE) (Partialvolumeneffekt: Abnahme der Intensität, wenn ein Objekt nicht vollständig in der Bildebene liegt) auf, so daß die Betrachtung der Intensität alleine nicht ausreicht, um die Objektgrenzen zuverlässig zu ermitteln. Zum anderen fallen die Unterschiede im Aussehen der Gefäße und der Gefäßstruktur bei verschiedenen Patienten auf, die die Verwendung von *a priori*-Kenntnissen über das Aussehen der Objekte erschweren. ISeg soll diesen Herausforderungen gerecht werden und so zuverlässig Grenzen der Objekte erkennen, die sich berühren und verschiedene Formen haben können.

2 Übersicht existierender Segmentierungsverfahren

Es existiert eine Reihe Segmentierungsmethoden, die im Folgenden kurz vorgestellt werden sollen.

2.1 Manuelle Segmentierung

Bei der manuellen Segmentierung markieren Mediziner die Objektgrenzen, sie treffen auch die Entscheidung, ob ein Bildbereich zu einem Objekt oder zu dem Hintergrund gehört. Dem Computer fällt dabei die Rolle eines Werkzeugs zu. Es gibt aber auch Segmentierungsumgebungen, wie zum Beispiel General Electric, Inc., Advantage Windows Workstation, die Segmentierung erleichtern, da sie Tools enthalten, die dem Segmentierer beispielsweise mögliche Objektgrenzen vorschlagen oder sie wenden vom Segmentierer vorgegebene Konturen auf eine Bildserie an. Die Entscheidung Objekt/Hintergrund wird aber auch in diesem Fall dem Mediziner überlassen. Manuelle Segmentierung ist zeitaufwendig, und damit auch teuer. Außerdem weisen die Segmentierungsergebnisse verschiedener Mediziner unter Umständen deutliche Unterschiede auf.

2.2 Modelle, die *a priori*-Kenntnisse nutzen

Einige Verfahren benutzen a-priori-Kenntnisse über die vorliegende Objekte. Diese Kenntnisse werden in Form von geometrischen oder Wahrscheinlichkeitsmodellen zur Verfügung gestellt. Bevor man mit einem solchen Verfahren arbeiten kann, müssen zunächst Modelle erstellt und parametriert werden. Das geschieht üblicherweise bei einem Trainigsprozess, bei dem Experten Bilder manuell segmentieren, der Algorithmus extrahiert daraus die Modellparameter. In den letzten Jahren wurde eine Reihe solcher Modelle entwickelt, die zunehmend bessere Resultate liefern. Leider verschlechtern sich die Ergebnisse mit der Abnahme des Verwandschaftsgrades der zu segmetierenden Bilder zu den Trainingsbildern. Außerdem sind gerade die CTA-Aufnahmen nur schwer modellierbar, denn die Variabilität im Aussehen der einzelnen Objekte ist zu hoch.

2.3 Semiautomatische Segmentierungsverfahren

Es gibt semiautomatische Segmentierungsverfahren, die den räumlichen Zusammenhang der Bildelemente untersuchen. Dabei sind die Intensitätswerte ausschlaggebend für die Entscheidung Objekt/Hintergrund. Die Benutzung eines Intensitätsschwellenwertes zum Treffen dieser Entscheidung ist einfach zu implementieren, leider haben Objekte auf den medizinischen Bildern jedoch oft überlappende Intensitätsverteilungen, so daß der Benutzer erst den optimalen Schwellenwert im “Try and Error”-Verfahren experimentell ermitteln muß.

Nach Meinung der Entwickler von ISeg können die Objekte der CTA-Aufnahmen mit den existierenden Verfahren nicht adäquat segmentiert werden, da sie einen hohen Benutzeraufwand erfordern und/oder nicht immer akzeptable Ergebnisse liefern. ISeg ist ein alternatives Segmentierungsverfahren, welches den Benutzeraufwand minimiert, und von den Abweichungen in der Erscheinung der Objekte nicht betroffen ist, da es keine Modelle zur Segmentierung verwendet.

3 Grundlagen

Dieses Kapitel stellt einzelne Schritte des Verfahrens genauer vor, das Zusammenspiel dieser Schritte wird im nächsten Kapitel vorgestellt. Grob läß sich die Vorgehensweise von ISeg folgendermaßen beschreiben: Es trennt ein Bild in verschiedene Regionen auf, ausgehend von der Intensität der Bildpunkte. Die Grenzen dieser Regionen, die Konturen, werden zur Erkennung von Objekten herangezogen, indem sie von innen nach außen miteinander verglichen werden. Beobachtungen zur Folge unterscheiden sich zwei Konturen stark im Aussehen, wenn die eine Kontur innerhalb des Objekts liegt und die andere Teile des Hintergrunds mit einschließt. Es liegt also nahe, diese Eigenschaften der Isolabel-Konturen zum Erkennen der Objektgrenzen einzusetzen, indem man je zwei benachbarte Konturen miteinander vergleicht und den Grad der Übereinstimmung ermittelt. ISeg benutzt Interpolation, um die Konturen zu glätten, und so das Rauschen zu minimieren. Das Ergebnis der Segmentierung ist um so genauer, je mehr Konturen zur Vefügung stehen, allerdings steigt mit deren Anzahl auch der Berechnungsaufwand.

3.1 Multilevel-Intensitätsschwellenwertverfahren

ISeg verwendet bei der Segmentierung Informationen über die Form der Objekte. Diese Informationen werden aus sogenannten Isolabel-Kontur Karten gewonnen. Zum Erstellen dieser Karten verwendet ISeg ein Multilevel-Intensitätsschwellenwertverfahren, welches im Gegensatz zu binärem Schwellenwertverfahren mit mehreren Schwellenwerten arbeitet. Bei dem Multilevel-Intensitätsschwellenwertverfahren wird der gesamte Intensitätsbereich I in disjunkte Teilbereiche $s_j = [i_j, k_j)$ unterteilt, es gilt: $\bigcup s_j = I$. Liegt die Intensität eines Punktes in einem Intensitätsteilbereich s_j, so wird er mit einem numerischen Label l_j ausgezeichnet, welches die Zugehörigkeit des Punktes zu dem Schwellenwert s_j wiedergibt. Zusammenhängende Punkte des Labels l_j bilden eine Region $r_{ji}, i = 1, \ldots, n$, wobei n die Anzahl der Regionen zu dem Label l_j ist.
Die Grenzen dieser Bereiche sind die Konturen, deren bestimmte Eigenschaften ISeg ausnutzt, um die Objektgrenzen zu ermitteln. Abbildung 3.1 zeigt die Entstehung einer Isolabel-Kontur Karte aus einem Bild.

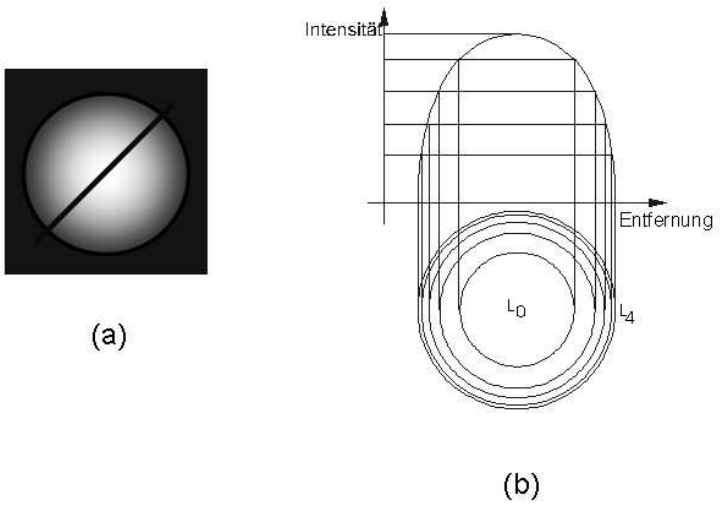


Abb. 3.1: Entstehung einer Isolabel-Kontur Karte (a) Ein Bild mit charakteristischen Merkmalen einer CTA-Aufnahme: dunkler Hintergrund und Intensitätsabfall von Objektmitte zu den Objektgrenzen hin. (b) Oben: Intensitätsprofil des Objekts aus dem Bild a entlang der Linie. Unten: Die Isolabel-Konturen des Objekts, l_0 ist die Kontur, die zu dem höchsten Schwellenwert t_0 gehört.

3.2 Eigenschaften der Isolabel-Kontur Karten

Das Muster pseudokonzentrischer Konturen, welches durch das Multilevel-Intensitätsschwellenwertverfahren entsteht, ähnelt den Höhenlinien geographischer Karten, weshalb auch die Bezeichnung Isolabel-Kontur Karte verwendet wird.
Das Aussehen der einzelnen Konturen einer Isolabel-Kontur Karte hängt im Wesentlichen von dem Grad des Bildrauschens, von der Intensitätsverteilung innerhalb des Bildes sowie von dem Ausmaß des PVE ab. Eine Eigenschaft der Isolabel-Konturen, die sich ISeg zunutze macht, um die Objektgrenzen zu ermitteln, ist, daß sich zwei Konturen, die innerhalb eines Objektes liegen, ähnlich zueinander sind. Sie können sich zwar in der Größe, in der relativen Lage innerhalb des Objektes unterscheiden, die Gestalt der Konturen bleibt aber ähnlich. Zwei Konturen, die Bildpunkte verschiedener Objekte oder auch Bildpunkte des Hintergrundes enthalten, unterscheiden sich auch in der Form. Abbildung 3.2 zeigt ein Beispiel einer Isolabel Kontur-Karte für eine CTA-Aufnahme.

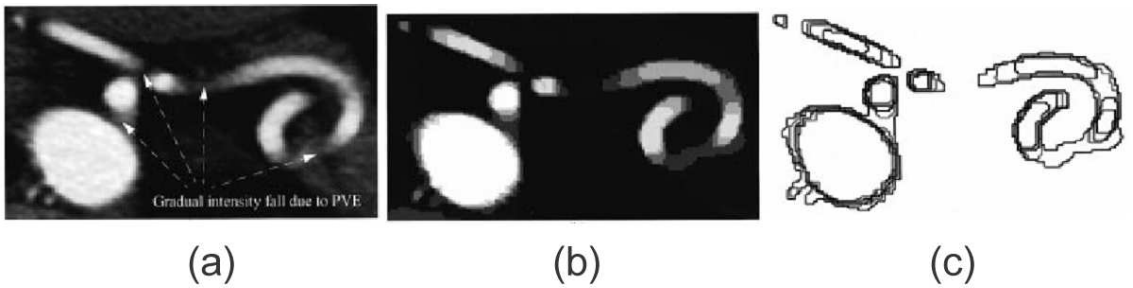


Abb. 3.2: Eine CTA-Aufnahme und die dazugehörige Isolabel-Kontur Karte. (a) CTA-Aufnahme mit mehreren Stellen, an denen die PVE auftreten. Durch den daraus resultierenden Intensitätsabfall wird es schwierig, das gewundene Gefäß als ein Objekt zu erkennen. (b) Die Isolabel-Kontur Karte. Verschiedene Grautöne stehen für verschiedene Labels. (c) Muster der Isolabel-Konturen. Man beachte die Aneinanderreihung der Konturen an Stellen mit abruptem Intensitätsgradienten und weite Abstände an den Stellen mit monotonen Intensitätsgradienten.

Die Extraktion der Konturen wird durchgeführt, indem ausgehend von den Grenzpunkten der zuletzt betrachteten Kontur jeweils deren acht Nachbarn betrachtet werden. Haben die Nachbarn das Label, das zu dem aktuell betrachteten Intensitätsschwellenwert gehört, wird die Suche ausgedehnt, bis Punkte entdeckt werden, die als Nachbarn Punkte mit dem Label des nächstniedrigeren Intensitätsschwellenwertes

haben. Diese Punkte werden in der Koordinatenliste der aktuellen Kontur gespeichert. Abbildung 3.3 zeigt die Extraktion einer Kontur ausgehend von einer anderen.Die Beziehung der Isolabel-Konturen untereinander läßt sich als eine Umschließen-Beziehung beschreiben. Zunächst gilt für je zwei Konturen einer Isolabel-Kontur Karte c_i, c_j, die die Regionen r_i, r_j umschließt, folgende Aussage:

$$\langle r_i \cap r_j = \emptyset \rangle \vee \langle r_i \cap r_j = r_i \rangle \vee \langle r_i \cap r_j = r_j \rangle \tag{3.1}$$

Diese Aussage bedeutet, daß je zwei Konturen entweder völlig disjunkt sind, oder eine der beiden die andere umschließt. Die Umschließen-Beziehung R_e läßt sich folgendermaßen definieren: seien r_{i+1} und r_i die Menge der Bildpunkte, die von den Konturen c_{i+1},c_i umschlossen werden, dann gilt $(c_{i+1}, c_i) \in R_e$ (d.h. c_{i+1} umschließt c_1) genau dann, wenn gilt: $r_{i+1} \supseteq r_i$. Diese Definition erlaubt die Darstellung der Beziehung der Konturen einer Isolabel-Kontur Karte durch einen Baum. Die innersten Konturen, die keine weitere Kontur beinhalten, sind dabei die Blätter. Diese umschließen die hellsten Objektbereiche. Abbildung 3.4 zeigt eine symbolische Isolabel-Kontur Karte mit dem dazugehörigen Baum der Konturen.

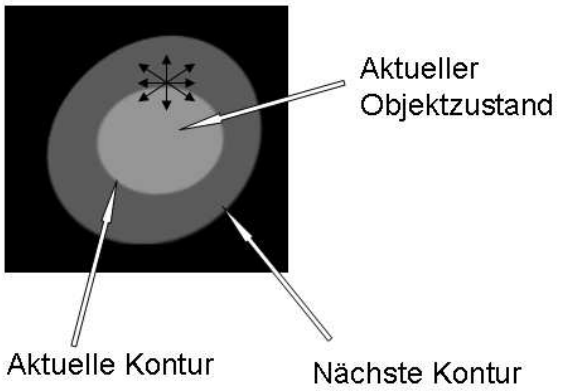


Abb. 3.3: Extraktion einer Kontur, ausgehend von einer anderen.

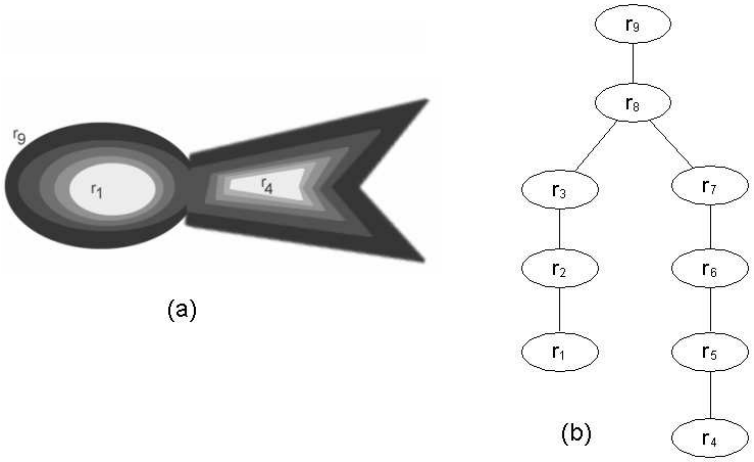


Abb. 3.4: Baumartige Anordnung der Konturen einer Isolabel-Kontur Karte. (a) Eine (hypothetische) Isolabel-Kontur Karte eines Bildes, welches zwei Objekte enthält, die nahe beieinander liegen und eine ähnliche Intenstätsverteilung haben. (b) Die dazugehörige baumartige Anordnung der Konturen. Die innersten Regionen r_4 und r_1 sind die Blätter des Baumes, die Region r_8, die beide Objekte umschließt, vereinigt die Zweige der Objekte zu einem Zweig.

Nachdem die Konturen extrahiert wurden, wird für je zwei benachbarte Konturen der Grad der Übereinstimmung ermittelt. “Benachbart“ bedeutet, daß zwischen diesen Konturen keine weitere Kontur verläuft, z.B. sind r_1 und r_2 in Abb. 3.4 benachbarte Konturen. Zur Analyse der Konturen wird folgende Definition

benutzt: gegeben ist eine Folge der Konturen $C = c_1, \ldots, c_{n-1}, c_n$ und das Ähnlichkeitsmaß $D(c_i, c_j)$, welches jedem Paar c_i,c_j der benachbarten Konturen den Grad der Übereinstimmung zuweist; C ist genau dann eine Folge ähnlicher Konturen, wenn für jedes Paar $c_{i-1}, c_i \in C, D(c_{i-1}, c_i) < T_m$ gilt, wobei T_m ein vordefinierter Schwellenwert ist. T_m muß so gewählt sein, daß zwei benachbarte Konturen als ähnlich gelten, wenn zwischen den beiden Konturen kein Übergang zu einem anderen Objekt oder zum Hintergrund stattfindet, auch wenn sie in der Größe, Ausrichtung und Lage Unterschiede aufweisen können. Abbildung 3.5 zeigt eine Folge benachbarter Konturen.

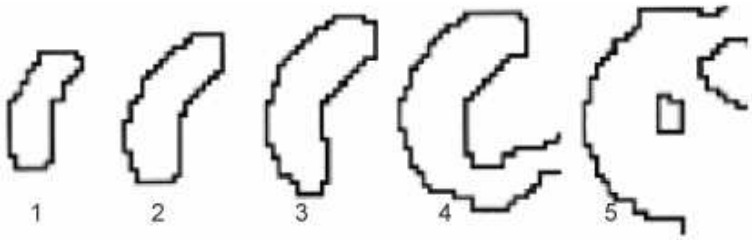


Abb. 3.5: Eine Folge der benachbarter Konturen der CTA-Aufnahme aus Abbildung 3.2. Man beachte die Ähnlichkeit der Konturen 1 bis 4 und den Unterschied in der Form zwischen den Konturen 4 und 5, der auf einen Übergang zu dem Hintergrund deutet.

3.3 Kubische Spline-Interpolation

Interpolation wird angewendet, um, ausgehend von einigen Stützpunkten, eine Funktion zu rekonstruieren. ISeg benutzt Interpolation zur Glättung der Konturen. Es gibt dabei mehrere Möglichkeiten, ISeg setzt die kubische Spline-Interpolation ein. Leider geben die Entwickler in ihrem Bericht nicht an, warum sie ausgerechnet diesem Verfahren den Vorzug geben.
Definition:

Sei Δ_n eine Unterteilung des Intervalls $[a, b]$:
$\Delta_n : a = x_0 < x_1 < \ldots < x_{n-1} < x_n = b$
mit Teilintervallen $[x_{j-1}, x_j]$, und seien $f_0, \ldots, f_n$ die Werte an den Stellen $x_0, \ldots, x_n$

Eine Funktion $S[a, b] \rightarrow I\!R$ heißt kubischer Spline, falls gilt:

- S ist über dem gesamten Intervall zweimal stetig differenzierbar
- S ist auf jedem Teilintervall ein Polynom 3.Grades:
  $S(x) = s_j(x) = a_j + b_j(x - x_{j-1}) + c_j(x - x_{j-1})^2 + d_j(x - x_{j-1})^3$ für $x \in [x_{j-1}, x_j]$ und $j = 1 \ldots, n$

Ein kubischer Spline hat demnach $4n$ Parameter, die folgendermaßen bestimmt werden:

- Interpolationseigenschaft:

$$s_j(x_{j-1}) = f_{j-1}, s_j(x_j) = f_j, \forall j = 1, \ldots, n \tag{3.2}$$

- Stetigkeit der ersten Ableitung:

$$s'_j(x_j) = s'_{j+1}(x_j), \forall j = 1, \ldots, n-1 \tag{3.3}$$

- Stetigkeit der zweiten Ableitung:

$$s''_j(x_j) = s''_{j+1}(x_j), \forall j = 1, \ldots, n-1 \tag{3.4}$$

(2) bis (4) liefern $4n-2$ Gleichungen, die restlichen zwei liefert die Art des kubischen Spline (natürlich/ periodisch/allgemein).
Definition:

Ein kubischer Spline heißt:

- **natürlicher Spline**, falls gilt: $S''(a) = S''(b) = 0$
- **periodischer Spline**, falls gilt: $S^{(i)}(a) = S^{(i)}(b), i = 0, 1, 2$
- **allgemeiner Spline**, falls (z.B.) gilt: $S'(a) = f'_0, S'(b) = f'_n$

3.4 Vergleich zweier Konturen

Das Mapping-Verfahren [2] kommt aus dem Bereich Mustererkennung und wird von ISeg dazu benutzt, den Ähnlichkeitsgrad zweier Konturen zu bestimmen. Auch hier geben die Entwickler nicht an, warum sie Mapping und kein anderes Verfahren einsetzen. Um zwei Objekte mit Hilfe von Mapping auf Übereinstimmung zu prüfen, müssen die Grenzen der Objekte als Winkelsequenzen vorliegen. Beliebige Konturen können durch eine Winkelfunktion beschrieben werden, die jedem Punkt einer Kontur den Winkel zuordnet, der die Neigung der Tangente gegenüber der x-Achse angibt [3]. Abbildung 3.6 zeigt ein Polygon und die dazugehörige Winkelfunktion. Eine Winkelsequenz ist eine Folge der Werte der Winkelfunktion an bestimmten Stellen.

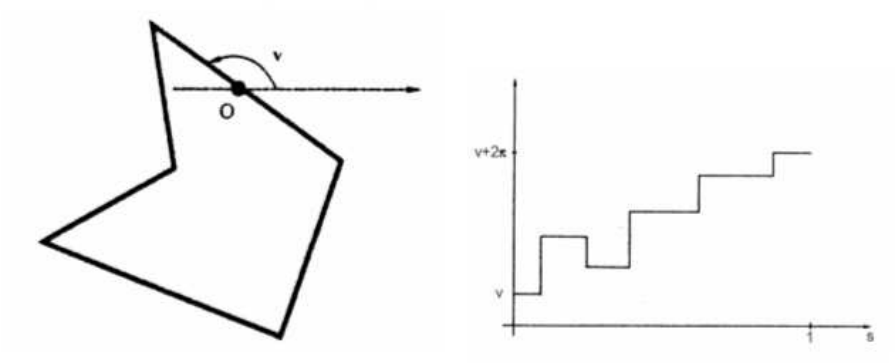


Abb. 3.6: Entstehung einer Winkelsequenz. Links: Ein Polygon, von dem die Winkelsequenz ermittelt werden soll, O ist der Anfangspunkt, v die Neigung der Tangente gegenüber der x-Achse. Rechts: die dazugehörige Winkelfunktion [3].

Ein Vergleich der Konturen findet durch ein Mapping der dazugehörigen Winkelsequenzen aufeinander statt.
Definition:

Ein (m, n)-Mapping ist eine Menge $M \subseteq \{1, 2, \ldots, m\} \times \{1, 2, \ldots, n\}$, wobei jedes Paar $\langle i, j\rangle \in M$ eine Kante, d.h eine Zuordnung eines Elements der ersten einem Element der zweiten Menge, darstellt. Es gilt:

- jedes Element der Menge $\{1, 2, \ldots, m\}$ ist die erste Komponente i mindestens einer Kante $\langle i, j\rangle \in M$
- jedes Element der Menge $\{1, 2, \ldots, n\}$ ist die zweite Komponente j mindestens einer Kante $\langle i, j\rangle \in M$
- Es gibt keine sich kreuzenende Kanten, d.h. es existieren keine i, i', j, j' so daß gilt: $i < i', j < j'$ und $\langle i, j'\rangle, \langle i', j\rangle \in M$

Ein (m, n)-Mapping ist *minimal*, falls keine echte Untermenge von M ein (m, n)-Mapping ist. Abbildung 3.7 zeigt ein minimales (9,9)-Mapping.
Um Aussagen über die Ähnlichkeit zweier Objekte zu treffen, ist ein Mapping der Konturen, die die Objekte umschließen, mit minimalen Kosten zu finden. Die Kosten eines Mappings sind dabei folgendermaßen definiert:

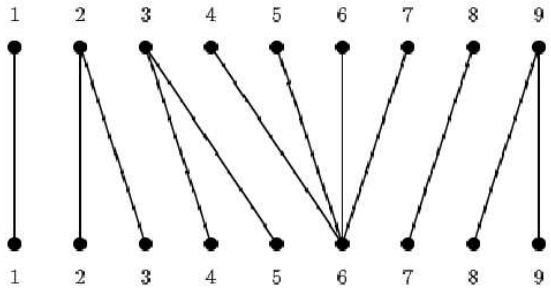


Abb. 3.7: Ein minimales (9,9)-Mapping.

Definition:

Seien $X = x_1, x_2, \ldots, x_m$ und $Y = y_1, y_2, \ldots, x_n$ zwei Winkelsequenzen und M ein (m, n)-Mapping. Die Kosten jeder Kante $\langle i, j \rangle$ sind:

$$cost(\langle i, j \rangle, X, Y) = \min(\mid x_i - y_j \mid, 2\pi - \mid x_i - y_j \mid) \tag{3.5}$$

Für die Kosten des Mappings gilt:

$$cost(M, X, Y) = \sum_{e \in M} cost(e, X, Y) \tag{3.6}$$

Um Aussagen über die Ähnlichkeit zweier Objekte zu treffen, muß ein Mapping der Konturen mit den minimalen Kosten gefunden werden. Dies ist mit dynamischer Programmierung in $O(mn)$ möglich [4]. Abbildung 3.8 zeigt zwei Winkelsequenzen und das dazugehörige Mapping mit den minimalen Kosten. Nach der Ermittlung des Mappings mit den minimalen Kosten werden die Kosten des Mappings mit dem experimentell ermittelten Schwellenwert verglichen, liegen sie unterhalb dieses Schwellenwertes, gelten die innerhalb der verglichenen Konturen liegenden Objekte als ähnlich.

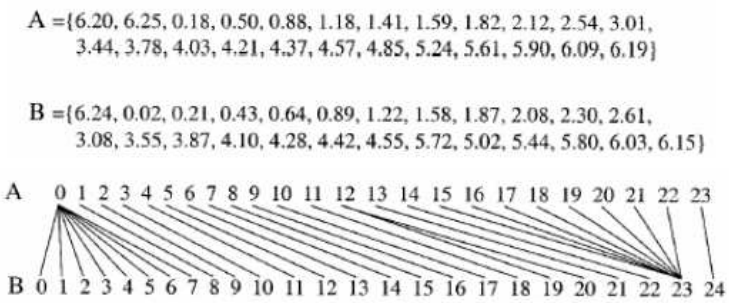


Abb. 3.8: Zwei Winkelsequenzen (Winkelangaben im Bogenmaß) und das dazugehörige Mapping mit den minimalen Kosten

4 ISeg: Arbeitsweise

Nachdem alle nötigen Grundlagen vorgestellt worden sind, kann die Arbeitsweise von ISeg nun genauer vorgestellt werden. Als ersten Schritt empfehlen die Entwickler die Vorbehandlung der Bilddaten mit einem Anisotrophic-Diffusion Filter [5], um den Einfluß des Bildrauschens zu reduzieren. Dieser Schritt ist jedoch kein Bestandteil des Verfahrens, deswegen wird in dieser Arbeit nicht näher darauf eingegangen. Im ersten Schritt wird eine Isolabel-Kontur Karte des Bildes mit dem Multilevel-Intensitätsschwellenwertverfahren erstellt. Die Anzahl der Schwellenwerte sollte hinreichend groß sein, denn die mit ISeg

ermittelten Objektgrenzen sind Näherungen, die um so genauer sind, je feiner die Intensitätsunterteilung bei der Erstellung der Isolabel-Kontur Karten ist. Dabei steigt aber natürlich auch die Anzahl der Konturvergleiche. Dann extrahiert ISeg die einzelnen Konturen und vergleicht die benachbarten Konturen miteinander. Es werden nur Konturen verglichen, deren Regionen nicht disjunkt sind, also Konturpaare, in denen eine Kontur die andere umschließt. ISeg geht dabei von innen nach außen vor, nähert sich also solange den Objektgrenzen, bis es eine zu große Abweichung in den benachbarten Konturformen entdeckt. An dieser Stelle bricht ISeg den Vergleich für das aktuelle Objekt ab, es wird angenommen, daß die vorletzte betrachtete Kontur die Objektgrenzen am besten wiedergibt. ISeg setzt jedoch den Vergleich fort, da es annimmt, daß es ein Oberobjekt geben kann, das mehrere Unterobjekte enthält, und die Änderung in der Form der Konturen deswegen so groß ist. Am Ende des Segmentierungsprozesses wird dem Benutzer die Möglichkeit gegeben, die Unterobjekte mit dem Oberobjekt zu verschmelzen.
Bei dem Vergleich der Konturen werden zunächst alle Konturen extrahiert, die zu dem höchsten Intensitätsschwellenwert gehören, dann die des nächstniedrigeren Schwellenwertes u.s.w. Die Arbeitsrichtung von ISeg kann man also als einen Bottom-Up–Graphendurchlauf des Baumes der Konturen (vgl. Abb. 3.4b) betrachten. Abbildung 3.9 zeigt die Reihenfolge der Konturextraktion. ISeg speichert jeden Zweig der benachbarten Konturen in einer Instanz der Datenstruktur, jede Instanz enthält die Liste der Koordinaten der zuletzt besuchten Kontur in diesem Pfad, diese Liste enthält die Koordinaten der einzelnen Punkte der Kontur, welche für die Extraktion der nächsten Kontur benötigt werden. Die zuletzt betrachtete Kontur (vgl. Abb. 3.3) stellt zu dem aktuellen Zeitpunkt den Status des Objekts dar, d.h. sollte die aktuelle Kontur ihr nicht mehr ähnlich sein, wird die zuletzt betrachtete Kontur für die Objektgrenzen stehen.

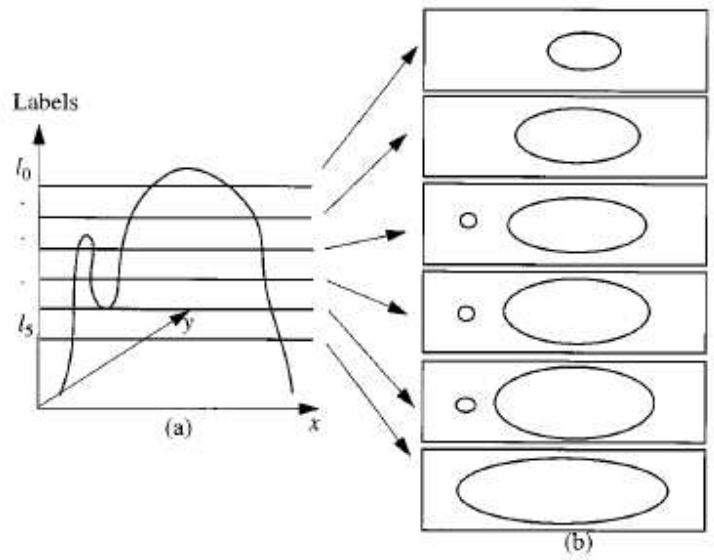


Abb. 3.9: Reihenfolge der Konturenextraktion von ISeg. (a) Das Intensitätsprofil eines Bildes. (b)Konturen, die ISeg extrahiert, die Richtung ist von oben nach unten.

Folgende Prozedur illustriert die Vorgehensweise von ISeg bei der Extraktion der Konturen für ein Label l_j:

- Finde für jede Datenstruktur-Instanz, ausgehend von den gespeicherten Koordinaten der Kontur zu dem Label l_{j+1}, die Kontur zu dem Label l_j und vergleiche diese miteinander.
- Alle noch nicht betrachteten Bildregionen mit dem Label l_j stehen für einen neuen Zweig in der Baumstruktur der Konturen, speichere für jede solche Region die dazugehörige Kontur in einer neuen Instanz der Datenstruktur ab.

Zum Vergleich zweier Konturen verwendet ISeg Mapping (vgl. 3.4). Dazu werden zu den Konturen zunächst die Winkelsequenzen berechnet. Im weiteren Schritt werden die Konturen geglättet, um den Einfluß des Bildrauschens und der geometrischen Verzerrungen zu reduzieren (dabei darf die Glättung

natürlich nicht die charakteristischen Merkmale einer Kontur zunichte machen). Dazu interpoliert ISeg die Kontur, in dem es von der n-elementigen Menge der Grenzpunkte einer Kontur $\lceil n/4 \rceil$ gleichmäßig auf der Kontur verteilte Punkte zufällig auswählt, und die restlichen interpoliert (vgl. 3.2). Um das Ergebnis der Interpolation unabhängig von der Wahl der Kontrollpunkte zu machen, führt ISeg diese Berechnung viermal mit verschiedenen Kontrollpunkten durch, die Ergebnisse werden gemittelt.
Durch die Ausnutzung der Baumstruktur der Konturen erzielt ISeg einen Effizienzvorteil, denn Iseg betrachtet jede Kontur höchstens zweimal. Ohne Ausnutzung dieser Eigenschaft beträgt die Anzahl der Vergleiche bei n Intensitätsschwellenwerten und q Konturen, die von jeder Kontur unmittelbar umschlossen werden

$$(n-2) \cdot q^{n-2}. \tag{3.7}$$

Bei der Ausnutzung der Baumstruktur ist die Anzahl der Vergleiche gleich der Anzahl der Verzweigungen in dem Baum

$$\sum_{i=1}^{n-2} q^i. \tag{3.8}$$

Beide Ausdrücke sind Summen der Potenzen von q, aber die Exponenten sind im zweiten Ausdruck (bis auf den letzten Summanden) kleiner als in dem ersten, daher gilt

$$\sum_{i=1}^{n-2} q^i < (n-2) \cdot q^{n-2}. \tag{3.9}$$

Also beträgt die maximale Ersparnis

$$(n-2) \cdot q^{n-2} - \sum_{i=1}^{n-2} q^i. \tag{3.10}$$

Diese Ersparnis läß sich in der Praxis nicht erreichen, denn die Anzahl der Verzweigungen variiert von Objekt zu Objekt. Für die Praxis nennen die Autoren eine Reduktion um den Faktor 4.5.
Abbildung 3.10 zeigt die Ergebnisse der Segmentierung der CTA-Aufnahme 3.2a. Hier stehen verschiedene Grautöne für verschiedene Objekte. Man erkennt auch Objekte, die mehrere Unterobjekte enthalten. Hier wird dem Benutzer die Möglichkeit gegeben, die Unterobjekte mit dem Oberobjekt zu verschmelzen.

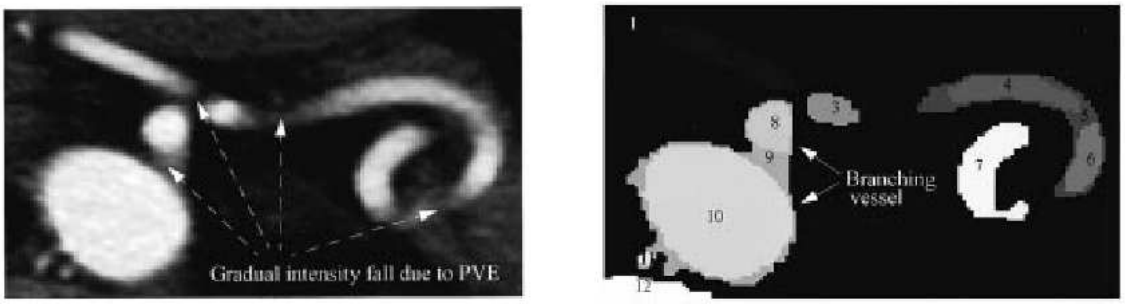


Abb. 3.10: Ergebnisse der Segmentierung. Links: Das Originalbild. Rechts: Das segmentierte Bild.

5 Leistungs- und Stabilitätsbewertung

Bei der Leistungs- und Stabilitätsbewertung haben die Entwickler eine Reihe von Versuchen durchgeführt, die dann statistisch ausgewertet wurden. Um die Ergebnisse besser verstehen zu können, werden im Folgenden die beiden statistischen Verfahren, mit denen die Ergebnisse ausgewertet wurden, vorgestellt.

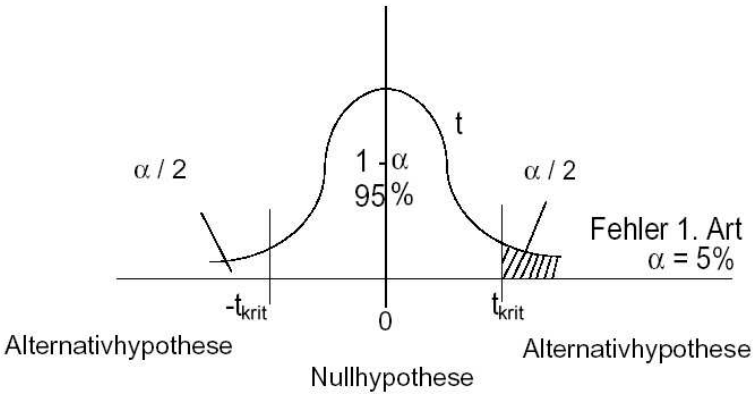


Abb. 3.11: Unterteilung des Wertebereichs in den kritischen und nichtkritischen Bereich. t ist dabei die Prüfgröße; liegt sie außerhalb von $(-t_{krit}, t_{krit})$, wird die Nullhypothese H_0 verworfen.

5.1 Grundlagen: Varianzanalyse, t-Test

Der statistische Test hat die Aufgabe, mit Hilfe von Stichprobenergebnissen eine Entscheidung darüber zu ermöglichen, ob eine Nullhypothese H_0 oder deren Alternativhypothese H_1 gilt. Dazu wird eine geeignete Prüfgröße benötigt. Der Wertebereich dieser Größe wird dabei in den kritischen und nichtkritischen Bereich unterteilt: liegt der aus einer Stichprobe ermittelte Wert der Prüfgröße innerhalb des kritischen Bereichs, wird die Nullhypothese abgelehnt, ansonsten gilt sie (vgl. Abbildung 3.11). Da die Entscheidung auf einer Stichprobe basiert, ist sie mit Fehlern behaftet. Zwei Arten sind möglich

- **Fehler 1.Art:** Die Nullhypothese wird verworfen, obwohl sie gilt.
- **Fehler 2.Art:** Die Nullhypothese wird nicht verworfen, obwohl sie falsch ist.

Übliche Schwellenwerte für die Wahrscheinlichkeit der Fehler 1.Art sind 0.05 oder 0.01, d.h. die Nullhypothese wird verworfen, wenn die Wahrscheinlichkeit des Fehlers 1.Art unter 0.05 bzw. 0.01 liegt.

Varianzanalyse

Varianzanalyse ist ein statistisches Verfahren, um Einflüsse qualitativer Faktoren $A, B, \ldots, K$ auf eine quantitative Zielgröße Y zu untersuchen. Die einzelnen Faktoren $A, B, \ldots, K$ können auch noch in verschiedenen Stufen $A_1, A_2, \ldots, A_a, B_1, B_2, \ldots, B_b, \ldots, K_1, K_2, \ldots, K_k$ auftreten. Die bei einer einzelnen Untersuchung y der Zielgröße Y auftretende Kombination der Stufen der zu untersuchenden Faktoren wird Faktorstufenkombination genannt. Zum Beispiel ist $A_3B_1C_2$ eine Faktorstufenkombination. Die vorliegenden N Ergebnisse $y_\kappa(\kappa = 1, \ldots, N)$ werden als Realisierungen der Zufallsvariablen $Y_\kappa(\kappa = 1, \ldots, N)$ aufgefaßt. Das der Auswertung zugrundeliegende Modell stellt jede Zufallsvariable Y_κ als Summe von Konstanten und/oder Zufallsvariablen dar, die die Einflüsse und die Wechselwirkungen der Faktoren beschreiben, und einer als Restabweichung bezeichneten Zufallsvariablen, die den durch die anderen Faktoren nicht beschreibbaren Einfluß beschreibt. Die Modelle werden folgendermaßen unterschieden:

- Anzahl der Faktoren $A, B, \ldots, K$.
- Verbindung der Faktoren untereinander (gekreuzt, geschachtelt).
- Zufälligkeit der Faktoren.
- Vollständigkeit
- Balance

Anschließend berechnet man für jede Zufallsvariable Y_κ die dazugehörige Varianz, die bei diesem Verfahren die Prüfgröße ist. Die Varianz wird dabei F-verteilt angenommen. Liegt die ermittelte Varianz innerhalb des kritischen Bereiches (die Grenzen des kritischen Bereiches für die vorgegebene Wahrscheinlichkeit des Fehlers der 1.Art lassen sich entweder mit Hilfe der F-Verteilung berechnen oder können anhand von Tabellen bestimmt werden [6]), wird die Nullhypothese abgelehnt.

Null-hypothese H_0	Alternativ-hypothehese H_1	Die Nullhypothese H_0 wird verworfen für: Prüfgröße	Grenzen des kritischen Bereiches für die vorgegebene Wahrscheinlichkeit des Fehlers 1.Art in Abhängigkeit von dem Stichprobenumfang n	
$\delta \leq 0$	einseitig	$\sqrt{n}\bar{d} / s_d$	$> t_{f;1-\alpha}$	mit $f = n - 1$
$\delta \geq 0$	einseitig	$\sqrt{n}\bar{d} / s_d$	$< -t_{f;1-\alpha}$	mit $f = n - 1$
$\delta = 0$	zweiseitig	$\sqrt{n}\lvert\bar{d}\rvert / s_d$	$= t_{f;1-\alpha/2}$	mit $f = n - 1$

Abb. 3.12: Berechnung der Prüfgröße bei dem paarweisen t-Test; δ ist der Erwartungs-, $\bar{d}$ der Mittelwert und s_d^2 die Varianz der Differenzen d_i. Die Grenzen des kritischen Bereiches ergeben sich je nach Wahl der Nullhypothese, sie hängen außerdem vom Umfang der Stichprobe und der gewünschten Wahrscheinlichkeit des Fehlers 1.Art ab.

Paarweiser Erwartungswertvergleich mit dem t-Test

Ein paarweiser Erwartungswertvergleich mit dem t-Test wird durchgeführt, wenn eine Stichprobe von n unabhängigen Wertepaaren $(x_i, x_{2i}), i = 1, 2, \ldots, n$ vorliegt. Jedes Wertepaar i besteht aus zwei Werten x_i, x_{2i}, die aus sachlichen Gründen zusammengehören (z.B. es liegen Ergebnisse zweier Testläufe mit verschiedenen Parametern vor). Die n Differenzen $d_i = x_i - x_{2i}$ sind unabhängige Werte einer Normalverteilung mit einer unbekannten Standardabweichung. Man errechnet Mittelwert $\bar{d}$ und Varianz s_d^2 der n Differenzen d_i:

$$\bar{d} = \frac{1}{n}\sum_{i=1}^{n} |x_i - x_{2i}|, \; s_d^2 = \frac{1}{n-1}\sum_{i=1}^{n}(d_i - \bar{d})^2. \tag{3.11}$$

Abbildung 3.12 zeigt die Berechnung der Prüfgröße.
Der Wert $t_{f;1-\alpha}$ läßt sich in einer Tabelle nachschlagen oder mit Hilfe der t-Verteilung (Student-Verteilung) berechnen [6]. Er ist sowohl von dem Stichprobenumfang als auch von der vorgegebenen Wahrscheinlichkeit des Fehlers der 1.Art abhängig.

5.2 Nachweis der Stabilität gegenüber Änderungen einzelner Parameter

In diesem Experiment sollte gezeigt werden, daß ISeg auch dann brauchbare Ergebnisse liefert, wenn verschiedene, für die Segmentierung wichtige, Parameter variieren. Diese Parameter sind insbesondere die erwartete niedrigste (LI) und höchste (UI) Intensität der zu segmentierenden Objekten, Anzahl der Intensitätsschwellenwerte (NT) sowie der bei dem Konturenvergleich benutzte Schwellenwert T_m (SDT). Als Referenzwerte wurden folgende Werte festgelegt: LI=49 HU, UI=598 HU, NT=170, SDT=0.048 (HU: Hounsfield-Einheit, relativer Wert, Einheit des CT-Wertes, orientiert an den relativen Dichtewerten von Wasser und Luft), die nach Meinung der Entwickler die besten Ergebnisse bei der Segmentierung geliefert haben. Leider erwähnen die Autoren nicht, ob die Referenzergebnisse auch von den unabhängigen Experten bestätigt wurden.
Mit diesen Referenzwerten wurden 33 Ausschnitte zufällig ausgewählter CTA-Aufnahmen segmentiert, die zu 11 CTA-Sätzen gehören. Die Ergebnisse der Segmentierung bilden die Referenz. Anschließend wurden diese Bilder mit veränderten Parametern segmentiert. Es gab 15 Parameterkombinationen, wobei jeder Parameter um 5% nach oben oder nach unten abweichen konnte. Im nächsten Schritt wurden die zu den 15 Parameterkombinationen gehörenden Ergebnisse mit den Referenzergebnissen verglichen und die Abweichungen in Form von NVM (= normalized voxel mismatch) bestimmt. NVM ist dabei definiert als

$$NVM = \frac{Anzahl\ falsch\ klassifizierter\ Voxel}{Anzahl\ der\ Voxel\ der\ zugehörigen\ Region}. \tag{3.12}$$

Die Ergebnisse wurden mit

Parameterabweichung oder deren Kombination	$P(\alpha)$	Parameterabweichung oder deren Kombination	$P(\alpha)$
LI	0.3156	LI∗SDT	0.3636
UI	0.3177	UI∗SDT	0.1722
LI∗UI	0.5552	LI∗UI∗SDT	0.3605
NT	0.2471	NT∗SDT	0.3030
LI∗NT	0.2237	LI∗NT∗SDT	0.2996
UI∗NT	0.2639	UI∗NT∗SDT	0.2668
LI∗UI∗NT	0.3189	LI∗UI∗NT∗SDT	0.3654
SDT	0.2123		

Tabelle 3.1: Ergebnisse der Varianzanalyse. Spalten 1 und 3: die einzelnen Faktorkombinationen. Spalten 2 und 4: Die Wahrscheinlichkeit für den Fehler der 1.Art.

Varianzanalyse ausgewertet. Die Nullhypothese H_0 dabei war: "Weder die Abweichung von 5% der einzelnen Parameter, noch die Kombination der Abweichungen beeinflußt signifikant die Segmentierung". Wie man der dritten und sechsten Spalte der Tabelle 3.1 entnehmen kann, ist die Wahrscheinlichkeit des Fehlers der 1.Art immer deutlich über 5%, so daß die Nullhypothese akzeptiert wird.

5.3 ISeg vs. konventionelles Intensitätsschwellenwertverfahren

Dieses Experiment soll belegen, daß ISeg robuster gegenüber der Wahl des Distanz-Schwellenwertes T_m ist als das konventionelle Intensitätsschwellenwertverfahren gegenüber der Wahl des Intensitätsschwellenwertes. Dabei wurde folgendes Experiment durchgeführt: Es wurden wieder dieselben CTA-Aufnahmen segmentiert wie in dem ersten Experiment. Der Referenzsatz für ISeg wurde ebenfalls übernommen, der Referenzsatz für das Intensitätsschwellenwertverfahren wurde mit dem Intensitätsschwellenwert 200 HU erzeugt. Anschließend wurden die Aufnahmen mit beiden Verfahren segmentiert, die Schwellenwerte wurden dabei in 5%-Schritten variiert, die Reichweite ging dabei von -25% bis +25% (ausgehend von dem Referenzwert). Aus den Stichproben wurden die beiden Zufallsvariablen NVM_{ISWV} und NVM_{ISeg} bestimmt und mit dem t-Test paarweise verglichen, die Nullhypothese lautete: "Die Ergebnisse von ISeg mit dem modifizierten Schwellenwert weichen weniger von den Referenzergebnissen ab, als die Ergebnisse des konventionellen Intensitätsschwellenwertverfahrens". Die Tabelle 3.2 zeigt den Mittelwert der Differenz $NVM_{ISWV} - NVM_{ISeg}$ der beiden Zufallsvariablen. Der Mittelwert der Differenz ist durchgehend positiv, was für ISeg spricht, da es demnach mit modifizierten Schwellenwerten kleinere Abweichungen von den Referenzergebnissen liefert, als das konventionelle Intensitätsschwellenwertverfahren. Als Kritik bleibt festzuhalten, daß, nach Schilderung der Entwickler, keine unabhängigen Experten bei dem Experiment dabei waren, und der Referenzschwellenwert bei dem Intensitätsschwellenwertverfahren von den Entwicklern selber eingestellt wurde. Dabei könnte eine schlechte Wahl desselben das Ergebnis maßgeblich beeinflussen.

5.4 ISeg vs. manuelle Segmentierung durch Experten

Im letzten Experiment sollte gezeigt werden, daß ISeg Ergebnisse liefert, die sich mit den Ergebnissen der manuellen Segmentierung durch Experten vergleichen lassen. Dazu wurden fünf Radiologen mit der Segmentierung eines Bildersatzes, bestehend aus 75 zufällig ausgewählten CTA-Schnitten, die von 16 Patienten stammten, beauftragt. Die Aufnahmen wurden in 6 Sätze unterteilt. Um die Ergebnisse von der benötigten Zeit und Reihenfolge unabhängig zu machen, bekamen die Experten ausreichend Zeit, sie segmentierten höchstens einen Teilsatz am Tag, in zufälliger Reihenfolge. Dabei wurden sie von Segmentierungstools unterstützt.
Es wurden zwei Vergleiche durchgeführt, zum einen wurden die Ergebnisse der Experten mit einem t-Test paarweise untereinander verglichen, dabei wurde der Mittelwert von NVM der Experten bestimmt sowie die Varianz. Im zweiten Vergleich wurden die Ergebnisse der Experten mit den Ergebnissen von ISeg paarweise mit einem t-Test verglichen, auch hier wurde der NVM-Mittelwert und -Varianz berechnet. Leider schweigen sich die Entwickler aus, wie die Vergleiche genau stattgefunden haben, ob nun jede Kombination je zwei Experten verglichen wurde, oder für jedes Objekt die Ergebnisse zweier zufällig ausgewählter Experten betrachtet wurde. Es ist ebenfalls nicht klar, ob die Ergebnisse von ISeg mit den

Abweichung des Intensitätsschwellenwertes in %	Mittelwert der Differenz $NVM_{ISWV} - NVM_{ISeg}$
-25	0.9838334
-20	0.3413646
-15	0.2854101
-10	0.2389239
-5	0.1093748
5	0.1104296
10	0.1542976
15	0.2101106
20	0.2240587
25	0.2201652

Tabelle 3.2: Der Mittelwert der Differenz $NVM_{ISWV} - NVM_{ISeg}$ bei verschiedenen Schwellenwertabweichungen im 2.Experiment.

	Experte-Experte NVM-Mittelwert	Experte-Experte NVM-Varianz	Computer-Experte NVM-Mittelwert	Computer-Experte NVM-Varianz
t-Test 1	0.29	0.19	0.5	6.56
t-Test 2	0.29	0.19	0.32	0.09

Tabelle 3.3: Auswertung der Ergebnisse des 3. Experiments. Man beachte den Rückgang der NVM-Varianz beim zweiten t-Test (hier wurde der Versuchsaufbau modifiziert: siehe Text), die Ergebnisse der zwei Vergleiche sind hier sehr ähnlich.

Ergebnissen jedes Experten verglichen wurden, oder ob auch hier für jedes Objekt ein Experte zufällig ausgewählt wurde. Die Nullhypothese lautete: "Die NVM-Mittelwerte der beiden Verfahren unterscheiden sich nicht nennenswert". Der t-Test wurde zweimal durchgeführt, weil ISeg bei einigen Objekte große Schwierigkeiten hatte, da die Anzahl solcher Objekte bei 0.5% der Gesamtanzahl lag, wurden beim zweiten t-Test diese Objekte nicht berücksichtigt, um die Ergebnisse des Vergleichs nicht zu verzerren. Tabelle 3.3 zeigt die Ergebnisse der Vergleiche. Abbildung 3.13 zeigt anhand eines Objektes, wie nah die Ergebnisse beieinander liegen.

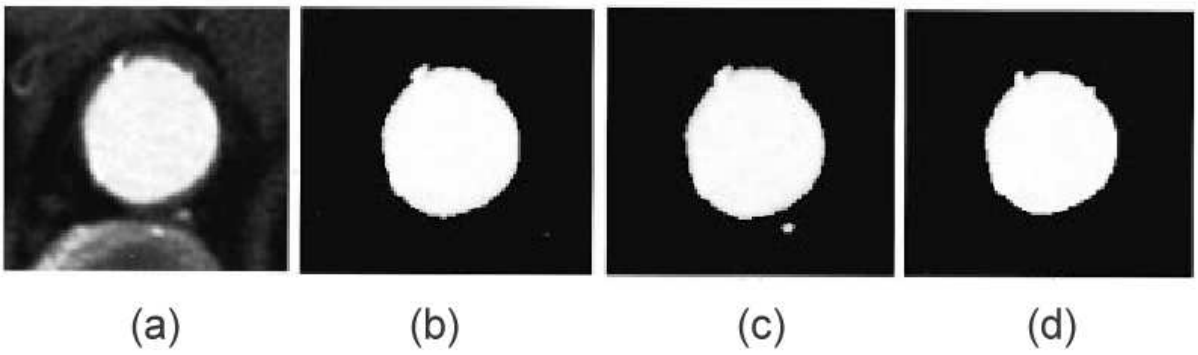


Abb. 3.13: Ergebnisse der Segmentierung. (a) CTA-Aufnahme der Aorta. (b)-(c) Ergebnisse manueller Segmentierung (d) Segmentierungsergebnis von ISeg

6 Zusammenfassung

ISeg ist aus der Notwendigkeit entstanden, visuelle Informationen aus den medizinischen Bilddaten zu extrahieren und so die Erkennung und Beurteilung der Ausmaße der Krankheiten zu vereinfachen und

eine bessere Behandlungsplanung zu ermöglichen. ISeg ahmt bei der Segmentierung die Vorgehesweise der Experten bei der manuellen Segmentierung nach. Die Experten versuchen nämlich einen Intensitätsschwellenwert zu finden, bei dem die Konturen eines Objekts am besten getroffen werden. ISeg setzt bei der Erzeugung der Isolabel-Kontur Karten eine Reihe der Intensitätsschwellenwerte ein, und findet durch den Vergleich der Konturen indirekt auch den besten Schwellenwert. Da dieser Vergleich von innen nach außen stattfindet, hat ISeg keine Probleme mit nah beieinander liegenden Objekten, denn die Kontur, die die Objekte zu einem Objekt vereinigt, weist eine große Änderung in der Form auf, so daß die von ISeg erkannt wird. ISeg reduziert den Einfluß des Bildrauschens, indem es die Glättung der Konturen durch die kubische Spline-Interpolation durchführt. ISeg nutzt die hierarchische Anordnung der Konturen in den Isolabel-Kontur Karten und bringt so einige Effizienzvorteile. Auch wenn ISeg keine manuelle Vorbehandlung der Bilder erfordert, so kann es vorkommen, daß manuelle Nachbearbeitung nötig ist. ISeg ist nicht auf a-priori-Wissen in Form von Modellen angewiesen. Es ist robust gegenüber Abweichungen einzelner Parameter bis zu 5% und es reagiert auch weniger sensibel auf die Änderung des Ähnlichkeitsschwellenwertes als das konventionelle Intensitätsschwellenwertverfahren, wie die ersten beiden Experimente gezeigt haben. Außerdem unterscheiden sich die Segmentierungsergebnisse nicht stärker von den Ergebnissen manueller Segmentierung als die Ergebnisse manueller Segmentierung verschiedener Experten untereinander.
Die Entwickler nennen zwei Voraussetzungen, damit ISeg zuverlässig funktioniert. Erstens muß der Kontrast hoch genug und das Bildrauschen niedrig sein. So konnte ISeg beispielsweise problemlos die Aorta von der Umgebung trennen, der Kontrast lag dabei bei etwa 200 HU, es scheiterte aber bei der Segmentierung der Thrombi in der Vena Cava, der Kontrast lag dabei bei etwa 20 HU. Die genauen Grenzwerte geben die Entwickler jedoch nicht an. Zweitens müssen die Aufnahmen der Objekte eine Fläche von mindestens 50 Pixel haben, denn bei kleineren Objekten ist die Intensitätsverteilung dadurch gestört, daß die Anzahl der Pixel nicht ausreicht, die Übergänge so weich darzustellen, wie sie in Wirklichkeit sind. In diesem Fall können die Änderungen der Konturen größer oder auch kleiner ausfallen als bei größeren Objekten. Auch wenn ISeg diesem Problem zum Teil durch den Einsatz der Konturglättung aus dem Weg geht, ist es bei so kleinen Objekten machtlos.

Literaturverzeichnis

[1] S. Shiffman, G. D. Rubin, S. Napel: Medical Image Segmentation Using Analysis of Isolable-Contour Maps. IEEE Trans. Med. Imag., vol. 19, no. 11, pp. 1064-1074, Nov. 2000.

[2] R. Fagin and L. Stockmeyer: Relaxing the triangle inequality in pattern matching. Int. J. Comput. Vision, vol. 30, no. 3, pp. 219-231, 1998.

[3] E. M. Arkin, L. P. Chew, D. P. Huttenlocher, K. Kedem, and J. S. B. Mitchell: An efficiently computable metric for comparing polygonal shapes. IEEE Trans. Pattern Anal. Machine Intell., vol. 13, pp. 209-211, Mar. 1991.

[4] G. Cortelazzo, G. A. Mian, G. Vezzi, and P. Zamperoni: Trademark shapes description by sting-matching techniques. Pattern Recogn., vol. 27, no. 8, pp. 1005-1018, 1994.

[5] P. Perona and J. Malik: Scale-space and edge detection using anisotropic diffusion. IEEE Trans. Pattern Anal. Machine Intell., vol. 12, pp. 629-639, July 1990.

[6] P.-T. Wilrich, H.-J. Henning, Graf, Ulrich: Formeln und Tabellen der angewandten mathematischen Statistik. Springer-Verlag, Berlin, 3.Auflage 1987.

Konnektivität in vollständigen Verbänden: Ein algebraisches Konzept zur formbasierten Bildverarbeitung

von Lutz Ißler

Betreuer: Christian Thies

Inhaltsverzeichnis

Zusammenfassung

Ein wichtiges Konzept in der Bildverarbeitung ist die Konnektivität. In dieser Seminararbeit wird ein formbasiertes (dh. morphologisches) Konzept zur Betrachtung von Konnektivität in den Grundzügen und einigen Anwendungen vorgestellt. Grundlage dieses Konzepts ist der algebraische Begriff des vollständiges Verbandes, der hier auf die Bildverarbeitung übertragen wird. Ausgehend von der Definition des vollständigen Verbandes werden morphologische Filter, Openings und Closings und Rekonstruktionen betrachtet und diese zur Konstruktion von Zusammenhangsklassen verwendet.

Keywords: Zusammenhang, Verband, Morphologie, Rekonstruktion

1 Einführung

Klassische Ansätze der Bildverarbeitung basieren u.a. auf signaltheoretischen Ansätzen und fassen ein Bild entsprechend als Signal auf. Die Verarbeitung von Bildern erfolgt dabei mit zum Teil komplexen Werkzeugen wie Filtern und Transformationen.
Dem gegenüber steht als vergleichsweise junger Ansatz der Bildverarbeitung die mathematische Morphologie. Hier werden Bilder als algebraische Konstrukte betrachtet und mit algebraischen Mengenoperatoren modifiziert. Die morphologische Bildverarbeitung verändert dabei nur den eigentlichen, sichtbaren Bildinhalt und nicht wie einige signaltheoretische Operatoren die Parameter des Bildes.
Grundlage des formbasierten Ansatzes ist der Begriff der Konnektivität, aus dem sich die morphologischen Operatoren ableiten. In dieser Seminararbeit sollen die grundlegenden Konzepte der morphologischen Bildverarbeitung und zwei der wichtigsten formbasierten Operatoren vorgestellt werden.
Konnektivität oder eben „Zusammenhang“ bedeutet intuitiv, dass sich in einem Bild zusammenhängende Bereiche identifizieren lassen. Diese Bereiche kann man mit geeigneten Werkzeugen erkennen, analysieren und zur weiteren Verarbeitung nutzen. In einem Bild erkannte Objekte sind nach diesem Verständnis aus einem oder mehreren zusammenhängenden Bereichen zusammengesetzt.
Die Definition des Zusammenhangs hängt stark von der gewünschten Anwendung ab: Bekannt sind der übliche graphentheoretische Zusammenhang mit 4er- und 8er-Nachbarschaft – darüberhinaus lassen sich aber noch weit mehr Zusammenhangsbegriffe definieren, z.B. Fuzzy-Zusammenhang und Hyper-Zusammenhang.
Ein allgemeiner Zusammenhangsbegriff führt in Verbindung mit dem Objektbegriff zu den folgenden Eigenschaften. Die Begriffe in Anführungszeichen sind zunächst lediglich informell zu verstehen und werden später exakt definiert:

(C1) Die „leere Menge“ ist zusammenhängend.

(C2) „Punkte“ sind zusammenhängend.

(C3) Die „Vereinigung“ von „überlappenden“ zusammenhängenden Objekten ist zusammenhängend.

Diese Eigenschaften implizieren insbesondere, dass verschiedene Zusammenhangskomponenten eines Objekts nicht „überlappen“ (da sie sonst zusammenfielen) und dass das ursprüngliche Objekt aus deren „Vereinigung“ besteht.
Eine Möglichkeit, Konnektivität zu definieren und dabei die Eigenschaften (C1)–(C3) als Axiome zu verwenden, ist die Theorie der Zusammenhangsklassen. Diese Theorie projiziert die Begriffe „leere Menge“, „Punkt“, „Vereinigung“ und „Überlappung“ auf die algebraische Theorie der Verbände (engl.: *lattice*). Die Familie von Objekten, die die Axiome (C1)–(C3) erfüllen, wird als *Zusammenhangsklasse* bezeichnet. Eine Zusammenhangsklasse kann durch eine Menge von algebraischen Operatoren, den sog. Zusammenhangs-Openings, die zur Extraktion von Zusammenhangskomponenten dienen, eindeutig beschrieben werden. Die in dieser Seminararbeit verwendete axiomatische Formulierung der Zusammenhangsklassen ist von Braga-Neto und Goutsias aus verschiedenen Arbeiten von Serra übernommen worden [BNG02a, Ser96, Ser98, Ser00].
Im Folgenden werden zunächst die notwendigen mathematischen Begriffe und Grundlagen vorgestellt (Abschnitt 2). Daran anschließend wird in Abschnitt 3 das Konzept der Zusammenhangsklassen eingeführt und in Abschnitt 4 die eindeutige Berechnung von Zusammenhangsklassen durch Zusammenhangs-Openings entwickelt. In Abschnitt 5 wird ein weiterer morphologischer Operator, die Rekonstruktion, vorgestellt und gezeigt, dass dieser ebenfalls zur Beschreibung von Zusammenhangsklassen dienen kann.

2 Grundlagen

Die vorgestellte Konnektivitätstheorie basiert auf den aus der Algebra bekannten vollständigen Verbänden. Diese sind wie folgt definiert:

Definition 2.1 *Ein* vollständiger Verband *(engl.* lattice*)* $(\mathcal{L}, \leq)$ *ist eine nichtleere Menge* $\mathcal{L}$ *mit einer partiellen Ordnung* $\leq$ *auf* $\mathcal{L}$ *(dh. einer binären Ordnungsrelation, die reflexiv, anti-symmetrisch und transitiv ist), so dass jede Teilmenge* $\mathcal{K} \subseteq \mathcal{L}$ *ein Infimum* $\bigwedge \mathcal{K}$ *und eine Supremum* $\bigvee \mathcal{K}$ *in* $\mathcal{L}$ *besitzt.*
Das kleinste Element *eines vollständigen Verbandes wird als* O *bezeichnet, das* größte Element *als* I*.*

In dieser Seminararbeit werden ausschließlich vollständige Verbände angenommen. Vollständige Verbände werden hier aus diesem Grund auch einfach als Verbände bezeichnet. Ebenso wird lediglich Bezug auf einen Verband $\mathcal{L}$ genommen, sofern die zugehörige partielle Ordnung klar oder nicht von ausdrücklicher Bedeutung ist.
Eine wichtige Eigenschaft algebraischer Verbände ist, dass Verbandelemente aus anderen Verbandelementen konstruiert werden können:

Definition 2.2 *Eine Teilmenge* $\mathcal{S} \subseteq \mathcal{L}$ *heisst* Supremum-erzeugende Familie *des Verbandes* $\mathcal{L}$*, wenn jedes Element von* $\mathcal{L}$ *als Supremum von Elementen aus* $\mathcal{S}$ *geschrieben werden kann. Ein Element einer Supremum-erzeugenden Familie* $\mathcal{S}$ *heisst* Supremum-erzeugendes Element*.*

Bei Betrachtungen in dieser Seminararbeit wird vorausgesetzt, dass O kein Supremum-erzeugendes Element ist (dh. $O \notin \mathcal{S}$).
Ein Verband kann verschieden Distributivitätseigenschaften aufweisen. Diese sind in der folgenden Definition festgelegt.

Definition 2.3 *Ein Verband* $\mathcal{L}$ *heisst* unendlich $\vee$-distributiv*, wenn gilt:*

$$A \wedge \bigvee B_\alpha = \bigvee (A \wedge B_\alpha) \quad \textit{für alle } A \in \mathcal{L} \textit{ und } \{B_\alpha\} \subseteq \mathcal{L}$$

Ein Verband $\mathcal{L}$ *heisst* unendlich $\wedge$-distributiv*, wenn gilt:*

$$A \vee \bigwedge B_\alpha = \bigwedge (A \vee B_\alpha) \quad \textit{für alle } A \in \mathcal{L} \textit{ und } \{B_\alpha\} \subseteq \mathcal{L}$$

Ein Verband $\mathcal{L}$ *heisst* unendlich distributiv*, wenn er sowohl unendlich* $\vee$*-distributiv als auch unendlich* $\wedge$*-distributiv ist.*

Zur Verdeutlichung der bisherigen Definitionen nun einige Beispiele.

Beispiel 2.4 *Die Menge* $\bar{\mathbb{R}} = \mathbb{R} \cup \{-\infty, \infty\}$ *der erweiterten reelen Zahlen und die Menge* $\bar{\mathbb{Z}} = \mathbb{Z} \cup \{-\infty, \infty\}$ *der erweiterten ganzen Zahlen sind Verbände (sogar Ketten, dh. Verbände mit totaler Ordnung). Die Ordnungsrelation ist dabei die übliche numerische Ordnung, und Infimum und Supremum sind das übliche numerische Infimum und Supremum. Informell gesprochen erweitert man also* $\mathbb{R}$ *(bzw.* $\mathbb{Z}$*) so, dass gilt* $-\infty \leq \mathbb{R} \leq \infty$ *(bzw.* $-\infty \leq \mathbb{Z} \leq \infty$*). Supremum-erzeugende Familien von* $\bar{\mathbb{R}}$ *bzw.* $\bar{\mathbb{Z}}$ *sind* $\mathbb{R}$ *bzw.* $\mathbb{Z}$*.* $\bar{\mathbb{R}}$ *und* $\bar{\mathbb{Z}}$ *sind unendlich distributiv.*

Beispiel 2.5 *Die Menge* $\mathcal{P}(E)$ *aller Teilmengen von* E *ist ein Verband. Die partielle Ordnungsrelation ist dabei die Mengeninklusion, das Infimum ist die Schnittmenge und das Supremum die Vereinigungsmenge. Eine Supremum-erzeugende Familie von* $\mathcal{P}(E)$ *sind die Punkte in* E*.*
Der Verband $\mathcal{P}(E)$ *repräsentiert mit* $E = \mathbb{Z}^2$ *ein zweidimensionales Binärbild. Jedes Element des Verbandes stellt praktisch eine mögliche Färbung von Punkten dar, wobei ein Punkt* (x, y) *in einem Element genau dann als gefärbt gilt, wenn das Tupel* (x, y) *in diesem Element enthalten ist.*

Beispiel 2.6 *Sei* $\mathcal{T}$ *ein Verband. Die Menge Fun*$(E, \mathcal{T})$ *aller Funktionen von einer Menge* E *in den Verband* $\mathcal{T}$ *ist ein Verband. Die partielle Ordnungsrelation ist dabei die partielle Ordnung* $\leq_{\mathcal{T}}$ *auf* $\mathcal{T}$*, übertragen auf die Funktionen:* $f \leq g \Leftrightarrow f(v) \leq_{\mathcal{T}} g(v)\ \forall v \in E$*. Infimum und Supremum sind das „punktweise" Infimum und Supremum, definiert durch* $(\bigwedge f_\alpha)(v) = \bigwedge f_\alpha(v)$ *bzw.* $(\bigvee f_\alpha)(v) = \bigvee f_\alpha(v)$*,* $\forall v \in E$ *(wobei Infimum und Supremum auf der rechten Seite jeweils Infimum und Supremum auf* $\mathcal{T}$ *sind).*

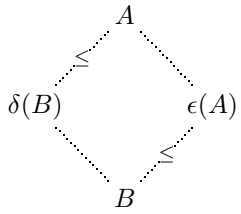


Abb. 4.1: Die für eine Adjunktion (ϵ, δ) geforderten Beziehungen (siehe Definition 2.9).

Sei $\mathcal{S}_\mathcal{T}$ eine Supremum-erzeugende Familie des Verbandes $\mathcal{T}$, und sei $O_\mathcal{T}$ das kleinste Element von $\mathcal{T}$. Die Supremum-erzeugende Familie des Verbandes Fun$(E, \mathcal{T})$ ist gegeben durch $\mathcal{S} = \{\delta_{v,t} | v \in E, t \in S_\mathcal{T}\}$, wobei

$$\delta_{v,t} = \begin{cases} t & \text{für } w = v \\ O_\mathcal{T} & \text{sonst} \end{cases}, \ w \in E$$

die Pulsfunktion *aus Fun$(E, \mathcal{T})$ bezeichnet.*
Der Verband Fun$(E, \mathcal{T})$ ist unendlich $\vee$- oder $\wedge$-distributiv (für beliebige E), sofern der Verband $\mathcal{T}$ unendlich $\vee$- oder $\wedge$-distributiv ist.
Der Verband Fun$(E, \mathcal{T})$ repräsentiert mit $E = \mathbb{Z}^2$ und $\mathcal{T} = \{0, \ldots, n-1\}$ ein Grauwertbild mit n verschiedenen Grauwerten. Dabei stellt jede Funktion $f : \mathbb{Z}^2 \to \{0, \ldots, n-1\}$, $f \in$ Fun$(\mathbb{Z}^2, \{0, \ldots, n-1\})$ eine Zuordnung von Grauwerten zu Punkten und damit ein diskretes Grauwertbild dar.

Nachdem klar ist, wie ein Verband aufgebaut ist, wird im nächsten Schritt formal erfasst, wie man Operationen auf beliebigen Verbänden durchführen kann.

Definition 2.7 *Seien $\mathcal{L}$ und $\mathcal{M}$ Verbände. Ein* Operator *auf den Verbänden $\mathcal{L}$ und $\mathcal{M}$ ist eine Abbildung $\psi : \mathcal{L} \to \mathcal{M}$.*

Operatoren lassen sich anhand ihrer Eigenschaften in Klassen einteilen. Die folgende Definition führt die Begriffe der Dilatation und der Erosion im Sinne der Verbandstheorie ein.

Definition 2.8 *Seien $A_i \in \mathcal{L}$, $i \in I \subseteq \mathbb{N}$ Teilmengen eines Verbandes $\mathcal{L}$.*
Eine Dilatation *ist ein Operator δ, für den gilt*

$$\delta(\bigvee_{i \in I} A_\alpha) = \bigvee_{i \in I} \delta(A_i).$$

Eine Erosion *ist ein Operator ϵ, für den gilt*

$$\epsilon(\bigwedge_{i \in I} A_i) = \bigwedge_{i \in I} \epsilon(A_i).$$

Erosionen und Dilatationen lassen sich jeweils paarweise zusammenfügen. Für derartige Operatorenpaare gelten besondere Eigenschaften. Zunächst die formale Definition:

Definition 2.9 *Eine* Adjunktion (ϵ, δ) *ist ein Operatorenpaar $\epsilon : \mathcal{L} \to \mathcal{M}$, $\delta : \mathcal{M} \to \mathcal{L}$, so dass gilt*

$$\delta(B) \leq A \Leftrightarrow B \leq \epsilon(A) \quad \text{für alle } A \in \mathcal{L}, \ B \in \mathcal{M}.$$

Die in Definition 2.9 geforderten Eigenschaften an die Operatoren sind nicht unmittelbar einsichtig. Abbildung 4.1 veranschaulicht die Beziehungen: Dabei werden die zunächst nur eingezeichneten beiden Beziehungen von der Definition gefordert. Sind zwei Operatoren adjunkt, bilden also zwei beliebige Objekte A, B jeweils gegenseitig Infimum und Supremum von auf die Objekte angewandten adjunkten Operatoren.
Eine Adjunktion besteht immer aus einer Erosion und einer Dilatation. Darüberhinaus ist eine Adjunktion durch die Angabe nur einer Komponente bereits eindeutig bestimmt, wie der folgende Satz zeigt. Für den hier fehlenden Beweis sei auf [Hei94] (Prop. 3.15) verwiesen.

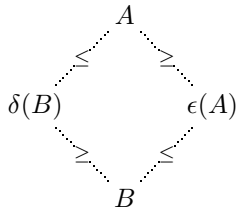


Abb. 4.2: Ist (ϵ, δ) eine Adjunktion auf $\mathcal{L}$, kann Abbildung 4.1 mit Definition 2.9 und Satz 2.10 um zwei Beziehungen ergänzt werden.

Satz 2.10 *Für Operatoren ϵ, δ gilt:*

(i) Sei (ϵ, δ) eine Adjunktion. Dann ist ϵ eine Erosion und δ eine Dilatation.

(ii) Zu jeder Erosion ϵ (Dilatation δ) gibt es genau eine Dilatation δ (Erosion ϵ), so dass (ϵ, δ) eine Adjunktion ist.

Im Allgemeinen sind Operatoren auf verschiedenen Verbänden $\mathcal{L}$ und $\mathcal{M}$ definiert. Für die Bildverarbeitung ist insbesondere der Fall $\mathcal{L} = \mathcal{M}$ interessant – schließlich soll ja bei der Anwendung eines Operators auf ein Bild wieder ein Bild mit den gleichen Abmessungen bzw. Dimensionen herauskommen. Aus diesem Grund wird in dieser Seminararbeit implizit immer von $\mathcal{L} = \mathcal{M}$ ausgegangen. Ein Operator $\psi : \mathcal{L} \to \mathcal{L}$ heisst dann auch vereinfacht ein Operator auf $\mathcal{L}$.
Sind in Satz 2.10 ϵ und δ Operatoren auf $\mathcal{L}$, so lässt sich Abbildung 4.1 zu Abbildung 4.1 ergänzen: Die beiden zusätzlichen Beziehungen ergeben sich offensichtlich aus der Tatsache, dass ϵ eine Erosion und δ eine Dilatation auf $\mathcal{L}$ ist.
Einige Operatoren haben die besondere Eigenschaft, ein Bild (bzw. den Verband) nur bei ihrer ersten Anwendung zu verändern. Jede iterierte Anwendung des Operators bewirkt dann keine Veränderung mehr. Diese Eigenschaft wird als Idempotenz bezeichnet. Desweiteren können Operatoren Wachstumseigenschaften aufweisen sowie die Objekte, auf die sie angewandt werden, „vergrößern“ oder „verkleinern“:

Definition 2.11 *Sei ψ ein Operator. ψ heisst*

- idempotent*, wenn gilt* $\psi\psi(A) = \psi(A),\ A \in \mathcal{L}$
- wachsend *(engl.* increasing*), wenn gilt* $A \leq B \Rightarrow \psi(A) \leq \psi(B)$
- extensiv*, wenn gilt* $\psi(A) \geq A,\ A \in \mathcal{L}$
- anti-extensiv*, wenn gilt* $\psi(A) \leq A,\ A \in \mathcal{L}$.

Wachsende und idempotente Operatoren sind für die Bildverarbeitung besonders bedeutend, weil sie die Eigenschaften des Bildinhalts weitgehend erhalten. Diese Operatoren werden auch besonders bezeichnet:

Definition 2.12 *Ein wachsender und idempotenter Operator wird als (morphologisches)* Filter *bezeichnet.*
Ein anti-extensives Filter heisst Opening*. Ein extensives Filter heisst* Closing*.*

Hat man eine Adjunktion gegeben, so lassen sich daraus ein Opening und ein Closing konstruieren. Dieser Satz soll hier ebenfalls ohne Beweis präsentiert werden. Der Beweis findet sich wiederum in [Hei94] (Th. 3.25, Ex. 6.19).

Satz 2.13 *Sei (ϵ, δ) eine Adjunktion auf $\mathcal{L}$. Dann ist $\theta = \delta\epsilon$ ein Opening auf $\mathcal{L}$ und $\phi = \epsilon\delta$ ein Closing auf $\mathcal{L}$.*

Zur Verdeutlichung des Konzepts von Erosionen, Dilatationen und Adjunktionen im Folgenden ein Beispiel.

Beispiel 2.14 *Sei $E \in \{\mathbb{R}^n, \mathbb{Z}^n\}$. Die Translation A_h einer Menge $A \in \mathcal{P}(E)$ ist die Menge $A_h = \{v + h | v \in A\} \in \mathcal{P}(E)$. Nimmt man ein* Strukturelement *$B \in \mathcal{P}(E)$ hinzu, lassen sich translationsinvariante Operatoren auf $\mathcal{P}(E)$ definieren:*

- $\epsilon_B(A) = A \ominus B = \{h \in E | B_h \subseteq A\}$ *(*translations-invariante Erosion *von* $A \in \mathcal{P}(E)$ *mit dem Strukturelement* B*) und*
- $\delta_B(A) = A \oplus B = \bigcup\{B_h | h \in A\}$ *(*translations-invariante Dilatation *von* $A \in \mathcal{P}(E)$ *mit dem Strukturelement* B*).*

Nach [Hei94] ist (ϵ_B, δ_B) *eine Adjunktion auf* $\mathcal{P}(E)$*. Das entsprechende Opening und Closing nach Satz 2.13 ist gegeben durch*

- $\theta_B = A \circ B = \delta_B \epsilon_B = (A \ominus B) \oplus B$ *(Opening) und*
- $\phi_B = A \bullet B = \epsilon_B \delta_B = (A \oplus B) \ominus B$ *(Closing).*

Eine wichtige Eigenschaft von Filtern ist die des starken Filters. Zunächst der Begriff des starken Filters:

Definition 2.15 *Sei* **id** *die Identitätsfunktion auf* $\mathcal{L}$ *(dh.* $\mathbf{id}(A) = A$*,* $A \in \mathcal{L}$*), und sei* ψ *ein Filter auf* $\mathcal{L}$*. Das Filter* ψ *heisst*

- Infimum-Filter*, wenn gilt* $\psi(\mathbf{id} \wedge \psi) = \psi$
- Supremum-Filter*, wenn gilt* $\psi(\mathbf{id} \vee \psi) = \psi$
- starkes Filter*, wenn* ψ *sowohl ein Infimum- als auch ein Supremum-Filter ist.*

Der folgende Satz zeigt, dass Openings und Closings starke Filter sind [Hei94] (Prop. 12.3):

Satz 2.16 *Openings und Closings auf einem Verband* $\mathcal{L}$ *sind starke Filter.*

Beweis. Sei α ein Opening auf $\mathcal{L}$. Wegen $\alpha(\mathbf{id} \wedge \alpha) = \alpha\alpha = \alpha$ (wg. Idempotenz) und $\alpha(\mathbf{id} \vee \alpha) = \alpha$ ist α sowohl ein Infimum- als auch ein Supremum-Filter und damit ein starkes Filter. Entsprechend ist das Closing β auf $\mathcal{L}$ wegen $\beta(\mathbf{id} \wedge \beta) = \beta$ und $\beta(\mathbf{id} \vee \beta) = \beta\beta = \beta$ (wg. Idempotenz) ein starkes Filter. □

3 Zusammenhangsklassen

Zusammenhangsklassen beschreiben den intuitiven „Zusamenhang“ in einem Bild (oder allgemeiner: einem Verband) formal. Eine Zusammenhangsklasse ist ein theoretisches Konstrukt, in dem die Forderungen (C1)–(C3) aus Abschnitt 1 als Axiome aufgefasst und damit dem gesamten Zusammenhang zugrunde gelegt werden.
In diesem Abschnitt geht es zunächst um die formale Definition des Begriffs der Zusammenhangsklasse. Zwei Möglichkeiten zur Beschreibung von Zusammenhangsklassen werden in den Abschnitten 4 und 5 vorgestellt. Dabei ist zu bemerken, dass diese beiden vorgestellten Beschreibungsmöglichkeiten äquivalent sind. Zumindest theoretisch kann man sich also frei für eine Beschreibungsart entscheiden – wobei die Wahl in der Praxis durch die verschiedenen Ausdrucksmöglichkeiten der Beschreibungen eingeschränkt sein dürfte.
Die folgende Definition überträgt die Forderungen (C1)–(C3) aus Abschnitt 1 in den Formalismus der Zusammenhangsklassen.

Definition 3.1 *Sei* $\mathcal{L}$ *ein Verband mit einer Supremum-erzeugenden Familie* $\mathcal{S}$*. Eine Familie* $\mathcal{C} \subseteq \mathcal{L}$ *heisst eine* Zusammenhangsklasse *in* $\mathcal{L}$*, wenn die folgenden Bedingungen erfüllt sind:*

(i) $O \in \mathcal{C}$

(ii) $\mathcal{S} \subseteq \mathcal{C}$

(iii) Es gilt $\bigvee C_\alpha \in \mathcal{C}$ *für eine Familie* $\{C_\alpha\} \in \mathcal{C}$ *mit* $\bigwedge C_\alpha \neq O$*.*

Die Familie $\mathcal{C}$ *erzeugt einen* Zusammenhang *auf* $\mathcal{L}$*. Die Elemente in* $\mathcal{C}$ *werden als* Zusammenhangselemente *bezeichnet.*

Mit dieser Definition entspricht die „leere Menge“ aus Abschnitt 1 dem kleinsten Element O eines Verbandes, die „Punkte“ sind die Supremum-erzeugenden Elemente, „Vereinigung“ ist das Supremum und „Überlappung“ meint, dass das Infimum von O verschieden ist.
Für die praktische Anwendung einer Zusammenhangsklasse in der Bildverarbeitung ist insbesondere die folgende Eigenschaft von Zusammenhangsklassen wichtig:

Definition 3.2 *Sei $\mathcal{C} \in \mathcal{P}(E), E \in \{\mathbb{R}^n, \mathbb{Z}^n\}$ eine Zusammenhangsklasse. $\mathcal{C}$ ist* translations-invariant*, wenn für alle $h \in E$ gilt*

$$A \in \mathcal{C} \Leftrightarrow A_h \in \mathcal{C}.$$

In der vorstehenden Definition steht A_h für die Translation, wie sie in Beispiel 2.14 verwendet wird.
Aus der Betrachtung einer Zusammenhangsklasse geht intuitiv der Begriff der Zusammenhangskomponente hervor: Augenscheinlich besteht jedes in einem Bild erkannte Objekt aus einem oder mehreren zusammenhängenden Teilen. Dies wird in der folgenden Definition formal gefasst.

Definition 3.3 *Sei $\mathcal{L}$ ein Verband und $\mathcal{C}$ eine Zusammenhangsklasse auf $\mathcal{L}$. Eine* Zusammenhangskomponente *von $A \in \mathcal{L}$ ist ein Element $C \in \mathcal{C}$, so dass*

(1) $C \neq O$

(2) $C \leq A$

(3) es gibt kein $C' \in \mathcal{C}$ mit $C \leq C' \leq A$.

Ist C eine Zusammenhangskomponente von A, so schreibt man $C < A$. Die Menge aller Zusammenhangskomponenten von A wird als $\mathcal{C}(A)$ bezeichnet.

Aus Definition 3.1 kann man zwei wesentliche Eigenschaften von Zusammenhangskomponenten ableiten: Die Zusammenhangskomponenten eines Objekts A dürfen sich nicht „überlappen“, und das Supremum (die „Vereinigung“) aller Zusammenhangkomponenten von A ergibt genau das Objekt A.
Abschließend stellt sich in diesem Abschnitt die Frage, wie man Zusammenhangsklassen angibt. Zunächst kann man natürlich alle Elemente einer Zusammenhangsklasse explizit notieren. Bequemer und eleganter ist es dagegen, mit ein wenig „Handwerkszeug“ die Konstruktion einer Zusammenhangsklasse festzulegen und damit die Klasse eindeutig zu beschreiben. Eine Möglichkeit hierzu sind Zusammenhangs-Openings, die im folgenden Abschnitt 4 vorgestellt werden und die im Wesentlichen genau die Zusammenhangskomponenten eines Objekts ermitteln.

4 Zusammenhangs-Openings

Hat man eine Zusammenhangsklasse auf einem Verband gegeben, so ist es wünschenswert, die Zusammenhangsklassen eines Objekts im Verband angeben zu können. Diese Angabe sollte möglichst allgemein erfolgen – ideal wäre hierfür ein Operator im Sinne von Abschnitt 2.7. Ein solcher Operator, der die Zusammenhangskomponenten aller möglichen Objekte in einem Verband und damit alle möglichen Zusammenhangskomponenten liefert, ist offenbar sogar geeignet, die Zusammenhangsklasse vollständig und eindeutig zu beschreiben, da eine Zusammenhangsklasse nach Definition 3.3 gerade aus der Vereinigung aller Zusammenhangskomponenten besteht. Dass dem tatsächlich so ist, wird in diesem Abschnitt klar.
Zunächst benötigt man also einen Operator, der zu einem gegebenen Objekt und einem „Punkt“ innerhalb dieses Objekts die zugehörige Zusammenhangskomponente liefert. Diese Forderung wird durch die folgende Definition erfüllt:

Definition 4.1 *Sei $\mathcal{C}$ eine Zusammenhangsklasse im Verband $\mathcal{L}$ mit einer Supremum-erzeugenden Familie $\mathcal{S}$. Ein* Zusammenhangs-Opening *ist der folgende Operator γ_x:*

$$\gamma_x(A) = \bigvee\{C \in \mathcal{C} | x \leq C \leq A\}, \quad A \in \mathcal{L}, x \in \mathcal{S}. \tag{4.1}$$

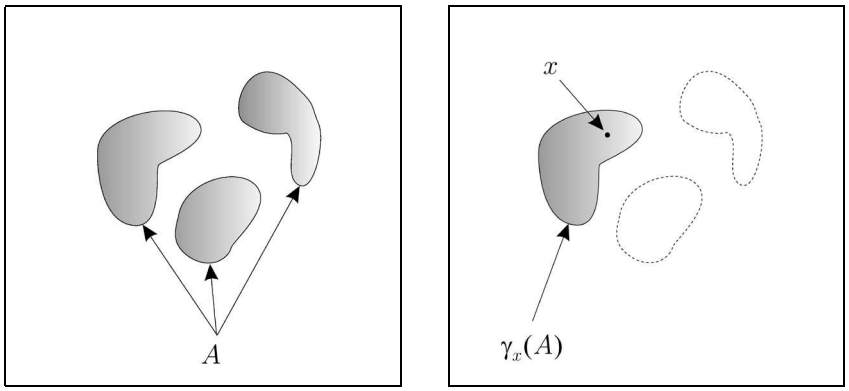


Abb. 4.3: Zusammenhangs-Opening
Links: Ein Bild mit einem Objekt A, bestehend aus Zusammenhangskomponenten (dunkelgrau). *Rechts:* Das Zusammenhangs-Opening $\gamma_x(A)$ liefert genau die Zusammenhangskomponente zurück, in der x liegt. Quelle: [BNG02b]

Aus Eigenschaft (iii) der Definition einer Zusammenhangsklasse (Definition 3.1) folgt, dass $\gamma_x(A) \in \mathcal{C}$ (für alle $A \in \mathcal{L}$ und $x \in \mathcal{S}$). Ausserdem gilt natürlich

$$C \in \mathcal{C} \Rightarrow \gamma_x(C) = C \quad \text{für alle } x \leq C, \tag{4.2}$$

dh. $\gamma_x(A)$ liefert für jeden Punkt x innerhalb eines Zusammenhangselements C der Zusammenhangsklasse $\mathcal{C}$ ebendieses Zusammenhangselement zurück. Dieser Sachverhalt wird in Abbildung 4.3 veranschaulicht. Aus diesen Beobachtungen ergibt sich zunächst:

Folgerung 4.2 *Für ein Zusammenhangs-Opening γ_x, eine Zusammenhangsklasse $\mathcal{C}$ und einen Verband $\mathcal{L}$ gilt:*

$$\mathcal{C} = \bigcup_{x \in \mathcal{S}} = \{\gamma_x(A) | A \in \mathcal{L}\}. \tag{4.3}$$

Beweis. Für jedes $x \in \mathcal{S}$ liefert $\gamma_x(A)$ das Supremum aller $C \in \mathcal{C}$ mit $x \leq C \leq A$. Ermittelt man zunächst die $\gamma_x(A)$ für alle $A \in \mathcal{L}$ mit festem x und wiederholt dieses für alle $x \in \mathcal{S}$, so erhält man offensichtlich alle in der Klasse $\mathcal{C}$ auf $\mathcal{L}$ möglichen Zusammenhangselemente. Die Vereinigung aller Zusammenhangselemente einer Zusammenhangsklasse ergibt natürlich gerade die Zusammenhangsklasse. □
Insbesondere ermittelt ein Zusammenhangs-Opening aber nicht nur die Zusammenhangselemente einer Klasse, sondern auch die Zusammenhangskomponenten eines Objekts. Dass heisst, formal gilt die im Gegensatz zu Gleichung 4.2 stärkere Aussage – und damit gleichzeitig die Tatsache, dass ein Zusammenhangs-Opening die Zusammenhangskomponenten eines Objekt eindeutig charakterisiert:

Folgerung 4.3 *Sei γ_x ein Zusammenhangs-Opening und sei $A \in \mathcal{L}$. Dann gilt:*

(i) $x \leq \gamma_x(A) \lessdot A$ *für alle* $x \leq A$

(ii) $C \lessdot A, x \leq \mathcal{C} \Rightarrow C = \gamma_x(A)$ *für alle Supremum-erzeugenden Elemente* x.

Beweis. Aussage (i) folgt direkt aus Definition 4.1, denn das Supremum über die Zusammenhangselemente C in einem Objekt A, die den „Marker“ x enthalten, ist gerade eine Zusammenhangskomponente nach Definition 3.3.
Ist wie in Aussage (ii) eine Zusammenhangskomponente $C <\cdot A$ gegeben, so liefert $\gamma_x(A)$ für ein Supremum-erzeugendes Element x, das innerhalb der Komponente C liegt ($x \leq C$), nach Aussage (i) genau die Komponente C. □
Nachdem Zusammenhangs-Openings Zusammenhangskomponenten eindeutig beschreiben können, ist die Frage zu stellen, ob damit auch die Beschreibung von Zusammenhangsklassen durch Zusammenhangs-Openings möglich ist. Der nächste Satz und eine Folgerung zeigen, dass dies zutrifft. Zunächst verleiht eine Zusammenhangsklasse den aus ihr abgeleiteten Zusammenhangs-Openings bestimmte Eigenschaften:

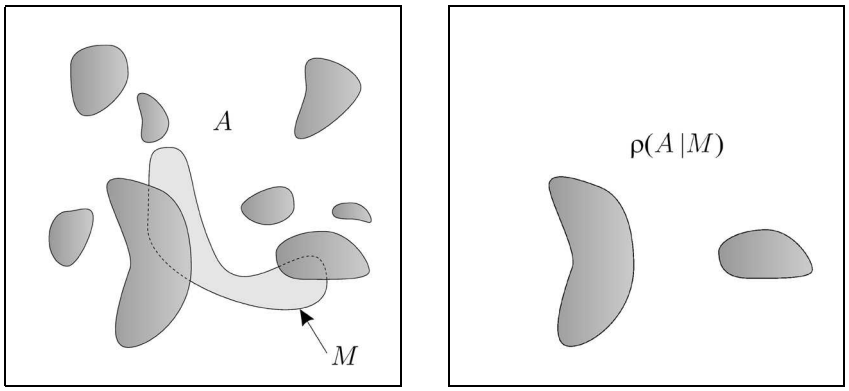


Abb. 4.4: Rekonstruktion
Links: Ein Bild mit einem Objekt A, bestehend aus Zusammenhangskomponenten (dunkelgrau). Ein Marker M ist hellgrau dargestellt. *Rechts:* Rekonstruktion $\rho(A|M)$ des Bildes mit dem Marker M. Das Ergebnis sind alle Komponenten von A, die von M überdeckt werden. Quelle: [BNG02a]

Satz 4.4 *Sei $\mathcal{L}$ ein Verband mit einer Supremum-erzeugenden Familie $\mathcal{S}$ und sei $Ccl(\mathcal{L})$ die Menge aller Zusammenhangsklassen von $\mathcal{L}$. Seien $\{\gamma_x|x \in \mathcal{S}\}$ die durch Definition 4.1 festgelegten, zu $\mathcal{C}$ gehörenden Zusammenhangs-Openings. Dann gilt:*

(i) $\gamma_x(x) = x$ *für alle* $x \in \mathcal{S}$

(ii) $x \not\leq A \Rightarrow \gamma_x(A) = O$

(iii) $\gamma_x(A) \wedge \gamma_y(A) \neq O \Rightarrow \gamma_x(A) = \gamma_x(A)$ *(dh.* $\gamma_x(A)$ *und* $\gamma_y(A)$ *sind entweder gleich oder disjunkt).*

Darüberhinaus beschreiben Zusammenhangs-Openings eindeutig die zugehörige Zusammenhangsklasse:

Folgerung 4.5 *Sei $Cop(\mathcal{L})$ die Menge aller Familien von Zusammenhangs-Openings $\{\gamma_x|x \in \mathcal{S}\}$, die die Eigenschaften (i)-(iii) aus Satz 4.4 erfüllen. Für eine Menge von Zusammenhangs-Openings $\{\gamma_x|x \in \mathcal{S}\} \in Cop(\mathcal{L})$ sei $\mathcal{C}$ durch Gleichung 4.3 gegeben. $\mathcal{C}$ ist dann natürlich eine Zusammenhangsklasse, dh. $\mathcal{C} \in Ccl(\mathcal{L})$.*

Es ist also möglich, eine Zusammenhangsklasse durch Zusammenhangs-Openings mit bestimmten – nicht stark einschränkenden – Eigenschaften (nämlich (i)–(iii) aus Satz 4.4) zu beschreiben und umgekehrt. Darüberhinaus ist diese Verbindung sogar eine Bijektion, so dass die Angabe einer Zusammenhangsklasse äquivalent entweder explizit oder mittels Zusammenhangs-Openings erfolgen kann und dass diese beiden Möglichkeiten beliebig austauschbar sind.
Im folgenden Abschnitt 5 wird ein weiterer Operator vorgestellt, nämlich die Rekonstruktion. Dieser Operator führt zu einer dritten Möglichkeit, Zusammenhangsklassen zu beschreiben.

5 Rekonstruktion

Eine wichtige Klasse von Operatoren in der morphologischen Bildverarbeitung sind Rekonstruktionsoperatoren. Ein solcher Operator ermittelt für ein gegebenes Objekt, das als *Marker* bezeichnet wird, die zugehörige Zusammenhangskomponente. Rekonstruktionsoperatoren stellen eine Erweiterung der Zusammenhangs-Openings dar: diese weisen das gewünschte Verhalten für einzelne Supremum-erzeugende Elemente bereits auf und müssen nur noch entsprechend auf Mengen erweitert werden.
Entsprechend dieser Überlegung formuliert die folgende Definition die wesentliche Eigenschaft eines Rekonstruktionsoperators mithilfe von Zusammenhangs-Openings:

Definition 5.1 *Sei $\mathcal{C}$ eine Zusammenhangsklasse auf dem Verband $\mathcal{L}$. Seien $\{\gamma_x|x \in \mathcal{S}\}$ die zu $\mathcal{L}$ gehörenden Zusammenhangs-Openings. Die* Rekonstruktion *$\rho(A|M)$ eines Elements $A \in \mathcal{L}$ mittels des* Markers *$M \in \mathcal{L}$ wird definiert durch*

$$\rho(A|M) = \bigvee_{x \leq M} \gamma_x(A). \tag{4.4}$$

Insbesondere gilt $\rho(A|O) = \bigvee \emptyset = O$ und $\rho(A|M) = \rho(A|M \wedge A)$, so dass aus $M \wedge A = O$ unmittelbar $\rho(A|M) = O$ folgt. Entsprechend nimmt man zumeist $M \leq A$ an.
Zusammenhangs-Openings auf einem Verband liefern nach Definition 5.1 einen Rekonstruktionsoperator. Umgekehrt können aus einer Rekonstruktion Zusammenhangs-Openings konstruiert werden, nämlich mittels

$$\gamma_x(A) = \begin{cases} \rho(A|X) & \text{für } x \leq A \\ O & \textit{sonst.} \end{cases} \tag{4.5}$$

Der Rekonstruktionsoperator $\rho(A|M)$ entspricht also nach Gleichung 4.4 einem einfachen Zusammenhangs-Opening, wenn der Marker M ein Supremum-erzeugendes Element x ist und $x \leq A$ gilt. Im Falle eines atomaren Verbandes (z.B. $\mathcal{P}(E)$) ist dies der Fall, so dass hier Rekonstruktion und Zusammenhangs-Opening für alle Supremum-erzeugenden Elemente gleichgesetzt werden können.
Der Rekonstruktionsoperator $\rho(A|M)$ bestimmt genau die Zusammenhangskomponenten von A, die der Marker M überdeckt:

Satz 5.2 *Für den Rekonstruktionsoperator $\rho(A|M)$ gilt*

$$\rho(A|M) = \bigvee \{C \lessdot A | C \wedge M \neq O\}. \tag{4.6}$$

Beweis. Sei $\lessdot A$ und $C \wedge M \neq O$. Dann kann man ein Supremum-erzeugendes Element x finden mit $x \leq C \wedge M \leq C$, so dass $x \leq M$ und $C = \gamma_x(A)$ gilt. Damit ist $\rho(A|M) = \bigvee_{x \leq M} \gamma_x(A) \geq \bigvee \{C < \cdot\, A | C \wedge M \neq O\}$ gezeigt.
Um die entsprechende Ungleichung der anderen Richtung zu zeigen, sei zunächst bemerkt, dass $\rho(A|M) = \rho(A|M \wedge A) = \bigvee_{x \leq M \wedge A} \gamma_x(A)$. Damit gilt $x \leq M \wedge A \Rightarrow x \leq A \Rightarrow x \leq \gamma_x(A) \lessdot A$ für $\gamma_x(A) \wedge M \geq x \neq O$. Daraus folgt, dass $\rho(A|M) \leq \bigvee \{C \lessdot A | C \wedge M \neq O\}$ gilt. □
Abbildung 4.4 veranschaulicht diese Auffassung des Rekonstruktionsoperators. In der Abbildung dargestellt ist ein Objekt A mit einem Marker M und die entsprechende Rekonstruktion $\rho(A|M)$.
Ebenso wie Zusammenhangs-Openings sind auch Rekonstruktionsoperatoren ein geeignetes Mittel, um Zusammenhangsklassen zu beschreiben. Dies erscheint im Hinblick auf die Definition des Rekonstruktionsoperators mithilfe von Zusammenhangs-Openings und mit Satz 4.4 nicht verwunderlich, gilt allerdings nur für unendlich $\vee$-distributive Verbände. Der folgende Satz formuliert genau die Eigenschaft von Rekonstruktionen, Zusammenhangsklassen vollständig zu beschreiben. Für den langwierigen Beweis des Satzes sei auf [BNG02a] verwiesen.

Satz 5.3 *Sei $\mathcal{L}$ ein unendlich $\vee$-distributiver Verband mit einer Supremum-erzeugenden Familie $\mathcal{S}$. Sei $\rho(A|M)$ der Rekonstruktionsoperator zu $\{\gamma_x | x \in \mathcal{S}\}$, $\{\gamma_x | x \in \mathcal{S}\} \in Cop(\mathcal{L})$. Dann gilt:*

(i) $M \wedge A \leq \rho(A|M)$

(ii) $\rho(\cdot|M)$ *ist ein Opening*

(iii) $\rho(A|\cdot)$ *ist wachsend und idempotent*

(iv) $\rho(A|\cdot)$ *ist symmetrisch (dh. es gilt* $y \leq \rho(A|x) \Leftrightarrow x \leq \rho(A|y)$ *für alle* $x, y \leq A$*)*

(v) $\rho(A|M) = \bigvee_{x \leq M} \rho(A|x)$.

Analog dazu sei $\mathrm{Rec}(\mathcal{L})$ die Menge aller Operatoren $\rho : \mathcal{L} \times \mathcal{L} \to \mathcal{L}$, die die Eigenschaften (i)-(v) erfüllen, und sei für die Rekonstruktion $\rho \in \mathrm{Rec}(\mathcal{L})$ die Menge $\{\gamma_x | x \in \mathcal{S}\}$ die Familie der durch Gleichung 4.5 bestimmten Openings. Dann ist $\{\gamma_x | x \in \mathcal{S}\}$ eine Familie von Zusammenhangs-Openings auf $\mathcal{L}$, dh. $\{\gamma_x | x \in \mathcal{S}\} \in \mathrm{Cop}(\mathcal{L})$. Der Rekonstruktionsoperator dieser Familie fällt darüberhinaus sogar mit ρ zusammen, so dass die Gleichungen 4.4 und 4.5 eine Bijektion zwischen $\mathrm{Cop}(\mathcal{L})$ und $\mathrm{Rec}(\mathcal{L})$ definieren.

Algorithmus 1 Propagierungsalgorithmus zur die Rekonstruktion aus Beispiel 5.4

$R \leftarrow M \cap A$
repeat
  for all Punkte in $A \setminus R$, die zu R adjazent sind **do**
    $R \leftarrow R \cup N$
  end for
until $N = \emptyset$
$\rho(A|M) \leftarrow R$

Die unendliche $\vee$-Distributivität des Verbandes $\mathcal{L}$ wird in Satz 5.3 lediglich für den Beweis der Idempotenz von $\rho(A|\cdot)$ benötigt. Insbesondere ist diese Eigenschaft damit nur notwendig, um zu zeigen, dass Gleichung 4.4 eine Abbildung von $\mathrm{Cop}(\mathcal{L})$ nach $\mathrm{Rec}(\mathcal{L})$ darstellt – der umgekehrte Weg mittels Gleichung 4.5 kommt ohne die unendliche $\vee$-Distributivität aus.
Da je eine Bijektion zwischen $\mathrm{Ccl}(\mathcal{L})$ und $\mathrm{Cop}(\mathcal{L})$ und zwischen $\mathrm{Cop}(\mathcal{L})$ und $\mathrm{Rec}(\mathcal{L})$ existiert, kann Zusammenhang auf einem Verband $\mathcal{L}$ kann auf drei – äquivalente – Arten beschrieben werden, nämlich mittels

- einer Zusammenhangsklasse $\mathcal{C} \in \mathrm{Ccl}(\mathcal{L})$,
- einer Familie von Zusammenhangs-Openings $\{\gamma_x | x \in \mathcal{S}\} \in \mathrm{Cop}(\mathcal{L})$
- oder einem Rekonstruktionsoperator $\rho \in \mathrm{Rec}(\mathcal{L})$.

Entsprechend können mittels der Bijektionen auch jeweils die beiden anderen Beschreibungswege aus einem gegebenen abgeleitet werden.
Im Folgenden nun drei Beispiele für Rekonstruktionsoperatoren:

Beispiel 5.4 *Sei $\mathcal{L} = \mathcal{P}(\mathbb{Z}^n)$. Die Supremum-erzeugenden Elemente sind genau die Punkte in $\mathbb{Z}^n$. Sei ausserdem eine Adjazenzrelation auf $\mathbb{Z}^n$ definiert, z.B. die durch die Kantenrelation des ungerichteten Graphen $G = (\mathbb{Z}^n, L)$ festlegte Adjazenzrelation. Sei der Operator $\rho : \mathcal{P}(\mathbb{Z}^n) \times \mathcal{P}(\mathbb{Z}^n) \to \mathcal{P}(\mathbb{Z}^n)$ definiert durch einen Propagierungsalgorithmus (Algorithmus 1:*
Wie man leicht sieht, gilt $\rho \in Rec(\mathcal{L})$. Der durch ρ definierte Zusammenhang auf $\mathbb{Z}^n$ entspricht dem graphentheoretischen Zusammenhang auf $G = (\mathbb{Z}^n, L)$, wobei L die angenommene Adjazenzrelation darstellt. In der Bildverarbeitung verwendet man für ein diskretes Grauwertbild üblicherweise $\mathbb{Z}^2$ mit der 4er- oder 8er-Nachbarschaft als Adjazenzrelation.

Beispiel 5.5 *Sei $\mathcal{L}$ ein unendlich $\vee$-distributiver Verband, und sei δ eine extensive Dilatation auf $\mathcal{L}$. Für ein gegebenes $A \in \mathcal{L}$ ist der Operator $\delta_A(M) = \delta(M) \wedge A$ offensichtlich eine Dilatation auf $\mathcal{L}$, die als* geodätische Dilatation *des Markers M innerhalb der Maske A bezeichnet wird. Definiere nun den Operator $\rho : \mathcal{L} \times \mathcal{L} \to \mathcal{L}$ durch*

$$\rho(A|M) = \bigvee_{i=1}^{\infty} \delta_A^i(M),$$

wobei ψ^i die i-fache Ausführung des Operators ψ bezeichnet. Der Operator $\rho(A|M)$ heisst geodätische Rekonstruktion *des Markers M innerhalb der Maske A. Offensichtlich ist $\rho(A|\cdot)$ eine Dilatation auf $\mathcal{L}$, da ρ als Supremum von Dilatationen definiert ist. Eigenschaft (v) einer Rekonstruktion nach Satz 5.3 ist also erfüllt. Mittels einiger einfacher Zusatzbedingungen aus [RS01] können auch die Eigenschaften (i)–(iv) des Satzen nachgewiesen werden, womit dann gilt $\rho \in Rec(\mathcal{L})$.*

Beispiel 5.6 *Beispiel 5.4 beschreibt einen einfachen Algorithmus zur Definition einer Rekonstruktion. Dieser hat eine Laufzeit von $O(n^3)$, die durch die Implementierung eines anderen Verfahrens zur Rekonstruktion unter Verwendung von Warteschlangen auf $O(n^2)$ als Worst-Case-Laufzeit reduziert werden kann [Vin93]. Für eine Implementierung des Algorithmus sei auf [Thi01] verwiesen.*

6 Ausblick

Diese Seminararbeit ist natürlich nicht in der Lage, mehr als einen ersten Überblick über ein so komplexes Thema wie die morphologische Bildverarbeitung zu geben. Allerdings ist zu bemerken, dass selbst mit den hier vorgestellten absoluten Grundlagen schon umfassende Möglichkeiten der Bildverarbeitung und die wesentlichen Operatoren der Morphologie zur Verfügung stehen.
Die Theorie der morphologischen Bildverarbeitung zeichnet sich gerade durch ihren im Vergleich zur Signaltheorie geringen Abstraktionsgrad aus. Morphologische Operatoren arbeiten auf dem sichtbaren Bild und den offensichtlichen und intuitiven Eigenschaften des Bildinhalts wie Formen und Zusammenhangskomponenten, wogegen klassische lineare Operatoren auch umfangreiche Modifikationen von für das Ziel der Operation wenig oder gar unbedeutenden Teilen des Bildsignals vornehmen. Beispielsweise manipuliert die Fourier-Transformation im Wesentlichen die Amplitude eines Signals, während die für die Bildinhaltsanalyse eigentlich interessante Ortsinformation in der Signalphase codiert ist. Ein geeigneter morphologischer Operator beschränkt sich auf die die Manipulation der sichtbaren Ortsinformation und weist darüberhinaus noch Eigenschaften wie die des starken Filters oder der Idempotenz auf, die die vorgenommene Modifikation des Bildes klar definiert.
Morphologische Operatoren bieten somit zwei wesentliche Vorteile gegenüber linearen signaltheoretischen Operatoren. Sie sind

- nachvollziehbar in ihren Auswirkungen und
- einfach parametrierbar.

Ein morphologischer Operator kann natürlich auch in seinem Aufbau und seinen Auswirkungen komplex sein, wie es sogar häufig der Fall ist. Jedoch ist selbst der komplexeste Operator aus atomaren, algebraisch definierten Operatoren wie Erosionen und Dilatationen zusammengesetzt und deshalb ein formal nachvollziehbares algebraisches Konstrukt, das genau auf die Bedürfnisse einer praktischen Anwendung zugeschnitten werden kann.
Letztlich schlägt sich dieser Vorteil der Morphologie natürlich auch in den Algorithmen nieder, die zur morphologischen Bildverarbeitung und -analyse eingesetzt werden. Diese sind häufig sehr einfach und effizient zu implementieren und – wichtig für die Wiederverwendbarkeit einer Implementierung – leicht zu modularisieren. Der Aufbau von komplexen Operatoren aus einfachen Einzelbausteinen entspricht den Ansätzen der modernen Softwaretechnik.
Einige Anwendungen der morphologischen Bildverarbeitung wurden bereits im Text genannt. Insbesondere ist die Morphologie natürlich – entsprechend ihrem Namen – formbasiert, was sie für die Auswertung von Struktur- und Forminformationen im Bild besonders geeignet macht. Als Beispiel für die Anwendung morphologischer Operatoren seien hier angeführt:

- Kantenerkennung,
- Kontrastverstärkung,
- Störungsbeseitigung,
- Skelettierung,
- Segmentierung und
- Distanztransformation.

Die Medizin macht sich die Möglichkeiten der Bild-Morphologie besonders im Bereich der automatischen Bildanalyse und Bildverbesserung zunutze. Bekannte Anwendungen in der medizinischen Bildanalyse sind die Erkennung pathologischer Zellveränderungen in Gewebeproben und die Ermittlung von Qualitäts- und Quantitätszahlen bei Blutuntersuchungen. Dabei steht insbesondere im Vordergrund, dass geeignete morphologische Operatoren das in der Medizin übliche heterogene Bildmaterial verarbeiten können und so in der Lage sind, beispielsweise Zellen eines bestimmten Typs in einem Bild weitestgehend unabhängig von Auflösung und Qualität des Bildes zu zählen.
Darüberhinaus spielt die Morphologie auch eine große Rolle in modernen Segmentierungsalgorithmen bei der Auswertung von Tomografiebildern und als unterstützende Disziplin in der aufbereiteten Visualisierung von medizinischen Bilddaten, z.B. virtuellen Flügen durch den menschlichen Körper.

Literaturverzeichnis

[BNG02a] Ulisses Braga-Neto and John Goutsias. Connectivity on complete lattices: New results. *Computer Vision and Image Understanding*, 2002.

[BNG02b] Ulisses Braga-Neto and John Goutsias. A theoretical tour of connectivity in image processing and analysis. *Journal of Mathematical Imaging and Vision*, 2002.

[Hei94] H. J. Heijmans. *Morphological Image Operators*. Academic Press, Boston, Massachusetts, 1994.

[RS01] C. Ronse and J. Serra. Geodesy and connectivity in lattices. *Fundamenta Informaticae*, 46:349–395, 2001.

[Ser96] J. Serra. *Mathematical Morphology and its Applications to Image and Signal Processing*, chapter Connectivity on complete lattices, pages 81–96. P. Maragos and R. W. Schafer and M. A. Butts, Eds. Kluwer, Boston, Massachusetts, 1996.

[Ser98] J. Serra. Connectivity on complete lattices. *Journal of Mathematical Imaging and Vision*, 9:231–251, 1998.

[Ser00] J. Serra. Connections for sets and functions. *Fundamenta Informaticae*, 41:147–186, 2000.

[Thi01] Christian Thies. Ein morphologisches Multiskalenverfahren zur regionenbasierten Segmentierung medizinischer Bilder. Master's thesis, RWTH Aachen, 2001.

[Vin93] L. Vincent. Morphological grayscale reconstruction in image analysis: Applications and efficient algorithms. *IEEE Transactions on Image Processing*, 2:176–201, 1993.

Erscheinungsbasierte 3-D Objekterkennung

von Andrea Wenning

Betreuer: Daniel Keysers

Inhaltsverzeichnis

Zusammenfassung

In dieser Ausarbeitung wird ein Verfahren zum automatischen Lernen von Objektmodellen für die Erkennung und die Positionsbestimmung vorgestellt. Hierbei wird ein Objekt durch eine Menge von 2-D Bildern beschrieben. Da für ein Objekt in einem 2-D Bild die Ausrichtung und die Beleuchtungsrichtung variabel sind, kann das Objekt bezüglich dieser beiden Parameter kompakt dargestellt werden. Als erstes wird der Eigenraum bestimmt. In diesem wird das Objekt durch eine parametrische Mannigfaltigkeit beschrieben. Für die Erkennung eines Objektes in einem Bild, wird das Bild in den Eigenraum projiziert. Dann wird, um das Objekt zu bestimmen, die Mannigfaltigkeit gesucht, die am nächsten an dem projizierten Bild liegt. Die Ausrichtung des Objektes wird durch die genaue Position auf der Mannigfaltigkeit wiedergegeben.
Das in dieser Ausarbeitung vorgestellte Verfahren zeigt gute Ergebnisse, jedoch ist zum einen die geeignete Wahl der Grösse des Eigenraums nicht bekannt und zum anderen ist der Einsatz in einer realen Anwendung schwierig.

Keywords: Eigenraum, Mannigfaltigkeit, Erscheinung, Objekterkennung, Modellierung

1 Einleitung

Im medizinischen Image Retrieval werden große Mengen von medizinischen Bildern, z.B. Röntgen-, CT- oder MR-Bilder, in Datenbanken verwaltet. Dazu werden Methoden bereit gestellt, um Bilder zu finden, Merkmale zu extrahieren und zu vergleichen. Ein Objekt, z.B. der menschliche Schädel, wird nicht kompakt dargestellt, sondern unabhängig durch einzelne Bilder. Abbildung 5.1 stellt verschiedene Röntgenbilder von Schädeln dar. Betrachtet man den Schädel von unterschiedlichen Seiten, so stellt man fest, daß der Schädel von hinten anders aussieht als von vorne oder von der Seite. Dieses gibt die Frage auf, wie diese Informationen über das Objekt "Schädel" geeignet darstellt werden können, um zum einen das Objekt selber zu erkennen und zum anderen die Ausrichtung des Objektes zu bestimmen, beim Schädel z.B. frontal oder seitlich.
Ein 3-D Objekt kann durch eine Menge von 2-D Bildern dargestellt werden. Diese Bilder werden in den Eigenraum projiziert. In dem das Objekt dann kompakt durch eine parametrische Mannigfaltigkeit repräsentiert wird, die von der Ausrichtung des Objektes im Bild und der Beleuchtungsrichtung abhängt. Für die Erkennung eines Bildes, wird das Bild in den Eigenraum projiziert. Dann wird, um das Objekt zu bestimmen, die Mannigfaltigkeit gesucht, die am nächsten an dem projizierten Bild liegt. Die Ausrichtung des Objektes wird durch die genaue Position auf der Mannigfaltigkeit wiedergegeben.

2 Ansatz zur Darstellung von 3-D Objekten

Ein Erkennungssystem für 3-D Objekte benötigt die Modelle der Objekte in seinem Speicher. In der Vergangenheit wurden z.B. geometrische Modelle erstellt, die die Gestalt der Objekte beschreiben. Hierbei wurden die Objekte mit Computer Aided Design (CAD) Modellen dargestellt. Das Problem dabei ist, daß man nicht alle Objekte, z.B. das Skelett eines Menschen, mit CAD-Modellen modellieren kann. Weiterhin muß für jedes Objekt ein Modell von Hand erstellt werden. Dieses ist aufwendig und unpraktisch, wenn man eine große Anzahl von Objekten darstellen möchte. Da jedes Modell erst von einem Menschen erstellt werden muß, ist ein System, welches auf diesem Ansatz basiert niemals in der Lage selbständig Modelle zu erstellen.
Betrachtet man einen Menschen, wie er sich ein 3 dimensionales Objekt visuell merkt, so beobachtet

Abbildung 5.1: Röntgenbilder von Schädeln

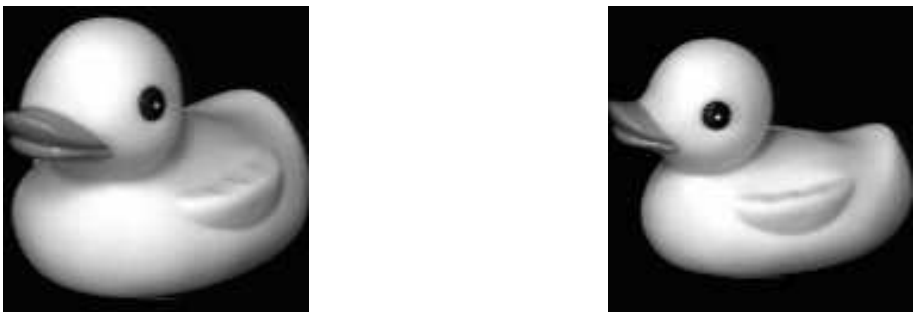


Abbildung 5.2: Ein Objekt aus der Datenbank

man, daß der Mensch das Objekt rotiert und die Erscheinung des Objektes von unterschiedlichen Richtungen betrachtet. Weiterhin hat man festgestellt, daß der Mensch wahrscheinlich 3-D Objekte als eine Menge von 2-D Bildern repräsentiert. Versucht man diese Beobachtung auf ein automatisches visuelles Lernsystem umzusetzen, so muß man sich als erstes überlegen, was die Erscheinung eines Objektes ist. Erscheinung kann man als Kombination von Gestalt, Reflektion, Ausrichtung in der Szene und Beleuchtungsbedingungen auffassen. Gestalt und Reflektion sind feste Parameter, die sich für ein Objekt nicht ändern, während sich die Ausrichtung und die Beleuchtung verändern können. Dieses wird in Abbildung 5.2 am Beispiel einer Ente gezeigt. Ein Objekt wird daher in dem hier vorgestellten Ansatz aus einer Menge von Bildern dargestellt, die unter verschiedenen Positionen und Beleuchtungen aufgenommen wurden [1].

3 Erscheinungsbasiertes Lernen

Jedes Objekt wird in diesem Ansatz durch einen Menge von Bildern dargestellt. Jedes dieser Bilder wird normalisiert und durch einen Merkmalsvektor, der die Pixelwerte enthält, dargestellt. Um die Dimension zu reduzieren wird die Hauptachsentransformation verwendet. Es werden dabei zwei Eigenräume bestimmt. Der universelle Eigenraum dient zur Klassifikation und der Objekteigenraum zur Positionsbestimmung. Die Objekte werden durch parametrische Mannifaltigkeiten repräsentiert, die mittels Splines bestimmt werden. Abbildungen 5.3 zeigt ein Objekt mit zugehöriger Mannigfaltigkeit. Die Mannigfaltigkeit wird zur Veranschaulichung im 3-D Raum dargestellt. Dieser wird durch die ersten drei Eigenvektoren des Eigenraums aufgespannt. Jedes in diesen Eigenraum projizierte Bild entspricht einem Punkt, welcher in diesem Fall nur die Ausrichtung als Parameter hat. Die Beleuchtungsrichtung wird hier als konstant angesehen. Interpoliert man diese Punkte so erhält man die dargestellte Kurve. Die Kurve ist geschlossen, da, wenn man das Objekt um 360 Grad gedreht hat, man wieder am Ausgangspunkt ankommt.

Abbildung 5.3: Ein Objekt mit zugehöriger Mannigfaltigkeit

3.1 Normalisierung

Jedes Objekt wird durch eine Menge von Bildern dargestellt. Um die Bilder unabhängig von der Größe und von der Helligkeit zu repräsentieren, werden die Bilder normiert.
Die Bilder werden als erstes in der Größe normiert. Hierbei werden sie in Hintergrund und Objektregion segmentiert. Dieses ist möglich, da vorrausgesetzt wird, daß der Hintergrund schwarz ist. Jedes normalisierte Bild wird als Merkmalsvektor aus Pixelwerten geschrieben. Damit die Variationen in der Helligkeit eines Bildes die Erkennung nicht beeinflußt, werden die Bilder in der Helligkeit normalisiert. Die Energie

in einem Bild wird dazu auf den Wert 1 normiert.
Jeder dieser normalisierten Bildervektoren wird durch $b_{r,l}^{(k)}$ beschrieben. Hierbei gibt r die Rotation oder die Ausrichtung, l die Beleuchtungsrichtung und k die Nummer des Objektes an. Die komplette Menge von Bildern für ein Objekt wird als Objektmenge bezeichnet:

$$\{b_{1,1}^{(k)}, ..., b_{R,1}^{(k)}, ..., b_{R,L}^{(k)}\} \tag{5.1}$$

R und L geben die Anzahl von diskreten Positionen und Beleuchtungsrichtung an. Fasst man alle Objekte in einer Menge zusammen, so erhält man die universelle Bildmenge, wobei K die Anzahl der Bilder darstellt:

$$\begin{aligned} \{&b_{1,1}^{(1)}, ..., b_{R,1}^{(1)}, ..., b_{R,L}^{(1)}, \\ &b_{1,1}^{(2)}, ..., b_{R,1}^{(2)}, ..., b_{R,L}^{(2)}, \\ &\qquad ... \\ &b_{1,1}^{(K)}, ..., b_{R,1}^{(K)}, ..., b_{R,L}^{(K)}\} \end{aligned} \tag{5.2}$$

3.2 Eigenraum

Die Bildmenge ist sehr groß. Betrachtet man ein Bild der Größe 128×128, so erhält man einen normalisierten Bildvektor $b_{r,l}^{(k)}$ mit mehr als 10000 Komponenten. Um eine effiziente Berechnung durchführen zu können, wird als erstes jedes Bild in einen niedriger dimensionalen Unterraum abgebildet. Hierzu wird die Hauptachsentransformation (PCA) verwendet [2][3]. Es werden zwei Unterräume bestimmt, der universelle Eigenraum und der Objekteigenraum. Der universelle Eigenraum wird auf allen Bildern berechnet und dient zur Klassifizierung eines Objektes. Der Objekteigenraum wird auf Bildern von einem Objekt berechnet, welcher zur Positionsbestimmung verwendet wird.

PCA

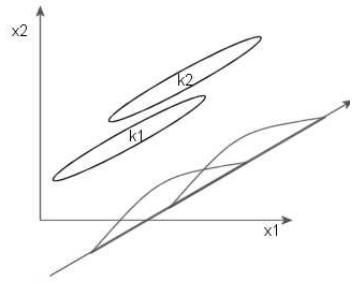


Abbildung 5.4: Hauptachsentransformation

Die Hauptachsentransformation projiziert die Merkmale in einen niedrieger dimensionalen Raum, so daß die Repräsentation der Klasse möglichst gut ist. Abbildung 5.4 zeigt dieses für einen 2 dimensionalen Fall. Dargestellt sind die Verteilungen der beiden Klassen $k1$ und $k2$. Mit der PCA wird der Unterraum bestimmt, der in diesem Fall durch die eingezeichnete Gerade repräsentiert wird. Werden die Elemente der Klassen in diesen Unterraum projiziert, so ergeben sich für die Klassen die eingezeichneten Verteilungen. Man sieht, daß die Gerade die Richtung der größten Streuung angibt.
Die Streuung der Merkmale ist durch die Streuungsmatrix gegeben:

$$S = \sum_{k=1}^{n} (x_k - m)(x_k - m)^T \tag{5.3}$$

$$m = \frac{1}{n} \sum_{k=1}^{n} x_k \tag{5.4}$$

n ist die Anzahl der Merkmalsvektoren und m ist der Mittelwert aller Merkmalsvektoren x_k. Gesucht wird nun eine Matrix V, die die Daten so in einen niedrig-dimensionalen Raum projiziert, daß die Streuung im Unterraum maximal wird. Die Dimension der Projektion wird als bekannt vorausgesetzt, bzw. ist variabel. Dazu maximiert man:

$$|V^T S V| \tag{5.5}$$

mit der Nebenbedingung:

$$V^T V = I \tag{5.6}$$

Dieses kann durch das Eigenwertproblem der Streuungsmatrix S gelöst werden. Die Eigenwerte und Eigenvektoren erfüllen dabei folgenden Zusammenhang:

$$\lambda e_i = S e_i \tag{5.7}$$

Die Eigenvektoren e_i bilden sortiert nach der Größe der zugehörigen Eigenwerte die Spalten der Matrix V. Die Stärke der Komprimierung ist variabel, je nach Wahl der Anzahl an Spalten von V, d.h. der Eigenvektoren. Werden wenige Eigenvektoren benutzt, so gehen viele Informationen verloren. Dadurch ist die Repräsentation nicht sehr genau. Nimmt man viele Eigenvektoren, so hat man viele Informationen und damit eine sehr genaue Beschreibung. Aber es sind nicht alle Informationen wichtig für die Repräsentation. Um eine geeignete Anzahl h von Eigenvektoren zu bestimmen, kann man den Quotienten aus der Summe der größten h Eigenwerte und der Summe aller Eigenwerte betrachten. In dem hier beschriebene Verfahren hat sich gezeigt, daß die Anzahl $h = 20$ ausreichend ist, um eine gut Erkennung zu erreichen.

$$V = [e_1, ..., e_h] \tag{5.8}$$

Universeller- und Objekteigenraum

Der universelle Eigenraum wird auf der gesamten Bildmenge berechnet. Die Streuungsmatrix der Bildmenge ist gegeben durch:

$$S_u = \sum_{k=1}^{K} \sum_{r=1}^{R} \sum_{l=1}^{L} (x_{r,l}^{(k)} - m)(x_{r,l}^{(k)} - m)^T \tag{5.9}$$

$$m = \frac{1}{n} \sum_{k=1}^{K} \sum_{r=1}^{R} \sum_{l=1}^{L} x_{r,l}^{(k)} \tag{5.10}$$

m ist der Mittelwert aller Bilder und $n = K \cdot R \cdot L$ die Anzahl aller Bilder. Von dieser Matrix S_u werden mittels PCA die Eigenvektoren e_i bestimmt. Die Eigenvektoren spannen den universellen Eigenraum auf. Der Objekteigenraum wird auf der Objektmenge bestimmt. Hierzu wird die Streuungsmatrix von allen Bildern zu einem Objekt bestimmt. Die Matrix ist gegeben durch:

$$S_o^{(k)} = \sum_{r=1}^{R} \sum_{l=1}^{L} (x_{r,l}^{(k)} - m^{(k)})(x_{r,l}^{(k)} - m^{(k)})^T \tag{5.11}$$

$$m^{(k)} = \frac{1}{n_k} \sum_{r=1}^{R} \sum_{l=1}^{L} x_{r,l}^{(k)} \tag{5.12}$$

Hierbei ist $m^{(k)}$ der Mittelwert der Bilder von einem Objekt k und $n_k = R \cdot L$ die Anzahl der Bilder von einem Objekt. Von dieser Matrix werden die Eigenvektoren e_i^k mittels PCA berechnet. Die Eigenvektoren spannen den Objekteigenraum auf. In dem hier beschriebenen Experiment wurde die Anzahl der Eigenvektoren $h = 20$ gewählt.

3.3 Repräsentation der Erscheinung

Die Repräsentation der Objekte findet in den Eigenräumen statt. In dem universellen Eigenraum wird die Erscheinungscharakteristik der Objekte beschrieben. Dazu wird jedes Bild aus der Bildmenge in den universellen Eigenraum projiziert. Als erstes wird von jedem Bild $x_{r,l}^{(k)}$ der Mittelwert m der Bildmenge abgezogen und dieses wird dann mit den Eigenvektoren multipliziert.

$$g_{r,l}^{(k)} = [e_1, ..., e_h]^T (x_{r,l}^{(k)} - m) = V^T (x_{r,l}^{(k)} - m) \tag{5.13}$$

Dadurch erhält man diskrete Punkte. Die Punkte beschreiben eine Mannigfaltigkeit.

$$g^{(k)}(\Theta_1, \Theta_2, ..., \Theta_m) \tag{5.14}$$

$\Theta_1, \Theta_2, ..., \Theta_m$ sind kontinuierliche Parameter für Ausrichtung und Beleuchtungsrichtung. Die Anzahl der Parameter kann variieren. Davon abhängig beschreibt das Mannigfaltigkeit eine Kurve, Oberfläche oder ein Volumen. Da im folgenden beschriebenen Experiment nur zwei Parameter, einer für die Ausrichtung und einer für die Beleuchtungrichtung, benutzt werden, wird die Mannigfaltigkeit gegeben durch:

$$g^{(k)}(\Theta_1, \Theta_2) \tag{5.15}$$

Die Mannigfaltigkeit wird mittels Cubic Spline bestimmt. Die Idee der Splines ist, daß man stückweise mit Polynomen eines festen Grades interpoliert. Hierbei wird nicht der Polynomgrad sondern die Anzahl der Stücke erhöht. Die diskreten Punkte dienen als Stützpunkte für die Interpolation. Da hier Cubic Splines verwendet werden, haben die Polynome den Grad drei.
Jedes Bild $x_{r,l}^{(k)}$ aus der Objektmenge wird in den Objekteigenraum projiziert. Dieses ist genauso wie bei der Projektion in den universellen Eigenraum. Es werden nur die Bilder zu einen Objekt betrachtet. Von diesen wird der Mittelwert $m^{(k)}$ der Objektbilder abgezogen.

$$f_{r,l}^{(k)} = [e_1^{(k)}, ..., e_h^{(k)}]^T (x_{r,l}^{(k)} - m^{(k)}) = {V^{(k)}}^T (x_{r,l}^{(k)} - m^{(k)}) \tag{5.16}$$

Von den diskreten Punkten wird mittels Cubic Spline Interpolation eine Mannigfaltigkeit bestimmt, die wie oben erwähnt hier nur von zwei Parametern abhängt, der Beleuchtungsrichtung und der Ausrichtung.

$$f^{(k)}(\Theta_1, \Theta_2) \tag{5.17}$$

Das Bild 5.5 zeigt Mannigfaltigkeiten, die zur Veranschaulichung nur bezüglich der ersten drei Eigenvektoren zu den größten Eigenwerten dargestellt sind. Jedes in diesen Eigenraum projizierte Bild entspricht einem Punkt, welcher hier die Ausrichtung Θ_1 und die Beleuchtungsrichtung Θ_2 als Parameter hat. Die Punkte werden interpoliert. Dadurch erhält man die dargestellten Kurven. Die Kurve ist entlang Θ_1 geschlossen, da, wenn man das Objekt um 360 Grad gedreht hat, man wieder am Ausgangspunkt ankommt. Da es für jede Ausrichtung mehrere Beleuchtungsrichtungen gibt, hat die Kurve eine gewisse Breite.

4 Objekterkennung und Bestimmung der Position

Die Erkennung und Positionsbestimmung eines Bildes findet in den Eigenräumen statt. Um das Bild zu erkennen, wird es in den universellen Eigenraum projiziert. Es wird dann das Objekt gesucht, dessen Mannigfaltigkeit am nächsten an dem projizierten Bild liegt. Hat man das Objekt bestimmt, so wird das Bild in den Objekteigenraum projiziert. Die Ausrichtung des Objektes wird durch die Position auf der Mannigfaltigkeit gegeben, die am nächsten an dem projizieren Bild liegt.

4.1 Abstände

Gegeben sei eine Bildmenge mit K Objekten. Man möchte ein Bild x klassifizieren. Gesucht wird die Klasse k, die bei einer gegebenen Verteilung p und gegebenen x am wahrscheinlichsten ist. Das bedeutet, es wird das k gesucht, welches die Verteilung $p(k|x)$ maximiert.

$$x \to r(x) = \underset{k}{\operatorname{argmax}}\{p(k|x)\} \tag{5.18}$$

Der Ausdruck ist auch bekannt als die Bayes'sche Entscheidungsregel [4]. Für die Bayes'sche Entscheidungsregel ist bekannt, daß sie die Fehlerrate minimiert, falls die tatsächliche Verteilungsfunktion bekannt

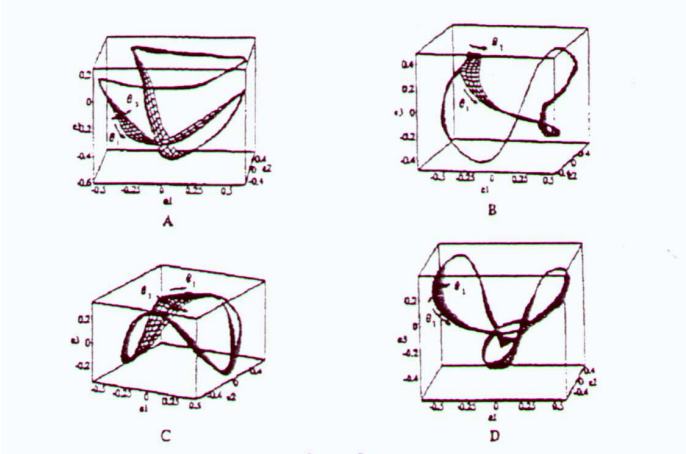


Abbildung 5.5: Mannigfaltigkeiten, die nur bezüglich der ersten drei wichtigsten Eigenvektoren dargestellt sind.

ist.
In dem vorgestellten Ansatz werden zwei Voraussetzungen gemacht. Die a-priori Wahrscheinlichkeit ist für alle Klassen gleich, d.h. jede Klasse ist gleich wahrscheinlich:

$$p(k) = \frac{1}{K} \tag{5.19}$$

Des weiteren ist $p(x|k)$ Gauß-verteilt.
Man kann die Bayes'sche Entscheidungsregel folgendermaßen umschreiben:

$$\underset{k}{\text{argmax}}\{p(k|x)\} = \underset{k}{\text{argmax}}\{\frac{p(k) \cdot p(x|k)}{p(x)}\} \tag{5.20}$$

Da $p(x)$ nicht von k abhängt, kann es bei der Berechnung vernachläßigt werden.

$$\underset{k}{\text{argmax}}\{p(k|x)\} = \underset{k}{\text{argmax}}\{p(k) \cdot p(x|k)\} \tag{5.21}$$

Durch die Annahme, daß alle Klassen gleich verteilt sind, kann $p(k)$ aus dem Term weggelassen werden.

$$\underset{k}{\text{argmax}}\{p(k|x)\} = \underset{k}{\text{argmax}}\{p(x|k)\} \tag{5.22}$$

Nach Annahme ist $p(x|k)$ Gauss-verteilt. Weiterhin wird für alle Klassen angenommen, daß die Kovarianzmatrizen gleich sind, d.h. $\Sigma_k = \Sigma$ für alle k.

$$\underset{k}{\text{argmax}}\{p(k|x)\} = \underset{k}{\text{argmax}}\{\exp[-\frac{1}{2}(x-\mu_k)^T\Sigma^{-1}(x-\mu_k)]\} \tag{5.23}$$

Da die Exponentialfunktion monoton ist, braucht nur der Exponent betrachtet werden.

$$\underset{k}{\text{argmax}}\{p(k|x)\} = \underset{k}{\text{argmax}}\{-\frac{1}{2}(x-\mu_k)^T\Sigma^{-1}(x-\mu_k)\} \tag{5.24}$$

Die Konstante $\frac{1}{2}$ spielt bei der Maximierung keine Rolle. Weiterhin wird der Ausdruck maximal, wenn $\frac{1}{2}(x-\mu_k)^T\Sigma^{-1}(x-\mu_k)$ minimal wird.

$$\underset{k}{\text{argmax}}\{p(k|x)\} = \underset{k}{\text{argmin}}\{(x-\mu_k)^T\Sigma^{-1}(x-\mu_k)\} \tag{5.25}$$

Ist Σ die Einheitsmatrix, so ergibt das den Euklidischen Abstand. Die Einheitsmatrix kann unter der Bedingung $\Sigma_k = \Sigma$ immer erreicht werden, indem die Merkmalsvektoren einer sogenannten "Whitening-Transformation" unterzogen werden. Die PCA hat hier die Eigenschaft, daß die entstandenen Merkmale unkorreliert sind [4].

$$\underset{k}{\text{argmax}}\{p(k|x)\} = \underset{k}{\text{argmin}}\{\|x-\mu_k\|^2\} \tag{5.26}$$

Die Erkennung kann dann auf Grund von Distanzberechnungen durchgeführt werden.
Ein Bild x_m kann durch seine Projektion angenähert werden.

$$x_m \approx \sum_{i=1}^{k} g_{m_i} e_i + c \tag{5.27}$$

Benutzt man in der Distanzberechnung zweier Bilder x_m und x_n diese Gleichung, so ergibt sich:

$$\begin{aligned} \|x_m - x_n\|^2 &\approx \|\sum_{i=1}^{h} g_{m_i} e_i - \sum_{i=1}^{h} g_{n_i} e_i\|^2 \\ &= \|\sum_{i=1}^{h} (g_{m_i} - g_{n_i}) e_i\|^2 \\ &= \sum_{i=1}^{h}\sum_{j=1}^{h} e_i^T e_j \cdot (g_{m_i} - g_{n_i})^2 \\ &= \|g_m - g_n\|^2 \end{aligned} \tag{5.28}$$

Die letzte Vereinfachung erfolgt durch die Tatsache, daß die Eigenvektoren orthogonal zueinander sind, d.h. $e_i^T e_j = 1$ wenn $i = j$, und 0 sonst. Die Distanz zweier Bilder kann also durch die Distanz ihrer Projektionen approximiert werden werden.

4.2 Erkennung

Das zu bestimmendes Bild x wird als erstes in den universellen Eigenraum projiziert.

$$x_u = [e_1, ..., e_h]^T (x - m) = V^T (x - m) \tag{5.29}$$

Gesucht wird das Objekt, dessen Mannigfaltigkeit am nächsten an x_u ist.

$$k^* = argmin_k \{ \min_{\Theta_1, \Theta_2} \|x_u - g^{(k)}(\Theta_1, \Theta_2)\| \} \tag{5.30}$$

Ist die minimale Distanz grösser als ein festgelegter Schwellwert, so wird angenommen, daß das Bild x von keinem Objekt ist, die das System gelernt hat. Liegt die Distanz unterhalb des Schwellwertes, so ist das Bild von dem Objekt k^*.
Steht das Objekt fest, so wird x in den Objekteigenraum projiziert.

$$x_o = [e_1^{(k^*)}, ..., e_h^{(k^*)}]^T (x - m^{(k^*)}) = V^{(k^*)^T} (x - m^{(k^*)} \tag{5.31}$$

Um die Position des Bildes zu bestimmen, wird die Lage auf der Mannigfaltikeit gesucht welche am nächsten zu x_o liegt.

$$i = argmin_{\Theta_1, \Theta_2} \|x_o - f^{(k)}(\Theta_1, \Theta_2)\| \} \tag{5.32}$$

Die Lage auf der Mannigfaltigkeit gibt dann die Parameter des Bildes x an. Θ_1 steht für die Ausrichtung und Θ_2 für die Beleuchtungrichtung.

4.3 Suchverfahren

Für die Suche nach dem nächsten Punkt auf der Mannigfaltigkeit wurden in der betrachteten Arbeit zwei verschiedene Verfahren verwendet. Das erste Verfahren basiert auf der binären Suche im hochdimensionalen Raum, während bei dem zweite Verfahren ein dreischichtiges neuronales Netz verwendet wurde.
Das erste Verfahren ist eine binäre Suche im hochdimensionalen Raum [5]. Um das Prinzip zu veranschaulichen, wird der 3-D Raum betrachtet. Es wird eine Punktmenge bestimmt, deren Punkte innerhalb eines Würfels der breite 2ϵ liegen. Das Zentrum dieses Würfels ist das in den Eigenraum projizierte Bild. In dieser Punktmenge wird dann im Sinne des Euklidischen Abstandes der am nächsten gelegene Punkt gesucht.

Die Bestimmung der Punktmenge erfolgt folgendermaßen. Zuerst werden die Punkte bestimmt, die zwischen zwei parallelen Ebenen X_1 und X_2 liegen. Diese Ebenen haben zu dem projizierten Bildpunkt jeweils den Abstand ϵ und sind senkrecht zur x-Achse. Diese Punkte werden in eine Kandidatenliste eingetragen. Als nächstes werden die Punkte aus der Kandidatenliste gestrichen, die nicht zwischen zwei weiteren Ebenen Y_1 und Y_2 liegen. Die Ebenen sind senkrecht zur y-Achse und haben ebenfalls den Abstand ϵ zu dem projizierte Bildpunkt. Dieses wird dann für die Ebenen Z_1 und Z_2 wiederholt. In der Kandidatenliste sind jetzt nur noch die Punkte enthalten, die innerhalb des Würfels mit der Seitenlänge 2ϵ und dem projizierten Bild als Zentrum.
Da die bestimmte Punktmenge klein ist, sind die Kosten der Suche nach dem am nächsten liegenden Punkt klein. Die Hauptkosten entstehen bei der Erstellung der Kandidatenliste. Um die Kandidatenliste zu erstellen wird in [5] eine Datenstruktur vorgestellt auf der man mit 1-D binärer Suche effizient die Punkte zwischen zwei parallelen Hyperebenen bestimmen kann. Hierbei werden die Punkte in k 1-D Arrays gespeichert. k ist die Dimension des Eigenraums. Das j-te Array enthält die j-te Komponente der Punkte. Jedes Array wird sortiert, wobei nachgehalten wird, welche Stelle im sortierten Array zu welchen Punkt gehört. In dem Array mit der ersten Koordinate jeden Punktes werden die Punkte mit binärer Suche bestimmt, dessen erste Koordinate innerhalb der ϵ-Umgebung der Koordinate des projizierten Bildpunktes liegen. Diese werden in die Kandidatenliste eingetragen. Für die entstandene Kandidatenliste wird dann dieses Verfahren für die weiteren Koordinaten iteriert. Dadurch erhält man die Kadidatenliste, die die Punkte in der ϵ-Umgebung des projizierten Bildpunktes enthält. Die Kosten liegen in $O(k \log_2 n)$, wobei k die Dimension des Eigenraumes und n die Anzahl der Punkte der Mannigfaltigkeit ist.
Bei dem zweiten Verfahren wird ein neuronales Netz [6] verwendet. Dabei wird das Zuordnen der Eingabepunkte und der Parameter der Mannigfaltigkeit gelernt. Dieses Netz basiert auf der Regularisationstechnik. Das Netzwerk hat die Eigenschaft die diskreten Bildpunkte implizit zu interpolieren. Dadurch wird die Cubic Spline Interpolation zur Bestimmung der Mannigfaltigkeit bei diesem Verfahren nicht benötigt. Durch die implizite Interpolation erreicht dieses Verfahren etwas genauere Ergebnisse in der Positionsbestimmung.

5 Experiment

Die Bilder der Objekte wurden in Abhängigkeit von der Ausrichtung und der Beleuchtungsrichtung mit einer Kamera aufgenommen.

5.1 Aufbau

Für die Repräsentation des Objektes werden 2-D Bilder des Objektes benötigt. Um diese Bilder zu erhalten, wird ein Objekt auf eine motorisierte Drehplatte gestellt. Diese Platte kann sich nur um eine Achse drehen. Daher wird nur ein Parameter für die Ausrichtung betrachtet. Die Beleuchtungsrichtung wurde mit einem Roboterarm variiert. Diese wurde durch den zweiten Parameter angegeben. Von einem Objekt werden dann mit einer Kamera Bilder unter verschiedenen Ausrichtungen und Beleuchtungsrichtungen aufgenommen.

5.2 Ergebnisse

Das System wurde auf zwei Objektmengen trainiert und getestet. Die Objektmenge A enthielt 4 Objekte die gleiche Reflektionseigenschaften haben, aber deren Gestalt in gewissen Positionen sehr ähnlich sind. Die Objektmenge B enthielt 4 Objekte mit einer komplexen Erscheinungscharakteristik. Zu jeder Objektmenge wurden zwei Trainingsmengen mit unterschiedlicher Anzahl an verwendeten Ausrichtungen definiert. Die Trainingsmenge I enthält für jedes Objekt 450 Bilder. Es wurden 5 Beleuchtungsrichtungen und 90 verschiedene Ausrichtungen gewählt. Die Trainingsmenge II enthält für jedes Objekt 90 Bilder. Für diese Trainingsmenge wurden wieder 5 Beleuchtungsrichtungen aber nur 18 Ausrichtungen verwendet. Die Testmengen bestanden wieder aus diesen 4 Objekten, wobei hier 3 Beleuchtungsrichtungen und 90 Positionen benutzt wurden. Die Positionen im Test waren andere als im Training.
Tabelle 5.1 zeigt die Testergebnisse. Die Ergebnisse stellen den durchschnittlichen absoluten Ausrichtungsfehler in Grad dar. Die beiden Objektmengen erziehlen auf der Trainingsmenge I eine durchschnittliche Abweichung von 0.5 Grad. Es zeigt, daß die Positionsbestimmung sehr genau ist. Selbst auf der Trainingsmenge II mit deutlich weniger Ausrichtungen, ist die Bestimmung der Ausrichtung nicht erheblich schlechter. Sie fällt auf ca. 1 Grad ab.

Tabelle 5.1: Durchschnittlicher absoluter Ausrichtungsfehler in Grad

Durchschnittlicher absoluter Ausrichtungsfehler in Grad		
	I	II
A	0.5	1.0
B	0.5	1.2

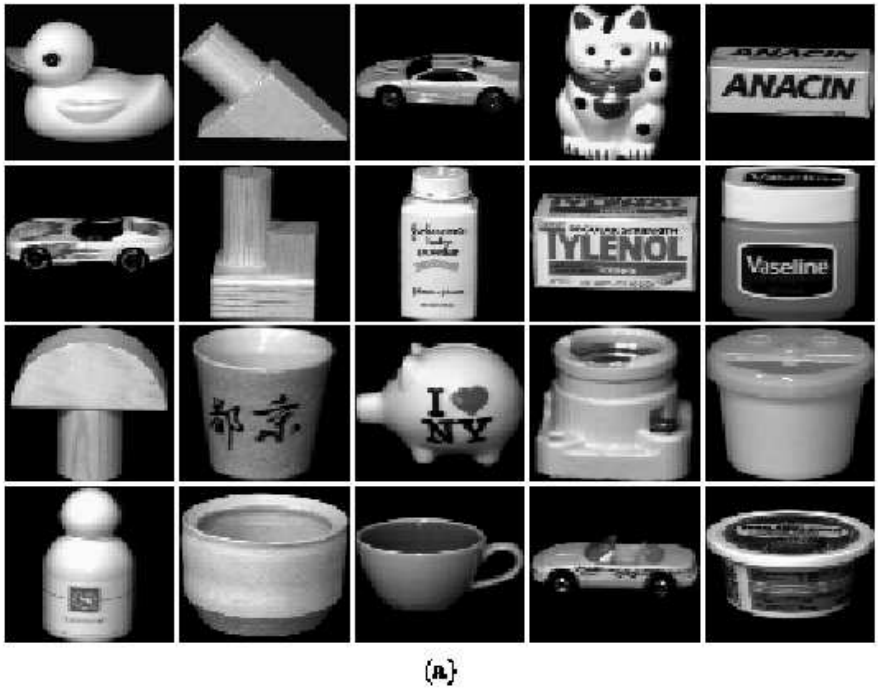


Abbildung 5.6: Objekte, die in diesem Experiment verwendet wurden

Abbildung 5.6 zeigt Objekte mit denen das System weiter getestet wurde. Die Erscheinung dieser Objekte wurde im 20 dimensionalen universellen Eigenraum dargestellt. Sowohl die Erkennung als auch die Positionsbestimmung fanden im universellen Eigenraum statt. Der komplette Erkennungsprozess, mit Segmentierung, Normalisierung, Projektion des Bildes in den universellen Eigenraum und die Suche nach dem nächsten Objekt und der Position, wurde in weniger als einer Sekunde auf einer Sun Workstation durchgeführt. Die Erkennung läuft also fast in Echtzeit ab. Die Bilder wurden alle richtig erkannt. Dieses Ergebnis ist subjektiv sehr gut, da wenn man sich die Objekte anschaut, man feststellt, daß einige Objekte, wie z.B. die Autos, sich sehr stark ähneln.

6 Zusammenfassung

Es wurde ein Verfahren zur erscheinungsbasierten 3-D Objekterkennung vorgestellt. Ein 3-D Objekt wird durch eine Menge von 2-D Bildern dargestellt. Es werden zwei Eigenräume mit der PCA bestimmt, in die die Bilder projiziert werden. Die projizierten Bilder eines Objektes werden mit Splines interpoliert. Die entstandene Kurve beschreibt eine Mannigfaltigkeit, auf der die Erkennung und Bestimmung der Ausrichtung stattfindet.
Das vorgestellte System hat auf den benutzten Bildern gute Ergebnisse erzielt. Es wird jedoch vorausgesetzt, daß die Bilder segmentiert sind. Des weiteren ist die geeignete Größe des Eigenraums unbekannt. Ein Objekt wurde bezüglich der zwei Parameter Ausrichtung und Beleuchtungsrichtung dargestellt. Um aber die Ausrichtung eines Objektes in drei Dimensionen zu beschreiben, benötigt man zwei Parameter für die Ausrichtung und ein Parameter für die Beleuchtungsrichtung. Obendrein ist es schwierig diesen Ansatz

in einer realen Anwendung einzusetzen, da man Bilder von den Objekten benötigt, die man erkennen möchte. Man kann z.B. in dem Bereich der Gesichtserkennung, nicht jeden Menschen auf eine Drehplatte setzen und die Bilder von seinem Gesicht machen. Im Bereich der Medizin wäre dieses Verfahren z.B. bei CT Bildern einsetzbar, da diese Bilder schon drei dimensional vorliegen.

Literaturverzeichnis

[1] H. Murase, S.K. Nayar: Visual Learning and Recognition of 3D Objects from Appearance. International Journal of Computer Vision, 1995; 14(1): 5–24.

[2] R.O. Duda, P.E. Hart, D.G. Storck: Pattern Classification. J. Wiley, New York, 2001.

[3] T. Lehmann, W. Oberschelp, E. Pelikan, R. Repges: Bildverarbeitung für die Medizin. Springer, 1997.

[4] H. Ney: Skript zur Vorlesung: Mustererkennung und Neuronale Netze. RWTH Aachen, 2002.

[5] S.A. Nene, S.K. Nayar: A Simple Algorithm for Nearest Neighbor Search in High Dimensions. Technical Report No. CUCS-030-95, 1995.

[6] T. Poggio, F. Girosi: A Theory of Networks for Approximation and Learning. 1989.

Kantenerkennung in medizinischen Bildern

von Markus Jogmin

Betreuer: Thomas Lehmann

Inhaltsverzeichnis

Zusammenfassung

In dieser Ausarbeitung wird eine Technik vorgestellt, die einen Vergleich von verschiedenen Verfahren zur Kantendetektion ermöglicht. Hierzu werden verschiedene ROC-Kurven erstellt und miteinander verglichen. Konkret werden 50 Objekt- und 10 Luftbilder mithilfe der drei Verfahren von Canny, Sobel und Heitger bearbeitet. Anschließend findet ein Vergleich der Ergebnisse statt.

Keywords: Kantendetektion, Evaluierung, Ground-Truth, ROC-Kurve, Gold Standard

1 Einleitung

Beim Titel *Kantenerkennung in medizinischen Bildern* stellt sich die Frage: „Was ist Kantenerkennung und wozu braucht man sie in der Medizin?“ Kantenerkennung dient dazu, in Bildern Kanten, d.h. Konturen und Linien zu erkennen und diese hervorzuheben. In der Medizin werden z.B. Ränder von Tumoren so deutlich gemacht, dass man ihre Größe und Form auf Bildern besser erkennen kann. Bei den Bildern handelt es sich in der Medizin oft um Sonographie- oder Röntgenbilder (Abb. 6.1). Gewünscht wird ein Verfahren, welches aus einem vorgegebenen Bild ein neues Bild erstellt, in dem die Kanten deutlich erkennbar sind. Hierzu ist eine geeignete Auswahl unter den Verfahren zur Kantendetektion notwendig. Im Folgenden werden drei Verfahren und eine Technik zur Bewertung bzw. zum Vergleich von Verfahren zur Kantenerkennung vorgestellt. Die vorgestellte Technik zum Vergleichen von Kantendetektionsverfahren ist einem Paper [1] entnommen, aus dem auch die später vorgestellten Ergebnisse stammen.

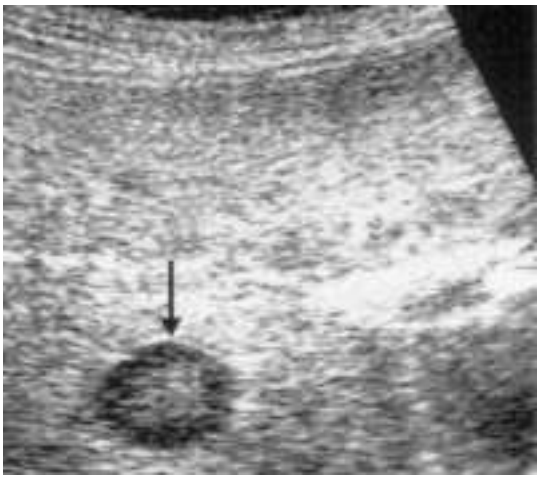


Abbildung 6.1: Sonographiebild (Abb. aus [2])

Nachfolgend wird zwischen Kanten und Linien unterschieden. Wie man aus Abbildung 6.2 ersieht, unterscheiden sich Kanten und Linien im Verlauf der Grauwerte. Kanten sind Übergänge (stetig oder abrupt) zwischen unterschiedlichen Grauwertniveaus, erkennbar am monoton steigenden bzw. fallenden Kurvenverlauf. Dagegen stellen Linien markante Änderungen zwischen benachbarten, fast gleichen Grauwertniveaus dar, was im lokalen Extremum im Kurvenverlauf deutlich wird.

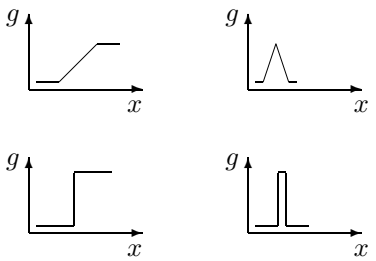


Abbildung 6.2: Beispiele für 1D-Grauwertprofile von Kanten (links) und Linien (rechts)

2 Verfahren zur Kantendetektion

2.1 Canny

Zur Kantenerkennung mit dem Canny-Operator wird zuerst die Bildfunktion $f(x, y)$ mit den Richtungsableitungen der Gauß-Funktion in x- und y-Richtung gefaltet:

$$D_x(x,y) = \frac{\partial}{\partial x}\left(G_\sigma(x,y) * f(x,y)\right) = \frac{\partial G_\sigma(x,y)}{\partial x} * f(x,y) \tag{6.1}$$

$$D_y(x,y) = \frac{\partial}{\partial y}\left(G_\sigma(x,y) * f(x,y)\right) = \frac{\partial G_\sigma(x,y)}{\partial y} * f(x,y) \tag{6.2}$$

Dabei ist

$$G_\sigma(x,y) = \exp\left(\frac{-(x^2+y^2)}{2\sigma^2}\right) \tag{6.3}$$

bis auf einen Skalierungsfaktor die Gauß-Funktion mit der Standardabweichung σ und dem Faltungsoperator $*$.
Damit berechnet sich die erste Ableitung der Gauß-Funktion in x-Richtung als:

$$\frac{\partial G_\sigma(x,y)}{\partial x} = \frac{-x}{\sigma^2}\exp\left(\frac{-(x^2+y^2)}{2\sigma^2}\right) \tag{6.4}$$

Der Gradient $\vec{D}$ der Gauß-geglätteten Bildfunktion an der Stelle (x, y) ergibt sich dann zu:

$$\vec{D}(x,y) = D_x(x,y)\vec{e}_x + D_y(x,y)\vec{e}_y \tag{6.5}$$

Dieser Gradient $\vec{D}$ zeigt stets in Richtung des größten Anstiegs der Gauß-geglätteten Bildfunktion und steht somit immer senkrecht zur Kantenrichtung an der Stelle (x, y). Der Betrag des Gradienten $\vec{D}$ ergibt sich also zu:

$$D(x,y) = \sqrt{D_x(x,y)^2 + D_y(x,y)^2} \tag{6.6}$$

Dieser Betrag entspricht der Größe der Änderung der Gauß-geglätteten Bildfunktion an der Stelle (x, y) und ist damit ein gutes Maß für die Stärke der Kante. Da wir an Kanten interessiert sind, die eine Breite von nur einem Pixel besitzen, muss das bisher erstellte Bild nochmals nachbearbeitet werden. Die Nachbearbeitungsschritte werden nun erläutert.

Non-Maxima-Suppression

Da die Bildfunktion und das Ergebnis der Filteranwendung nur auf einem diskreten Gitter vorliegen und die Gradientenrichtung im Allgemeinen nicht genau in Richtung eines Gitterpunktes zeigt, werden zwei Gradientenwerte, die der Gradientenrichtung am Nächsten liegen, durch lineare Interpolation berechnet. Hierzu wird zuerst aus dem Gradienten $\vec{D}$ die Normale $\vec{n}$ zur Kante bestimmt:

$$\vec{n}(x,y) = \frac{\vec{D}}{\sqrt{D_x(x,y)^2 + D_y(x,y)^2}} = n_x\vec{e}_x + n_y\vec{e}_y \tag{6.7}$$

Entsprechend dem Beispiel in Abbildung 6.3 berechnen sich dann die interpolierten Gradienten D_1 und D_2 zu:

$$D_1 = \frac{n_x}{n_y}D(x+1,y-1) + \frac{n_y-n_x}{n_y}D(x,y-1) \tag{6.8}$$

$$D_2 = \frac{n_x}{n_y}D(x-1,y+1) + \frac{n_y-n_x}{n_y}D(x,y+1) \tag{6.9}$$

Analog kann man die Werte D_1, D_2 für die anderen sieben möglichen Richtungen bestimmen, wobei man die Richtung in jedem Punkt durch eine Fallunterscheidung anhand der Normalenrichtung ermittelt. Ein Wert $D(x, y)$ ist dann ein Maximum, wenn sowohl $D(x, y) \geq D_1$ als auch $D(x, y) \geq D_2$ gilt. Nach Anwendung eines Schwellwertverfahrens auf die so vorselektierten Kantenpunkte erhält man 1-Pixel breite Kanten, die an dem Ort der größten Grauwertänderung liegen.
Die so berechneten Punkte haben immer noch eine Grauwert-Wertigkeit, die binarisiert werden muss.

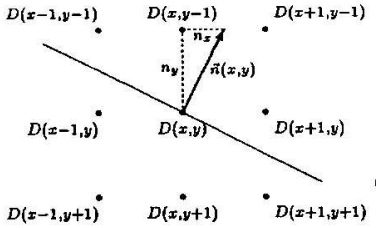


Abbildung 6.3: Suche nach dem Maximum in Gradientenrichtung (aus [3])

Hysteresis-Threshold

Das *Hysteresis-Threshold*-Verfahren ist ein Schwellwertverfahren, bei dem man nicht nur einen Schwellwert, sondern ein Intervall verwendet, welches durch einen hohen Schwellwert T_{high} und einen niedrigen Schwellwert T_{low} begrenzt wird.
Punkte, an denen der Gradientenbetrag größer ist als der hohe Schwellwert T_{high}, werden verwendet, um neue Konturen zu beginnen, während Punkte, an denen der Gradientenbetrag größer ist als der untere Schwellwert T_{low} nur zur Fortsetzung von Kanten dienen. Punkte, deren Gradientenbetrag unterhalb des unteren Schwellwertes liegt, werden verworfen. Somit lässt sich das *Hysteresis-Threshold*-Verfahren als ein einfaches Verfahren zur Konturverfolgung ansehen.

Beispiel

Abbildung 6.4 zeigt, wie aus einem Original a) nach Ermittlung der vertikalen Kanten in b) und der horizontalen Kanten in c) sowie nach der Normierung des Gradienten in d) und nach Anwendung der Nachbearbeitungsverfahren Non-Maxima-Suppression und Hysteresis-Threshold schließlich die Endversion unter e) entsteht.

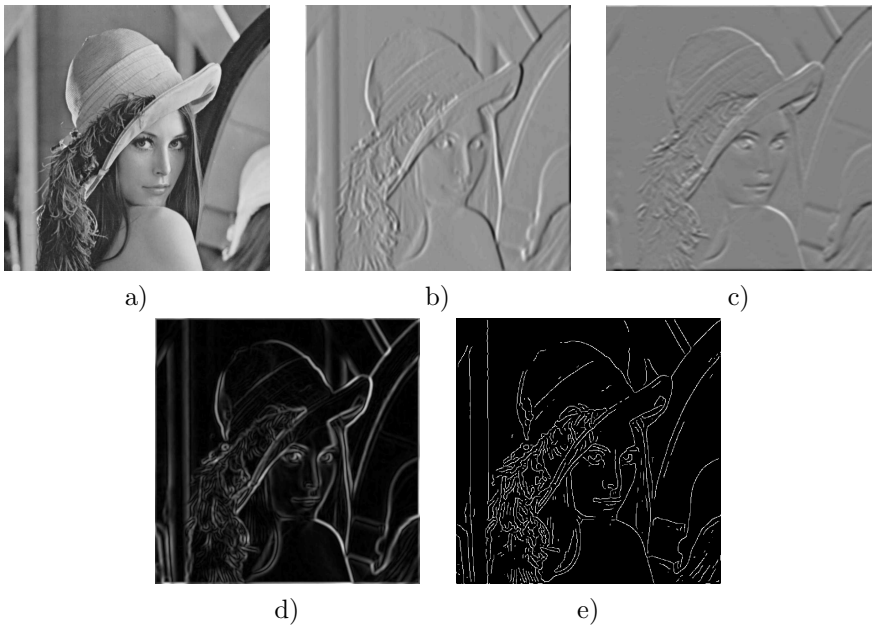


Abbildung 6.4: Zwischenschritte beim Canny-Verfahren

2.2 Sobel

Das Sobel-Verfahren basiert auf der diskreten Faltung. Hierbei wird die in Abbildung 6.5 dargestellte Maske H wie folgt mit der Bildfunktion $f(x, y)$ zur Funktion $s(x, y)$ verknüpft:

	h(-1,-1)	*h(-1,0)*	*h(-1,1)*
H:	*h(0,-1)*	*h(0,0)*	*h(0,1)*
	h(1,-1)	*h(1,0)*	*h(1,1)*

Abbildung 6.5: Maske

$$s = f * H = \sum_{(i,j)\in H} f(x+i, y+j)h(i,j) \tag{6.10}$$

Der Sobel-Operator ist ein Differentiationsoperator, der senkrecht zur Differenzierungsrichtung eine Glättung mit einer (1 2 1) Binomial-Filtermaske enthält. Die Binomial-Verteilung wird hier als eine diskrete Approximation der Gauß-Funktion verwendet. So ergeben sich die in Abbildung 6.6 angegebenen Filtermasken für die x- und y-Richtung. Die ebenfalls in Abbildung 6.6 angegebenen Masken für die diagonalen Richtungen folgen aus Rotationsüberlegungen.

$S_y = \frac{1}{4}$

1	2	1
0	0	0
-1	-2	-1

$S_x = \frac{1}{4}$

1	0	-1
2	0	-2
1	0	-1

$S_/ = \frac{1}{4}$

0	-1	-2
1	0	-1
2	1	0

$S_\backslash = \frac{1}{4}$

-2	-1	0
-1	0	1
0	1	2

Abbildung 6.6: Sobel-Operator

Der kombinierte Sobel-Operator lautet dann:

$$\widehat{S} = \max\{|S_x|, |S_y|, |S_/|, |S_\backslash|\} \tag{6.11}$$

Das Verfahren liefert breite Kanten, wie man an Abbildung 6.7 erkennt. Deshalb werden in der Version des Sobel-Operators, die hier zum Vergleichen benutzt wird, zusätzlich noch die Methoden *Non-Maxima-Suppression* und *Hysteresis-Threshold* des Canny-Verfahrens verwendet.

0	0	10	20	30	40	40	40
0	0	10	20	30	40	40	40
0	0	10	20	30	40	40	40
0	0	10	20	30	40	40	40
0	0	10	20	30	40	40	40
0	0	10	20	30	40	40	40

$\xrightarrow{S_x}$

0	10	20	20	20	20	0	0
0	10	20	20	20	20	0	0
0	10	20	20	20	20	0	0
0	10	20	20	20	20	0	0
0	10	20	20	20	20	0	0
0	10	20	20	20	20	0	0

Abbildung 6.7: Ausschnitt aus einem Bild; die Zahlen stehen für Grauwerte

2.3 Heitger

Beim Verfahren von Heitger [4] werden Kanten und Linien erkannt und unterschiedlich behandelt. Der Nachteil der bisherigen Methoden besteht darin, dass Linien im Original nach der Bearbeitung zu Doppellinien werden; siehe hierzu das Kreuz in Abbildung 6.8 rechts. Die gleichzeitige Behandlung von Kanten und Linien liefert dagegen einfache Linien, wie man in Abbildung 6.8 an dem Kreuz im mittleren Bild sehen kann. Heitger verwendet für die gleichzeitige Behandlung von Kanten und Linien einen Bandpassfilter, der aus der Fourier-Transformierten F zu der Bildfunktion $f(x, y)$ den „interessanten“ Frequenzbereich herausfiltert. Der Bandpassfilter besteht aus zwei Filtern $\mathcal{F}_o$ und $\mathcal{F}_e$, welche die ungeraden (odd) und geraden (even) Anteile der Frequenzfunktion liefern. In Polarkoordinaten arbeitet er im zweidimensionalen

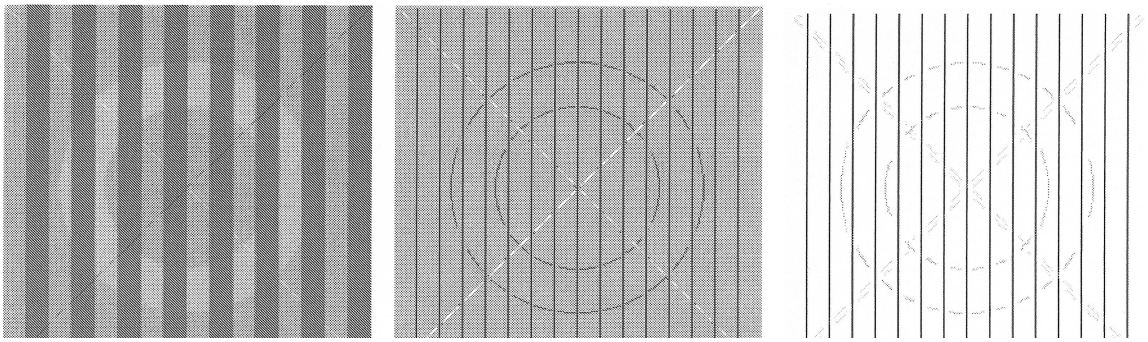


Abbildung 6.8: Vergleich von Heitger und Canny: Orginal links, Heitger Mitte und Canny rechts (aus [4])

Fall mit

$$\mathcal{F}_o(r,\theta) = \mathcal{H}_o \cdot \cos^{2n}(\theta - \theta_0) \quad \text{und} \quad \mathcal{F}_e(r,\theta) = \mathcal{H}_E \cdot \cos^{2n}(\theta - \theta_0), \tag{6.12}$$

wobei $\mathcal{H}_o$ und $\mathcal{H}_e$ die Komponenten darstellen, die vom Radius r abhängen, während die Winkelkomponente θ von der Potenz der Kosinusfunktion abhängig ist. Die Faltung eines Bildes mit dem ungeraden und geraden Filter wird im Frequenzraum durch eine komplexe Multiplikation mit den Funktionen $\mathcal{F}_o$ und $\mathcal{F}_e$ ausgeführt. Anschließend führt man eine inverse Fourier-Transformation durch. Zum Ortsvektor $\vec{r}$ und zur Filterrichtung θ_n, wobei N Filterrichtungen durch $\theta_n = \frac{\pi n}{N}$ gegeben sind, erhält man als Ergebnisse der Faltung mit den ungeraden und geraden Filtern die Funktionen $\mathcal{R}_{o,n}(\vec{r})$ und $\mathcal{R}_{e,n}(\vec{r})$. Diese werden kombiniert zu einer Antwortfunktion, die wie folgt definiert wird:

$$\mathcal{M}_n(\vec{r}) = \sqrt{[\mathcal{R}_{e,n}(\vec{r})]^2 + [\mathcal{R}_{o,n}(\vec{r})]^2} \tag{6.13}$$

Weiterhin sind nach Heitger noch die 1. und 2. Ableitung der Antwortfunktion $\mathcal{M}_n(\vec{r})$ nach der Orthogonalrichtung von θ_n zu beachten:

$$\text{1. Ableitung:} \quad O_n^{(1)}(\vec{r}) = \frac{\partial \mathcal{M}_n(\vec{r})}{\partial \theta_{n_\perp}} \tag{6.14}$$

$$\text{2. Ableitung:} \quad O_n^{(2)}(\vec{r}) = \frac{\partial^2 \mathcal{M}_n(\vec{r})}{\partial \theta_{n_\perp}^2} \quad \text{mit } \theta_{n_\perp} = \theta_n + \frac{\pi}{2} \tag{6.15}$$

Hieraus entwickelt Heitger die Filterfunktionen $\mathcal{S}_n(\vec{r})$ und $\mathcal{C}_n(\vec{r})$, die auf Kanten (im ungeraden Fall) und Linien (im geraden Fall) reagieren und somit entsprechende Eigenschaften dieser Objekte gezielt unterdrücken bzw. verstärken.

$$\mathcal{S}_n(\vec{r}) = \left[(\mathcal{R}_{o,n}(\vec{r}))^2 - \alpha \left(O_n^{(1)}(\vec{r}) \right)^2 - \beta \left(O_n^{(2)}(\vec{r}) \right)^2 \operatorname{sign} \left(O_n^{(2)}(\vec{r}) \right) \right]^+ \tag{6.16}$$

$$\mathcal{C}_n(\vec{r}) = \left[(\mathcal{R}_{e,n}(\vec{r}))^2 - \alpha \left(O_n^{(1)}(\vec{r}) \right)^2 - \beta \left(O_n^{(2)}(\vec{r}) \right)^2 \operatorname{sign} \left(O_n^{(2)}(\vec{r}) \right) \right]^+ \tag{6.17}$$

Hierbei unterdrückt die 1. Ableitung die unerwünschten Eigenschaften für Kanten und Linien und die 2. Ableitung verstärkt die gewünschten Eigenschaften. Zu beachten ist, dass alle Funktionen wegen des Quadrates positive Ergebnisse liefern. α und β kontrollieren das Verhältnis zwischen der 1. und 2. Ableitung und $[]^+$ bedeutet, dass alle negativen Ergebnisse zu Null gesetzt werden.
Für eine positive Linie bei $y = 0$ zeigt Abbildung 6.9, welche Beiträge bei der Verstärkung und Unterdrückung der entsprechenden Eigenschaften die ungeraden und geraden Filter $\mathcal{R}_{o,n}(\vec{r})$ und $\mathcal{R}_{e,n}(\vec{r})$, die Antwortfunktion $\mathcal{M}_n(\vec{r})$ und deren 1. und 2. Ableitung liefern.
Nach der Berechnung von $\mathcal{S}_n(\vec{r})$ und $\mathcal{C}_n(\vec{r})$ ist nun zu entscheiden, ob eine Linie oder Kante vorliegt. Hierzu führt Heitger aus, dass man besonders ausgeprägte Merkmale für die Stärken von Linien bzw. Kanten an einer Stelle $\vec{r}$ erreicht, indem man die 1. Fourier-Frequenz des n-Tupels aller Ergebnisse von $\mathcal{S}_n(\vec{r})$ und $\mathcal{C}_n(\vec{r})$ aller n Richtungen betrachtet. Die Näherung hebt lokale Eigenschaften hervor, die aus einer Richtung resultieren, und vermindert Merkmale, die sich aus verschiedenen Richtungen ergeben. Die Auswertungen von $\mathcal{S}_n(\vec{r})$ und $\mathcal{C}_n(\vec{r})$ bestimmen die Stärke, den Typ und die Orientierung der zugrunde

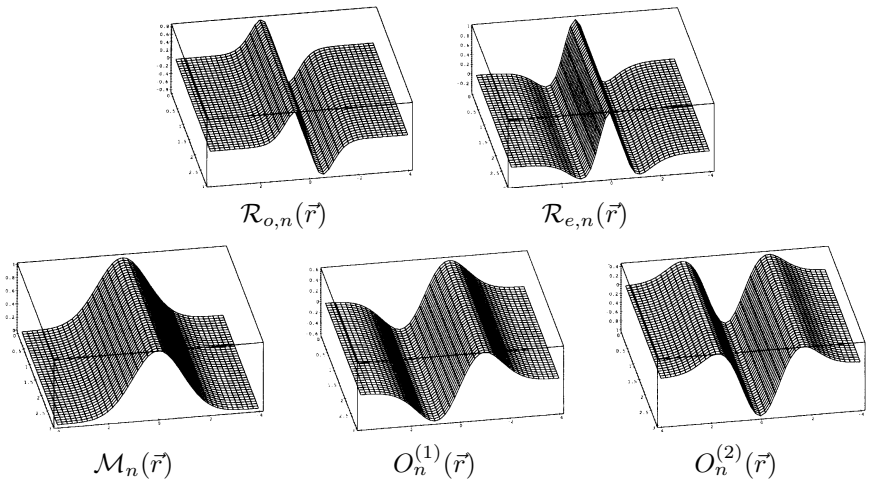


Abbildung 6.9: Graphische Darstellung der Ergebnisse der Filterfunktionen nach Heitger

liegenden Struktur (Kante oder Linie). Die 1. Fourier-Frequenz lässt sich durch ihre reellen und imaginären Koeffizienten präsentieren: $\Re_S(\vec{r})$ und $\Im_S(\vec{r})$ bzw. $\Re_C(\vec{r})$ und $\Im_C(\vec{r})$. Für die Amplituden erhält man für Kanten bzw. Linien:

$$\mathcal{A}_S(\vec{r}) = \sqrt{\Re_S(\vec{r})^2 + \Im_S(\vec{r})^2} \quad \text{bzw.} \quad \mathcal{A}_C(\vec{r}) = \xi\sqrt{\Re_C(\vec{r})^2 + \Im_C(\vec{r})^2} \tag{6.18}$$

Die Stärke des Bildpunktes bei $\vec{r}$ ergibt sich als das Maximum dieser beiden Berechnungen. Weiterhin gilt: Ist $\mathcal{A}_S(\vec{r}) > \mathcal{A}_C(\vec{r})$, so handelt es sich um eine Kante, ansonsten um eine Linie. Durch

$$\theta(\vec{r}) = \frac{1}{2}\tan^{-1}\left(\frac{\Im_S(\vec{r})}{\Re_S(\vec{r})}\right) \quad \text{bzw.} \quad \theta(\vec{r}) = \frac{1}{2}\tan^{-1}\left(\frac{\Im_C(\vec{r})}{\Re_C(\vec{r})}\right) \tag{6.19}$$

wird die lokale Richtung festgelegt, wobei $\theta(\vec{r}) \in [-\frac{\pi}{2}, \frac{\pi}{2}]$ ist.
Zum Schluss wird beim Verfahren von Heitger noch eine eigene Version eines Non-Maxima-Suppression-Verfahrens angewendet, das für Kanten und Linien unterschiedlich ist. Dieses Verfahren funktioniert ähnlich wie das Non-Maxima-Suppression-Verfahren nach Canny. An dem Beispiel in Abbildung 6.10 kann man gut sehen, dass die Kanten im Bild als schwarze und die Linien als weiße Linien dargestellt werden.

3 Vergleichsmaterial

Die hier verwendeten Bilder lassen sich in zwei Gruppen gliedern: Objektbilder und Luftbilder. Hier werden 50 Objektbilder verwendet, die sowohl Objekte im Freien als auch Objekte innerhalb von Räumen zeigen. Außerdem werden 10 Luftbilder benutzt. Die Abbildungen 6.11 und 6.12 zeigen vier Beispiele.

4 Vergleichstechnik

Der Vergleich der Verfahren in [4] basiert auf den in der Medizin üblichen Bewertungskonzepten von *Sensitivität*, *Spezifität* und ROC-Kurven.

4.1 Sensitivität, Spezifität und ROC-Kurven

Das Konzept von *Sensitivität* und *Spezifität* ist in der Medizin verbreitet, um die Güte eines medizinischen Tests zu bewerten. Hierzu wird aus den Häufigkeiten für richtig positive und falsch negative Tests eine Vierfeldertafel (Abb. 6.13) aufgebaut. Es gilt für:
die Sensitivität S_e:

$$S_e = \frac{rp}{rp + fn} = \frac{\text{richtig positiv}}{\text{wirklich positiv}} \tag{6.20}$$

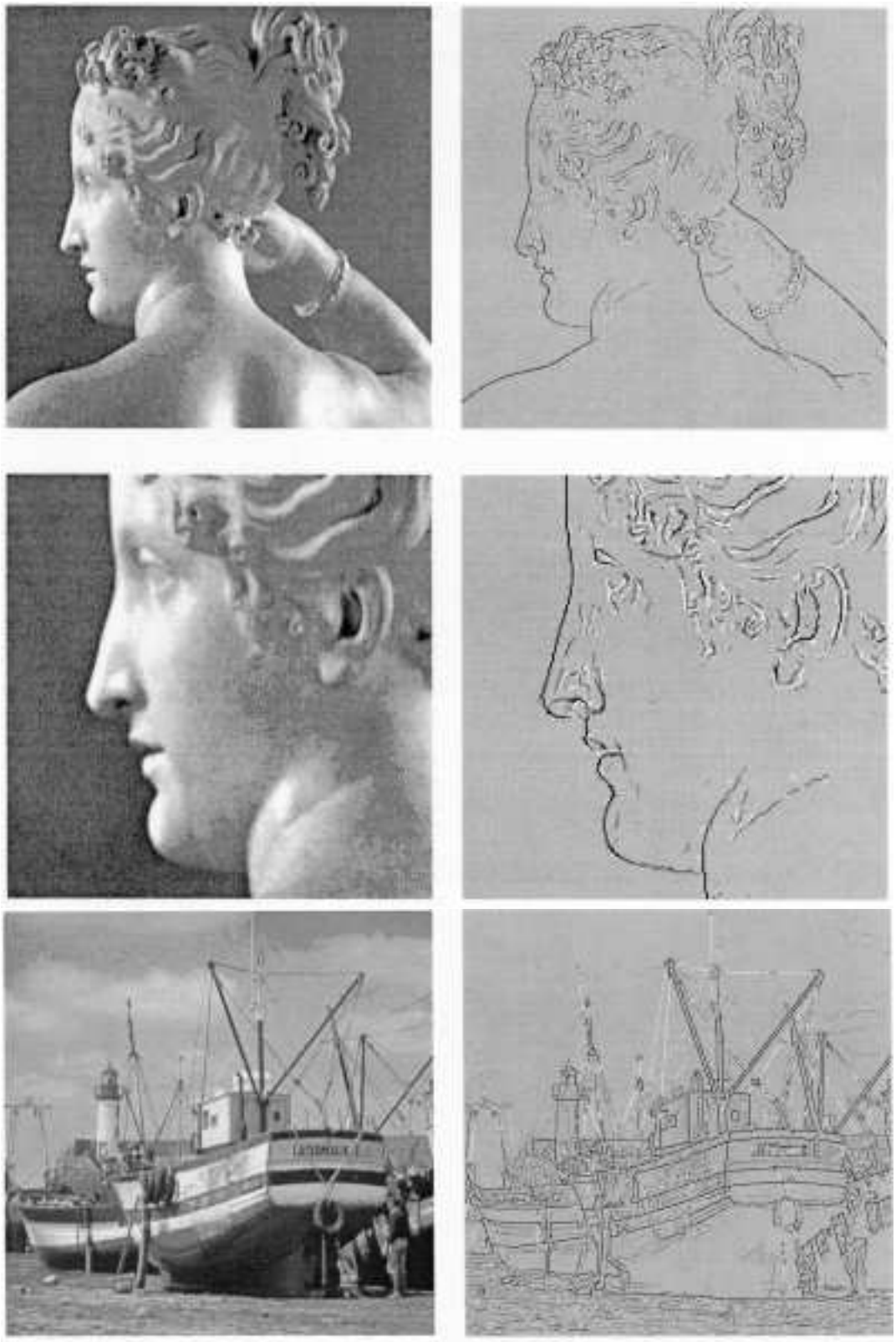


Abbildung 6.10: Beispiel für das Heitger-Verfahren (aus [4]): Linien werden weiß und Kanten schwarz dargestellt. Man sieht jeweils links die Originalbilder und rechts die Bilder, die das Verfahren von Heitger liefert.

und die Spezifität S_p:

$$S_p = \frac{rn}{rn + fp} = \frac{\text{richtig negativ}}{\text{wirklich negativ}} \tag{6.21}$$

Bei nicht-dichotonen Merkmalen ist es nötig, dass man eine Schwelle festlegt, ab der etwas als positiv bzw. negativ angesehen wird. Wenn jetzt die Spezifität und die Sensitivität möglichst hoch sein sollen, muss man die Schwelle so wählen, dass die Anzahl der falsch positiven und falsch negativen Ergebnisse minimiert wird, was jedoch nicht gleichzeitig möglich ist. Wenn man wie in Abbildung 6.14 fp über $fn = \overline{rp}$ aufträgt, ist ein Test besser, wenn die Fläche unter der Kurve kleiner ist. Diese Kurven heißen ROC-Kurven (receiver operating characteristic).

4.2 Ground Truth

Um die Anzahl der richtig detektierten Kantenpunkte bestimmen zu können, benötigt man ein „Referenzbild“, in dem die „wirklichen“ Kanten eingezeichnet sind. Bei dem hier vorgestellten Vergleich werden diese Bilder als *Ground Truth* bezeichnet. Diese *Ground Truth*-Bilder werden per Hand erstellt, indem man die verschiedenen Regionen des Bildes (Abb. 6.15) wie folgt markiert:

- schwarz sind Kanten,
- grau repräsentiert eine Region, in der keine Kanten liegen und
- weiß sind die Bereiche, die nicht interessieren.

Diese Bilder werden manuell von einer Person erstellt. Nach den Aussagen im Paper [1] hat diese Art der Erstellung keinen Einfluss auf das relative Abschneiden der Verfahren. Zu diesem Zweck wurden im Paper 5 Ground-Truth-Bilder von drei verschiedenen Personen erstellt. Für die Erstellung dieser Bilder benötigten diese Personen zwischen 30 und 90 Minuten. Abbildung 6.16 zeigt einige Beispiele hierzu, die verdeutlichen, dass sich die ROC-Kurven zwar absolut verschieben, aber ihre relativen Lagen zueinander beibehalten.

4.3 Art des Vergleichs mit Ground Truth

Nachfolgend wird die Art und Weise vorgestellt, mit der die Bilder mit den *Ground Truth*-Bildern verglichen werden. Es werden nun zwei Zähler „True Positive“ (TP) und „False Positive“ (FP) eingeführt, die wie folgt benutzt werden:

- Wenn ein Kantenpixel des Bildes in eine graue Region fällt, in der keine Kanten liegen sollen, so wird FP um 1 erhöht.
- Liegt das Kantenpixel innerhalb eines Abstandes T_{match} von einer Ground Truth-Kante, so wird TP um 1 erhöht, und dieses Pixel markiert, sodass es nicht mehr für weitere Matchingversuche zur Verfügung steht. Wenn innerhalb von T_{match} mehrere Pixel in Frage kommen, dann wird das Pixel ausgewählt, welches der Ground Truth-Kante am Nächsten liegt.
- Wenn ein Kantenpixel in einen weißen Bereich fällt, der als nicht interessant gekennzeichnet ist und nicht zu einer Ground Truth-Kante gematcht wird, ändern sich weder TP noch FP.

4.4 Trainings ROC-Kurven

Bei einem gegebenen Detektor wird für ein vorliegendes Bild eine Trainings ROC-Kurve ermittelt, indem man den Parameterraum des Detektors abfragt. Falls der Detektor P Parameter besitzt, so enthält der Parameterraum bei zunächst vier vorgegebenen, gleichmäßig verteilen Werten insgesamt 4^P Punkte. Nun wird für jeden Parameterpunkt das Kantenbild mit dem Ground Truth-Bild verglichen und ein Paar (TP,FP) bestimmt. Alle so erhaltenen Paare werden prozentual in ein Koordinatensystem eingetragen. Auf der x-Achse werden die Prozente für die undetektierten Ground Truth-Kanten und auf der y-Achse die Prozente für FP verzeichnet. Hierbei werden die Punkte verbunden, die den Achsen am Nächsten liegen. Anschließend werden die Parameter solange schrittweise verbessert, bis die Änderung der Fläche unter der ROC-Kurve unterhalb von 5% liegt. Auf jeden Fall werden jedoch mindestens zwei Iterationen durchgeführt.

Abbildung 6.17 a) zeigt die zeichnerische Auswertung von 64 Parameterpunkten für das Canny-Verfahren; hierzu wurde das Flugzeugbild (Abb. 6.15) benutzt. Die Abbildungen 6.17 b) - d) stellen zwei Iterationsstufen und die endgültige ROC-Kurve dar.
So erhält man für jedes Bild und Verfahren eine ROC-Kurve. Auf diese Weise bekommt man hier 50 bzw. 10 Trainings ROC-Kurven für die zwei Bildklassen. Diese Kurven spiegeln die Leistung eines Verfahrens wider, wenn man das Verfahren für ein spezielles Bild optimiert.

4.5 Test ROC-Kurven

Die Verfahren werden jetzt jeweils, nachdem sie auf ein Bild optimiert wurden, auf alle anderen Bilder der gleichen Kategorie angewendet. So erhält man zu jedem Verfahren und jedem Bild 49 bzw. 9 Test-Kurven. Es ergeben sich also $50 \cdot 49 = 2450$ bzw. $10 \cdot 9 = 90$ Test-Kurven. Diese geben Auskunft über die Leistung eines Verfahrens, wenn dieses nicht speziell für Bilder, auf die es angewendet wird, optimiert ist.

4.6 Gemittelte Kurven

Da man für jedes Verfahren 50 bzw. 10 Trainings- und 2450 bzw. 90 Test-Kurven erhält, kann man leicht einsehen, dass diese für einen direkten Vergleich ungeeignet sind. Aus diesem Grund bildet man nun für die Trainingskurven zu einem Verfahren eine gemittelte Kurve über alle Bilder einer Kategorie; für die Test-Kurven mittelt man analog. Da man die beiden Kategorien der Bilder getrennt betrachtet, erhält man zu jedem Verfahren 4 Kurven. Die gemittelten Kurven werden erzeugt, indem man für eine bestimmte Menge von TP-Werten die dazugehörigen FP-Werte mittelt und dann den gemittelten Wert in die Kurve einträgt.

5 Ergebnisse des Vergleichs

In Abbildung 6.18 kann man erkennen, dass ein hoher Prozentsatz von TP alleine nicht sehr aussagekräftig ist. Man sieht an diesem Beispiel deutlich, dass bei 90% TP bei beiden Bildern ganz gut festzustellen ist, was für ein Objekt auf dem Bild zu sehen ist. Jedoch ist das Sobel-Bild bei 95,5% TP unbrauchbar, hingegen ist auch bei 95,5% TP das Heitger-Bild noch gut zu erkennen.
Im Weiteren werden neben den gemittelten ROC-Kurven noch zwei Merkmale betrachtet: Die Häufigkeit, in der eine hohe Prozentzahl von Ground Truth-Kanten gematcht wird, und die Übereinstimmung von detektierten Kanten.

5.1 Gemittelte Kurven

In den Diagrammen (Abb. 17-19) sind zur Übersicht alle 11 Verfahren aufgeführt, die in [1] verglichen wurden. In Abbildung 6.19 sind die Trainings ROC-Kurven für die 11 Verfahren dargestellt, wobei die Verfahren in zwei Graphiken getrennt wurden, um die Übersichtlichkeit zu erhalten. Dabei ist das Verfahren von Canny in beiden Graphiken eingetragen, um einen Vergleich zwischen den Graphiken herzustellen. In Abbildung 6.20 sind die Test ROC-Kurven dargestellt. In beiden Darstellungen kann man erkennen, dass von den drei hier vorgestellten Verfahren das von Heitger die besten Ergebnisse liefert, und das Verfahren von Canny schneidet besser ab als das von Sobel.

5.2 Häufigkeit von Ground Truth-Matching

Für die Auswahl bzw. Bewertung eines Detektors ist es wichtig zu wissen, in welcher Häufigkeit eine hohe Prozentzahl von Ground Truth-Kanten bei einem annehmbaren Wert von FP detektiert wird. Hierzu ist in Abbildung 6.21 für die verschiedenen Verfahren die Anzahl von Bildern in Abhängigkeit von den Prozenten der nichtdetektierten Ground Truth-Kanten aufgetragen. Bei der Analyse der Graphen stellt man fest, dass alle Verfahren bei einem akzeptablen Wert von FP 84% der Ground Truth-Kanten matchen. Die größten Probleme bei der Übereinstimmung von Ground Truth-Kanten mit dem nach dem entsprechenden Verfahren gelieferten Bild zeigen sich bei Sobel. Ausgezeichnete Werte werden dagegen mit 99,75% nach der von Heitger vorgeschlagenen Methode erreicht.

5.3 Gleichheit von Kanten

Es ist sehr interessant zu betrachten, inwieweit zwei Verfahren die selben Ground Truth-Kanten matchen. Für den Fall, dass zwei Verfahren viele unterschiedliche Kanten matchen, könnte man darüber nachdenken, aus den beiden Verfahren ein kombiniertes zu entwerfen, das dann mehr Kanten erkennt.
Jedes der hier verglichenen Verfahren matcht 80% der Kanten, sodass zumindest 60% der Kanten von beiden Verfahren gemeinsam erkannt werden. Für die restlichen 40% können 0%-ige bis 100%-ige Übereinstimmung erzielt werden. Die Ergebnisse für die drei hier näher betrachteten Verfahren sind in Abbildung 6.22 dargestellt. Aus diesen Tabellen kann man entnehmen, dass die Übereinstimmungen hoch sind. Deshalb ist es auch nicht sinnvoll, einen komplexen Detektor aus zwei anderen Detektoren zu entwerfen.

6 Fazit

Das Verfahren zum Vergleichen liefert meiner Meinung nach ein stimmiges Gesamtbild ab. An einigen Stellen habe ich aber noch Schwierigkeiten, mich der Meinung der Autoren anzuschließen.
Zuerst scheint mir die Auswahl der Bilder recht eingeschränkt zu sein: ich verstehe nicht, wie man gerade auf 50 bzw. 10 Bilder gekommen ist. Desweiteren ist die Art der Bilder für eine Anwendung in der Medizin auch nicht unbedingt repräsentativ, da man in der Medizin eigentlich immer ziemlich verrauschte Bilder (z.B. Röntgen- oder Sonographiebilder) hat. Deswegen scheint mir eine unmittelbare Übertragung der Ergebnisse auf die Kantenerkennung in der Medizin recht schwierig.
Überdies finde ich, dass der Einfluss der Ground Truth-Bilder eigentlich größer sein müsste, als hier dargestellt. Die Bewertung der ROC-Kurven, die über die 5 Ground Truth-Bilder von 3 Testpersonen erstellt wurden, würde glaubhafter und aussagekräftiger erscheinen, wenn man sowohl die Anzahl der Beteiligten als auch die Zahl der Ground Truth-Bilder erhöhen würde. Außerdem hätte man dies für verschiedene Originale testen sollen. So jedenfalls erscheint mir die Auswertung wenig überzeugend.

Literaturverzeichnis

[1] Bowyer K, Kranenburg C und Dougherty S: Edge Detector Evaluation Using Empirical ROC Curves. Computer Vision and Understanding 84, Volume 1, Seiten 77-103 (2001)

[2] Block B: Der Sono-Trainer. Thieme Verlag, Stuttgart-New York, 2000

[3] Steinbrecher R: Bildverarbeitung in der Praxis. Oldenbourg, München, 1993

[4] Heitger F: Feature Detection Using Suppression and Enhancement, TR 163, Image Science Lab, ETH-Zürich, 1995

Abbildung 6.11: Beispiele für Luftbilder

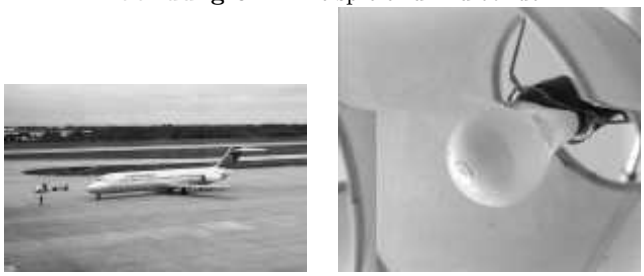


Abbildung 6.12: Beispiele für Objektbilder

	„wirklich“ positiv	„wirklich“ negativ	Summe
Test positiv	richtig positiv (rp)	falsch positiv (fp)	$(rp + fp)$
Test negativ	falsch negativ (fn)	richtig negativ (rn)	(fn+rn)
Summe	$(rp + fn)$	$(fp + rn)$	

Abbildung 6.13: Vierfeldertafel

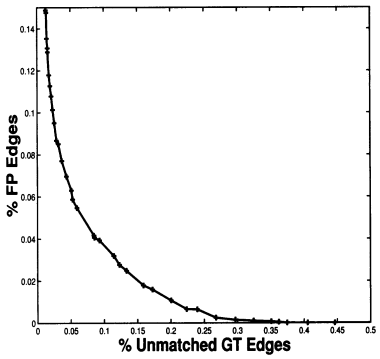


Abbildung 6.14: Beispiel einer ROC-Kurve, fp aufgetragen über $fn = \overline{rp}$

Abbildung 6.15: Beispiel für ein Bild und das dazugehörige Ground Truth-Bild

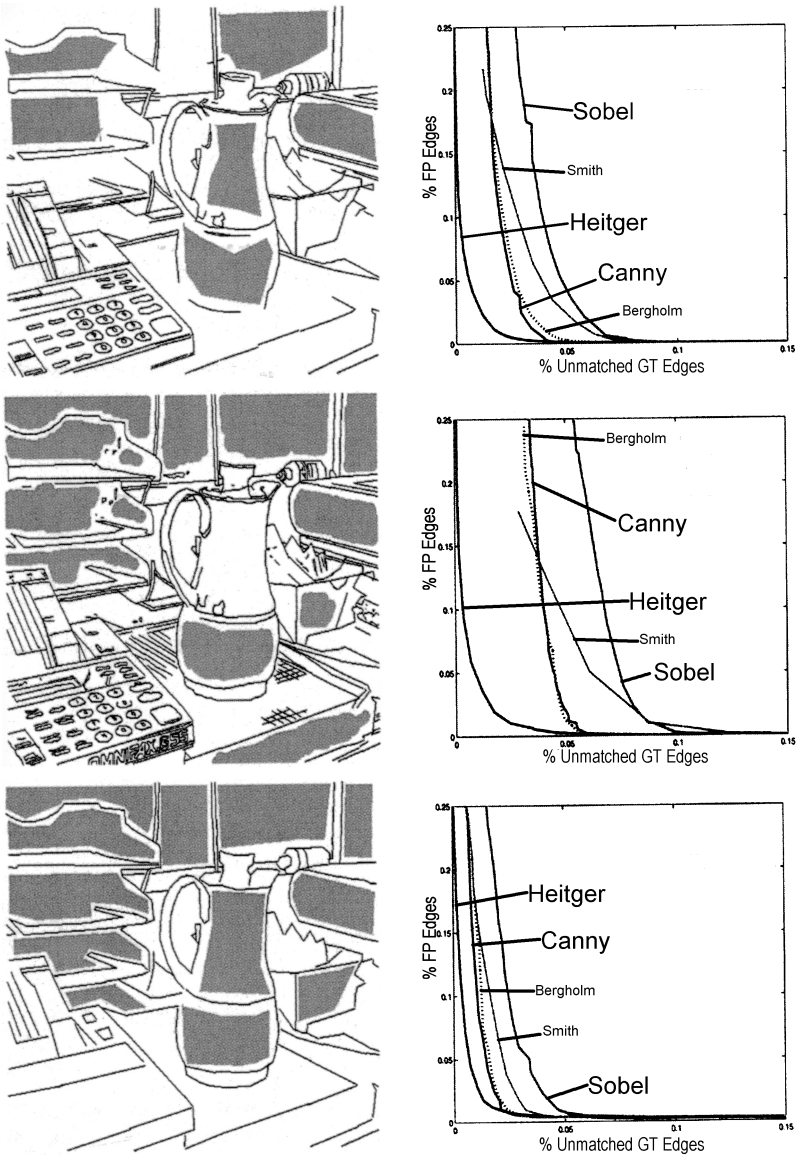


Abbildung 6.16: Verschiedene Ground Truth-Bilder und die dazugehörigen Auswertungskurven

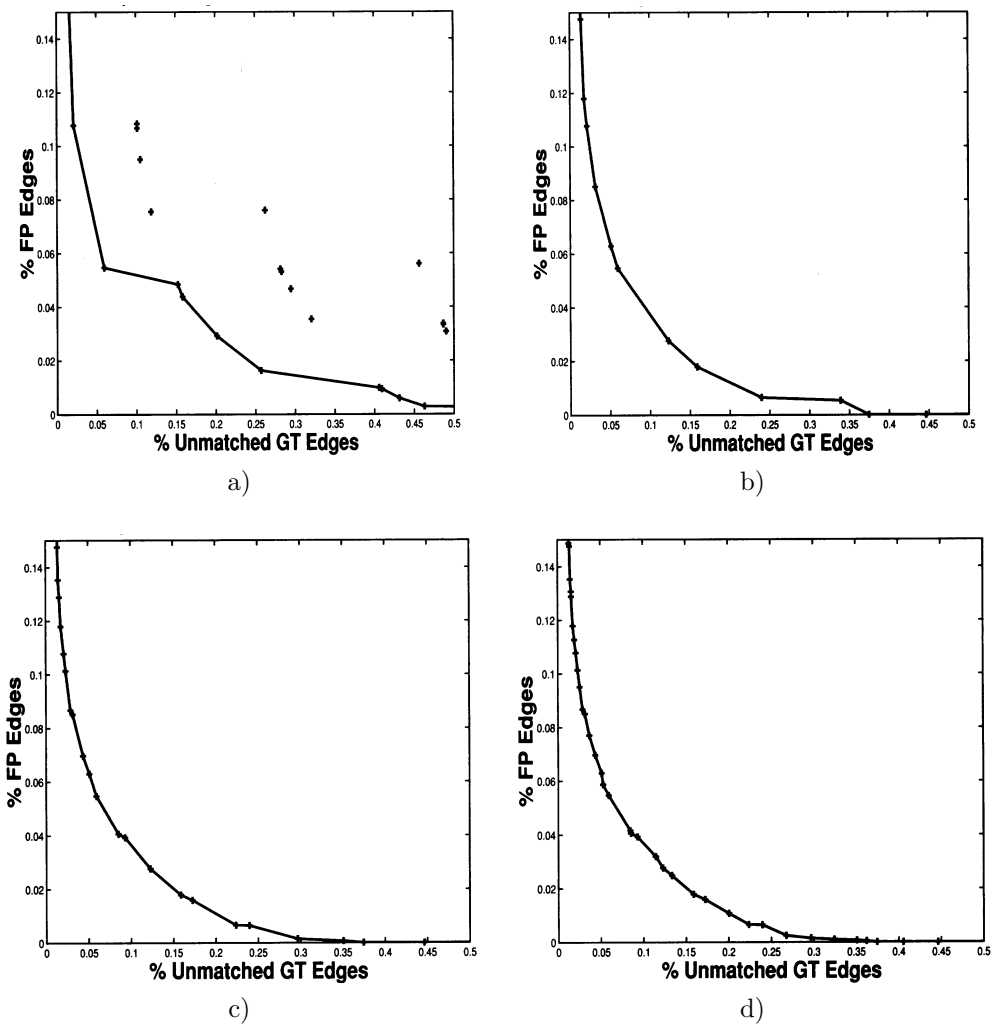


Abbildung 6.17: Der Parameterraum für das Bild aus Abb. 6.15 wird schrittweise für das Canny-Verfahren angepasst: a) Anfangszustand, b) 2. Iteration, c) 3. Iteration, d) endgültige ROC-Kurve.

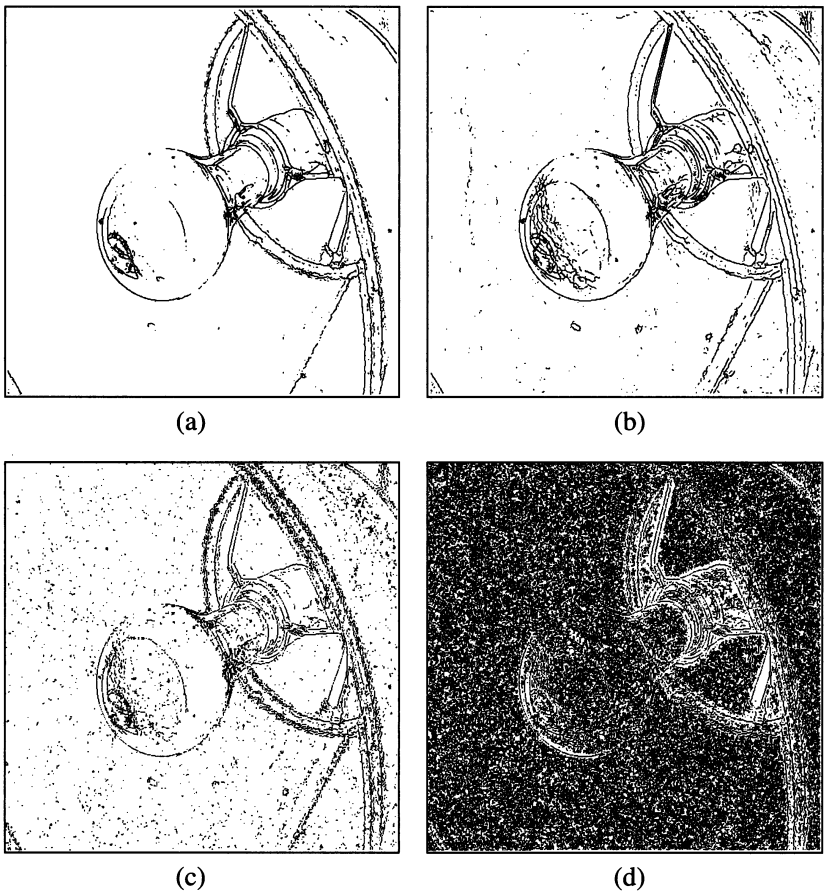


Abbildung 6.18: Kantenbilder für Heitger und Sobel: (a) Heitger bei 90% TP; (b) Heitger bei 95,5% TP; (c) Sobel bei 90% TP; (d) Sobel bei 95,5% TP.

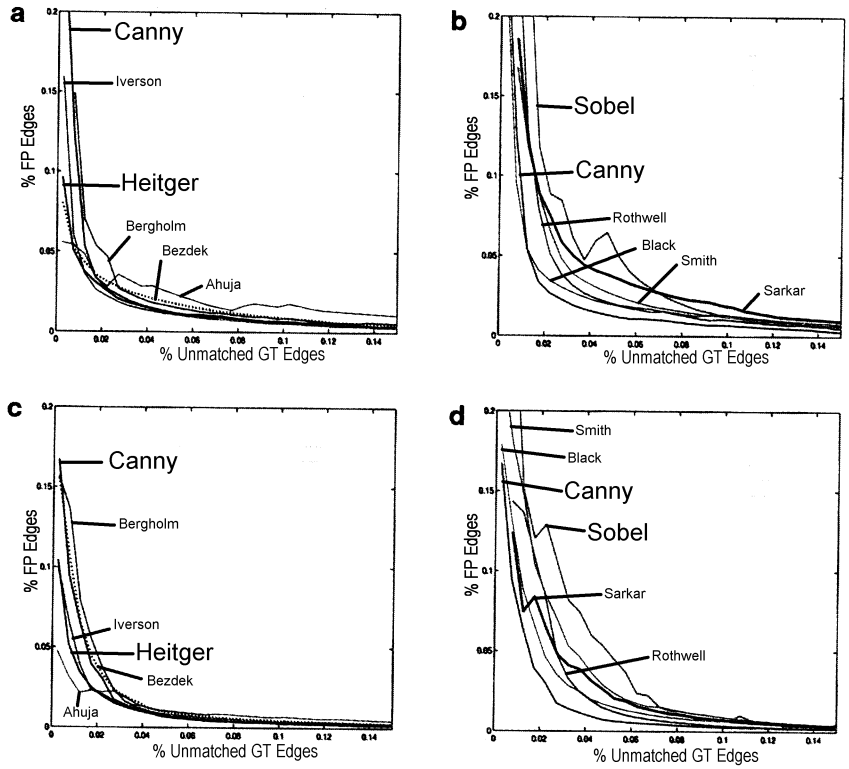


Abbildung 6.19: Gemittelte Trainingskurven für Objekt- und Luftbilder: (a) und (b) Luftbilder; (c) und (d) Objektbilder

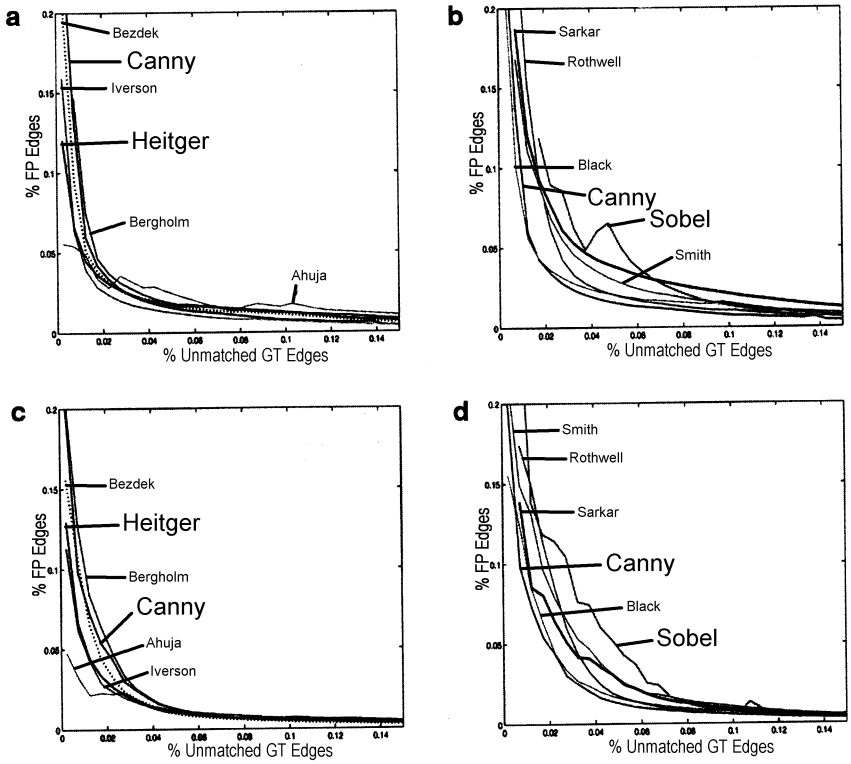


Abbildung 6.20: Gemittelte Testkurven für Objekt- und Luftbilder: (a) und (b) Luftbilder; (c) und (d) Objektbilder

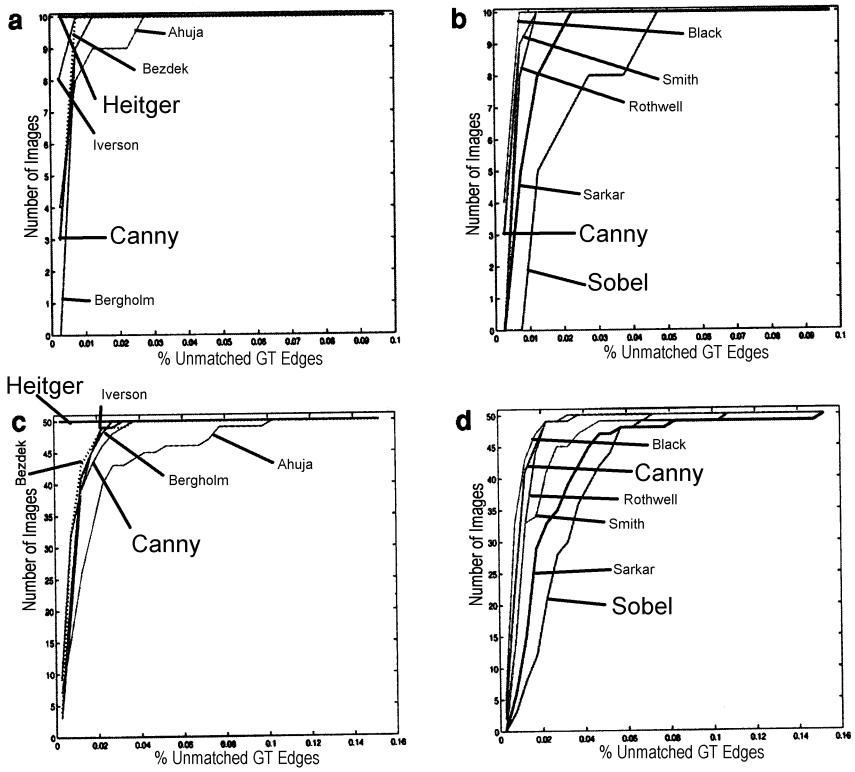


Abbildung 6.21: Häufigkeit von gematchten GT-Kanten: (a) und (b) Luftbilder; (c) und (d) Objektbilder

	Canny	Heitger
Canny	*	-
Heitger	74%	*
Sobel	61%	65%

50 Objektbilder

	Canny	Heitger
Canny	*	-
Heitger	79%	*
Sobel	64%	59%

10 Luftbilder

Abbildung 6.22: Freie Übereinstimmung in Prozent

Morphologische Multiskalen Bildanalyse auf der Basis von Levelings

von Jörg Freudenstein

Betreuer: Christian Thies

Inhaltsverzeichnis

Zusammenfassung

In dieser Ausarbeitung wird der Aufsatz aus [1] vorgestellt, in dem Levelings eingeführt werden. Für die Identifizierung von verschiedenen Informationen aus ein und demgleichen (medizinischen) Bild eignet sich besonders die Anwendung von Multiskalen. Dabei wird für jedes zu betrachtende Objekt auf dem Bild eine andere, jeweils geeignete Auflösungsstufe gewählt. Levelings stellen dann ein ideales Hilfsmittel in der Kette der morphologischen Bildsegmentierung dar. Sie ermöglichen eine totale Partitionierung eines Bildes in Flat Zones *und Übergangsbereiche, ohne die Lage dieser Bereiche zu verändern oder sie unscharf zu gestalten. Für die weitergehende Objekterkennung ermöglichen Levelings die deutliche Visualisierung von vormals unscharfen Objektkanten.*

Keywords: Morphologie, Levelings, Skala, Erosion, Dilatation

1 Einleitung

In der Medizinischen Bildverarbeitung entstehen die betrachteten Bilder fast ausschließlich durch Messungen technischer Geräte. Dabei werden zwar inhaltlich alle relevanten Details (wie z.B. Kanten, Objekte, Extrema, Regionen, Strukturen ...) abgebildet, diese sind aber für das menschliche Auge nicht oder nicht unmittelbar zu erkennen. Das linke Bild in Abbildung 7.1 zeigt einen solchen Fall. Mit Hilfe ein-

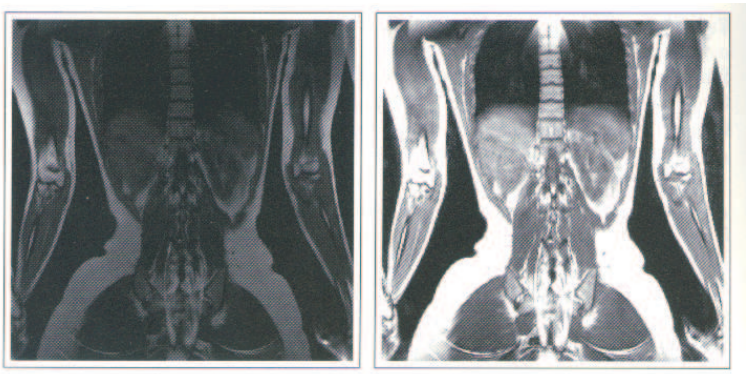


Abbildung 7.1: Grauwert-Transformation: 256 Töne links, 569 Grauwerte rechts

facher Grauwert-Transformationen (wie in Abb. 7.1) oder *morphologsichen Operationen* (siehe Kapitel 2.2) versucht man nun, den Bildgehalt für seine Bedürfnisse zu optimieren. Diese Bedürfnisse können je nach Situation völlig unterschiedlich sein: So kann man globale Zusammenhänge in einer gröberen Auflösungsstufe besser erkennen, in welcher (redundante oder störende) Informationen reduziert und die Kernaussage besser fokussiert wurden; Für Details eignet sich entsprechend eine feinere Auflösungsstufe besser. Generell ist jedoch die *Kausalität* und *Orts-Invarianz* von Erscheinungen im Bild wichtig: Wenn die Wirbel in Abb. 7.1 zu sehen sind, dann müssen sie sich "innerhalb" der Wirbelsäule befinden; Für andere Zwecke genügt vielleicht die bloße Darstellung der Wirbelsäule in Form einer senkrechten Linie.

2 Die Erzeugung von Skalenräumen

Verschiedene Auflösungsstufen (*Skalen*) von dem gleichen (medizinischen) Bild ergeben in ihrer Gesamtheit einen Skalenraum S_n. Generell entstehen die Skalen s durch die Anwendung eines Filters ϕ aus dem ursprünglichen Bild u oder aus einer bereits erzeugten Skala:

$$S_n = \left\{ s_i \,\middle|\, \begin{array}{lcll} s_i & = & \phi\ s_{i-1} & \text{für } i = \{1 \dots n\} \\ s_o & = & u & \end{array} \right\}$$

Jedoch dürfen Folgende Einwirkungen die Ergebnisse der Informations-Extraktion nicht beeinflussen:

- Translation (Verschiebung)

- Drehung
- Beleuchtung

2.1 Filter

Als Filter bezeichnet man beliebig geartete, nach mathematischen Grundsätzen vorgenommene Transformation eines Bildes. Der ursprüngliche Begriff und die Anwendung des Filters stammt aus der Signaltheorie. Im Falle der Bildverarbeiutng dienen sie in der Regel drei verschiedenen Zielen:

1. Bildverbesserung
   Hierbei werden z.B. Farb- oder Bildfehler beseitigt oder die Helligkeit korrigiert. Bildelemente werden entzerrt oder hervorgehoben oder das Rauschen verringert.

2. Quantitative Auswertung
   Bei der Quantitativen Auswertung werden z.B. Tiefeninformationen aus Stereobildern extrahiert oder die Bewegung in Bildfolgen bestimmt.

3. Bildgestaltung und Bildmanipulation
   In diesen Bereich fallen vor allem die Anwendung von Effekten, das Retuschieren oder die Kombination mehrerer Bilder.

Zu unterscheiden sind ferner

- *globale* Filter
  Hier beeinflussen alle Punkte eines Bildes die Änderung eines Bildpunktes (z.B. Fourier-Transformation)
- *lokale* Filter
  Nur unmittelbare Nachbarn beeinflussen die Änderung eines Pixels
- sowie homogene Punktoperatoren
  der betrachtete Bildpunkt wird ohne Beeinflussung durch andere verändert (z.B. bei der Grauwertmanipulation oder Normierungen)

Die praktische Durchführung lokaler Bildoperationen gelingt mit Hilfe von Filtermasken. Filtermasken ähneln Bildmatritzen, sind quadratisch und haben immer eine ungerade Zeilen- und Spaltenzahl. Die kleinstmögliche Maske besteht aus je drei Zeilen und Spalten. Die Bildmatrix wird mit der Maske verknüpft, indem die Maske über die Bildmatrix geschoben wird. Dabei werden jedes Maskenelement und der darunterliegende Bildpunkt miteinander verrechnet, danach wird die Maske weiterverschoben.

Lineare Filter

In der herkömmlichen, frequenzorientierten Bildverarbeitung werden vor allem *lineare Filter* eingesetzt (z.B. Hochpassfilter, Tiefpassfilter). Ein linearer Filter ϕ, angewendet auf die Funktionen f und g hat die Form

$$\phi\left(a_1 f + a_2 g\right) = a_1\,\phi\left(f\right) + a_2\,\phi\left(g\right) \tag{7.1}$$

(mit $a_1, a_2 \in \mathbb{R}$). Die Wirkung von linearen Filtern kann je nach dem Filtertyp sehr unterschiedlich sein. Man kann damit

- relevante Bildinhalte hervorheben, z.B. Kanten
- Störungen beseitigen, z.T. auch Rauschen
- Die visuelle Beurteilung erleichtern, z.B. durch Erhöhen der Bildschärfe

Trotzdem können lineare Filter nicht den lokalen Kontext in Betracht ziehen, also prinzipiell nicht besonders gut zwischen interessierender Information und Rauschen unterscheiden. Insbesondere liefern sie auch keine stabile Ortsinformation; dies ist einer der größten Nachteile, der, wie später zu sehen, durch Levelings "behoben" werden kann.

2.2 Morphologische Filter

Aufgrund ihrer nichtlinearen Eigenschaften bieten *morphologische Filter* eine bessere Grundlage. Bei ihrem Einsatz lassen sich Objekte leichter segmentieren; Die Kausalität von Erscheinungen im Bild wird nun berücksichtigt (vgl. dazu nochmals Abb. 7.1): Wenn ein Wirbel in einer groben Auflösung im Bild zu sehen ist, dann muß er auch in einer feineren Auflösungsstufe bereits an dieser Stelle sichtbar gewesen sein (siehe auch Abschnitt 3; zur genauen Definition eines morphologischen Filters siehe Kap. 4.4).

Der Begriff Morphologie

Der Begriff Morphologie (Formenlehre) wird in der Biologie zur Beschreibung der Formen von Pflanzen und Tieren, insbesondere in Hinblick auf ihre Skelette (siehe Kap. 2.3), verwendet. Es lassen sich aber auch geometrisch-mathematische Operationen angeben, mittels derer sich eine geometrische Form z.B. auf ihren Kern, ihr Skelett, zurückführen bzw. aus diesem weitgehend wieder rekonstruieren läßt. Hiermit bietet sich die Möglichkeit zu einer Modifikation von Formen. Wenn man das medizinische Bild als Funktionen mit variablen Amplitudenwerten (Grauwertgebirge) ansieht, können morphologische Filter aber vor allem zur nichtlinearen Kontrastverstärkung oder -elimination, zur Segmentierung (siehe Kap. 2.4) und zur Auffindung von Objektgrenzen in Bildsignalen eingesetzt werden.

Erosion, Dilatation

Die beiden Basisoperatoren in der morphologischen Filterung sind die dualen Filter *Erosion* (Verkleinerung, Ausdünnung) und *Dilatation* (Vergrößerung, Aufblähung). Abbildung 7.2 zeigt den binären Fall: Zu

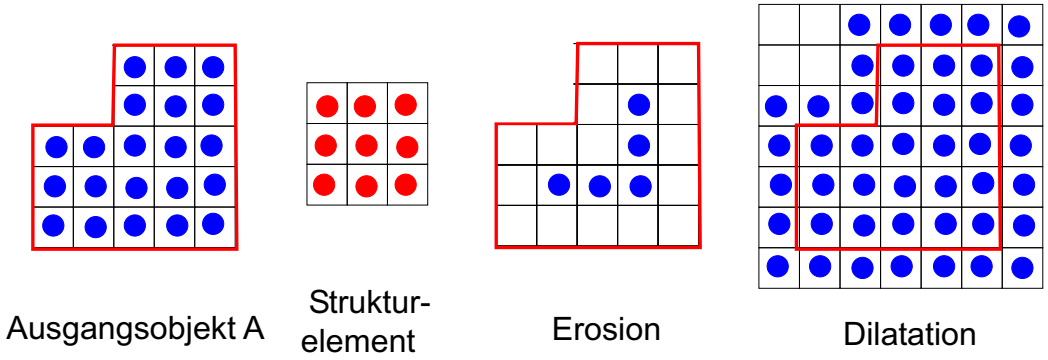


Abbildung 7.2: Morphologische Operationen: Erosion und Dilatation

sehen sind das Ausgangsbild A und ein Strukturelement τ der Größe 3×3. Dieses Strukturelement wird - ähnlich wie eine Filtermaske bei der linearen Filterung - über dem Objekt A verschoben, wobei jeweils die Position des Mittelpunktes des Strukturelements als Referenz dient. Als Ausgangsform der Erosion ergibt sich nun ein "ausgedünntes Objekt", bei welchem nur diejenigen Punkte übrigbehalten werden, an denen alle Punkte des Strukturelements mit einem Punkt des ursprünglichen Objektes zusammenfallen. Umgekehrt ergibt sich durch Dilatation ein "aufgeblähtes Objekt", bei dem diejenigen Punkte hinzukommen, an denen mindestens ein Punkt des Strukturelements mit einem Punkt des ursprünglichen Objektes zusammenfiel:

Erosion:

$$E_\tau(f(x)) = \min\left(f(x+i)\,|\,i \in \tau\right) \tag{7.2}$$

Dilatation:

$$D_\tau(f(x)) = \max\left(f(x+i)\,|\,i \in \tau\right) \tag{7.3}$$

Die Verfahren sind invers zueinander und verlustbehaftet, in der Regel kann also nicht wieder das Ausgangsbild hergestellt werden. Erosion und Dilatation können auch abwechselnd auf ein Bild angewendet werden. Dabei nennt man die Ausführung von Erosion $\Longrightarrow$ Dilatation *Opening*, die Anwendung Dilatation $\Longrightarrow$ Erosion hingegen *Closing*. Bei der Anwendung von Openings werden in der Regel hervorstehende Bildbereiche "abgeschnitten", beim Closing fallen ggf. vorhandene "Einstülpungen" weg. Openings bzw. Closings sind *idempotente* Filter, d.h. die mehrmalige Anwendung des Filters verändert

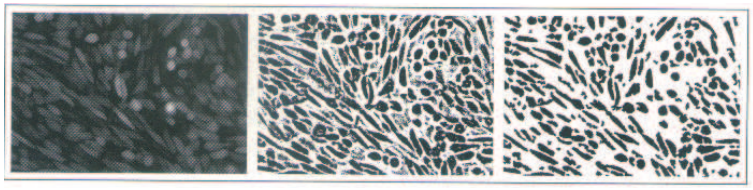


Abbildung 7.3: Anwendung des Opening-Filters

das Ergebnis nicht mehr. Bereits nach der ersten Anwendung eines Openings oder Closings ist also der sog. *Fixpunkt* erreicht. Es gilt daher: $E_\tau, D_\tau \in \{\, \phi \,|\, \phi\ \phi = \phi \,\}$.
In Abbildung 7.3 wurde das Ausgangsbild (links) zunächst binarisiert (mitte) und dann durch eine Opening-Operation "geöffnet" (rechts). So wurden die meisten Zellen zu isolierten Objekten, die weiteranalysiert werden können.

2.3 Skelettierung

Eine wichtige Operation ist auch die Skelettierung. Sie ist von Bedeutung, wenn einzelne Unregelmäßigkeiten nicht von Interesse sind, sondern allein die innere Form des Abgebildeten nutzbringend ist. Eine einheitliche Definition für Skelette gibt es nicht, aber man kann sich auf folgende grundlegende Forderungen einigen ([2]):

- ein Skelett muß aus Linien von der Breite eines Pixels bestehen
- zusammenhängende Objekte müssen nach der Skelettierung auch noch zusammenhängen
- die Skelettlinien müssen ungefähr in der Mitte des ursprünglichen Objektes verlaufen, d.h. sie befinden sich jeweils gleich weit vom Rand das Ausgangsobjektes entfernt
- Endpunkte dürfen nicht entfernt werden
- topologische Eigenschaften, wie z.B. Löcher, bleiben erhalten
- es dürfen keine zusätzlichen Verzweigungen entstehen

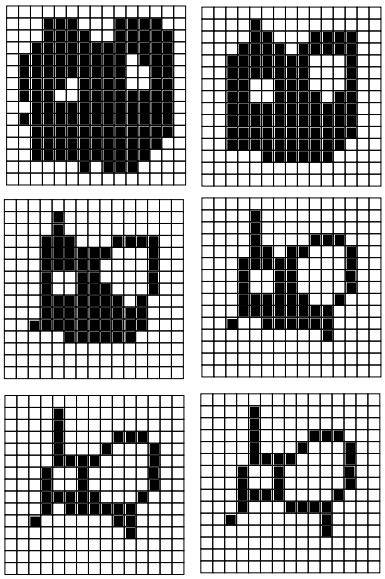


Abbildung 7.4: Skelettierung

Abbildung 7.4 zeigt fünf Schritte bis zur Skelettierung des Ausgangsmotives oben rechts. Verwendet wurde der sog. HIT-AND-MISS-Algorithmus, der eine Folge von variierenden Strukturelementen benützt.

2.4 Segmentierung

Die Segmentierung eines Bildes bedeutet die Zusammenfassung “inhaltlich zusammengehöriger Regionen” (aus [2]). Besonders bei diesem Punkt zeigt sich, daß es sinnvoll ist, mit verschiedenen Auflösungsstufen zu arbeiten, je nachdem, welches Ziel man verfolgt: Gröbere Skalen sind für weiträumigere Regionen interessant, während feinere Skalen mehr detaillierte Informationen enthalten. Generell gibt es für die Segmentierung verschiedene Ansätze:

1. Die Ermittlung der gleichen Grauwertstufen. Dies ist aber ein sehr einfaches Schwellwert- Verfahren, daß nur bei rein punktorientierten Bildern sinnvoll ist.
2. Das kantenorientierte Verfahren. Häufig lassen sich Objekte über ein “Abtasten ihrer Außenkante” am besten erkennen.
3. Ermittlung der Regionen über “zusammenhängende Pixel”. Wie zusammenhängend definiert ist, kann unterschiedlich sein. *Einfach zusammenhängend* ist eine Region, wenn alle Pixel mindestens direkt rechts, links, oben oder unten einen Nachbarn in dieser Region haben.
4. Die Einbeziehung von Texturen. Durch den (schwierigen) Vergleich von Oberflächenstrukturen können ebenfalls Regionen voneinander unterschieden werden.

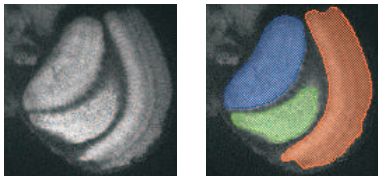


Abbildung 7.5: Segmentierung

Die Abbildung 7.5 zeigt auf der rechten Seite 3 segmentierte Regionen (aus [3]).
In der Praxis wird die Segmentierung häufig aus prozeßtechnischer Sicht betrachtet: Für den Anwender soll die Segmentierung (und anschließende Klassifizierung) ein bestimmtes, fest vorgegebenes Objekt im Bild lokalisieren (z.B. die Niere in einem Ultraschallbild). Mit den sog. *Levelings* wird aber ein etwas anderer Ansatz verfolgt: Die Levelings bilden eine weitere, neu eingeschobene Abstraktionsschicht. Sie sind somit nur *ein Teil* eines Segmentierungsverfahrens. In dieser Vorbereitung zur Segmentierung werden *alle* im Bild vorhandenen Objekte vorbehandelt, unabhängig davon, ob der Anwender anschließend nach ihnen sucht oder nicht (die Anfragen des Anwenders (”Wo ist die Niere?“ ... Leber? etc.) können somit anschließend wesentlich schneller beantwortet werden; dieser Schritt zählt dann aber nicht mehr zu den Levelings).

3 Levelings

3.1 Beschaffenheit von Levelings

Levelings sind ein starker morphologischer Filter (siehe Kap. 4.4). Sie bilden ein wesentliches Hilfsmittel zur Partitionierung von Bildern und sind somit ein Teil der Segmentierung. Es handelt sich um eine weitere Abstraktionsschicht, die in die Kette der morphologischen Bildanalyse eingefügt werden kann. Mit Hilfe von Levelings werden Kanten sowie regionale Extrema verstärkt. Im Gegensatz zu anderen verfahren bleiben diese Maxima und Minima aber ortsinvariant.
Für die hier betrachteten Multiskalenräume, die mit Hilfe von *Levelings* erzeugt werden sollen, werden die Voraussetzungen für die Skalentransformation noch enger gefaßt:

- Die Transformation soll zur Detailreduzierung im Bild führen, d.h. die Änderungen brauchen nicht mehr reversibel zu sein
- Es soll das Maximum-Prinzip gelten: In der gröberen Skala ist der maximale Helligkeitswert höchstens so groß wie in der feineren Skala, der minimale Helligkeitswert mindestens so groß wie in der feineren Skala [4].

- Gegebenheiten in der gröberen Skala müssen kausal von Gegebenheiten in der feineren Skala abhängen [5].
- In den gröberen Skalen dürfen keine neuen Strukturen und neuen regionalen Extrema enstehen [6].

3.2 Flat-Zones

In einen Bild (dargestellt durch eine Funktion f) beschreibt eine *Flat-Zone* eine Menge von benachbarten Pixeln mit demselben Grauwert. Eine *R-Flat-Zone* (oder *quasi-Flat-Zone*) ist eine Menge benachbarter Pixel, die jeweils paarweise einer Relation R genügen: $f_p \; R \; f_q$ für alle zusammenhängenden benachbarten Pixel (p, q).
Wählt man folgende Relation R : $f_p \approx f_q$: $|f_p - f_q| \leq \lambda$, so nennt man das Ergebnis eine quasi-Flat-Zone mit maximaler Neigung λ.

3.3 Monotone Abbildung

Eine Skala g ist eine *monotone Abbildung* einer Skala f genau dann wenn für alle benachbarten Pixel (p, q) gilt: $g_p > g_q \Rightarrow f_p > f_q$. Eine monotone Abbildung erzeugt keine neuen regionalen Minima oder Maxima: Ist z.B. g_i ein Minimum, haben alle seine Nachbarn eine größere Amplitude. Nach der Voraussetzung sind dann aber auch alle Nachbarn von f_i größer, so daß das Minimum in der Quellskala bereits existierte (analog für Maximum).

3.4 Definition von Levelings

Ein Spezialfall der Monotonen Abbildung ist nun das Leveling:

Definition 1 (Levelings) *Eine Skala g ist ein* Leveling *einer Skala f genau dann wenn für alle Nachbarpixel (p, q) gilt:* $g_p > g_q \Rightarrow f_p \geq g_p$ *und* $g_q \geq f_q$

Dabei gilt:

- konstante Funktionen sind ein Leveling jeder anderen Funktion.
- $g_1 \wedge g_2$ sowie $g_1 \vee g_2$ sind Levelings von f, falls g_1 und g_2 für sich genommen bereits Levelings von f sind.
- Die Relation $\{g$ ist ein Leveling von $f\}$ ist eine Preorder (reflexiv und transitiv).
- falls g ein Leveling von f ist (außer konstanten Funktionen) und f ein Leveling von g, dann ist $f = g$.

Die Definition der Levelings kann aufgespalten werden in zwei weniger strenge Teile:

Definition 2 (Lower-Leveling) *Eine Funktion g ist ein* Lower-Leveling *einer Funktion f genau dann wenn für alle benachbarten Pixel (p, q) gilt: $g_p > g_q \Rightarrow g_q \geq f_q$*

Definition 3 (Upper-Leveling) *Eine Funktion g ist ein* Upper-Leveling *einer Funktion f genau dann wenn für alle benachbarten Pixel (p, q) gilt: $g_p > g_q \Rightarrow g_p \leq f_p$*

Damit ist eine Funktion g ist ein *Leveling* einer Funktion f genau dann, wenn sie sowohl ein Lower-Leveling als auch ein Upper-Leveling der Funktion f ist.

Falls (p, q) benachbarte Pixel sind, so gilt ferner

$$\left.\begin{matrix} g_q > f_q \\ g_p > f_p \end{matrix}\right\} \Rightarrow g_p = g_q \quad \text{falls } g \text{ ein Upper-Leveling von } f \text{ ist.} \qquad \left.\begin{matrix} g_q < f_q \\ g_p < f_p \end{matrix}\right\} \Rightarrow g_p = g_q \quad \text{falls } g \text{ ein Lower-Leveling von } f \text{ ist.} \tag{7.4}$$

An allem Stellen, an denen g größer bzw. kleiner als f ist haben sich also Flat-Zones herausgebildet. Wählt man $\delta_p g$ als das Maximum der Amplituden aller Nachbarn von p in g (p inbegriffen) bzw. $\varepsilon_p g$

als ein solches Minimum, dann kann man die Bedingungen für Lower- bzw. Upper-Levelings wie folgt umformulieren:

Lower-Leveling nach Def. 2	$g_p > g_q$	$\Rightarrow$	$g_q \geq f_q$	(7.5)
mit Hilfe von δ	$\delta_q g > g_q$	$\Rightarrow$	$g_q \geq f_q$	(7.6)
boolsche Umformung	$g_q \geq \delta_q g$	*oder*	$g_q \geq f_q$	(7.7)
oder	g_q	$\geq$	$\min(\delta_q g\ ,\ f_q)$	(7.8)
und analog für Upper-Levelings	$g_p > g_q$	$\Rightarrow$	$g_p \leq f_p$	(7.9)
	$g_p > \varepsilon_p g$	$\Rightarrow$	$g_p \leq f_p$	(7.10)
	$g_p \leq \varepsilon_p g$	*oder*	$g_p \leq f_p$	(7.11)
	g_p	$\leq$	$\max(\varepsilon_p g\ ,\ f_p)$	(7.12)

Mit den Gleichungen (7.8) und (7.12) lassen sich Levelings nun alternativ definieren: Eine Funktion g ist ein Leveling einer Funktion f genau dann, wenn

$$\min(f\ ,\ \delta g) \quad \leq \quad g \quad \leq \quad \max(f\ ,\ \varepsilon g) \tag{7.13}$$

4 Ein Algorithmus zum Erzeugen von Levelings

Im folgenden wird ein sehr einfacher Algorithmus eingeführt, mit dem man zu einer beliebigen Funktion f (*Referenzfunktion*) ein Leveling erzeugen kann. Man bedient sich dazu einer weiteren Funktion g (*Markerfunktion*, siehe Abb. 7.6), die so modelliert wird, daß sie am Ende ein gültiges Leveling von f

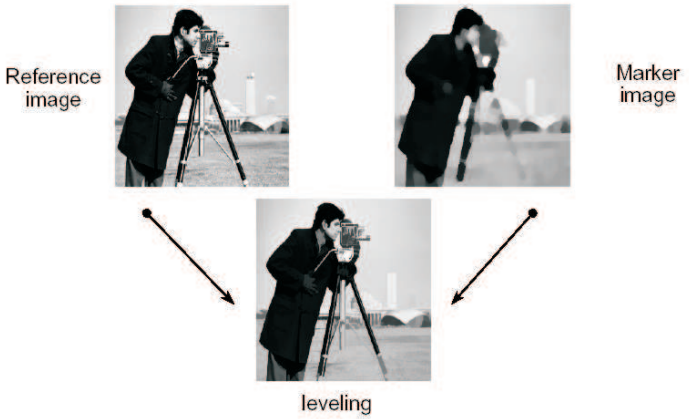


Abbildung 7.6: Aus dem Marker und dem Referenzbild wird ein Leveling erzeugt

darstellt. Der Charakter des Markers spielt zwar für das erzeugte Leveling eine wichtige Rolle, ist aber dennoch beliebig wählbar. Der triviale Fall wäre $g = f$, damit wäre schon ein gültiges Leveling gefunden. Der Algorithmus wird zunächst getrennt für Lower- und Upper-Levelings (siehe Kap. 3.4) formuliert und anschließend zusammengesetzt:

4.1 Konstruktion von Lower-Levelings

Falls die Markerfunktion g (noch) kein Lower-Leveling von f (siehe Def. 2) ist, existiert ein kleineres Nachparpixel:

$$\{\text{g ist kein Lower-Leveling von f}\} \quad \Leftrightarrow \quad \exists \text{ Nachbarn } (p,q) \text{ mit } g_p < \min(f_p\ ,\ g_q) \tag{7.14}$$

Der kleinstmögliche Wert, für den ein Lower-Leveling erreicht wird, lautet daher $g'_p = \min(f_p\ ,\ g_q)$. Daraus ergibt sich folgender Algorithmus Niv^-:

- für alle benachbarten Pixel (p, q) bei denen $g_p < f_p$ und $g_p < g_q$ setze $g_p = \min(f_p\ ,\ g_q)$

Mit Hilfe der maximalen Amplitude δ läßt sich der äquivalente Algorithmus Niv^{δ} formulieren:

- falls $g < f$ setze $g = \min(f\ ,\ \delta g)$

Dieser Algorithmus wird nun solange über das ganze Bild hinweg iteriert (z.B. sequentiell oder mit Hilfe einer Queue) bis das Kriterium überall erfüllt ist.

4.2 Konstruktion von Upper-Levelings

Analog lassen sich nun auch Upper-Levelings (siehe Def. 3) konstruieren. Es ergbit sich der Algorithmus Niv^{+}:

- für alle benachbarten Pixel (p, q) bei denen $g_p > f_p$ und $g_p > g_q$ setze $g_p = \max(f_p\ ,\ g_q)$

Mit Hilfe der manimalen Amplitude ε läßt sich nun wiederum der äquivalente Algorithmus Niv^{ε} formulieren:

- falls $g > f$ setze $g = \max(f\ ,\ \varepsilon g)$

4.3 Konstruktion von Levelings

Um nun das eigentliche Leveling zu erzeugen, werden die beiden Algorithmen Niv^{δ} und $\mathrm{Niv}^{\varepsilon}$ parallel auf g angewendet. Wenn sich dabei die Funktion g nicht mehr ändert, ist sie ein Leveling von f.

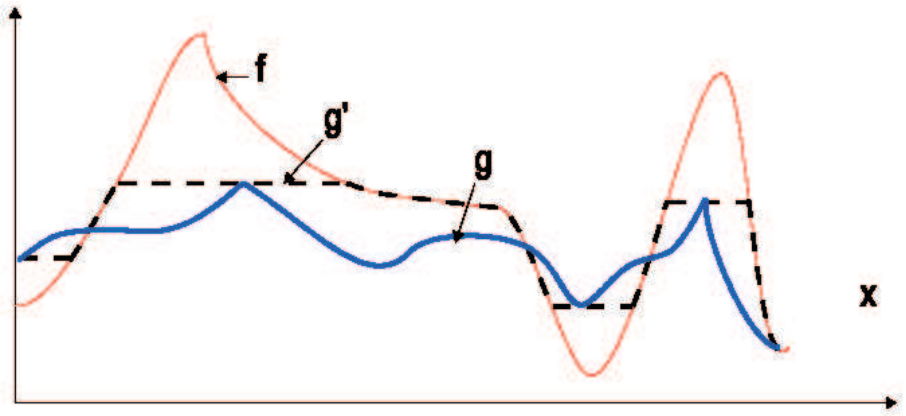


Abbildung 7.7: Aus dem Marker g wurde g' erzeugt, ein Leveling von f

In Abbildung 7.7 ist eine Marker-Funktion g aufgezeigt, die mit Hilfe des Leveling-Verfahrens in ein Leveling g' der Funktion f überführt wird. Für $\{g < f\}$ wird g um den kleinstmöglichen Betrag angehoben. Dies geschieht solange, bis eine Flat-Zone erschaffen wurde oder die Funktion f erreicht wurde. Stellen, an denen $\{g' < f\}$ sind daher immer eine Flat-Zone.
Für $\{g > f\}$ wird g um den kleinstmöglichen Betrag abgesenkt. Dies geschieht wiederum solange, bis eine Flat-Zone erschaffen wurde oder die Funktion f tangiert wurde. Auch solche Stellen, an denen $\{g' > f\}$ sind daher wieder eine Flat-Zone.
Die Abbildung 7.8 zeigt nun eine ähnliche Situation, allerdings wurden hier Quasi-Flat-Zones anstatt der strikten Flat-Zones zugrunde gelegt: Für $\{g < f\}$ wird g um den kleinstmöglichen Betrag angehoben, bis eine Quasi-Flat-Zone erschaffen wurde oder die Funktion f tangiert wurde. Stellen, an denen $\{g' < f\}$ sind daher immer eine Quasi-Flat-Zone (für $\{g > f\}$ gelten die Ausführungen analog zu Bild 7.7). Der Algorithmus für die Quasi-Flat-Zones unterscheidet sich etwas von dem bisher verwendeten. Die Minimum/Maximum-Operatoren ε und δ werden durch allgemeinere Operatoren ersetzt: β stellt einen Stauchungsfaktor $\beta g \leq g$ dar und α einen Dehnungsfaktor $\alpha g \geq g$.
Der Algorithmus läßt sich besonders gut am Beispiel einer diskretisierten Grauwertfunktion verdeutlichen (Abb. 7.9): In Bild a) ist zunächst die Referenzfunktion f (schwarze Linie) und die Markerfunktion g (grauer Bereich) dargestellt. Dann folgen die zwei Hilfsfunktionen: In b) wurde die Erosion ε von f

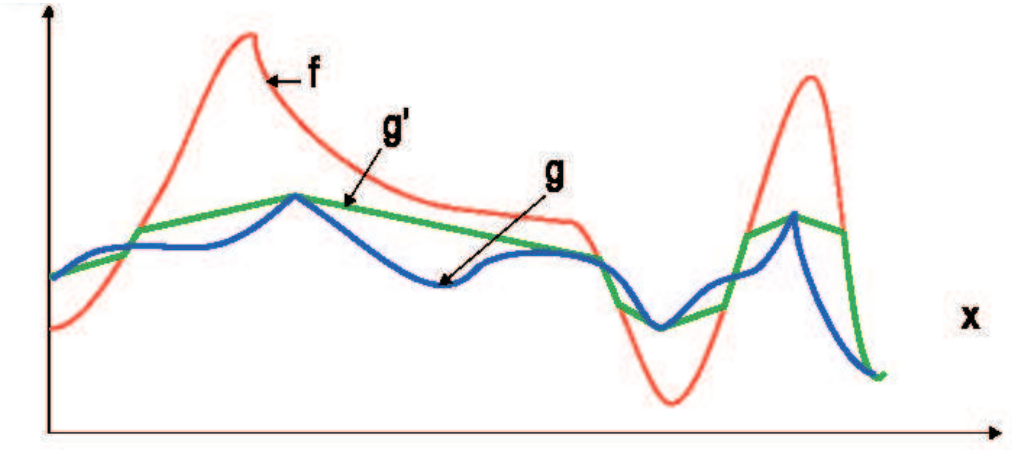


Abbildung 7.8: Extended Levelings arbeiten mit Quasi-Flat-Zones

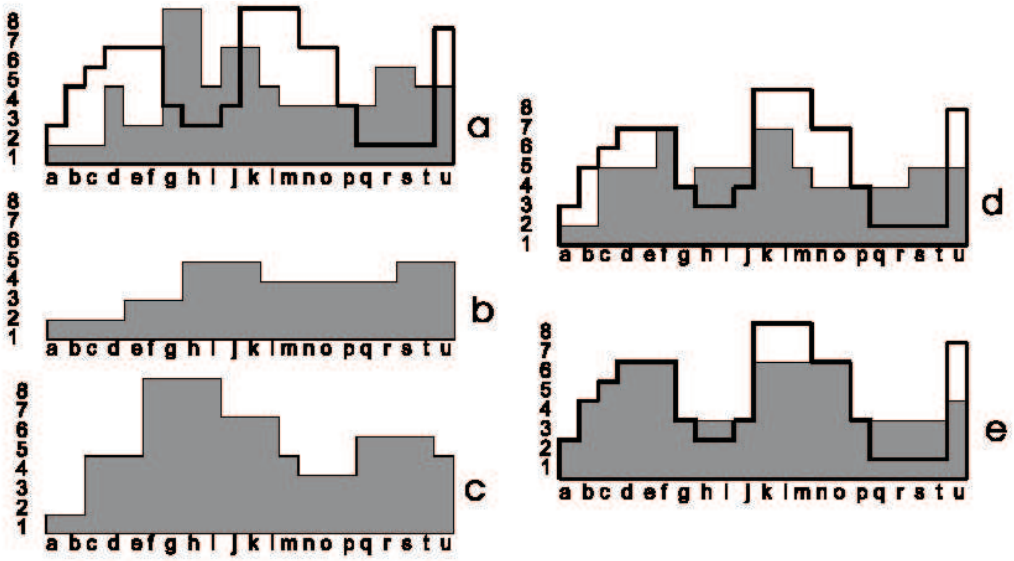


Abbildung 7.9: Levelings am Beispiel einer Grauwertfunktion

berechnet, in c) die Dilatation δ von f (jeweils mit einem Strukturelement der Größe 3). Das Bild d) ist nun durch Anwendung des Algorithmus aus a), b) und c) entstanden:

- falls $g(x) < f(x) : g'(x) = \min(\delta g(x)\ ,\ f(x))$
- falls $g(x) > f(x) : g'(x) = \max(\varepsilon g(x)\ ,\ f(x))$
- falls $g(x) = f(x)$: nichts zu tun

Der Endzustand in Bild e) schließlich ist durch mehrfache Wiederholung dieser Schritte aus Bild d) entstanden. Dabei können die Iterationen können parallel, sequentiell oder hierarchisch angewendet werden, die Reihenfolge hat keinen Einfluß auf das Ergebnis.

4.4 Robustheit von Levelings

Ein spezieller Fall des morphologsichen Filters ist der *starke* morphologische Filter. Die vorgestellten Levelings sind ein solcher starker morphologischer Filter, der im folgenden noch genau definiert wird: Für die Eigenschaft des *morphologischen Filters* müssen folgende Kriterien eines Filters erfüllt sein:

- er muß *ansteigend* sein: $g > h \Rightarrow \phi\ g > \phi\ h$
- er muß *idempotent* sein: $\phi\ \phi = \phi$

Der Filter ist ferner ein *starker* morphologsicher Filter, falls

- $\phi \min(ID\ ,\ \phi) = \phi \max(ID\ ,\ \phi) = \phi$
  Funktionen, deren Werte also zwischen der Urprungsfunktion und der gefilterten Funktion liegen, ergeben also bei Anwendung des Filters dasselbe Ergebnis.

Aufgespannter Skalenraum

Wir betrachten nun zum Abschluß einen durch Levelings aufgespannten Skalenraum. Dabei ersetzen wir (wie aus Kapitel 4.3 bekannt) die Erosion ε durch einen allgemeineren anti-extensiven Operator $\beta g \leq q$ und die Dilatation δ durch einen allgemeineren extensiven Operator $\alpha g \geq g$. Für eine Folge solcher Operatoren und $i > j$ gelte $\alpha_i < \alpha_j$ und $\beta_i > \beta_j$. Sei ferner Λ_i ein Leveling, entstanden mit α_i und β_i. Dann ist $\Lambda_i(f, g)$ ein Leveling von $\Lambda_j(f, g)$ und die Menge der Λ_i spannen einen Skalenraum auf.

5 Zusammenfassung

Levelings bilden ein abgeschlossenes algebraisches Konzept. Sie stellen ein in Ihrer Art recht einfaches morphologisches Hilfsmittel dar. Als starker morphologischer Filter sind sie in der Lage, bei der Partitionierung von medizinischen Bildern wertvolle Dienste zu leisten. Die Levelings stellen eine zusätzliche Abstraktionsebene im Verlauf einer Segmentierung dar; mit ihrer Hilfe wird das Bild vor-bearbeitet. Ohne spezielle Anforderung an die zu findenden Objekte wird das medizinische Material so präpariert, daß in weiterführenden Schritten jede Anfrage nach Inhalten wesentlich schneller und effizienter beantwortet werden kann als bei der herkömmlichen Segmentierung.
Im einzelnen werden Kanten verdeutlicht sowie lokale Minima und Maxima verstärkt, aber ortsinvariant belassen. Je nach Anforderungsprofil können später grobe Skalen verwendet werden, um globale Zusammenhänge zu erkennen oder feinere Skalen, um Details einzelner Objekte zu ergründen.

Literaturverzeichnis

[1] Meyer F, Maragos P: *Morphological Scale-Space Representation with Levelings.* Second International Conference, Scale-Space'99, Corfu, Greece, September 1999, LNCS 1682, p. 187 ff. (1999).

[2] Lehmann T, Oberschelp W, Pelikan E, Repges R: *Bildverarbeitung für die Medizin.* Springer Verlag (1997).

[3] Lange T, Westerhoff N: *Segmentation-Project.* Konrad-Zuse-Zentrum für Informationstechnik Berlin, Abteilung Visualisierung; Segmentation (1998).

[4] Hummel RA, Gidas BC: *Zero crossings and the heat equation.* Technical report, New York Univ., Courtant Institute of Math. Science, Computer Science Division, (1984).

[5] Koenderink JJ: *The structure of images.* Biol. Cybern., 50:363-370 (1984).

[6] Lindeberg T: *On scale selection for differential operators.* In B. Braathen K. Heia, K.A. Hogdr, editor, Proc. 8th Scandinavian Conf. Image Analysis, Trømso, Norway (1993). Norwegian Society for Image Processing and Pattern Recognition.

Automatische Detektion und Verfolgung von Zungenkonturen

von Florian Hasibether

Betreuer: Mark Oliver Güld

Inhaltsverzeichnis

Zusammenfassung

Die vorliegende Arbeit beschäftigt sich mit einem Verfahren zur Detektion und Verfolgung von Konturen der Zunge auf Grundlage von Ultraschall-Bildmaterial. Hierzu findet eine diskrete Darstellung von verformbaren Konturen, „Snakes“, Anwendung. Die Arbeit stellt das Konzept der verformbaren Konturen vor, diskutiert Varianten zu ihrer Modellierung und Optimierung und skizziert ein Verfahren zur Verfolgung von Zungenkonturen, welches abschließend Bewertung findet.

Keywords: Snakes, verformbare Konturen, Zunge, Tracking, Motion Analysis

1 Einleitung

Zur Untersuchung der Funktion der Zunge bei Sprachproduktion und beim Schlucken ist eine Modellierung und Verfolgung ihrer Kontur über die Zeit von Interesse. Dabei wird nach einem Verfahren gesucht, welches imstande ist, unter Kenntnis physiologischer Gegebenheiten der Zunge und mit Hilfe initialer Benutzerinteraktion die Kontur der Zunge automatisch zu erkennen und in ein Modell abzubilden, welches für weitere Rückschlüsse auf die Stellung der Zunge im Mundraum ausgewertet werden kann.

Die *Phonetik* untersucht artikulierte Laute hinsichtlich der Art und Weise, wie sie über die an der Lautbildung beteiligten Organe realisiert werden [10]. Die Zunge ist das wesentliche Organ des Menschen zur Produktion distinktiver Laute. Zur Artikulation eines Lautes wird in der Lunge ein Luftstrom erzeugt, dieser an den Stimmlippen zur Schwingungserzeugung genutzt und der entstandene Ton im Mundraum moduliert. Abbildung 8.1 zeigt die Stellung der Zunge für die Vokale des Deutschen. Die Luft strömt weitgehend ungehindert aus, Lautproduktion geschieht durch die Position der Zunge.

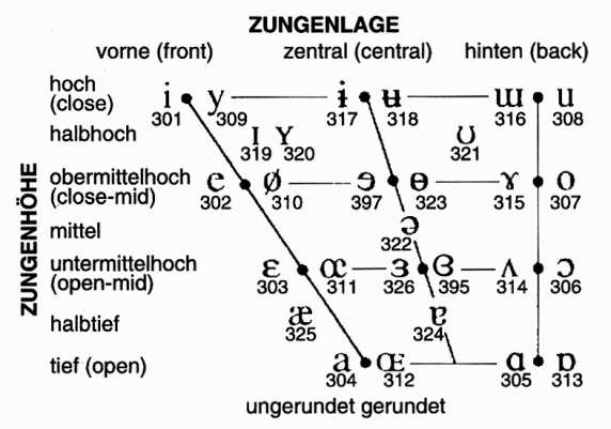


Abbildung 8.1: Vokale im Deutschen

Es ist daher naheliegend, mittels bildgebender Verfahren der Medizin die Stellung der Zunge bei der Sprach- bzw. Lautproduktion aufzunehmen und Modelle für ihre Lage zur entsprechenden Lautkonfiguration zu erstellen. Stone und Lundberg rekonstruieren in [8] für 18 Laute des Englischen 3D Modelle aus Ultraschallbildern der Zunge. In dieser Arbeit wird zunächst auf die Eigenschaften von Ultraschallbildern eingegangen, anschließend eine Einführung in *Aktive Konturmodelle* gegeben und schließlich ein automatisches Verfahren zur Detektion und Verfolgung der Zunge basierend auf der Arbeit von Akgul, Kambhametta und Stone [1] vorgestellt und dessen Ergebnisse bewertet. Ein automatisches Verfahren ist der manuellen Extraktion insofern überlegen, als daß die Genauigkeit über die Bearbeitungsdauer konstant bleibt und die gewonnenen Ergebnisse unter Anwendung des gleichen Verfahrens reproduzierbar sind.

2 Aufnahme des Bildmaterials

2.1 Auswahl der Untersuchungsmethode

Die medizinische Diagnostik stellt eine Reihe bildgebender Verfahren zur Verfügung, die sich für die digitale Weiterverarbeitung eignen [9]. Dazu zählen radiologische Verfahren, insbesondere die digitale Subtraktionsangiographie zum Sichtbarmachen von Blutgefäßen und die Computertomographie, welche durch Rotation der Röntgenquelle und Integration über die gewonnenen Profile Schichtbilder des Körpers zu liefern imstande ist. Magnetresonanztomographie ist unter den bildgebenden Verfahren das neueste, ihre Anwendung die teuerste [9]. Dabei werden die Kerne der Wasserstoffatome im Körper mit Hilfe eines Magnetfeldes in eine Richtung ausgelenkt. Diese Auslenkung kann durch Radiowellen gezielt beeinflusst werden. Die Veränderung zurück in Richtung des Magnetfeldes bei Abschalten der Radiowellen führt zu Signalen, die zu Bildmaterial verarbeitet werden können. Zuvor genannte Methoden sind jedoch hinsichtlich der Kosten, Strahlungsbelastung und Geschwindigkeit weniger geeignet als die Sonographie. Dieses mit Ultraschall arbeitende Verfahren bedient sich piezoelektrischer Kristalle, welche in der Lage sind, elektrische in akustische Energie umzuwandeln. Vergleichbar mit Radar- oder Sonarmessungen wird bei der Ultraschalluntersuchung ein Signal, hier Wellen im Bereich von 2 bis 10 Mhz, ausgesendet und am Untersuchungsgegenstand reflektiert. Die über die Laufzeit und die Güte der Reflektion gewonnenen Informationen werden im Ultraschallbild wiedergegeben. Dieses Verfahren zeichnet sich durch die Eigenschaft aus, daß es in der Lage ist, Bildfolgen in Echtzeit (d.h. 30 Bilder pro Sekunde) zu liefern. Für eine Aufnahme des Mundraumes bieten sich überdies nichtinvasive Verfahren an. Wenngleich Ultraschall, etwa bei der Echokardiographie, auch zur Messung im Körper verwendet werden kann, stellt der folgende Abschnitt einen Aufbau eigens zur Ultraschallaufnahme der Zunge vor, der das Sprechen so wenig wie möglich behindert.

2.2 System zur Bildaufnahme

Bedingt durch die herkömmlichen klinischen Anwendungen sind Ultraschallgeräte nicht auf die Umstände bei der Aufnahme von Bildern der Zunge eingerichtet. Bei Geräten, die für den Handbetrieb vorgesehen sind, ist es nahezu unmöglich, den Schallkopf komplett ruhig unter dem Kinn zu halten. Das hier vorgestellte Verfahren arbeitet mit einem System, dem *Head and Transducer Support System (HATS)* [7], welches den Ultraschallkopf unter dem Kinn fixiert (vgl. Abbildung 8.2).

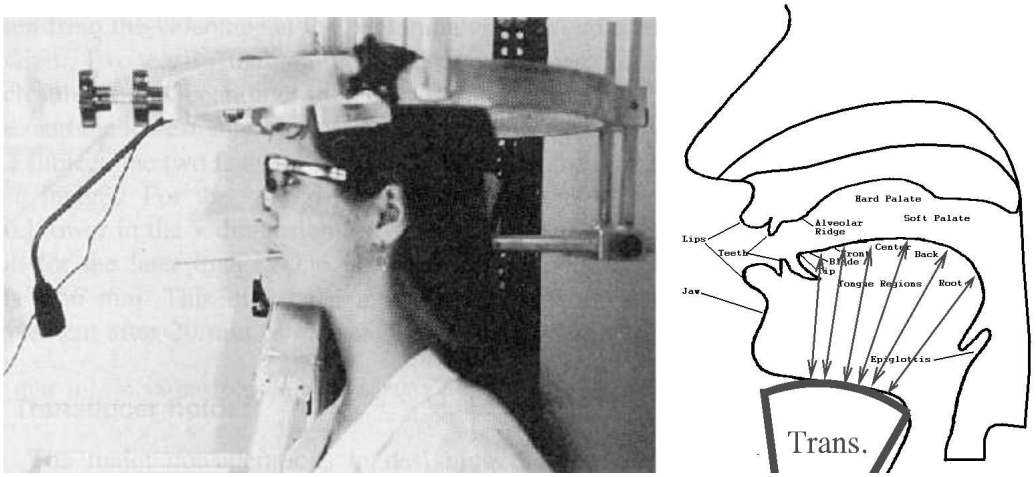


Abbildung 8.2: Head and Transducer Support System

Die aufkommenden Probleme bei der Aufnahme von Ultraschallbildern zur digitalen Weiterverarbeitung und insbesondere zum Zweck des Verfolgens der Zungenkontur über eine Bildsequenz hinweg, resultieren vornehmlich aus dem Umstand, daß bei kleinsten Drehungen oder Verschiebungen des Schallkopfes eine andere Schicht des Mundraumes aufgenommen wird. Ergebnisse zeigen damit weder einen über die Zeit konstanten Bildausschnitt, noch lassen sie sich reproduzieren.

Dem begegnet das HATS, indem drei Ursachen für dieses Problem gelöst werden: Fixierung des Kopfes, Stabilisierung des Ultraschallkopfes und Vermeidung des Zusammenpressen des Unterkiefers bei Abwärtsbewegung des Kinns hin zum Schallkopf. Über eine geeignete Einrichtung wird der Kopf über Klammern in Position gehalten, während sich der Unterkiefer frei bewegen kann. Darunter ist in geeignetem Abstand und vorbestimmter Neigung der Schallkopf fixiert. Es werden damit nicht ausschließlich die Zungenbewegung, sondern vielmehr die Bewegung der Zunge mit dem Unterkiefer aufgenommen.

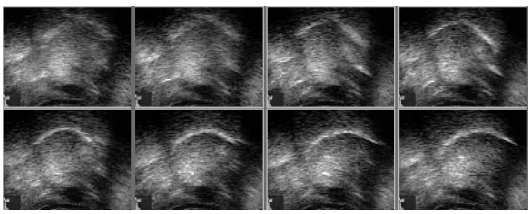
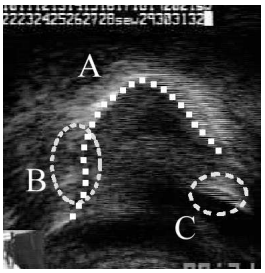


Abbildung 8.3: Mit HATS gewonne Sequenz von Ultraschallbildern

Abbildung 8.3 zeigt eine Sequenz von Ultraschallbildern, die mit Hilfe von HATS aufgenommen werden können. Dabei ist die Spitze der Zunge am rechten Bildrand zu suchen. Der nächste Abschnitt geht auf Besonderheiten dieses Bildmaterials für die Bildverarbeitung ein.

2.3 Eigenschaften von Ultraschallbildern

Ziel des hier vorgestellten Verfahrens ist es, die Oberfläche der Zunge in den vorliegenden Bildsequenzen zu erkennen und die Verschiebung dieser Kontur zu verfolgen. Das Verfahren muß demnach in der Lage sein, die als stark ausgeprägte Kante dargestellte Oberfläche sicher im Bild zu erkennen. Problematisch ist jedoch das starke Bildrauschen (vgl. Abbildungen 8.3, 8.4). Die Ultraschalldiagnostik kennt eine Reihe typischer Probleme, die bei diesem Verfahren auftreten: Bereiche, die akustisch abgeschattet sind, Brechung der Ultraschallwellen und unerwünschte Reflektionen verursachen Artefakte, die die Auswertung des Bildmaterials behindern.



A: Zungenkontur
B: Unterbrechung der Kontur
C: Unerwünschte Reflektion

Abbildung 8.4: Zungenkontur und problematische Bildbereiche

In Abbildung 8.4 ist die zu extrahierende Kontur mit A bezeichnet. Beispiele für die genannten Bildstörungen sind mit B (Unterbrechung der Kontur durch akustischen Schatten) und C (Reflektion einer nicht zur Zunge gehörenden Stelle) markiert. In Abbildung 8.3 zeigen sich weitere Schwierigkeiten, die durch ein Extraktionsverfahren angegangen werden müssen: Während in der unteren Bildsequenz die Zungenkontur trotz Rauschen relativ gut zu erkennen ist, zeigen die Bilder eins und zwei der Sequenz kaum auswertbare Reflektionen der Ultraschallwellen. Dies macht bei Betrachtung eines Bildes $I_t(x, y)$ zum Zeitpunkt t die Auswertung der Informationen aus den Bildern $I_{t-1}(x, y)$ und $I_{t+1}(x, y)$ notwendig. Ein geeignetes Verfahren stellen aktive Konturen dar, die Wissen über ein zugrundeliegendes Modell einbeziehen und auf das aktuell vorliegende Bildmaterial optimiert werden. Eine Einführung in aktive Konturen gibt das folgende Kapitel.

3 Einführung in verformbare Konturen

Das Prinzip der *aktiven Konturmodelle*, wegen ihrer stückweisen Annäherung an eine optimale Lage auf dem Bildmaterial auch *Snakes* genannt, wurde 1988 durch Kass, Witkin und Terzopoulos [5] eingeführt. Seit dem findet es in weitreichenden Anwendungen der Bildverarbeitung Verwendung zur Segmentierung im 2D und 3D Bereich, zum Mustervergleich und zum Verfolgen und Analysieren von Bewegungen [6].
Snakes sind in der Lage, im Bild vorhandene Kanten nicht allein aufgrund des Bildgradienten an einem bestimmten Punkt, sondern auch durch Auswerten räumlicher Zusammenhänge zu finden. Ihr grundsätzlicher Vorteil gegenüber anderen Verfahren zur Merkmalsextraktion ist das Einbeziehen von Bilddaten, einer initialen Ausgangsposition für die Snake, Bedingungen hinsichtlich der gewünschten Kontureigenschaften sowie aus dem Anwendungsbereich abgeleitete Vorgaben zur Form in einen einzigen Extraktionsprozeß [4].
Im Zusammenhang mit dem hier vorgestellten Verfahren zur Extraktion der Zungenkontur ist Interaktionsfähigkeit mit dem Benutzer von erwähnenswerter Bedeutung. Dieser gibt eine Ausgangskontur vor, welche schrittweise verbessert wird. Die Technik, die hier zum Tragen kommt, ist Energieminimierung. Die jeweils aktuelle Konfiguration der Snake birgt in sich ein gewisses Maß an Energie, welches über Vorgaben durch das Modell sowie Gegebenheiten des Bildes bestimmt wird. Die Optimierung der Snake erfolgt dahingehend, daß die Snakeenergie ein Minimum annehmen soll. Der Term *aktive* Kontur hat also seinen Ursprung in dem Umstand, daß die Lage der Kontur sich während der Optimierungsschritte verändert. Die nachfolgenden Abschnitte stellen ausgehend von der ursprünglichen Formulierung durch Kass et.al. [5] Methoden zur Beschreibung von Snakes und deren Optimierung dar.

3.1 Verformbare Konturen mit numerischer Optimierung

Kass et.al. definieren eine aktive Kontur als einen parametrisierten Vektor $v(s)$, der die Bildkoordinaten zu einem Punkt auf der Snake angibt, wobei s die Position auf der Snake bezeichnet mit $s = 0$ für deren Anfang und $s = 1$ für deren Endpunkt (8.1).

$$v(s) = (x(s), y(s)), \quad s \in [0, 1] \tag{8.1}$$

Die gesamte Energie der Snake erhält man durch Integration einer Energiefunktion über die Länge der Snake:

$$E^*_{\text{snake}} = \int_0^1 E_{\text{snake}}(v(s))ds \tag{8.2}$$

Die wesentliche Leistung der aktiven Konturen ist, daß sie sowohl Informationen über die Kontur selbst, über das Bild, sowie über externe Einflüsse, etwa anwendungsspezifisches Wissen, nutzt, um sich an gewünschte Merkmale im Bild anzulegen. Die Snakeenergie setzt sich daher aus Termen zusammen, die die interne Energie der Kontur beschreiben sowie Bildenergie und Energie aus Modelleinflüssen einbeziehen:

$$\int_0^1 E_{\text{int}}(v(s)) + E_{\text{image}}(v(s)) + E_{\text{con}}(v(s))ds \tag{8.3}$$

Die Minimierung dieses Terms charakterisiert das Verhalten der Snake. Er bestimmt, wie sich die Snake verhält und an welche Merkmale im Bild sie sich anlegt. Der Ansatz von Kass et.al. schlägt eine Zerlegung der aus dem Bildmaterial gewonnenen Energie E_{image} in einen Term, der die Bildintensität berücksichtigt, $E_{\text{line}} = I(x, y)$, einen Term der auf Kanten reagiert, $E_{\text{edge}} = -|\nabla I(x, y)|^2$ und einen Term für Linienendpunkte E_{term} vor. Die interne Energie hingegen ist gegeben durch

$$E_{\text{int}}(v(s)) = \alpha(s)|v'(s)|^2 + \beta(s)|v''(s)|^2 \tag{8.4}$$

Hierbei sind $v'(s)$ und $v''(s)$ die erste bzw. die zweite Ableitung von $v(s)$ nach s. Der Faktor α wichtet die Elastizität der Kontur, d.h. je größer er ist, desto stärker widersetzt sich die Snake einer Ausdehnung. Der

Faktor β hingegen bestimmt den Einfluß der Starrheit der Snake. Je höher β ist, desto stärker widersteht die Snake ihrer Krümmung.
Die meisten Anwendungen fassen die Energieterme zusammen und unterscheiden lediglich E_{int} und E_{ext}, womit die Energie der Snake geschrieben werden kann als

$$E^*_{\text{snake}} = \int_0^1 E_{\text{snake}} ds = \int_0^1 E_{\text{int}}(v(s)) + E_{\text{ext}}(v(s)) ds \tag{8.5}$$

Schließlich wird jedoch das Minimum von E^*_{snake} gesucht. Aus der Variationsrechnung stammt die Euler-Lagrange-Bedingung, welche besagt, daß für eine Funktion $F = F(x, y(x), y'(x))$ ein Minimum des Integrals

$$E = \int_{x_1}^{x_2} F(x, y(x), y'(x)) \tag{8.6}$$

der folgenden Bedingung genügen muß:

$$\frac{d}{dx}\frac{\partial F}{\partial y'} - \frac{\partial F}{\partial y} = 0 \tag{8.7}$$

Mit der Formulierung wie in (8.5) und unter Verwendung von (8.4) für E_{int} ergibt sich beim Einsetzen in (8.6) die Darstellung

$$-\frac{d}{ds}(\alpha(s)v'(s)) + \frac{d^2}{ds^2}(\beta(s)v''(s)) + \nabla E_{\text{ext}}(v(s)) = 0 \tag{8.8}$$

Unter der Bedingung, daß $\alpha(s)$ und $\beta(s)$ Konstanten sind, kann $v(s)$ in seine Komponenten x und y aufgeteilt und folgende Euler-Gleichungen notiert werden [5],[2],[6].

$$-\alpha\frac{d^2x}{ds^2} + \beta\frac{d^4x}{ds^4} + \frac{\partial E_{\text{ext}}(v(s))}{\partial x} = 0 \tag{8.9}$$

$$-\alpha\frac{d^2y}{ds^2} + \beta\frac{d^4y}{ds^4} + \frac{\partial E_{\text{ext}}(v(s))}{\partial y} = 0 \tag{8.10}$$

Für eine numerische Lösung dieser Gleichungen werden die Ableitungen in dem Term der internen Energie E_{int} über die Methode der finiten Differenzen approximiert:

$$\begin{aligned} &v_i'(s) = |\tfrac{dv_i}{ds}| \approx |v_i - v_{i-1}|^2 = (x_i + x_{i-1})^2 + (y_i - y_{i-1})^2 \\ &\text{bzw.} \\ &v_i''(s) = |\tfrac{d^2v_i}{ds^2}| \approx |v_{i-1} - 2v_i + v_{i+1}|^2 = (x_{i-1} - 2x_i + x_{i+1})^2 + (y_{i-1} - 2y_i + y_{i+1})^2 \end{aligned} \tag{8.11}$$

Dabei bezeichnet in Gleichung (8.11) das $v_i = (x_i, y_i)$, $i = 1, \ldots, n$ einen Punkt auf der Kontur. In dieser diskreten Darstellung ergibt sich für E_{snake} in (8.5) durch Einsetzen von (8.11) in (8.8)

$$\begin{aligned} &\alpha_i(v_i - v_{i-1}) - \alpha_{i+1}(v_{i+1} - v_i) + \beta_{i-1}(v_{i-2} - 2v_{i-1} + v_i) \\ &-2\beta_i(v_{i-1} - 2v_i + v_{i+1}) + \beta_{i+1}(v_i - 2v_{i+1} + v_{i+2}) + (f_x(i), f_y(i)) = 0 \end{aligned} \tag{8.12}$$

Hier ist $f_x(i) = \partial E_{\text{ext}}/\partial x_i$ bzw. $f_y(i) = \partial E_{\text{ext}}/\partial y_i$. Die Eulergleichungen (8.9), (8.10) können schließlich mit (8.12) in Matrixform als

$$Ax + f_x(x, y) = 0 \tag{8.13}$$
$$Ay + f_y(x, y) = 0 \tag{8.14}$$

geschrieben werden und mit Hilfe der impliziten Euler-Methode gelöst werden [5],[2]:

$$x_t = (A + \gamma I)^{-1}(\gamma x_{t-1} - f_x(x_{t-1}, y_{t-1})) \tag{8.15}$$
$$y_t = (A + \gamma I)^{-1}(\gamma y_{t-1} - f_x(x_{t-1}, y_{t-1})) \tag{8.16}$$

3.2 Energieminimierung mittels dynamischer Programmierung

Die oben skizzierte numerische Lösung der Euler-Lagrange-Gleichung zur Minimierung der Snakeenergie birgt jedoch einige Probleme. Bei der Gleichung handelt es sich um eine notwendige Bedingung für das Vorhandensein eines Minimum. Amini et.al [2] weisen darauf hin, daß bei Verwendung der Euler-Lagrange-Gleichung möglicherweise ein Maximum anstatt eines Minimum gefunden wird. Weitere Kritikpunkte betreffen die Konvergenz und Stabilität des Verfahrens. Es wird daher ein Lösungsansatz vorgeschlagen, der auf dem Prinzip der dynamischen Programmierung beruht.
Amini et.al. gehen wiederum von einer Diskretisierung der internen Energie wie in (8.12) aus und schreiben die Snakeenergie als

$$E^*_{\text{snake}} = \sum_{i=0}^{n-1} E_{\text{int}}(v_i) + E_{\text{ext}}(v_i). \tag{8.17}$$

Die Minimierung von (8.17) kann als schrittweiser, mehrstufiger Entscheidungsprozeß angesehen werden. Der Ansatz mittels dynamischer Programmierung stellt eine Verbindung her zwischen der Minimierung der totalen Snakeenergie und der Minimierung einer Funktion der Form

$$E(v_1, v_2, \dots, v_n) = E_1(v_1, v_2) + E_2(v_2, v_3) + \cdots + E_{n-1}(v_{n-1}, v_n) \tag{8.18}$$

wobei jedes v_i nur m verschiedene Werte annehmen kann. Die dynamische Programmierung arbeitet mit einer Kostenfunktion, für die schrittweise die minimalen Gesamtkosten bestimmt werden können. Als Beispiel für (8.18) mit $n = 5$:

$$\begin{aligned} s_1(v_2) &= \min_{v_1} E_1(v_1, v_2) \\ s_2(v_3) &= \min_{v_2}\{s_1(v_2) + E_2(v_2, v_3)\} \\ s_3(v_4) &= \min_{v_3}\{s_2(v_3) + E_3(v_3, v_4)\} \\ \min_{v_1,\dots,v_4} E(v_1, v_2, v_3, v_4, v_5) &= \min_{v_4}\{s_3(v_4) + E_4(v_4, v_5)\} \end{aligned} \tag{8.19}$$

Generell also

$$s_k(v_{k+1}) = \min_{v_k}\{s_{k-1}(v_k) + E_k(v_k, v_{k+1})\}. \tag{8.20}$$

Berücksichtigt man zunächst nur den Term für die erste Ableitung in E_{int} (vgl. (8.8) bzw. (8.12)), so ergibt sich bei der Anwendung auf die aktive Kontur

$$s_k(v_{k+1}) = \min_{v_k}\{s_{k-1}(v_k) + E_{\text{ext}}(v_k) + |v_{k+1} - v_k|^2\}. \tag{8.21}$$

Neben einer Matrix, die diese Kosten aufnimmt, wird eine Positionsmatrix benötigt, welche für Schritt k den Wert von v_k aufnimmt, der (8.21) minimiert. Kandidaten für v_k werden in einer $m \times m$-Umgebung gesucht.
Berücksichtigt man auch den Engergieterm zweiter Ordnung in (8.8), so müssen drei adjazente Punkte für (8.18) betrachtet werden:

$$E_{i-1}(v_{i-1}, v_i, v_{i+1}) = E_{\text{ext}}(v_i) + E_{\text{int}}(v_{i-1}, v_i, v_{i+1}) \tag{8.22}$$

Für die Kostenfunktion ergibt sich in diesem Fall:

$$S_i(v_{i+1}, v_i) = \min_{v_{i-1}}\{S_{i-1}(v_i, v_{i-1}) + \alpha(|v_i - v_{i-1}|)^2 + \beta|v_{i+1} - 2v_i + v_{i-1}|^2 + E_{\text{ext}}(v_i)\} \tag{8.23}$$

Für jede Stufe hat die DP Tabelle m^2 Einträge. Einträge in der i+1-ten Zeile repräsentieren feste Werte für v_{i+1} und v_i. Minimierung geschieht über v_{i-1} und wird bei i+1 eingetragen. Über Bracktracking in der DP Tabelle kann die minimale Energie $E^*_{\text{snake}}(t) = \min_{v_n} S_{n-1}(v_n)$ zum Zeitpunkt t gefunden werden. Dieser

Schritt kann wiederholt werden, bis die Energie sich nicht weiter verbessern lässt [2]. Die Lösung mittels dynamischer Programmierung hat eine Komplexität von $O(nm^3)$, wobei n die Länge der Kontur und m die Anzahl der Wahlmöglichkeiten für ein v_k ist. Die hier vorgeschlagene Lösung bedient sich überdies der Randbedingung, daß zwei adjazente Knoten auf der Kontur sich nicht näher als einen Abstand d kommen dürfen. Minimierungen von $S_i(v_{i+1}, v_i)$, die diese Bedingung verletzen, werden verworfen.

3.3 Energieminimierung durch einen Greedy Algorithmus

In [11] finden die zuvor dargestellten Methoden zur Energieminimierung Bewertung. Dabei wird für den Ansatz von Kass et.al. [5] vor allem die numerische Instabilität, das Fehlen von Bedingungen an den Abstand der Punkte auf der Snake und für den Ansatz von Amini et.al. [2] die hohe Laufzeit bemängelt. Zu beiden Algorithmen wird bemerkt, daß die Approximation der Ableitungen des internen Energieterms über die Methode der endlichen Differenzen geschieht. Hier setzen Williams und Shah an und stellen eine Lösung mittels eines Greedy Algorithmus vor [11]. Um die interne Energie abzubilden, werden verschiedene Ansätze zur Bewertung der Krümmung einer aktiven Kontur aus diskreten Punkten diskutiert und verglichen. Hinsichtlich der Berechnungseffizienz günstig ist die Bestimmung der Differenz zweier Vektoren $\vec{u_1}$ und $\vec{u_2}$ mittels

$$|\vec{u_2} - \vec{u_1}|^2 \tag{8.24}$$

um die Krümmung der Kontur zu bewerten. Hierbei sind die $\vec{u_i} = (x_i - x_{i-1}, y_i - y_{i-1})$. Durch Normalisierung wird der Längenbezug ausgeschaltet und lediglich die relative Richtung bewertet:

$$|\vec{u_{i+1}}/|\vec{u_{i+1}}| - \vec{u_i}/|\vec{u_i}||^2 \tag{8.25}$$

Die für den Greedy Ansatz verwendete Darstellung zerlegt die interne Energie in Terme für Kontinuität und Krümmung. Die externe Energie wird über den Bildgradienten definiert. Die Autoren schildern die Beobachtung, daß die Wahl von $|v_i - v_{i-1}|^2$ für den Kontinuitätsterm (vgl. (8.11)) zu einem Schrumpfen der Kontur führt, da hier schließlich die Distanz zwischen zwei Punkten minimiert wird, und schlagen $\bar{d} - |v_i - v_{i-1}|$ vor. Dies mißt den Abstand der durchschnittlichen Distanz $\bar{d}$ zu der Distanz der aktuell betrachteten Punkte. Hinsichtlich des Terms zweiter Ordnung (vgl. (8.4)) für die Krümmung wird die Approximation (8.11) aus Abschnitt 3.1 gewählt.

```
1  do
2    /* loop to move points to new location */
3    for i=0 to n   /* point 0 is first and last one processed */
4      E_Min = MAXINT
5      for j= 0 to m-1   /* m is size of neighborhood */
6         E_j = α_i E_cont,j + β_i E_curv,j + γ_i E_image,j
7         if E_j < E_Min then
8            E_Min = E_j
9            jmin = j
10     Move point v_i to location jmin
11     if jmin not current location, ptsmoved += 1   /* count points moved */
12   /* process determines where to allow corners in the next iteration */
13   for i = 0 to n-1
14     c_i = |u_i/|u_i| - u_{i+1}/|u_{i+1}||^2      /* compute curvature */
15   for i= 0 to n-1
16     if (c_i > c_{i-1} and c_i > c_{i+1})    /* if curvature is larger than neighbors */
17       and c_i > threshold1                 /* and curvature is larger than threshold */
18       and mag(v_i) > threshold2            /* and edge strength is above threshold */
19         then β_i = 0                       /* relax curvature at point i */
20 until ptsmoved < threshold3
```

Abbildung 8.5: Pseudocode des Greedy Algorithmus zur Energieminimierung

Der Algorithmus geht nach dem *Greedy Verfahren* vor (vgl. Abbildung 8.5). Hierbei wird die Snakeenergie für Punkte einer Umgebung um einen Punkt auf der Snake berechnet und die lokal beste Lösung gewählt (vgl. Zeilen 5-9). Zu Beginn eines Durchlaufs werden α_i, β_i und γ_i für alle i zu 1 initialisiert. Zeilen 13-19 bestimmen den Einfluß auf die Gesamtenergie und relaxieren bei starker Krümmung gegebenenfalls den Krümmungsterm durch $\beta_i = 0$. Die Autoren weisen darauf hin, daß dieser Algorithmus keine Optimalität garantiert, jedoch für eine Kontur mit n Punkten und eine $m \times m$ Umgebung mit einer Laufzeit von $O(nm)$ sehr günstig abschneidet. Einen vergleichbaren Ansatz findet man in [3].

4 Verfolgung der Zunge mittels Snakes

Die vorhergehenden Kapitel haben einen Einblick in den Untersuchungsgegenstand und das Prinzip aktiver Konturen gegeben. Im folgenden wird ein Verfahren [1] vorgestellt, welches sich die dargestellten Vorteile und Methoden von Snakes zu Nutze macht, um die Kontur der Zunge aus einer Ultraschall Bildsequenz zu erkennen.

4.1 Modellierung der Zunge

Die Zungenkontur wird modelliert durch eine diskrete Variante der Snakes, gegeben durch eine Menge V von geordneten Punkten $v_i = (x_i, y_i)$ und einer initialen Modellkontur S.

$$\begin{aligned} V &= [v_1, v_2, \ldots, v_n] \\ S &= [s_1, s_2, \ldots, s_n] \end{aligned} \tag{8.26}$$

Die Energie der Snake wird aufgefasst wie zuvor als die durch α und β gewichtete Summe von interner, d.h. der durch das Snakemodell unter Berücksichtigung der Initialkontur bestimmten, Energie $E_{\text{int}}(v_i, S)$ und externer Energie $E_{\text{ext}}(v_i, I)$. Damit ergibt sich als zu minimierender Term

$$E_{\text{snake}}(V, S, I) = \sum_{i=1}^{n} \alpha E_{\text{int}}(v_i, S) + \beta E_{\text{ext}}(v_i, I). \tag{8.27}$$

Interne Energie

In die Modellierung der Snake gehen drei Faktoren ein, die durch λ gewichtet sind: Ein Term $E_{\text{smo}}(v_i)$ für die *Glattheit*, ein Term $E_{\text{sim}}(v_i, S)$ für die *Übereinstimmung* mit dem vorgegebenen Modell S und ein Term $E_{\text{dist}}(v_i)$, der den *Abstand* der Punkte auf der Snake zueinander bemißt.

$$E_{\text{int}}(v_i, S) = \overrightarrow{\lambda} \cdot \begin{bmatrix} E_{\text{smo}}(v_i) \\ E_{\text{sim}}(v_i, S) \\ E_{\text{dist}}(v_i) \end{bmatrix} \tag{8.28}$$

$$\overrightarrow{\lambda} = [\lambda_1 \lambda_2 \lambda_3], \quad 0 \leq \lambda_1, \lambda_2, \lambda_3 \leq 1, \quad \lambda_1 + \lambda_2 + \lambda_3 = 1 \tag{8.29}$$

Der Term für die Glattheit steht in direkter Verbindung mit der Krümmung in (8.4). Anstatt sich jedoch der zweiten Ableitung zu bedienen, wird durch Akgul et.al. die Wahl des Winkel zwischen zwei Vektoren, die durch einen Punkt v_i auf der Snake und seine Nachbarn v_{i-1} und v_{i+1} gegeben sind, vorgeschlagen. Die aus der Geometrie bekannte Gleichung zur Berechnung des Winkels zwischen zwei Vektoren führt auf

$$E_{\text{smo}}(v_i) = 1 - \cos\theta_i = 1 - \frac{\overrightarrow{v_{i-1}v_i} \cdot \overrightarrow{v_i v_{i+1}}}{| \overrightarrow{v_{i-1}v_i} || \overrightarrow{v_i v_{i+1}} |} \tag{8.30}$$

Dabei wird der Wert dieses Ausdrucks umso geringer, je mehr die Snake einer geraden Strecke entspricht. Bei der hier verwendeten Darstellung handelt es sich um eine nicht geschlossene Snake. Daher wird

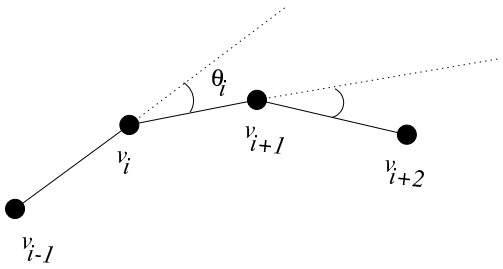


Abbildung 8.6: Berechnung von θ_i als Maß für die Krümmung der Snake

$E_{\text{smo}}(v_1) = E_{\text{smo}}(v_2)$ und $E_{\text{smo}}(v_n) = E_{\text{smo}}(v_{n-1})$ definiert und die Gleichung für $i = 2 \dots (n-1)$ angewendet. Dieses Modell zeigt Abbildung 8.6.
Eine entscheidende Rolle in dieser Anwendung spielt der Term, welcher die Ähnlichkeit zu einer vorgegebenen Kontur bewertet. Zu Beginn der Extraktion wird diese Modellkontur durch den Benutzer vorgegeben, um den interessanten Bildbereich einzugrenzen und damit den geschilderten Schwierigkeiten von unerwünschten Reflektionen in den Ultraschallbildern beizukommen. Durch die Bewertung der Krümmungsänderung zwischen einer neu extrahierten Kontur und dem Modell (vgl. Gleichung (8.31)) wird die Snake gezwungen, ihre Form an die bestehende Kontur anzupassen. Die Ähnlichkeit in einem Punkt ist gegeben durch die Differenz zwischen der Krümmungsenergie im Punkt v_i und der Krümmungsenergie im korrespondierenden Punkt s_i des Modells.

$$E_{\text{sim}}(v_i, S) = \left| \left(1 - \frac{\overrightarrow{v_{i-1}v_i} \cdot \overrightarrow{v_iv_{i+1}}}{| \overrightarrow{v_{i-1}v_i} || \overrightarrow{v_iv_{i+1}} |} \right) - \left(1 - \frac{\overrightarrow{s_{i-1}s_i} \cdot \overrightarrow{s_is_{i+1}}}{| \overrightarrow{s_{i-1}s_i} || \overrightarrow{s_is_{i+1}} |} \right) \right| \tag{8.31}$$

In vorhergehenden Arbeiten [2],[11] wurde festgestellt, daß ein gleichbleibender Abstand zwischen den Punkten einer Snake beachtet werden muß, um dem Phänomen vorzubeugen, daß es zu einer Aggregation von Punkten kommt. Der Abstand zwischen zwei benachbarten Punkten $|v_i - v_{i-1}|$ wird daher hinsichtlich seiner Übereinstimmung mit einem durchschnittlichen Abstand über alle Punkte bewertet und erreicht für ein v_i einen umso kleineren Energiezustand, je näher er an diesem Durchschnitt ist.

$$E_{\text{dist}}(v_i) = \left| \frac{|v_i - v_{i-1}|}{\frac{1}{n-1} \sum_{j=2}^{n} |v_j - v_{j-1}|} - 1 \right| \tag{8.32}$$

Experimentelle Untersuchungen der Autoren zeigten, daß für den Wichtungsparameter-Vektor λ eine automatische Wahl mit Hilfe des sogenannten *min-max Prinzips* möglich ist. Das min-max Prinzip maximiert über λ und minimiert über mögliche Positionen für v_i. Dies trägt dem Wunsch Rechnung, daß Terme in (8.28), die besonders hohe Energie an einem Punkt haben, bei der Minimierung der Gesamtenergie stärker einbezogen werden müssen. Für eine Einstellung der Parameter α und β in (8.27) ist dies nicht ohne weiteres möglich.

Externe Energie

Ziel der Anwendung ist es, die Kontur der Zunge zu finden. Diese stellt sich im Bild als eine mehr oder weniger stark ausgeprägte Kante dar. Die externe Energie verknüpft die Snake mit Merkmalen im Bild. Für $E_{\text{ext}}(v_i, I)$ wird hier der Bildgradient, welcher die Intensitätsunterschiede an einem Punkt v_i bewertet, gewählt.

$$E_{\text{ext}}(v_i, I) = -|\nabla I(v_i)| \tag{8.33}$$

Die Autoren weisen darauf hin, daß der Bildgradient allein nicht als ein robustes Maß erscheinen mag und geben daher eine *coarse-to-fine* Strategie an. Auf der gröberen Ebene wird zunächst nur die Stärke des Bildgradienten betrachtet. Es sei V_0 die initiale Kontur in Bild I_0 und $v_0 = (x_0, y_0)$ das erste Element

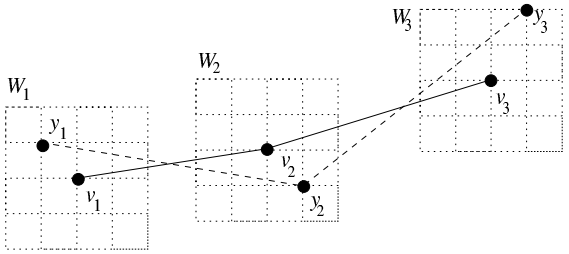


Abbildung 8.7: Suchfenster W_i für mögliche Nachfolger y_i von v_i

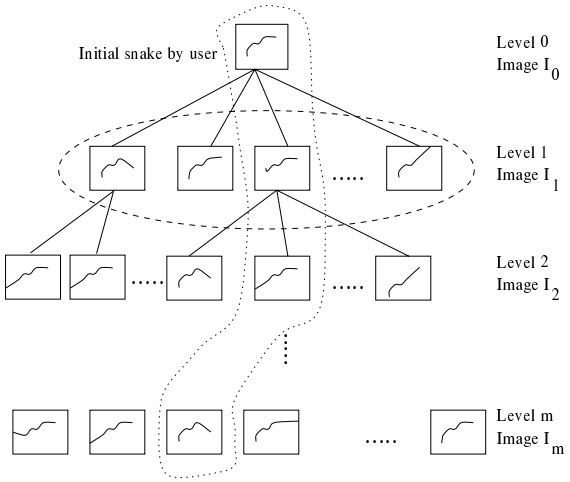


Abbildung 8.8: Baumrepräsentation der Konturen für eine Bildsequenz

von V_0. Dann bezeichnet W_0 eine Suchregion um v_0 im Bild I_1. Aufgrund des Bildgradienten werden hier die besten Kandidaten für v_0 gesucht. Die Verfeinerung ist nun, die Richtung des Gradienten einzubeziehen und aus den gefundenen Kandidaten diejenigen auszusuchen, deren Gradientenrichtung ähnlich ist. Erweitert werden kann diese Methode dadurch, den Kandidaten $\nabla I_1(x_1, y_1)$ mit der Normalen der Zungenkontur im Punkt v_0 zu vergleichen. Hier geht die Annahme ein, daß Bildgradienten von Punkten auf der Zungenkontur orthogonal zu der Snake orientiert sind. Mit Hilfe dieses Verfahrens können aus einem $l \times l$ Suchfenster für W_i, d.h. l^2 möglichen Kandidaten, die besten l bis $2l$ Stück ausgewählt werden.

4.2 Verfolgung der Zungenkontur

Ausgehend von der durch den Experten vorgegebenen Kontur soll die Snake in der Lage sein, eine optimale Lage auf dem Bild hinsichtlich der die Zunge darstellenden Kante zu finden. Die optimierte Kontur stellt wiederum die Initialkontur für das nächste Bild der Sequenz dar. Dieser Prozeß wird bis zur Abarbeitung aller Bilder verfolgt. Es sei $V = [v_1, v_2, \ldots, v_n]$ eine Snake im Bild I und $Y = [y_1, y_2, \ldots, y_n]$ eine weitere Snake, die die Energie möglicherweise verbessert und deren y_i in den $l \times l$ Suchfenstern W_i um $v_i, i = 1 \ldots n$ liegen. Die Menge der *möglichen Snakes* $P(V, I)$ für V in Bild I beinhaltet alle denkbaren Y. Abbildung 8.7 zeigt ein Beispiel für $n = 3$. Die Kardinalität von $P(V, I)$ ist in diesem Falle $l^{2^n} = 5^{2^3}$. Für die Gesamtenergie einer optimierten Menge von Snakes ergibt sich unter Zuhilfenahme von $P(V, I)$ bei einer Bildsequenz aus m Bildern

$$E_{\text{Sequence}} = \min \sum_{k=1}^{m} E_{\text{Snake}}(V_k, V_{k-1}, I_k). \tag{8.34}$$

Dabei ist $V_k \in P(V_{k-1}, I_k)$ und V_0 die Initialkontur. Als Modell für die aktuelle Snake V_k im Bild I_k (mit Ausnahme der Startkontur) wird also die Snake V_{k-1} aus dem vorhergehenden Bild zugrundegelegt und diese auch für die Auswahl der Kandidaten zur Energieoptimierung verwandt. Abbildung 8.8 stellt dieses Optimierungsproblem als Suchbaum dar. Die gestrichelte Linie schließt die Elemente von $P(V_0, I_0)$ ein, die gepunktete Linie markiert die optimalen Snakes. Jede Ebene bildet ein Bild der Sequenz ab und jeder Knoten in dem Baum hat l^{2^n} direkte Nachfolger. Eine erschöpfende Suche zum Finden der gepunktet markierten Snakes ist daher äußerst ineffizient.

4.3 Energieoptimierung

Das Kapitel 3 hat einige Methoden zur Optimierung vorgestellt. Unter Rücksichtnahme auf Laufzeit und numerische Stabilität wählen Akgul et.al. [1] ein Verfahren, was sich an den vorgestellten Greedy Algorithmus anlehnt. Dabei wird zu Bestimmung der Gesamtenergie lediglich das lokale Minimum für V_k mit $k = 1 \ldots m$ bestimmt (vgl. (8.34)).

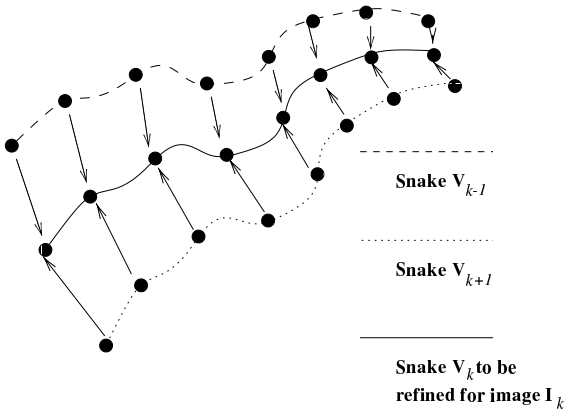


Abbildung 8.9: Verfeinerung der Kontur in Bild k unter Zuhilfenahme von Bild $k-1$ und Bild $k+1$

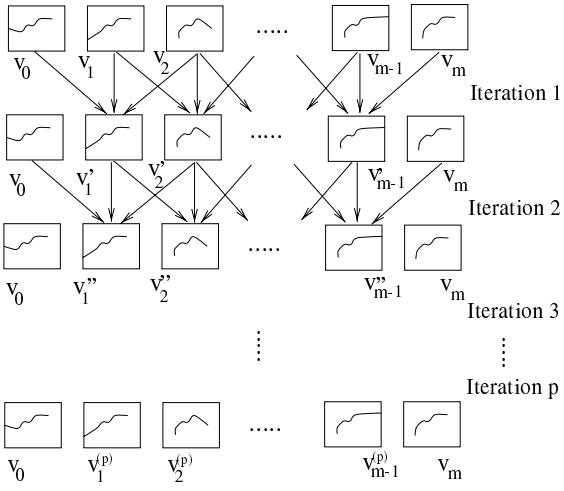


Abbildung 8.10: p-fache Anwendung des Nachverarbeitungsschritts

$$E_{\text{Sequence}} = \sum_{k=1}^{m} \min E_{\text{Snake}}(V_k, V_{k-1}, I_k) \tag{8.35}$$

Hierbei ist jedoch zu beachten, daß, ähnlich dem Ansatz von Williams und Shah [11], keine optimale Lösung wie in [2] garantiert wird. Bei Auswahl des lokalen Minimums werden nur noch die Söhne des entsprechenden Knotens betrachtet. Intuitiv bedeutet das, daß für die Entscheidung auf Ebene i (vgl. Abbildung 8.8) keine Informationen von Ebene $i+1$ und darunter zu Rate gezogen werden. Die Folge sind Probleme hinsichtlich der räumlich-zeitlichen Konsistenz, obwohl gerade dies durch die Verfolgung der Zunge über die Zeit gewährleistet werden soll. Die Autoren weisen darauf hin, daß insbesondere in der Anwendung der Sprachanalyse eine Betrachtung der Vorgänger- und Nachfolge-Bilder obligatorisch ist und schlagen daher einen Nachverarbeitungsschritt zur Wahrung räumlich-zeitlicher Konsistenz vor.
Zunächst wird die approximative Energieoptimierung mittels (8.35) berechnet. Der Nachverarbeitungsschritt bedient sich einer zusätzlichen Energieoptimierung, bei der in den Term der internen Energie eine zusätzliche Energie $E_{\text{stc}}(v_i^k)$ eingefügt wird, um den Forderungen an räumlich-zeitliche Konsistenz Rechnung zu tragen.

$$E_{\text{stc}}(v_i^k) = 1 - \frac{\overrightarrow{v_i^{k-1} v_i^k} \cdot \overrightarrow{v_i^k v_i^{k+1}}}{\mid \overrightarrow{v_i^{k-1} v_i^k} \mid\mid \overrightarrow{v_i^k v_i^{k+1}} \mid} \tag{8.36}$$

Diese Gleichung definiert Vektoren zwischen der Position v_i auf einer Snake in den Bildern $k-1$ und k bzw. k und $k+1$ (v_i^k ist das i-te Element der Kontur in Bild k) und bewertet deren Winkel zueinander analog zu Gleichung (8.30). Hier wird also die kontinuierliche Verschiebung eines Knotens über die Zeit bewertet.
Auf Grundlage eines Verfahrens von Gunn und Nixon werden dann die gefundenen Positionen für die Snakes verbessert. In [4] werden zur Segmentierung geschlossene duale aktive Konturen benutzt, die sich der interessanten Region von außen und innen nähern und ihr Optimum erreichen, wenn beide Snakes dieselbe Kontur gefunden haben. Hier werden auf eine ähnliche Weise die Snakes V_{k-1} und V_{k+1} benutzt, um V_k zu verbessern. Entlang der Kontur V_k werden Kandidatenpunkte für V_{k-1} ausgesucht und die Energie unter Verwendung von E_{stc} erneut optimiert, analog für V_{k+1}. Abbildung 8.9 zeigt, wie die Vorgänger- und Nachfolgesnakes verwendet werden, um V_k zu verbessern. Dieser Prozeß stoppt, wenn für die Verbesserung von beiden Seiten die gleiche Kontur für V_k gefunden wird.
Der Nachverarbeitungsschritt findet Anwendung für alle m Snakes $V_1, \ldots, V_{m-1}$ der Bildsequenz und wird für die gesamte Sequenz iteriert, bis die totale Energie nicht mehr verbessert werden kann. Diesen Prozeß illustriert die Abbildung 8.10. Dabei kann man erkennen, daß beim ersten Iterationsschritt Informationen

des Vorgängers und Nachfolgers einfließen, bei den weiteren Schritten sich jedoch auch der Einfluß weiterer Nachbarn auswirkt. Da für z.B. im ersten Schritt für V_2' Informationen aus V_1, V_2 und V_3 einflossen, profitiert V_2'' im nächsten Iterationsschritt schon von Einflüssen aus V_0 bis V_4, da für V_1' auch V_0,V_1,V_2 bzw. für V_3' auch V_2,V_3,V_4 zu Rate gezogen wurden. Damit ist dem in diesem Abschnitt eingangs erwähnten Problem des mangelnden Bezugs zu anderen Brüdern eines Knotens bei der Wahl des Minimums zu großen Teilen abgeholfen.

5 Diskussion und Bewertung

Die vorangegangenen Abschnitte haben das Vorgehen bei der Anwendung des Prinzips der aktiven Konturen auf das Problem der Erkennung und Verfolgung von Konturen der Zunge dargestellt. Abschließend soll erläutert werden, wie die Ergebnisse des Verfahrens durch die Autoren von [1] überprüft wurden. Auf einem 195Mhz SGI Rechner benötigte die Extraktion der Kontur eines Bildes etwa 10 Sekunden. Die Autoren weisen darauf hin, daß ein Sprachforscher zur manuellen Extraktion der Kontur mehrere Minuten benötige. Hinsichtlich der Validierung wurden die Ergebnisse von 20 Sequenzen zunächst visueller und qualitativer Kontrolle in Zusammenarbeit mit Experten und anschließend numerischen Vergleichen unterzogen. Dabei ist jedoch zu beachten, daß es für die Kontur, die in einem Bild zu sehen ist, keine *ground truth* gibt, d.h. daß kein extrahiertes Ergebnis, weder durch den Experten, noch durch das Verfahren, als absolut exakt eingestuft werden kann. Zur Überprüfung der Ergebnisse wird daher der Ansatz verfolgt, die Abweichung gewonnener Konturen voneinander zu vergleichen. Nachfolgende Gleichung gibt die Summe der quadrierten Abweichungen (mean sum of squared differences) zweier verformbarer Konturen $U = [u_1, u_2, \dots, u_n]$ und $V = [v_1, v_2, \dots, v_n]$ als Maß deren Übereinstimmung an.

$$\mathrm{MSSD}(U,V) = \frac{1}{2n}\left(\sum_{i=1}^{n}\min_j(v_i - u_j)^2 + \sum_{i=1}^{n}\min_j(u_i - v_j)^2\right)$$

Für den Vergleich wurden zwei Sequenzen mit 15 Bildern herangezogen, wobei die erste, links in Abbildung 8.11 einen schwereren und die andere einen typischen Testfall darstellt. Die Abbildung zeigt die Auftragung der einzelnen Fehlersummen im Vergleich und ermöglicht Schlüsse auf die Güte des Verfahrens relativ zu den Abweichungen der beiden Experten. Es zeigt sich, daß mit Ausnahme eines Ausreißers in Bild 6 des typischen Falls, die Abweichung zwischen System und Experten nicht wesentlich verschieden ist von der Abweichung der Experten untereinander.

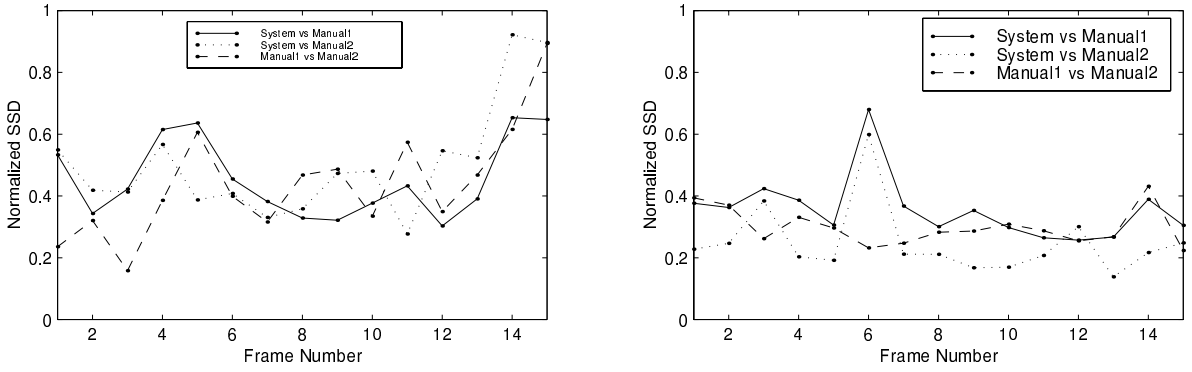


Abbildung 8.11: Normalisierte Mittlere Summe quadratischer Abweichungen

6 Schlußbetrachtung

Diese Arbeit hat ein Verfahren zur Detektion von Konturen der Zunge aus Ultraschallbildern vorgestellt. Das Bildmaterial wurde dabei durch ein eigens entwickeltes System aufgenommen, welches Reproduzierbarkeit der Bildsequenzen ermöglicht. Es wurde auf Schwierigkeiten bei der Auswertung von Ultraschallbildern eingegangen und in dem Prinzip der aktiven, verformbaren Konturen ein geeignetes Werkzeug zu deren Überwindung gefunden. Das vorgestellte Verfahren macht sich die Vorteile dieses

Prinzips zunutze, einerseits um ein Modell der Zunge in die Segmentierung einzubeziehen, andererseits um eine kontinuierliche Verfolgung unter Berücksichtigung von Vorgänger- und Nachfolgerbildern einer Sequenz zu gewährleisten. Wenngleich die Möglichkeit der Einbeziehung anwendungsspezifischen Wissens erwähnt wird, zeigen Akgul et.al. jedoch nicht, wie dies in der algorithmischen Realisierung verwirklicht werden soll. Das Verfahren bedient sich eines laufzeiteffizienten Greedy Algorithmus, dessen Resultate über räumlich-temporale Informationen verbessert werden. Die Genauigkeit des Ansatzes wird durch Vergleich mit der Extraktionsleistung von Experten bemessen und es zeigt sich, daß die Fehler innerhalb der interpersonellen Abweichungen der Experten liegen.

Literaturverzeichnis

[1] Y.S. Akgul, C. Kambhamettu, and M. Stone. Automatic extraction and tracking of the tongue contours. *IEEE Transactions on Medical Imaging*, 18(10):1035–1045, 1999.

[2] A.A. Amini, T.E. Weymouth, and R.C. Jain. Using dynamic programming for solving variational problems in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(9):855–867, 1990.

[3] D. Geiger, A. Gupta, L.A. Costa, and J. Vlontzos. Dynamic programming for detecting, tracking and matching deformable contours. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(3):294–302, March 1995.

[4] S.R. Gunn and M.S. Nixon. A robust snake implementation; a dual active contour. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(1):63–68, January 1997.

[5] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1988.

[6] T. McInerney and D. Terzopoulos. Deformable models in medical image analysis: A survey. *Medical Image Analysis*, 1(2):91–108, 1996.

[7] M. Stone and E.P. Davis. A head and transducer support system for making ultrasound images of tongue/jaw movement. *The Journal of The Acoustical Society of America*, 98(6):3107–3112, December 1995.

[8] M. Stone and L. Lundberg. Three-dimensional tongue surface shapes of english consonants and vowels. *The Journal of The Acoustical Society of America*, 99(6):1–10, 1996.

[9] J.H. van Bemmel and M.A. Musen. *Handbook of Medical Informatics*, chapter 9, pages 127–146. Springer, Heidelberg, Germany, second edition, 2000.

[10] U. Willi. Phonetik und Phonologie. In Angelika Linke, Markus Nussbaumer, und Paul R. Portmann, Herausgeber, *Studienbuch Linguistik*, Band 121 der Reihe *Germanistische Linguistik*, Seiten 401–435. Niemeyer, Tübingen, Germany, dritte Auflage, 1996.

[11] D.J. Williams and M. Shah. A fast algorithm for active contours and curvature estimation. *Computer Vision, Graphics, Image Processing*, 55:14–26, 1992.

Semiautomatische Bildsegmentierung mit “Live-Wire”-Verfahren

von Michael Sauren

Betreuer: Michael Kohnen

Inhaltsverzeichnis

Übersicht

Ziel dieser Ausarbeitung ist es, eine Einführung in die semiautomatische Bildsegmentierung anhand des Prinzips der Live-Wire Technik zu geben. Hierzu betrachten wir Intelligent Scissors, ein Verfahren, welches auf dieser Technik basiert. Die Idee der Live-Wire Technik ist, durch grobes Umfahren mit der Maus ein Objekt innerhalb eines Bildes zu segmentieren. Das Problem der Kantendetektion wird auf die Suche nach einem optimalen Pfad innerhalb eines gewichteten Graphen zurückgeführt. Zur weiteren Verbesserung der Ergebnisse werden die Konzepte des On-the-fly Training und des Abkühlens der Kontur eingeführt. Diese ermöglichen dem Benutzer ein wesentlich freieres Bewegen des Mauszeigers, so dass die Segmentierung schneller durchgeführt werden kann.

Keywords: Bildsegmentierung, Live-Wire, Intelligent Scissors, On-the-fly Training, Abkühlen der Kontur

1 Einführung

Die herkömmlichen Methoden der Bildsegmentierung haben einige Defizite:

- Die manuelle Segmentierung ist zwar ziemlich zuverlässig, erfordert aber einen sehr hohen Zeitaufwand des Benutzers. Weiterhin liegen ihre Schwächen in der Reproduzierbarkeit, wir werden später darauf zurückkommen.
- vollständige automatische Segmentierung ist zwar möglich, funktioniert aber nur in bestimmten eingeschränkten Umgebungen, das heisst, diese Modelle sind nicht in einem breiten Anwendungsfeld einsetzbar.
- Die bisherigen Ansätze zur semiautomatischen Segmentierung erfordern häufig viel Interaktion seitens des Benutzers. Für gewöhnlich wird eine Initialkontur, ein 2D-Template, eine Entscheidungsfunktion oder Ähnliches vorgegeben, woraufhin der entsprechende Algorithmus startet. Häufig ist das Ergebnis dennoch nicht befriedigend, was darin resultiert, dass der Benutzer seine ursprüngliche Eingabe entsprechend verfeinern oder abändern muss. Dies wird solange iteriert, bis ein zufriedenstellendes Ergebnis erzielt wird.

Der Ansatz der Live-Wire Technik ist eine *interaktive* halbautomatische Segmentierungstechnik, bei der der Benutzer in Echtzeit das Ergebnis seiner Aktionen sieht, das heisst die momentan gewählte Kontur. Hierzu wird das Bild als gewichteter Graph betrachtet, wobei die Pixel durch Knoten des Graphen dargestellt werden, mit gerichteten, gewichteten Kanten zu ihren jeweiligen 8 Nachbarpixeln. Die Gewichtung der Kanten setzt sich hierbei aus einer Reihe von Kostenfunktionen zusammen. Mithilfe einer leicht modifizierten Version des Algorithmus von Dijkstra wird ein optimaler Pfad von einem festen Saatpunkt zu jedem anderen Pixel des Bildes berechnet. Hierbei ist zu bemerken, dass die Berechnung in linearer Zeit durchgeführt wird, so dass die Echtzeit-Interaktion gewährleistet werden kann.
Um die benötigte Benutzerinteraktion weiter zu verringern, werden durch “Abkühlen der Kontur” (engl. boundary cooling) neue Saatpunkte entlang der Kontur automatisch generiert. Dies geschieht immer dann, wenn ein Abschnitt der momentan aktiven Teilkontur sich über längere Zeit nicht verändert.
Weiterhin wird das Training dynamisch implementiert, was dazu führt, dass die Kriterien für die Kantendetektion den Gegebenheiten des Bildes angepasst werden, um so auch die gewünschte Kontur zu erhalten, wenn sich ggf. stärkere Kanten oder Rauschen in der unmittelbaren Umgebung befinden.

2 Grundlagen

In diesem Kapitel werden einige wesentliche Grundlagen der Bildverarbeitung kurz erläutert, die zum Verständnis des Verfahrens notwendig sind, ausführlicher werden diese in [1] vorgestellt.

2.1 Die 2D-Bildgeometrie

In der medizinischen Bildverarbeitung arbeitet man mit diskretisierten Bildern, das heisst das originale, kontinuierliche Bild, wird übersetzt in ein diskretes Bild, um darauf mit algorithmischen Techniken arbeiten zu können. Es finden zwei Diskretisierungen des Bildes statt, eine Orts- und eine Wertediskretisierung

(Quantisierung). Erstere teilt das Bild in ein Raster aus N × M Quadraten auf, die Pixel. Hierbei steht N für die Breite, M für die Höhe des Bildes. Insgesamt lässt sich dieses Vorgehen als Abbildung

$$f : M \times N \rightarrow G$$

beschreiben, wobei G die Grauwerte der Bildpunkte darstellt.
In der Pixelebene kann nun die Nachbarschaft eines Pixels definiert werden. So bilden beispielsweise die vier horizontal und vertikal angrenzenden Pixel (auch *direkte Nachbarn* des Pixels genannt) die sogenannte Vierernachbarschaft. Nimmt man auch noch die diagonal angrenzenden Pixel (die *indirekten Nachbarn*) hinzu erhält man die Achternachbarschaft. Zu beachten ist hierbei, dass der Pixel selbst nicht Mitglied seiner Nachbarschaft ist.
Die zweifache Diskretisierung des Bildes bringt natürlich einige Probleme mit sich, da das diskrete Bild immer nur eine Approximation an das kontinuierliche Bild sein kann. Weiterhin gelten die Gesetze der Euklidischen Geometrie im allgemeinen nicht für die Pixelebene. Zum Beispiel schneiden sich zwei nichtparallele Geraden nicht unbedingt in einem gemeinsamen Schnittpunkt. In der 2D-Bildgeometrie ist die Entfernung eines horizontal bzw. vertikal angrenzenden Pixels vom Ursprungspixel 1, die eines diagonal angrenzenden Pixels $\sqrt{2}$.

2.2 Bildfilterung mit Templates

Im Normalfall ist es wünschenswert, das diskretisierte Bild einer Transformation zu unterziehen, wodurch die gewünschte Information nachher einfacher zu extrahieren ist. Ein Beispiel hierfür ist die Verstärkung des Kontrastes eines Bildes. Aufgrund ihrer Eigenschaften bezeichnet man solche Transformationen als *Filter*. Hierzu gibt es z.B. sogenannte Punktoperatoren zur Grauwerttransformation. Wir betrachten hingegen lokale Operatoren auf der Basis von Masken.
Anhand des Nachbarschaftskonzepts kann man leicht sogenannte Masken (engl. templates), durch die Nachbarschaft eines Pixels einschliesslich des Zentralpixels selbst definieren. Wie auch die Nachbarschaft eines Pixels sind Masken nicht auf die direkten und indirekten Nachbarn beschränkt. Es können durchaus auch solche Pixel dazu gehören, die nicht unmittelbar an das mittlere Pixel angrenzen, z.B. bei einer 5 × 5 Maske.
Jeder Stelle der Maske wird ein Gewicht $h(i, j)$ zugeordnet. Der Aufpunkt (der Mittelpunkt der Maske) erhält die Koordinaten $(i, j) = (0, 0)$ und die restlichen Punkte nach folgendem Schema

h(-1,-1)	h(-1,0)	h(-1,1)
h(0,-1)	h(0,0)	h(0,1)
h(1,-1)	h(1,0)	h(1,1)

Um mit verschiedenen Masken erzielte Ergebnisse vergleichen zu können, wird die Summe der positiven Maskenelemente für gewöhnlich normiert.
Die Transformation mit einer Maske M auf ein Bild f deuten wir als Übergang zum Bild s mittels der Transformationsgleichung:

$$s(k, l) = \sum_{(i,j)\in M} f(k + i, l + j)h(i, j) \tag{9.1}$$

der Korrelation von f und s. Hierbei kann natürlich das Problem auftreten, dass eine solche Maske über den Bildrand hinausragt. Es gibt verschiedene Möglichkeiten, dieses Problem zu behandeln, zum Beispiel:

- Randpunkte werden nicht transformiert. Der Nachteil ist, dass das Bild bei wiederholter Transformation kleiner wird.
- Das Bild wird nach aussen hin extrapoliert. Bei der Iteration der Transformation können sich Extrapolationsfehler nach innen fortsetzen.
- Die Maske wird eingeschränkt, so dass sie nicht über den Bildrand hinausragt.
- Das Bild wird periodisch fortgesetzt (wrap-around), das heisst wenn die Maske z.B. über den linken Bildrand herausragt, wird sie am rechten Bildrand fortgesetzt. Dieses Verfahren ist problematisch bei Bildern, für die Periodizität auch nicht nur annähernd gegeben ist.

Von Interesse sind hier insbesondere der Gauss- und der Laplacefilter.
Der Gaussfilter ist eine diskrete Version der Dichtefunktion der Normalverteilung. Durch Einsatz des Gaussfilters kann Rauschen unterdrückt sowie Störungen und Kanten geglättet und Oszillation abgeflacht werden. Für geradzahlige n werden Filtermasken mithilfe der Binomialkoeffizienten $\binom{n}{k}$ folgendermassen definiert:

$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}, \text{ für } n = 2$$

In der ersten Zeile und Spalte stehen jeweils die Binomialkoeffizienten, die Eintragungen im Innern der Maske ergeben sich als Produkt der Zahlen am oberen und linken Rand. Der Normierungsfaktor ist $1/4^n$. Gebräuchlich ist auch der Einsatz von eindimensionalen Gaussfiltern, z.B.:

$$\frac{1}{64}\begin{bmatrix} 1 & 6 & 15 & 20 & 15 & 6 & 1 \end{bmatrix}$$

Der Laplacefilter wird eingesetzt, um vorhandene steile Kanten zu verstärken. Bereiche mit konstantem Grauwert oder konstanter Grauwertsteigerung (sog. Rampen) werden von ihm ignoriert. Im Kontinuierlichen ist der Laplaceoperator definiert durch die Summe der zweiten partiellen Abbildungen:

$$\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \tag{9.2}$$

Im diskreten Fall entspricht dies:

$$f(k+1,l) - 2f(k,l) + f(k-1,l) + f(k,l+1) - 2f(k,l) + f(k,l-1) \tag{9.3}$$

Man erhält also das diskrete Laplace Template:

$$\mathrm{L} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

3 Intelligent Scissors

Wie erwähnt wird das Bild bei der Live-Wire Technik als gewichteter Graph aufgefasst, mit Pixeln als Knoten und gerichteten, gewichteten Kanten, die jedes Pixel mit seinen 8 angrenzenden Nachbarn verbinden. Im folgenden Abschnitt werden wir betrachten, wie sich diese Gewichtung laut [2] errechnet.

3.1 Gewichtung des Graphen, Kostenfunktionen

Seien p und q zwei benachbarte Pixel innerhalb des Bildes. Dann beschreibt $l(p,q)$ die lokalen Kosten der gerichteten Kante von p nach q. Bei der lokalen Kostenfunktion handelt es sich um die gewichtete Summe aus Kostenfunktionen über die in Tabelle 9.1 aufgelisteten Bildeigenschaften.

Diese Komponenten fassen wir zu einer Kostenfunktion der Form:

$$l(p,q) = \omega_Z \cdot f_Z(q) + \omega_G \cdot f_G(q) + \omega_D \cdot f_D(q) + \omega_P \cdot f_P(q) + \omega_I \cdot f_I(q) + \omega_O \cdot f_O(q) \tag{9.4}$$

zusammen, wobei jedes ω die Gewichtung der entsprechenden Funktion ist. Aus Erfahrung haben sich folgende Gewichtungen bewährt: $\omega_Z = 0.3$, $\omega_G = 0.3$, $\omega_D = 0.1$, $\omega_P = 0.1$, $\omega_I = 0.1$, $\omega_O = 0.1$. Diese Werte sind voreingestellt, können aber bei Bedarf leicht angepasst werden.

Bildeigenschaft	Bezeichnung
Laplacescher Nulldurchgang	f_Z
Gradientenbetrag	f_G
Gradientenrichtung	f_D
Kanten-Pixelwert	f_P
"Innen"-Pixelwert	f_I
"Aussen"-Pixelwert	f_O

Tabelle 9.1: Kostenfunktionen

Der Laplacesche Nulldurchgang f_Z, sowie die Gradientenrichtung f_D, besitzen statische Kostenfunktionen, das heisst sie können ohne a priori Wissen über den Bildinhalt berechnet werden. Hingegen besitzen die Pixeleigenschaften f_P, f_I und f_O, dynamische Kostenfunktionen, die erst durch Training berechnet werden können. Der Gradientenbetrag f_G besitzt als einziges Merkmal sowohl statische als auch dynamische Kostenfunktionen. Für die Pixeleigenschaften f_P, f_I und f_O war es nicht möglich, sinnvolle statische Kostenfunktionen zu formulieren, weshalb sie erst nach dem Training eine Bedeutung haben. Dies führt dazu, dass ihre Gewichte gleich Null sind, wenn Training ausgeschaltet ist, oder aber keine Trainingsdaten vorhanden sind. Sowohl der Laplacesche Nulldurchgang, als auch der Gradientenbetrag benutzen zur Bildfaltung verschiedene Maskengrössen, wodurch es ihnen möglich ist, sich einer grossen Vielfalt von verschiedenen Bildtypen anzupassen. Hierzu wird automatisch für jedes Pixel die Maskengrösse gewählt, die am besten der Punktantwort des Aufnahmegerätes entspricht.

Laplacesche Nulldurchgänge

Die Hauptaufgabe des Laplaceschen Nulldurchgangs ist Kantendetektion. Hierzu werden, wie bereits erwähnt, Laplacefilter verschiedener Grössen verwendet, die jeweils einer anderen Standardabweichung einer zweidimensionalen Gaussverteilung entsprechen (siehe [3]). Die verwendeten Standardabweichungen liegen zwischen $\frac{1}{3}$ Pixel und jeweils in $\frac{1}{3}$-Schritten, 2 Pixeln. Hierbei entspricht eine Standardabweichung von $\frac{1}{3}$ Pixel einer 5×5 Maske und eine Abweichung von 2 Pixeln einer 15×15 Maske. Der Grund für den Einsatz Laplacefilter verschiedener Grössen ist, dass kleine Masken sensibler im Bezug auf Details sind, wohingegen grosse Masken Rauschen unterdrücken. Bewährt haben sich 5×5 und 9×9 Masken, welche auch standardmässig verwendet werden. Natürlich können bei kontrastarmen, verrauschten Bildern auch grössere Masken eingesetzt werden. Mithilfe des Laplaceschen Nulldurchgangs wird eine binäre Kostenfunktion realisiert. Liegt ein Pixel auf einem Nulldurchgang, so sind die Teilkosten für Kanten zu diesem Pixel niedrig, ansonsten hoch. Das bedeutet:

$$f_Z(q) = \begin{cases} 0 & , wenn \quad L(q) = 0 \\ 1 & , wenn \quad L(q) \neq 0 \end{cases} \tag{9.5}$$

wobei $I_L(q)$ die Laplacetransformation des Ursprungsbilds I an der Stelle q ist.
Hierbei ist zu bemerken, dass bei einer diskreten Laplacetransformation, wenn überhaupt, nur wenige Pixel genau auf einem Nulldurchgang liegen. Deshalb definiert man den Nulldurchgang hier als Vorzeichenwechsel zwischen zwei Pixeln. Der Pixel, dessen Betrag am kleinsten ist, wird als Nulldurchgang gewählt.
Da man für die Berechnung von f_Z verschiedene Maskengrössen verwendet, hat jede der entsprechend resultierenden Kostenfunktionen ein Gewicht, mit dem sie zu f_Z beiträgt. Standardmässig wird die 5×5 Maske mit 0.45 etwas leichter gewichtet, als die 9×9 Maske mit 0.55. Daraus folgt, dass die Nulldurchgangs-Kosten für ein Pixel genau dann Null sind, wenn der Pixel bei jeder Maskengrösse als Nulldurchgang gewählt wurde. Genauso sind die Kosten genau dann 1, wenn bei keiner der Maskengrössen dort ein Nulldurchgang gefunden wurde. In jedem anderen Fall ist $0 < f_Z < 1$. Da es sich hier um eine binäre Eigenschaft handelt, unterscheidet der Laplacesche Nulldurchgang nicht zwischen "starken" und "schwachen" Kanten (also Kanten mit hohen bzw. niedrigen Gradienten). Hierzu verwendet man den Gradientenbetrag.

Gradientenbetrag

Zur Berechnung des Gradientenbetrags werden die partiellen Ableitungen des Bildes nach x und y gebildet, mithilfe von Gaussfiltern verschiedener Grösse. Auf diese Weise erhält man die partiellen horizontalen (I_x) und vertikalen (I_y) Gradientenbeträge des Bildpunktes. Der Gradientenbetrag G eines Bildpunktes wird durch $G = \sqrt{I_x^2 + I_y^2}$ approximiert. Nun sollen die statischen Gradientenbetragskosten aber niedrig sein für hohe Gradienten (also starke Kanten) und hoch für niedrige Gradienten (schwache Kanten). Dies erreicht man, indem der Gradientenbetrag vom maximal erreichbaren Gradientenbetrag subtrahiert wird. Weiterhin dividiert man noch durch das Maximum, um die Kosten auf 1 zu normieren, bevor man mit der Gewichtung ω_G multipliziert. So ergeben sich die statischen Kosten zu:

$$f_G = \frac{max(G') - G'}{max(G')} = 1 - \frac{G'}{max(G')} \tag{9.6}$$

mit $G' = G - min(G)$.
Wie auch beim Laplaceschen Nulldurchgang werden zur Berechnung der Gradientenbetragskosten verschiedene Maskengrössen verwendet. Damit man die verschieden Resultate vergleichen kann, werden die Masken ebenso normalisiert. Im Gegensatz zur Vorgehensweise beim Laplaceschen Nulldurchgang werden die Kosten hier nicht als gewichtete Summe der Einzelkosten berechnet. Vielmehr wird die Maske verwendet, welche die Originaleigenschaften der Kante am Besten annähert. Zur Entscheidung, welche Maske das ist, gibt es zwei Möglichkeiten. Hier wird die Methode verwendet, bei der die Grösse des Laplacefilters, der das steilste Gefälle an einem Nulldurchgang produziert, als Maskengrösse benutzt wird.
Als Alternative dazu kann man für jedes Pixel die Maske wählen, die für diesen Pixel den grössten Gradientenbetrag hervorruft. Diese Methode bietet sich an für kontrastarme, stark verrauschte Bilder, bei denen die Informationen über den Nulldurchgang unzuverlässig sind.

Gradientenrichtung

Mithilfe der Gradientenrichtung ordnet man starken Richtungsänderungen eines Kantenverlaufs hohe Kosten zu. Hierzu betrachtet man den Einheitsvektor $D(p)$ der Gradientenrichtung am Punkt p. Um die Gradientenrichtung an einem Pixel p zu bestimmen, ist es zum Beispiel möglich, die partiellen Ableitungen nach x und y zu berechnen, diese Anhand ihres Gradientenbetrags zu gewichten und anschliessend den Mittelwert zu berechnen. Man definiert nun $D'(p)$ als den zu $D(p)$ senkrechten Einheitsvektor. Weiterhin definiert man nun den normalisierten Einheitsverbindungsvektor zwischen p und q als:

$$L(p,q) = \frac{1}{\|p-q\|} \begin{cases} q-p & ,wenn \quad D'(p)\cdot(q-p) \geq 0 \\ p-q & ,wenn \quad D'(p)\cdot(q-p) < 0 \end{cases} \tag{9.7}$$

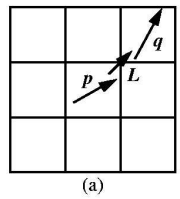
bei dem die Richtung so gewählt wird, dass die Differenz zwischen p und der Richtung des Vektors minimiert wird. Weiterhin definiert man nun die Skalarprodukte

$$\begin{aligned} d_p(p,q) &= D'(p)\cdot L(p,q) \\ d_q(p,q) &= L(p,q)\cdot D'(q) \end{aligned} \tag{9.8}$$

und erhält als Formel für die Gradientenrichtungkostenfunktion:

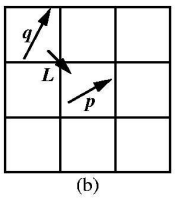
$$f_D(p,q) = \frac{2}{3\pi}\{\arccos[d_p(p,q)] + \arccos[d_q(p,q)]\} \tag{9.9}$$

Der Verbindungsvektor hat die Eigenschaft, einer Kante zwischen zwei benachbarten Pixeln, die ähnliche Gradientenrichtungen besitzen, aber (fast) senkrecht zu ihrem Verbindungsvektor stehen, hohe Kosten zuzuordnen. Wenn die Gradientenrichtungen der Pixel gegenseitig und zum Verbindungsvektor ähnlich sind, sind die Kosten niedrig. Insgesamt soll gewährleistet werden, dass der aktuellen Kante gefolgt wird.



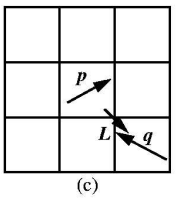
(a)

$D'(p) = [0.870, 0.492]$
$D'(q) = [0.473, 0.881]$
$L(p,q) = [0.707, 0.707]$
$d_p(p,q) = D'(p) \cdot L(p,q) = 0.964$
$d_q(p,q) = L(p,q) \cdot D'(q) = 0.957$
$f_D(p,q) = \frac{2}{3\pi}\{\text{acos}[d_p(p,q)] + \text{acos}[d_q(p,q)]\} = 0.120$



(b)

$D'(p) = [0.870, 0.492]$
$D'(q) = [0.473, 0.881]$
$L(p,q) = [0.707, -0.707]$
$d_p(p,q) = 0.267$
$d_q(p,q) = -0.288$
$f_D(p,q) = 0.671$



(c)

$D'(p) = [0.870, 0.492]$
$D'(q) = [-0.881, 0.473]$
$L(p,q) = [0.707, -0.707]$
$d_p(p,q) = 0.267$
$d_q(p,q) = 0.957$
$f_D(p,q) = 0.880$

Abbildung 9.1: Gradientenrichtung

Pixelwerte

Pixelwerte als dynamische Kostenfunktionen haben, wie bereits erwähnt, erst nach dem Training eine Bedeutung. Als Kanten-Pixelwert wird der Wert genommen, den das Pixel im Originalbild hatte und normiert. Typische Grauwerte liegen zwischen 0 und 255, deshalb skaliert man mit dem Faktor $\frac{1}{255}$. Also ist der Kanten-Pixelwert an einem Punkt p

$$f_P(p) = \frac{1}{255} \cdot I(p) \tag{9.10}$$

wobei $I(p)$ dem Wert von p im Originalbild entspricht. Für die "Innen"- bzw. "Aussen"-Pixelwerte verfährt man ähnlich. Hier wird als Wert wieder der Wert eines Pixels aus dem Ursprungsbild gewählt, allerdings um einen bestimmten Betrag entfernt. Für den "Innen"-Pixelwert $f_I(p)$ von p wird der Wert genommen, der in einer Entfernung k in Gradientenrichtung vorliegt, für den "Aussen"-Wert $f_O(p)$ wird die Entfernung -k (also entgegen der Gradientenrichtung) genommen. Als Berechnungsvorschrift ergibt sich also

$$\begin{aligned} f_I(p) &= \tfrac{1}{255} \cdot I(p + k \cdot D(p)) \\ f_O(p) &= \tfrac{1}{255} \cdot I(p - k \cdot D(p)) \end{aligned} \tag{9.11}$$

wobei $D(p)$ der bereits eingeführte Einheitsvektor der Gradientenrichtung ist. Die Entfernung k ist entweder ein konstanter, vom Benutzer festgelegter Wert, oder sie wird 1 Pixel grösser gewählt, als die Hälfte der optimalen Maskengrösse am Punkt p. Zu beachten ist, dass die errechnete "Innen"- bzw. "Aussen"-Position im Normalfall nicht genau in der Mitte eines Pixels liegt. Deshalb kann man entweder den am nächsten gelegenen Pixel nehmen, oder aus den umliegenden vier Pixeln interpolieren.

3.2 Training

Ein häufig auftretendes Problem ist, dass das Objekt, das man segmentieren möchte, keine starken Konturen hat. Die statischen Kostenfunktionen sind aber derart beschaffen, dass sie starke Kanten gegenüber schwachen favorisieren. Um solche Objekte dennoch erkennen zu können, verwendet man ein sogenanntes *On-the-fly Training*. Das heisst, das Training wird hierbei nicht in einer separaten Phase, sondern in Echtzeit während des Segmentierens durchgeführt. Das hat den Vorteil, dass die dynamischen Kostenfunktionen sich den Gegebenheiten des untersuchten Objektes anpassen. Betrachten wir hierzu Abbildung 9.2. Zu sehen ist hier eine CT Aufnahme des linken Herzventrikels. Man beachte, dass der Rand des Ventrikels, im Gegensatz zum äusseren Rand des Herzens, einen eher niedrigen Gradientenbetrag besitzt. Wie wir gesehen haben, werden bei den statischen Gradientenbetragskosten niedrige Kosten mit hohen

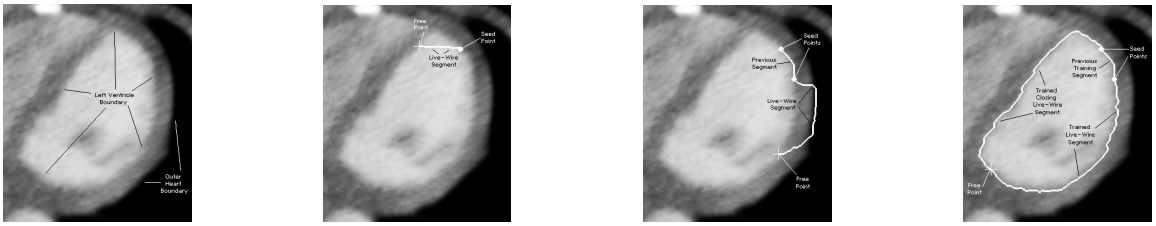


Abbildung 9.2: Aufnahme des linken Herzventrikels

Gradientenbeträgen assoziiert, hingegen haben niedrige Gradientenbeträge hohe Kosten. Dies führt im Fall des Herzventrikels dazu, dass das aktive Live-Wire Segment nicht den gewünschten Ventrikelrand überdeckt, sondern an der äusseren Herzwand (die einen höheren Gradientenbetrag hat) verläuft (siehe Abbildung). Um das zu verhindern, wird beim *On-the-fly Training* eine sogenannte Histogrammanpassung durchgeführt. Das heisst, dass die trainierte Kostenfunktion solche Pixel favorisiert (ihnen niedrige Kosten zuordnet), die ähnlich den Pixeln entlang des Trainingssegments sind. Siehe hierzu auch Abbildung 9.2. Man sieht in Abbildung 9.2 (d), dass bereits ein kurzes Trainingssegment ausreicht, damit die gewünschte, schwächere Kontur gewählt wird.
Um das Training effizienter zu gestalten, werden für die trainierbaren Kostenfunktionen f_P, f_I, f_O und f'_G im Voraus Wertebilder berechnet. Hierbei ist $f'_G = \frac{G'}{max(G')}$ der skalierte Gradientenbetrag. Diese Wertebilder dienen dazu, während des Trainings untersuchte Pixelwerte als Indizes für das entsprechende Komponentenhistogramm zu wählen. Die Bilder werden berechnet, indem f_P, f_I, f_O und f'_G wie folgt skaliert und gerundet werden. Seien I_P, I_I, I_O und I_G jeweils die zu f_P, f_I, f_O und f'_G entsprechenden Wertebilder, dann werden diese berechnet durch

$$\begin{aligned} I_P &= \lfloor (n_P - 1) f_P + 0,5 \rfloor \\ I_O &= \lfloor (n_I - 1) f_I + 0,5 \rfloor \\ I_I &= \lfloor (n_O - 1) f_O + 0,5 \rfloor \\ I_G &= \lfloor (n_G - 1) f'_G + 0,5 \rfloor \end{aligned} \tag{9.12}$$

wobei $n_P, n_I, n_O = 256$ und $n_G = 1024$ die Anzahl der Einträge des entsprechenden Histogramms sind. Die Komponentenbilder haben zwei Aufgaben. Zum einen, dynamische Histogramme zu erstellen, zum anderen dienen sie als Indizes für die dynamischen "Cost Maps" zur Berechnung von Kantenkosten. Damit das Training sich an stetige Änderungen der Kantencharakteristik anpassen kann, wird für die trainierten Kostenfunktionen nur der letzte Teil der definierten Objektkontur betrachtet. Die Trainingslänge t beschreibt, wieviele der Konturpixel vor dem letzten gesetzten Saatpunkt für die Generierung der Trainingsstatistiken herangezogen werden. Eine monoton sinkende Gewichtsfunktion w bewirkt, dass der zuletzt untersuchte Pixel im Verhältnis zu den anderen am meisten Einfluss hat. Der Trainingsalgorithmus untersucht die Wertebilder entlang der letzten t Pixel des Kantensegments und erhöht die Einträge im Komponentenhistogramm entsprechend des Pixelgewichts, um so ein Histogramm für jede Trainingskomponente zu generieren.
Zu bemerken ist, dass das Training am effektivsten für Objekte ist, deren Kanteneigenschaften sich nicht allzusehr, oder wenigstens nicht allzu schnell ändern. Bei Objekten mit plötzlichen, drastischen Änderungen der Kanteneigenschaften kann Training sogar kontraproduktiv sein. Für diesen Fall ist es dem Benutzer möglich, das Training jeder Zeit aus- und wieder einzuschalten. Beispielsweise würde man das Training kurzzeitig ausschalten, ehe man an eine plötzliche Änderung gelangt. Die Trainingslänge t ist mit typischerweise 32 bis 64 Pixeln relativ kurz, was häufig zu verrauschten Verteilungen führen kann. Um die Komponentenhistogramme zu entrauschen kann ein eindimensionaler Gaussfilter auf sie angewendet werden. Danach wird jedes Histogramm skaliert und invertiert, durch die daraus resultierenden Histogrammwerte werden Kostengrössen für die "Innen"- und "Aussen"-Pixelwerte dargestellt (Cost Map). Die maximalen lokalen Verbindungskosten M seien definiert als die grösstmöglichen Kosten, die man durch Summation über die Komponentenkosten erhalten kann. Die maximalen Verbindungskosten einer Komponente sind das Produkt aus dem Gewicht ω der Komponente und der maximalen Verbindungskosten M. So sind zum Beispiel die maximalen Gradientenbetrag-Verbindungskosten $M_G = \omega \cdot M$. Diese maximalen Komponentenkosten werden beim Überführen eines Histogramms in eine Komponenten-Cost Map als Skalierungsfaktoren verwendet. Sei nun h_G das untersuchte und geglättete Histogramm zum Gradientenbetrag. Dann ergibt sich die dynamische Gradientenbetrags-Cost Map, m_G wie folgt:

$$m_G = \left\lfloor \frac{max(h_G) - h_G}{max(h_G)} M_G + 0,5 \right\rfloor = \left\lfloor M_G(1 - \frac{h_G}{max(h_G)} \right\rfloor \tag{9.13}$$

Die Division durch $max(h_G)$ dient der Skalierung des Histogramms. Für die anderen dynamischen Komponenten-Kostenkarten m_P, m_I und m_O gilt jeweils eine entsprechende Gleichung, die man durch Substitution von h_G durch h_P, h_I und h_O sowie von M_G durch M_P, M_I und M_O erhält. Da der Gradientenbetrag sowohl statische, als auch dynamische Kostenfunktionen hat, ist es von Vorteil, diese zu kombinieren. Dieser Fall tritt zum Beispiel ein, wenn nur weniger als t Konturpixel zur Verfügung stehen, wodurch die Verteilung noch stärker verrauscht und dadurch weniger aussagekräftig ist. Aus diesem Grund führt man eine minimale Grösse s ein, die besagt, wieviele Konturpixel für eine verlässliche Untersuchung notwendig sind. Wenn ein Kontursegment weniger als s Pixel enthält, sind nicht genügend Daten vorhanden, um eine verlässliche dynamische Gradientenbetrags-Cost Map zu erstellen. In diesem Fall wird die statische Gradientenbetragskostenfunktion mit der untersuchten Cost Map m_G kombiniert. Hierzu werden die minimale Grösse s und die Anzahl der tatsächlich untersuchten Pixel $t_s \leq t$ verwendet. Eine angeglichene statische Gradientenbetragskostenfunktion wird berechnet, indem s und t_s mit der Gradientenbetrags-Cost Map m_G kombiniert werden. Daraus ergibt sich die kombinierte Gradientenbetrags-Cost Map m'_G zu

$$m'_G(x) = \begin{cases} \min(m_G(x), \left\lfloor M_G[1 - \frac{x(s-t_s)}{(n_G-1)s} + 0,5] \right\rfloor & , wenn \quad t_s < s \\ m_G(x) & , wenn \quad t_s \geq s \end{cases} \tag{9.14}$$

wobei $x = 0, 1, ..., n_G - 1$ der Definitionsbereich von m'_G ist. Zu beachten ist, dass im Falle $t_s = 0$ (das heisst, wenn keine Trainingsdaten verfügbar sind) m'_G einfach die unangeglichene statische Kostenfunktion berechnet. m'_G berechnet also sowohl die statische als auch die dynamische Gradientenbetragskostenfunktion. Letztendlich erhält man, bei einem Input von t_s zusammenhängenden Punkten und p_i, mit $i = 0, 1, ..., t_s - 1$, so dass $p_i \neq p_i + 1$ und $\|p_{i+1} - p_i\| \leq \sqrt{2}$, den folgenden Trainingsalgorithmus:

Input:
- $t_s \leq t$ {Anzahl der untersuchten Konturpunkte }
- p_i für $i = 0, 1, ..., t_s - 1$ {zusammenhängende Punktsequenz }
- σ {Maskengrösse }

Datenstrukturen:
- w {Vektor der Trainingsgewichte }
- h_G, h_P, h_I, h_O {Komponentenhistogramme }

Output:
- m'_G, m_P, m_I, m_O {Cost Maps der trainierten Komponenten}

Algorithmus:

```
Initialisiere Histogramme h_G, h_P, h_I, h_O als leer
FOR i = 0 TO t_s − 1 DO
  Trage jeweiligen Wert am Punkt p_i entsprechend seines Gewichts w(i) in jeweiliges Histogramm ein
OD
Glätte die Histogramme mithilfe von σ
Skaliere und invertiere die Histogramme
IF s > t_s THEN
  FOR v = 0 TO n_G − 1 DO
    Kombiniere statische und dynamische Gradientenbetragskomponente
  OD
FI
```

Da Training für den Laplaceschen Nulldurchgang und die Gradientenrichtung nicht zur Verfügung steht, werden diese Kosten im Voraus berechnet und zu einer statischen Nachbarschaft-Cost Map kombiniert. Dadurch spart man aufwendige Berechnungen während der interaktiven Bildsegmentierung. Diese kombinierten Kosten werden für jede Kante berechnet, indem die skalierten und gerundeten lokalen statischen

Kostenfunktionen summiert werden. Für einen gegeben Punkt p und einen benachbarten Punkt q ist die statische Verbindungs-Cost Map l_S definiert durch

$$l_S(p,q) = \lfloor M_Z \cdot f_Z(q) + 0,5 \rfloor + \lfloor M_D \cdot f_D(p,q) + 0,5 \rfloor \tag{9.15}$$

wobei M_Z und M_D die maximalen Kosten für den Laplaceschen Nulldurchgang bzw. die Gradientenrichtung repräsentieren. Da jeder Pixel acht Nachbarn besitzt, benötigt die vorberechnete Kostenkarte l_S für N Bildpixel $8N$ Kostenwerte.
Zum Schluss werden die Gradientenbetragswerte noch entsprechend der euklidischen Entfernung gewichtet. Gradientenbetragskosten zu direkten Nachbarn werden mit 1 gewichtet, zu indirekten Nachbarn mit $\sqrt{2}$. Damit ergibt sich folgende Gewichtungsfunktion w_N für einen Nachbarn q von p

$$w_N(p,q) = \begin{cases} \sqrt{2} & ,wenn \quad L_x(p,q) \neq 0 \wedge L_y(p,q) \neq 0 \\ 1 & ,wenn \quad L_x(p,q) = 0 \vee L_y(p,q) = 0 \end{cases} \tag{9.16}$$

wobei L_x und L_y die horizontale bzw. vertikale Komponente des eingeführten Verbindungsvektors L beschreiben.
Aufbauend auf Gleichung (4) für die lokale Kostenfunktion l, definieren wir eine aktualisierte Kostenfunktion l, mit einem Wertebereich zwischen 0 und $M-1$. Diese berücksichtigt das Training, die vorberechnete statische Cost Map l_S sowie die Gewichtungsfunktion bezüglich der euklidischen Entfernung. Insgesamt ergibt sich als Kostenfunktion für einen Nachbarn q eines Pixels p

$$l(p,q) = l_S(p,q) + w_N(p,q) \cdot m_G(I_G(q)) + m_P(I_P(q)) + m_I(I_I(q)) + m_O(I_O(q)) \tag{9.17}$$

3.3 Graphsuche

Im Gegensatz zu vielen anderen Ansätzen, verwendet Intelligent Scissors eine andere Formulierung für den Graphen sowie einen anderen Algorithmus zur Graphsuche. Alternativ zu anderen Ansätzen basiert die Graphformulierung hier auf Pixeln und nicht auf Zwischenräumen zwischen den Pixeln. Weiterhin wird eine modifizierte Version des Algorithmus von Dijkstra (siehe [4]) verwendet. Diese Unterschiede bringen einige Vorteile mit sich:

- Es werden keine Beschränkungen für die Suche oder Untersuchung formuliert. Dies garantiert mehr Freiheit und Vielfältigkeit für die Objektkonturen.
- Die aktive Liste wird durch ein angepasstes Bucket Sort Verfahren in linearer Zeit sortiert.
- Es werden keine a priori Pixel bzw. Zielpunkte spezifiziert. Da optimale Pfade zu allen Pixeln berechnet werden, ist die Echtzeit-Interaktion erst möglich.

Der Algorithmus wird initialisiert mit einem Start- bzw. Saatpunkt s mit kumulativen Kosten von 0, in einer ansonsten leeren Liste L, der sogenannten aktiven Liste. Ein Punkt p wird in diese Liste eingefügt entsprechend seiner gesamten bzw. kumulativen Kosten $g(p)$. Alle anderen Punkte des Bildes werden effektiv mit unendlichen Kosten initialisiert. Nach der Initialisierung erzeugt der Algorithmus, basierend auf der lokalen Kostenfunktion l einen minimalen (bezogen auf die Kosten) Spannbaum. In jeder Iteration wird der Punkt p mit den minimalen kumulativen Kosten (also der Punkt am Anfang der aktiven Liste) aus der Liste entfernt und expandiert. Dies geschieht, indem die Gesamtkosten zu jedem noch nicht expandierten Nachbarn von p berechnet werden. Für jeden Nachbarn q von p sind die kumulativen Kosten zu q die Summe aus den bisherigen Gesamtkosten zu p und der lokalen Verbindungskosten von p nach q, also $g_{tmp} = g(p) + l'(p,q)$. Sind die neu berechneten Kosten niedriger als die bisherigen, also $g_{tmp} < g(q)$, so werden $g(q)$ die neuen, niedrigeren Kosten zugewiesen, und ein "optimaler Pfad pointer" von q nach p gesetzt. Sind die kumulativen Kosten zu allen nicht expandierten Nachbarn von p errechnet und alle nötigen Pointer gesetzt, wird p als expandiert markiert. Es wird weiter iteriert, bis alle Punkte als expandiert markiert sind. Die aktive Liste wird implementiert als Array von Listen. Die Grösse des Arrays ist hierbei der Wertebereich der diskreten lokalen Kosten M. Jede der Sublisten enthält Punkte mit den gleichen kumulativen Kosten, wobei die Reihenfolge der Punkte innerhalb der Subliste keine

Rolle spielt. Folglich sind diese Sublisten einfach zusammenhängende Stacks. Mit $L(i) \downarrow q$ beschreiben wir, dass ein Punkt q mit kumulativen Kosten c in die aktive Liste einsortiert wird, indem q bei Index $i = c \bmod M$ auf den Stack gepusht wird. Vorausgesetzt, dass M konstant gleich 2 ist, kann die Modulo-Berechnung durch eine schneller zu berechnende, bitweise UND-Verknüpfung realisiert werden, nämlich $i = c \wedge (M-1)$. Folglich benötigt man, um einen Punkt in die aktive Liste einzusortieren, eine bitweise UND-Verknüpfung, das Finden des Index in der Liste sowie zwei Zeigerzuweisungen. Sei nun $N(p)$ die Menge der Nachbarpixel von p, $e(p)$ eine boolesche Funktion, die beschreibt, ob p bereits expandiert wurde, und $ptr(q)$ der "Optimale Pfad"-Pointer für Punkt q. Dann ergibt sich der folgende Algorithmus zur uneingeschränkten Graphsuche:

Input:
s {Start- bzw. Saatpunkt}
$l(p,q)$ {Kostenfunktion für die lokalen Verbindungskosten zwischen p und q}
Datenstrukturen:
L {den totalen Kosten nach sortierte Liste der aktiven Pixel (wird leer initialisiert) }
$N(p)$ {Die Achter-Nachbarschaft von p }
$e(p)$ {Boolesche Funktion, die aussagt, ob p expandiert ist }
$g(p)$ {Funktion der kumulativen Kosten vom Saatpunkt nach p }
Output:
ptr {Pointer von jedem Pixel, der den optimalen Pfad angibt}
Algorithmus:
Initialisiere aktive Liste L mit Saatpunkt s und Kosten von s gleich Null
WHILE $L \neq \emptyset$ DO {Solange nichtexpandierte Punkte existieren }
  Entferne Punkt p mit minimalen Kosten aus der aktiven Liste
  Markiere p als expandiert
  FÜR JEDES NICHT EXPANDIERTE $q \in N(p)$
    Berechne kumulativen Kosten $g_tmp(q)$ zu Nachbar q
    IF $q \in L \wedge g_tmp(q) < g(q)$ THEN
      entferne q aus der Liste und füge ihn, entsprechend $g_tmp(q)$ neu ein.
    FI
    IF $q \notin L$ THEN
      trage q entsprechend $g_tmp(q)$ in L ein und setze $ptr(q) = p$
    FI
OD

Dieser Algorithmus existiert in zwei Implementationen, abhängig davon, ob das Training ein- oder ausgeschaltet ist. Bei eingeschaltetem Training werden die lokalen Verbindungskosten $l(p,q)$ berechnet wie definiert. Ohne Training lautet die entsprechende Funktion

$$l'(p,q) = l_S(p,q) + w_N(p,q) \cdot m_G(I_G(q)) \tag{9.18}$$

Hierbei berechnet die Gradientenbetrags-Cost-Map-Funktion einfach die statische inverse lineare Rampe. Gleichung (18) wird bei ausgeschaltetem Training verwendet, da so die Effizienz bei der Echtzeit-Interaktion verbessert wird.
Um den nächsten Punkt mit minimalen kumulativen Kosten c aus der aktiven Liste zu entfernen (dargestellt durch $p \leftarrow \min(L)$), ist es notwendig das Array von Sublisten zu durchsuchen, bis man an die erste Subliste gelangt, die mindestens einen Punkt enthält. Begonnen wird die Suche an der Stelle, die den Kosten des letzten expandierten Punkts entspricht, wenn das Ende das Arrays erreicht wird, wird am Anfang des Arrays fortgefahren, solange bis man eine nichtleere Subliste gefunden hat. Dies leistet folgender Algorithmus

$c = c - 1$ { kompensiert das Inkrementieren in der Schleife}
REPEAT { Suchen des nächsten nichtleeren Stack mit niedrigsten Kosten }
  $c = c + 1$ { Inkrementieren von c auf nächsthöheren kumulativen Kosten}
  $i = c \wedge (M-1)$ { Berechnen des Array-Index}
UNTIL $L(i) \neq \emptyset$
$p \uparrow L(i)$ { Entfernen des nächsten Punktes mit minimalen Kosten von Stack i }

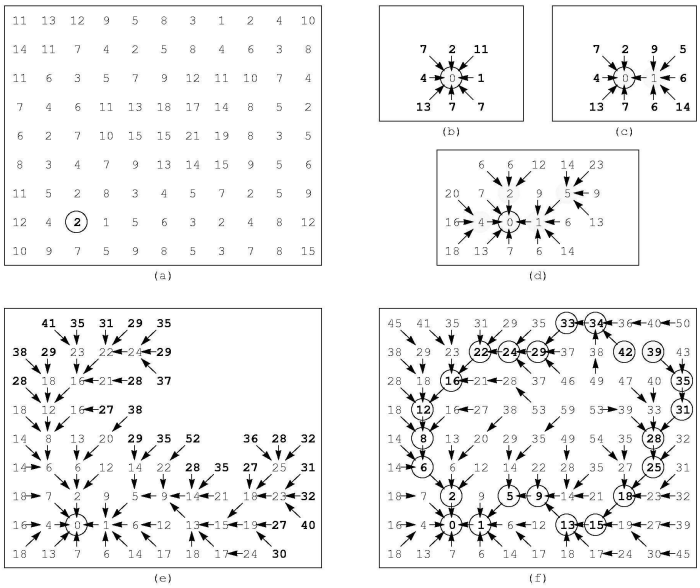


Abbildung 9.3: (a) Anfängliche lokale Kostenmatrix. (b) Saatpunkt wurde expandiert. (c) 2 Punkte expandiert. (d) 5 Punkte expandiert. (e) 47 Punkte expandiert. (f) Berechnung der kumulativen Kosten abgeschlossen, mit 2 eingezeichneten Beispielpfaden.

wobei $p \uparrow L(i)$ heisst, dass der Punkt p vom Stack, der sich bei Index i befindet, entfernt wird. Das Problem beim Entfernen des nächsten Punkts ist, dass es nicht in konstanter Zeit geschieht. Im schlimmsten Fall benötigt man $M-1$ Iterationen, um den nächsten Punkt zu finden. Wenn wir aber davon ausgehen, dass die Wahrscheinlichkeit, dass ein Punkt in die Liste aufgenommen wird für jeden Index gleich gross ist, kann man erwarten, dass sich die Punkte auf der aktiven Liste im Normalfall an den Indizes befinden, die knapp über dem Index des zuletzt expandierten Punkts liegen. Als Vergleich hierzu stelle man sich einen Schneepflug vor, der in einem Schneesturm ständig im Kreis fährt. Dann befindet sich der tiefste Schnee immer genau vor dem Pflug. Für den Fall, dass für einen Punkt, der sich bereits in der aktiven Liste befindet, neue, niedrigere kumulative Kosten berechnet werden, muss dieser Punkt aus der Liste entfernt und an einem entsprechenden neuen Index wieder eingefügt werden. Das Entfernen eines Punkts q vom Stack bei Index i (dargestellt als $q \leftarrow L(i)$) geschieht, wie das Hinzufügen eines Punkts zur Liste, in konstanter Zeit. Um aufwendiges Suchen und Neuzuordnen in den Stacks (als einzeln verkettete Liste implementiert) zu vermeiden, werden die Daten des zu entfernenden Punktes einfach mit denen des obersten Stackelements überschrieben und dann das oberste Element entfernt. Da die Reihenfolge der Punkte in den Sublisten nicht von Belang ist, ist dies ohne weiteres möglich. Für jeden Pixel speichern Pointer die Position jedes Punkts auf der aktiven Liste. Ebenso werden die kumulativen Kosten für jeden Pixel gespeichert und damit der Index des entsprechenden Stacks.
Zur Veranschaulichung wie der Graphsuchalgorithmus arbeitet, betrachten wir Abb. 9.3.
In Abbildung 9.3(a) ist die anfängliche Karte der lokalen Kosten zu sehen, der Saatpunkt ist eingekreist. Die lokalen Kosten sind hier der Einfachheit halber pixel- anstelle von kantenbasiert dargestellt. In Abbildung 9.3(b) ist der Saatpunkt expandiert worden. Es ist gut zu erkennen, wie die Kosten der diagonalen Nachbarn des Saatpunktes entsprechend der euklidischen Distanz gewichtet wurden. Durch die Expansion weiterer Punkte können sich die kumulativen Kosten von aktiven Punkten ändern, wenn niedrigere kumulative Kosten errechnet werden (siehe Abb. 9.3(c)). Die Abbildungen 9.3(d-f) zeigen die Karte der kumulativen Kosten sowie die optimalen Pfade zu verschiedenen Zeitpunkten der Berechnung.

3.4 Erweiterung des Verfahrens

Nach der Generierung der Zeiger für die optimalen Pfade kann man sich mittels eines “freien Punktes” (entspricht der Position des Mauszeigers) Kontursegmente anzeigen lassen. Der optimale Pfad vom freien Punkt zum Saatpunkt wird als “Live Wire” angezeigt, der bei entsprechender Positionierung des Mauszeigers in der Nähe der gewünschten Kontur an dieser entlang läuft. Jetzt kann es natürlich vorkommen,

dass der Live Wire irgendwann von der gewünschten Kontur abweicht. In diesem Fall setzt man einen neuen Saatpunkt, kurz vor der Stelle an der die Abweichung auftritt. Das führt dazu, dass das bisher gewählte Kontursegment festgelegt und nicht weiter betrachtet wird und der Graphsuchenalgorithmus mit dem neuen Saatpunkt iteriert wird, also für jeden Punkt einen optimalen Pfad zu diesem berechnet. Da es für jeden Punkt nur einen optimalen Pfad geben kann, ist es nicht möglich, ein Objekt mit nur einem Saatpunkt zu umranden, man benötigt immer mindestens zwei Saatpunkte. Wenn ein neuer Saatpunkt gesetzt wird, wird die Karte der optimalen Pfade zum Startpunkt beibehalten. Dies dient dazu, vom momentanen freien Punkt, zusätzlich zum optimalen Pfad zum Saatpunkt, einen optimalen Pfad zurück zum Startpunkt zu berechnen. Dadurch wird das Abschlusssegment der Objektkontur automatisch generiert und muss nicht vom Benutzer definiert werden.

Cursor Snap

Cursor Snap ist eine Eigenschaft, die es dem Benutzer erlaubt, Start- bzw. Saatpunkte nicht genau auf der Kante eines Objekts platzieren zu müssen. Das genaue Platzieren ist häufig schwierig und kostet viel Zeit, was dem Prinzip von Live Wire widerspricht. Ungenaue Platzierung hingegen führt zu unerwünschten Spitzen in der Kontur. Um dies zu vermeiden, bzw. das Setzen von Saatpunkten zu vereinfachen, bewirkt Cursor Snap, dass der Mauszeiger auf den Pixel mit dem höchsten Gradientenbetrag innerhalb einer festgelegten Nachbarschaftsumgebung “springt”. Die Grösse dieser Umgebung kann vom Benutzer bestimmt werden und zwischen 1×1 (es findet kein Cursor Snap statt) und 19×19 (der Zeiger kann jeweils bis zu 9 Pixel horizontal und vertikal springen). Der freie Punkt springt also immer auf den Punkt innerhalb seiner Nachbarschaft, der den “besten Kantenpunkt” repräsentiert.

Abkühlen der Kontur

Obwohl es prinzipiell möglich ist, eine geschlossene Kontur durch zwei Saatpunkte (genauer: einen Start- und einen Saatpunkt) zu definieren, reicht dies in der Praxis selten aus. Für einfache Objekte benötigt man typischerweise zwei bis fünf Saatpunkte, bei komplexeren Objekten mitunter wesentlich mehr. Das Setzen der Punkte wird durch Cursor Snap zwar vereinfacht, aber bei vielen benötigten Punkten wird trotzdem ein Grossteil der Zeit auf das Setzen dieser Saatpunkte verwendet. Um dem Benutzer das häufige Setzen neuer Saatpunkte zumindest teilweise abzunehmen, “kühlt” das momentan aktive Kontursegment ab, was letzten Endes dazu führt, dass ein Pixel dieses Segments automatisch als neuer Saatpunkt gewählt wird. Eine Tatsache, die sich das Abkühlen zu Nutze macht, ist, dass unterschiedliche Pfade ein Teilstück gemeinsam mit anderen Pfaden haben. Besonders wenn der Mauszeiger in der Nähe des Objektes entlanggeführt wird, sieht man, dass sich der optimale Pfad in der Nähe des freien Punktes ändert, aber der Rest unverändert bleibt (siehe Abbildung 9.4). Wenn also die optimalen Pfade zweier verschiedener Pixel einen gemeinsamen Punkt haben, dann ist der Teilpfad von diesem Punkt zurück zum Saatpunkt für beide identisch. Je weiter man den freien Punkt vom Saatpunkt entfernt, desto länger wird also der Teil des Pfades, der sich nicht ändert.
Das Abkühlen geschieht, indem man einen Pixel der aktiven Kontur mit “stabiler Vergangenheit” findet, das heisst einen Pixel, ab dem sich der Pfad zurück zum Saatpunkt seit längerer Zeit nicht geändert hat. Hierzu werden für alle Pixel des Bildes zwei Zähler geführt. Der eine besagt, wie lange (in Milisekunden) ein Pixel Teil der aktiven Kontur ist, der zweite zählt, wie oft der Pixel “neugezeichnet” wurde. Der Zeitzähler bewirkt, dass ein Pixel immer mehr abkühlt, bis er schliesslich “einfriert” und automatisch einen neuen Saatpunkt generiert. Die “Neuzeichnungsvergangenheit” eines Pixels basiert ausschliesslich auf Ereignissen, das heisst wenn der freie Punkt sich ändert, wird diese Vergangenheit für jeden Pixel, der Teil der neuen aktiven Kontur ist, aktualisiert, sprich der Zähler inkrementiert. Bei der zeitlichen Vergangenheit wird in diesem Fall der Zähler sowohl um die Zeit, die der Pixel dargestellt wurde, erhöht, als auch um den skalierten Gradientenbetrag des Pixels. Das führt dazu, dass Pixel an starken Kanten schneller abkühlen als andere. Für beide Zähler existieren zwei Schwellwerte. Der untere Schwellwert bestimmt, ob ein Pixel zum Kandidaten für die automatische Selektion wird, der obere, ob ein Kontursegment korrekt ist. Der erste Pixel, bei dem beide Zähler die untere Schwelle erreichen, wird zum Kandidaten und der erste Kandidat auf einem Kontursegment, das einen Pixel enthält, dessen Zähler den oberen Schwellwert erreichen, wird als neuer Saatpunkt ausgewählt.
Das Ziel hierbei ist, dass neue Saatpunkte möglichst weit entfernt vom letzten Saatpunkt und möglichst nah am freien Punkt generiert werden. Letzteres wird erreicht, indem man den unteren Schwellwert relativ niedrig setzt, das heisst Pixel werden schnell zu Kandidaten. Das alleine würde aber dazu führen, dass neue Saatpunkte in der Nähe von manuell plazierten Punkten gesetzt würden, und so immer nur kurze

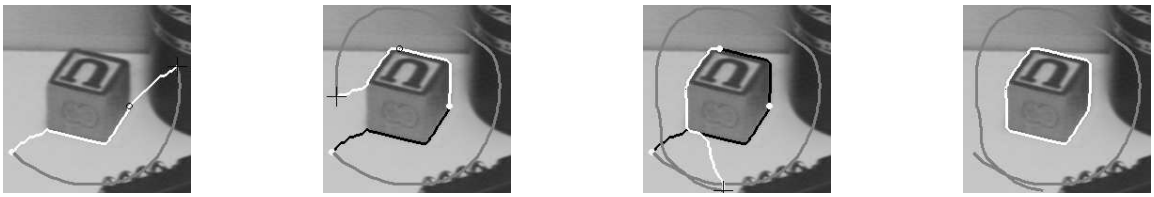


Abbildung 9.4: Mithilfe des Abkühlens der Kontur kann der Benutzer das Objekt sehr frei umfahren. Das aktuelle Live-Wire Segment ist weiss, sobald es abkühlt und einen neuen Saatpunkt erzeugt wird es schwarz dargestellt.

Live Wire Segmente definiert werden könnten, bevor automatisch ein neuer Saatpunkt gesetzt wird. Aus diesem Grund wählt man den oberen Schwellwert verhältnismässig hoch, so dass die "eingefrorenen" Kontursegmente möglichst lang sind. Für den Fall, dass ein Saatpunkt generiert wird, der nicht auf der gewünschten Kontur liegt, steht eine Backup-Funktion bereit. Hierfür werden die Daten (beispielsweise Karte der kumulativen Kosten und der optimalen Pfade) alter Saatpunkte beibehalten. Wird die Backup-Funktion angewendet, so wird der letzte gesetzte Saatpunkt samt seiner Daten gelöscht und der letzte Saatpunkt davor wird zum aktuellen Saatpunkt. Durch die Möglichkeit, den Mauszeiger sehr frei (ähnlich einem Lasso) und dementsprechend schnell um ein Objekt zu führen (siehe Abbildung 9.4), ist es möglich, dass einige unerwünschte Saatpunkte hintereinander gesetzt werden. In solchen Situationen ist es möglich, sukzessive Backups anzuwenden, um mehrere Saatpunkte wieder zu entfernen.

Expansion des Saatpunktes durch A^*-Suche

Aufgrund der interaktiven Natur von Live Wire, sind Verzögerungen durch Berechnungen nicht wünschenswert. Das Berechnen der optimalen Pfade jedes Pixels zum Saatpunkt kann aber bereits einige Sekunden in Anspruch nehmen. Auf einer 99 MHz 735 Unix Workstation dauert dieser Vorgang bei einer Bildgrösse von 512×512 Pixeln bei eingeschaltetem Training ca. 2,8 Sekunden und ohne Training 1,5 Sekunden. Gerade im Bezug auf das Abkühlen der Kontur ist es ungünstig, auch nur kurze Verzögerungen in Kauf nehmen zu müssen. Diese lassen sich aber durch die A^*-Suche fast komplett vermeiden. Wir erinnern uns an die, sich wellenförmig ausbreitende, Berechnung der optimalen Pfade (siehe Abb. 9.3). Pixel innerhalb der Wellenfront besitzen bereits alle einen optimalen Pfad. Wenn der Mauszeiger sich also im Innern der Wellenfront befindet, kann schon ein optimaler Pfad angezeigt werden, ohne dass die Berechnungen für das gesamte Bild abgeschlossen sein müssen. Sollte der Mauszeiger aus der Wellenfront heraus bewegt werden, wird solange kein optimaler Pfad angezeigt, bis die Wellenfront sich soweit ausgebreitet hat, dass der Cursor sich wieder in ihrem Innern befindet. Typischerweise breitet die Wellenfront sich schnell innerhalb des interessanten Gebiets (das Gebiet, in dem die Mausbewegungen stattfinden) aus, so dass es nur selten vorkommt, dass das Live Wire Segment ausgeblendet wird.

4 Mit Intelligent Scissors erzielte Ergebnisse

In diesem Kapitel betrachten wir die Ergebnisse, die Intelligent Scissors bei verschiedenen Bildtypenerzielt. In den Abbildungen 9.5 bis 9.10 sind zum Vergleich neben den weissen Live Wire-Konturen in Schwarz die "idealen" Konturen eingezeichnet (die natürlich subjektiv sind). Wo beide deckungsgleich sind, ist nur die weisse Kontur zu sehen. Bei Abbildung 9.5 handelt es sich um ein synthetisches Testbild, das erstellt wurde, um die Leistung von Intelligent Scissors bei Auftreten von unscharfen Kanten und weissem Rauschen zu untersuchen. Weiterhin enthält das Bild verschiedene Formen, um zu testen, inwieweit das Live Wire Segment z.B. scharfen Ecken oder Kurven folgen kann (man beachte auch die "Kammstruktur" oben rechts an der geschlossenen Kurve). Da dieses Bild synthetisch erstellt wurde, sind die optimalen Konturen bekannt und werden zum Vergleich verwendet. Zur Definition der Kontur wurden für das Polygon 4,3 Sekunden und für die geschlossene Kurve 8,3 Sekunden benötigt.
Abbildung 9.6 zeigt eine gestellte Situation mit verschiedenen Gegenständen, die bewusst einige Schwierigkeiten enthält, zum Beispiel, dass sich das Taschenmesser und die Büroklammerdose berühren.
Bei Abbildungen 9.7 und 9.8 handelt es sich um medizinische Bilder. Abbildung 9.7 zeigt die CT-Schichtaufnahme eines Rückenwirbels. Man sieht, dass an einer Stelle die Live Wire Kontur deutlich von der "idealen" Kontur abweicht (auch wenn die Live Wire Kontur korrekter erscheint). Zur Definition der äusseren Kontur des Wirbels wurden 5,9 Sekunden und 5 Saatpunkte benötigt.

Abbildung 9.8 zeigt eine Aufnahme eines Koronargefässes. Man sieht, dass die Live Wire Konturen sehr gut mit den “idealen” Konturen übereinstimmen. Für den linken Rand des Gefässes wurden 2,6 Sekunden und 3 Saatpunkte benötigt, für den rechten 1,9 Sekunden und lediglich ein Paar bestehend aus Saatpunkt und freiem Punkt.
Die Abbildungen 9.9 und 9.10 sind Farbbilder, die komplexe reale Szenen zeigen und gut geeignet sind, die Vielseitigkeit von Intelligent Scissors zu verdeutlichen. Sie enthalten teilweise sehr starke Kanten, die man mit langen Live Wire Segmenten und wenigen Saatpunkten definieren kann, andererseits aber auch Regionen die mehr Aktion seitens des Benutzers fordern, da man dort nur kurze Live Wire Segmente benutzen kann. Die hier vorgestellten Kostenfuntkionen gelten in dieser Form für Grauwertbilder. Sie sind aber auch bei Farbbildern einsetzbar wobei der Laplaceschen Nulldurchgang sowie der Gradientenbetrag für jeden Kanal eines RGB-Bildes berechnet werden und anschliessend das jeweilige Maximum verwendet wird. Für die Pixelwerte wird anstelle des Grauwertes jeweils die Helligkeit des Pixels verwendet.
Tabelle 9.2 zeigt die für jedes Objekt benötigten Zeiten und die Anzahl der Saatpunkte.

Objekt	Zeit in Sekunden	Anzahl der Saatpunkte
Büroklammerdose	3,6	2
Würfel	2,4	2
Taschenmesser	4,6	4
Tipp-Ex	5,1	4
Löffel	9,8	8

Tabelle 9.2: Benötigte Zeiten und Anzahl verwendeter Saatpunkte für verschiedene Szenen

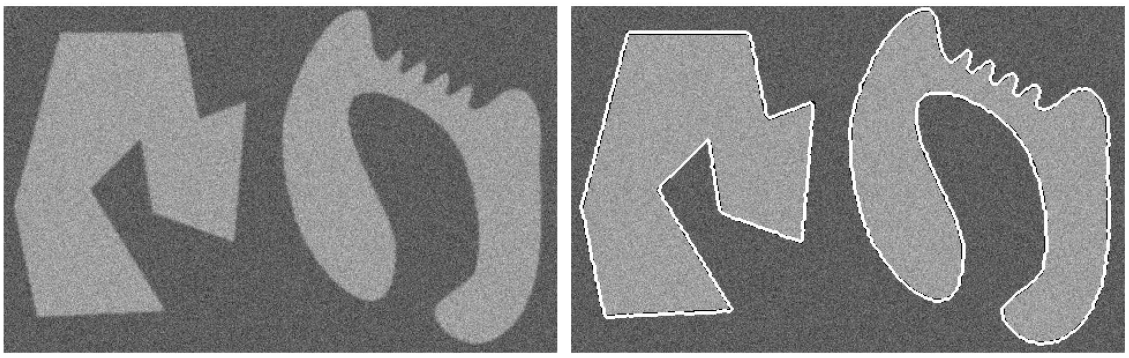


Abbildung 9.5: Synth. Testbild

4.1 Genauigkeit, Reproduzierbarkeit, benötigte Zeit

Die folgenden Tabellen vergleichen die mit Live Wire erzielten Ergebnisse im Gegensatz zu manuellem Tracing anhand verschiedener Objekte. Verglichen werden die zur Konturdefinition benötigte Zeit, die erreichte Genauigkeit und die Reproduzierbarkeit. In den Tabellen 9.3 bis 9.6 wurden die Ergebnisse von 8 Benutzern gemittelt. Nachdem jeder Benutzer sich eine Zeit lang mit den entsprechenden Tools eingearbeitet hat, hat er die fünf Objekte je dreimal manuell nachgezeichnet und je fünfmal mit Hilfe von Intelligent Scissors. Tabelle 9.3 zeigt die durchschnittliche, zur Konturdefinition benötigte Zeit. Wie man sieht, schneidet die Live Wire Technik wesentlich besser ab.
In Tabelle 9.4 ist die durchschnittlich erzielte Genauigkeit gezeigt. Bei den synthetischen Objekten (Polygon und geschlossene Kurve) ist die optimale Lösung bekannt und die erzielten Ergebnisse wurden mit einer euklidischen Entfernungskarte des Originalbildes verglichen. In der Entfernungskarte stellt der Grauwert jedes Pixels den euklidischen Abstand zum nächstliegenden Kantenpixel dar. Für die realen Objekte wurde eine Entfernungskarte anhand der “idealen” Konturen erstellt. Der Vergleich fand pixelbasiert statt und die Ergebnisse wurden zu jedem Objekt über alle Konturen und Benutzer gemittelt. Auch hier ist zu sehen, dass die Live Wire Technik in allen Punkten besser abschneidet.

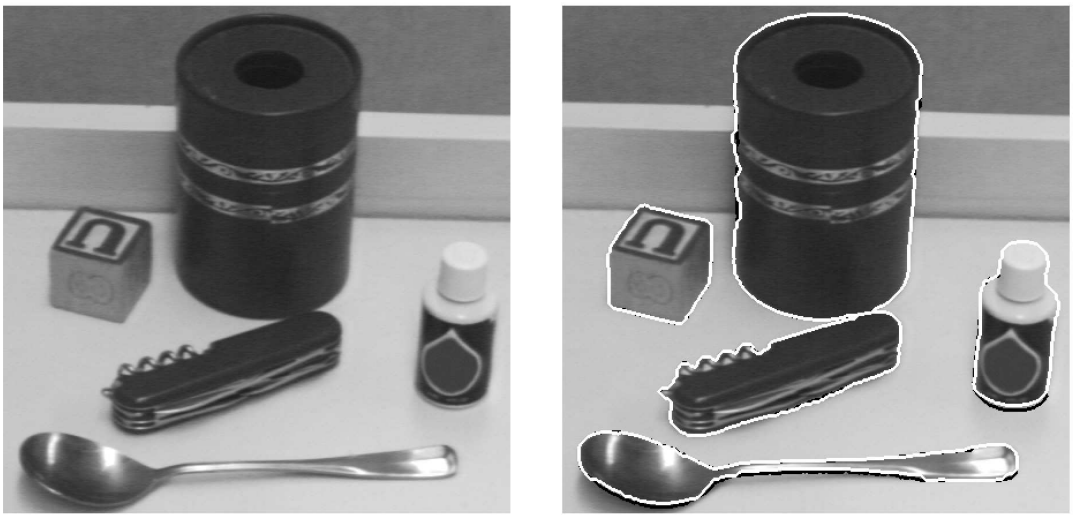


Abbildung 9.6: Schreibtischszene

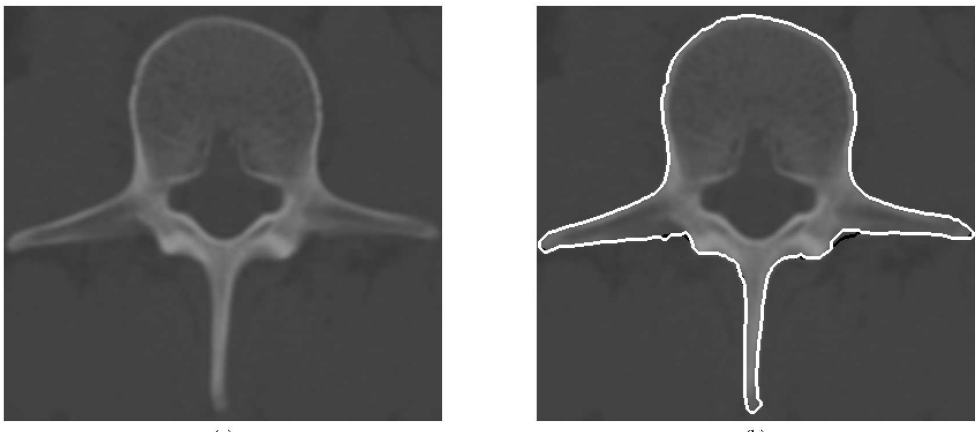


Abbildung 9.7: CT-Schichtaufnahme eines Rückenwirbels

Tabellen 9.5 und 9.6 beschreiben jeweils die Intra- bzw. Inter-Benutzer-Varianz. Für die Intra-Benutzer-Varianz wird jede vom Benutzer definierte Kontur eines Objektes mit jeder anderen, von ihm definierten, Kontur für dieses Objekt verglichen. Dies geschieht für jedes Objekt und alle Benutzer, so dass man letztendlich die durchschnittliche Intra- Benutzer-Varianz erhält. Für die Inter-Benutzer-Varianz wird jede mit einer bestimmten Technik definierte Kontur für jeden Benutzer mit jeder anderen Kontur des gleichen Objektes (von allen Benutzern, mit der gleichen Technik) verglichen.
Zu bemerken ist, dass selbst die Inter-Benutzer-Varianz bei Live Wire besser ist, als die Intra-Benutzer-Varianz bei manuellem Tracing. Die Abbildung 9.11 stellt die Ergebnisse aus den Tabellen noch einmal grafisch dar.

4.2 Komplexität

Ein Problem bei früheren Ansätzen zur Kantendetektion mithilfe von Graphsuche in Verbindung mit dynamischer Programmierung war der hohe Rechenaufwand. Durch die Einschränkung der lokalen Kosten auf Integerwerte innerhalb eines festgelegten Bereichs ist es möglich, ein Bucket Sort Verfahren mit Komplexität $O(N)$ bei der Graphsuche zu verwenden. N ist hierbei die Anzahl der Bildpixel für die ein optimaler Pfad zurück zum Saatpunkt berechnet worden ist. Wie bereits erwähnt (siehe Abschnitt 3.3) benötigt das Einfügen eines Punktes in die aktive Liste konstante und das Entfernen eines Punktes

Abbildung 9.8: Angiogramm einer Koronararterie

Abbildung 9.9: Realaufnahme von zwei Papageien

nahezu konstante Zeit. Insgesamt ergibt sich für die Graphsuche also bei N Pixeln eine Komplexität von $O(N)$. Betrachten wir dazu eine worst case Situation. Wenn ein Pixel von der aktiven Liste entfernt wird, wird er expandiert, indem die kumulativen Kosten zu all seinen nicht expandierten Nachbarn berechnet werden. Im schlimmsten Fall ist noch kein Nachbarpixel expandiert, das heisst es müssen die Kosten zu allen acht Nachbarn berechnet werden. Bei N Bildpüixeln ergibt dies also schlimmstenfalls $8N$ Kostenberechnungen. Allerdings kann nicht jeder Punkt acht noch nicht expandierte Nachbarn haben. Abgesehen vom Saatpunkt muss jeder Punkt mit kumulativen Kosten mindestens einen bereits expandierten Nachbarn besitzen. Eine Kante zwischen zwei Pixeln, die einmal verwendet wurde, um die kumulativen Kosten aufgrund der Expansion von einem der Pixel zu berechnen, wird tatsächlich danach nicht mehr für Kostenberechnungen herangezogen. Es wird also jede Kante zwischen zwei Pixeln nur genau einmal betrachtet. Da jeder Pixel je eine Kante zu allen acht Nachbarn besitzt, jede Kante aber wiederum von zwei Pixeln geteilt wird, ergeben sich also insgesamt $\frac{8}{2}N = 4N \in O(N)$ kumulative Kostenberechnungen.

5 Weitere Ansätze auf der Grundlage von Live-Wire

Aufbauend auf der Idee, die hinter Live Wire steckt, gibt es noch weitere Ansätze, von denen hier drei kurz vorgestellt werden sollen.

1. Tobogganing Live Wire: Hierbei werden die Pixel zu Regionen zusammengefasst, was in einer schnelleren Graphsuche resultiert.

2. Live Lane: Der Benutzer setzt nur den Startpunkt, während er den Mauszeiger innerhalb eines Korridors bewegt, werden neue Punkte automatisch generiert.

Abbildung 9.10: Realaufnahme einer Familie

Objekt	Live Wire (in Sek.)	Manuelles Tracing (in Sek.)
geschl. Polygon	11,7	20,9
geschl. Kurve	25,2	66,3
Büroklammerdose	10,0	23,5
Taschenmesser	13,7	34,8
Rückenwirbel	17,0	49,1

Tabelle 9.3: Vergleich der zur Segmentierung benötigten Zeit bei Live Wire und manuellem Tracing.

3. Adaptive Lane: Eine Version von Live Lane, bei der die Breite des Korridors entsprechend der Geschwindigkeit und Beschleunigung der Bewegungen angepasst wird.

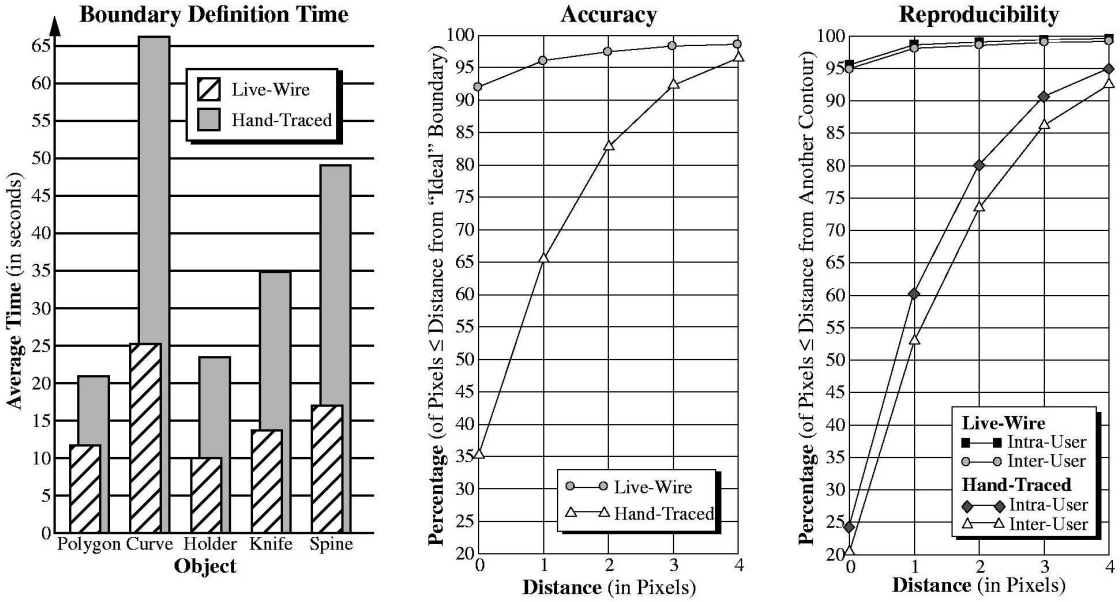


Abbildung 9.11: Vergleich

Objekt	Technik	mittl. Entf. (Pixel)	Standardabw. (Pixel)	Prozent exakt	Prozent ≤ 1 Pixel	Prozent ≤ 4 Pixel
geschl. Polygon	Live-Wire	0,018	0,161	97,55	99,78	99,98
	man. Tracing	0,333	0,649	69,14	92,33	99,73
geschl. Kurve	Live-Wire	0,026	0,311	97,12	99,68	99,88
	man. Tracing	0,862	1,235	44,62	73,53	97,68
Büroklammerdose	Live-Wire	0,504	1,438	82,90	88,99	95,40
	man. Tracing	1,505	1,257	19,79	55,30	96,11
Taschenmesser	Live-Wire	0,244	0,716	83,72	93,30	99,58
	man. Tracing	1,661	1,314	17,17	48,05	97,38
Rückenwirbel	Live-Wire	0,203	0,861	91,63	95,33	98,15
	man. Tracing	1,608	1,368	19,77	52,70	94,65
alle Objekte	Live-Wire	0,172	0,806	92,09	96,15	98,67
	man. Tracing	1,157	1,304	35,42	65,66	96,71

Tabelle 9.4: Vergleich der erzielten Genauigkeit bei Verwendung von Live Wire und manuellem Tracing.

Objekt	Technik	mittl. Entf. (Pixel)	Standardabw. (Pixel)	Prozent exakt	Prozent ≤ 1 Pixel	Prozent ≤ 4 Pixel
geschl. Polygon	Live-Wire	0,040	0,280	97,05	99,35	99,91
	man. Tracing	0,916	1,006	35,88	78,47	98,63
geschl. Kurve	Live-Wire	0,058	0,451	96,34	99,28	99,75
	man. Tracing	1,671	1,664	20,28	53,26	92,85
Büroklammerdose	Live-Wire	0,194	0,945	92,63	96,54	98,45
	man. Tracing	1,311	1,328	24,80	66,20	96,73
Taschenmesser	Live-Wire	0,118	0,632	93,30	97,83	99,49
	man. Tracing	1,359	1,288	25,83	60,72	95,94
Rückenwirbel	Live-Wire	0,079	0,588	95,89	98,81	99,61
	man. Tracing	1,700	1,512	19,32	51,67	92,84
alle Objekte	Live-Wire	0,114	0,594	95,46	98,58	99,50
	man. Tracing	1,455	1,461	24,10	60,13	94,82

Tabelle 9.5: Vergleich der Intra-Benutzer-Varianz bei Live Wire und manuellem Tracing.

5.1 Tobogganing Live Wire

Die wesentliche Idee bei Tobogganing Live Wire [5] ist, das Bild zunächst mit einem regionenbasierten Verfahren zu segmentieren und den gewichteten Graphen nicht pixelbasiert zu definieren, sondern regionenbasiert. Da die Knoten hier Regionen sind, ist der Graph insgesamt wesentlich kleiner, was natürlich in einer schnelleren Graphsuche resultiert.

Die Partitionierung des Bildes in Regionen geschieht mittels sog. *Tobogganing* (deutsch: Rodeln). Hierzu wird jedem Pixel eine “Rutschrichtung” zugeordnet, mit Ausrichtung auf den Pixel in der Vierer-Nachbarschaft mit dem höchsten Gradientenbetrag. Danach werden Pixel, die zum gleichen Pixel “rutschen” zu Regionen zusammengefasst und die Regionen mit einem Label versehen. Die Regionen, die so entstehen, sind effektiv die gleichen wie bei der Wasserscheiden-Transformation. Der Vorteil von Tobogganing ist die bessere Effizienz im Gegensatz zur Wasserscheiden-Transformation.

Nach der Partitionierung des Bildes wird ein gewichteter Graph erstellt. Hierzu betrachten wir zunächst die Grenzen zwischen den Regionen. Diese verlaufen jeweils zwischen zwei benachbarten Pixeln und werden als die Kanten des Graphen aufgefasst. Knoten sind Stellen, an denen sich drei oder vier Regionengrenzen treffen. Da die Knoten immer in der Mitte von vier Pixeln liegen, werden sie mit dem Pixel identifiziert, dessen obere linke Ecke dem Knoten entspricht. Knoten werden also durch Pixelpostionen repräsentiert und Kanten durch ein Viertupel aus Anfangsknoten, Endknoten, Label der linken Region und Label der rechten Region.

Durch den kleineren Graphen erfolgt die Berechnung der optimalen Pfade, je nach Durchschnittsgröße der Regionen, zwischen zwei und zehn mal so schnell wie bei herkömmlichem Live Wire.

Objekt	Technik	mittl. Entf. (Pixel)	Standardabw. (Pixel)	Prozent exakt	Prozent ≤ 1 Pixel	Prozent ≤ 4 Pixel
geschl. Polygon	Live-Wire	0,046	0,295	96,61	99,25	99,91
	man. Tracing	1,060	0,977	28,70	72,15	98,76
geschl. Kurve	Live-Wire	0,059	0,467	96,37	99,26	99,73
	man. Tracing	1,862	1,683	17,99	47,32	90,44
Büroklammerdose	Live-Wire	0,307	1,204	90,19	94,24	96,88
	man. Tracing	1,416	1,368	24,84	62,00	94,79
Taschenmesser	Live-Wire	0,144	0,698	92,22	97,17	99,37
	man. Tracing	1,631	1,436	19,60	51,44	93,66
Rückenwirbel	Live-Wire	0,087	0,634	95,63	98,68	99,56
	man. Tracing	1,990	1,634	16,30	43,43	89,75
alle Objekte	Live-Wire	0,114	0,700	94,79	98,06	99,20
	man. Tracing	1,676	1,539	20,50	52,98	92,59

Tabelle 9.6: Vergleich der Inter-Benutzer-Varianz bei Live Wire und manuellem Tracing.

5.2 Live Lane

Der Ansatz von Live Lane ist, dass der Benutzer den Startpunkt festlegt, danach mit der Maus einfach in der Nähe der gewünschten Kontur entlangfährt und zum Schluss, wenn er wieder in der Nähe des Startpunkts angelangt ist, eine "Schliessen"-Funktion initiiert. Hierzu wird eine quadratische Pixelmaske S, um den Startpunkt herum, definiert. Innerhalb von S wird ein optimaler Pfad vom Startpunkt zur aktuellen Position berechnet und als Live Wire Segment angezeigt. Wird der Cursor über den Rand von S hinausbewegt, so wird der optimale Pfad vom Startpunkt zum letzten, in S befindlichen, Punkt automatisch als Kontur gewählt. Dieser Punkt wird zum neuen Startpunkt und das Feld S wird um diesen zentriert. Das wiederholt sich solange, bis das aktuelle S den ersten Startpunkt enthält und der Benutzer die Schliessen-Funktion wählt, woraufhin der optimale Pfad vom letzten Punkt zum Startpunkt berechnet und ausgewählt wird. Die Vorteile von Live Lane gegenüber Live Wire sind unter anderem der wesentlich geringere Rechenaufwand. Während bei Live Wire optimale Pfade zu allen Pixeln berechnet werden, werden bei Live Lane hingegen nur kleine Umgebungungen betrachtet, in denen die Berechnungen stattfinden. Das führt dazu, das Live Lane einen so geringen Rechenaufwand hat, dass die erzielten Ergebnisse mehr oder weniger unabhängig von der Rechenleistung des Computers sind. Die Grösse der Maske S ist vom Benutzer frei wählbar zwischen 5 und 100 Pixeln. Bei kleinem S ähnelt Live Lane manuellem Tracing (sozusagen Live Lane mit $S = 1 \times 1$). Welche Maskengrösse optimal ist, hängt immer von den gerade gegebenen Eigenschaften der Kontur ab. Bei unscharfen, schwachen Kanten sollte die Maske möglichst klein sein, in Regionen mit klaren, starken Kanten sollte sie gross sein. Deshalb wäre es wünschenswert, die Maskengrösse automatisch den aktuellen Gegebenheiten anzupassen. Dieser Ansatz wird bei Adaptive Lane verwendet.

5.3 Adaptive Lane

Bei dieser Erweiterung von Live Lane besitzt S keine feste Grösse, sondern passt sich dem Verhalten des Benutzers an. Das Ziel ist, dass S in Regionen mit starken, klaren Kanten gross und in unklaren Regionen mit schwachen Kanten klein ist. Hierzu geht man davon aus, dass der Benutzer die Maus in klaren Regionen schnell und in unklaren langsam bewegt. Neue Punkte werden weiterhin generiert, wenn der Cursor S verlässt, aber auch, wenn die Geschwindigkeit der Mausbewegung einen bestimmten Schwellwert erreicht. In diesem Fall wird der neue Punkt gesetzt und die Grösse von S angepasst (je schneller die Bewegungen, desto grösser S). Als zweites Kriterium wird die Beschleunigung der Bewegungen in Betracht gezogen, da häufig abrupte Änderungen zwischen klaren und unklaren Regionen auftreten. Je nach Beschleunigung wird die Grösse von S aktualisiert. Damit geht man darauf ein, dass Benutzer die Maus in klaren Regionen mit hoher Geschwindigkeit und niedriger Beschleunigung, in unklaren Regionen mit niedriger Geschwindigkeit und Beschleunigung und mit hoher/negativer Beschleunigung in Übergangsregionen bewegt.

5.4 Vergleich

In [6] werden ausführliche Untersuchungen dargestellt, die unternommen wurden um Live Wire, Live Lane, Adaptive Lane und manuelles Tracing hinsichtlich der Geschwindigkeit und Reproduzierbarkeit zu vergleichen. Die Geschwindigkeit (gemessen in Schnitte pro Minute) variiert stark je nach Benutzer. Insgesamt ergibt sich die Reihenfolge:
Live Lane (am schnellsten) → Adaptive Lane → Live Wire → manuelles Tracing.
Hinsichtlich der Reproduzierbarkeit ergibt sich:
Live Wire → Live Lane → Adaptive Lane → manuelles Tracing,
wobei die Unterschiede der drei anderen Techniken gegenüber manuellem Tracing gross, untereinander allerdings gering sind.

5.5 Segmentierung von 3D-Objekten mithilfe von Live-Wire

Stellen wir uns ein dreidimensionales Objekt vor, als eine Menge $C_{(m)} = \{C_1, C_2, ..., C_m\}$ von zweidimensionalen, parallelen Schnitten. Die dreidimensionale Kontur des Objektes wird repräsentiert, durch gegebenenfalls mehrere, zweidimensionale Konturen (die geschlossen, zusammenhängend und orientiert sind) in jedem Schnitt C_i. Beim bisherigen Ansatz muss der Benutzer diese Konturen in jedem Schnitt mit 2D Live Wire definieren. In [7] wird ein neuer Ansatz zur Segmentierung von 3D Objekten vorgestellt, der diese wesentlich vereinfacht und verkürzt. Wir definieren *orthogonale Schnitte* C_o als solche, die senkrecht zu den axialen Schnitten C_i verlaufen. Auch in den orthogonalen Schnitten wird das 3D Objekt als Menge von geschlossenen, zusammenhängenden und orientierten 2D-Konturen repräsentiert. Die Idee ist nun, dass der Benutzer einige orthogonale Schnitte auswählt, um diese dann zu segmentieren. Bei geschickter Wahl sollten dann genügend Punkte der 3D-Kontur zur Verfügung stehen, dass das Live Wire Verfahren in allen parallelen Schnitten automatisch stattfinden kann.
Zu Beginn wird das Objekt vom Benutzer in Blöcke (Mengen von parallelen Schnitten) gleicher Topologie aufgeteilt, das heisst innerhalb eines Blocks sind die zweidimensionalen Regionen zusammenhängend. Nach der Aufteilung in Blöcke werden innerhalb eines jeden Blocks Strukturen definiert. Dies geschieht, indem ein paralleler Schnitt des Blocks ausgewählt wird und innerhalb der Struktur S ein Punkt c gesetzt wird. Der Schnittpunkt zwischen einer senkrecht zum Schnitt stehenden Gerade Z_c, die c enthält, und jedem anderen Schnitt innerhalb des Blocks, sollte wieder in der Struktur S liegen. Als nächstes werden einige orthogonale Schnitte gewählt, die sich alle in Z_c schneiden. Bei jedem dieser Schnitte wird vom Benutzer per Live Wire die 2D-kontur definiert. Von diesen Konturen werden jeweils die zwei am weitesten von einander entfernt liegenden Punkte, die sich auf der 2D-Kontur des parallelen Schnitts befinden, gewählt. Jeder orthogonale Schnitt steuert also zwei Punkte zur Gesamtkontur bei. Wurden für eine gegebene 3D-Struktur k orthogonale Schnitte ausgewält, so werden diese Punkte in kartesische Koordinaten umgewandelt (der Ursprungspunkt ist c) und in zwei Gruppen aufgeteilt, die Eingangs- und Ausganspunkte. Hierbei macht man sich den Umlaufsinn der 2D-Konturen zu Nutze, die so definiert sind, dass das Innere der Kontur immer als links befindlich angenommen wird. Eingangspunkte sind die, an denen die orthogonale 2D-Kontur den parallelen Schnitt C_i “betritt”, Ausgangspunkte die, an denen sie ihn verlässt. Danach werden die Punkte entsprechend ihres Winkels gegen den Uhrzeigersinn geordnet und zu Paaren zusammengefasst. Die ideale 2D-Kontur eines parallelen Schnitts C_i wird automatisch durch Verbinden der Punkte durch Live Wire generiert. Dies geschieht für alle parallelen Schnitte. Untersuchungen haben ergeben, dass dieser dreidimensionale Ansatz von Live Wire 2 bis 6 mal schneller ist, als der bisherige, und 3 bis 15 mal schneller als manuelles Tracing. Weiterhin besitzt 3D-Live Wire eine höhere Reproduzierbarkeit.

6 Zusammenfassung

Das in dieser Arbeit vorgestellte Verfahren hat einige wesentliche Vorteile gegenüber bisherigen Ansätzen zur semiautomatischen Segmentierung. Der Wichtigste ist ohne Frage die Laufzeitbeschränkung des Algorithmus auf Linearzeit. Erst dies ermöglicht die Echtzeit-Interaktion von Intelligent Scissors. Der Benutzer sieht unmittelbar das Resultat seiner Aktionen und kann direkt darauf eingehen. Um die Arbeit des Benutzers weiterhin zu erleichtern sind Konzepte wie Cursor Snap, das die Platzierung von Punkten vereinfacht, eingeführt worden, oder das Abkühlen der Kontur, das dem Benutzer das Setzen von Saatpunkten (teilweise) abnimmt. Das Prinzip des On-the-fly Trainings macht eine der Segmentierung vorangehende Trainingsphase überflüssig. Es wurde gezeigt, dass Intelligent Scissors, verglichen mit dem

manuellen Tracing schneller, einfacher, genauer und besser reproduzierbar ist. Hierbei sollte jedoch angemerkt werden, dass die Genauigkeit nicht wirklich vergleichbar ist, da, abgesehen von den synthetischen Testbildern, kein Goldstandard zur Verfügung steht, anhand dessen man die Ergebnisse bewerten kann. Die zum Vergleich verwendeten "idealen" Konturen wurden per Hand definiert, also durch manuelles Tracing, weswegen sie eigentlich nicht als sinnvoller Vergleich geeignet sind. Zur weiteren Leistungssteigerung (das heisst Verminderung des Rechenaufwands) sind die Ansätze des Tobogganing Live Wire, sowie Live Lane und Adaptive Lane vorgestellt worden, die jeweils in erster Linie auf der Reduzierung des Graphen, auf dem die Pfadsuche stattfindet, beruhen. Einerseits durch Aufteilen des Bildes in Regionen, andererseits durch Beschränken der Suche auf den gerade aktuellen Teil des Bildes. Zum Schluss wurde noch ein Ansatz vorgestellt, der die Segmentierung von 3D-Objekten wesentlich beschleunigt und vereinfacht.
Obwohl Intelligent Scissors die semiautomatische Bildsegmentierung stark vereinfacht, gibt es nichtsdestotrotz noch einige denkbare Erweiterungen. Die Sub-Pixelrepräsentation von Konturen würde zum Beispiel das Problem beheben, zu entscheiden, ob ein gewählter "Kantenpixel" selbst zur Kante gehört oder nicht. Zur 3D-Bildsegmentierung existieren weiterhin Ansätze, die beispielsweise dreidimensionale Filtermasken einsetzen, oder möglicherweise ein Live Surface Tool, das es ermöglicht, optimale Flächen auszuwählen. Hierbei würden allerdings auch viele neue Probleme auftreten, da kein dreidimensionaler Tracer-Algorithmus existiert. Diese Erweiterungsmöglichkeiten, verbunden mit der bisherigen Leistung, haben zur Folge, dass Intelligent Scissors eine bereits grosse Bedeutung für die semiautomatische Segmentierung besitzt, und dies wohl auch weiterhin der Fall sein wird.

Literaturverzeichnis

[1] T. Lehmann, W. Oberschelp, E. Pelikan, R. Repges: Bildverarbeitung für die Medizin, Springer-Verlag, 1997

[2] E.N. Mortensen, W.A. Barrett: Interactive Segmentation with Intelligent Scissors, Graphical Models and Image Processing, Vol. 60, pp. 349-384, 1998

[3] J. Canny: A Computational Approach to Edge Detection, IEEE: Transactions on pattern analysis and machine intelligence, Vol. PAMI-8 No.6, pp. 679-697, 1986

[4] N. Wirth: Algorithmen und Datenstrukturen, B.G. Teubner Stuttgart, 1995

[5] E.N. Mortensen, W.A. Barrett: Toboggan-Based Intelligent Scissors with a Four Parameter Edge Model, Proc. IEEE: Computer Vision and Pattern Recognition (CVPR '99), Vol. II, pp. 452-458, Fort Collins, CO, June 1999.

[6] A.X. Falcão, J.K. Udupa, S. Samarasekera, S. Sharma: User-Steered Image Segmentation Paradigms: Live Wire and Live Lane, Graphic Models and Image Processing, Vol. 60, pp. 233-260, 1998

[7] A.X. Falcão, J.K. Udupa: Segmentation of 3D Objects using Live Wire, Proc. SPIE 3034, pp. 228-239, 1997