Ausarbeitung:

Segmentierung 2a
(modellbasiert, interaktiv)

Autor:
Matthias Frank

Matrikelnummer:
2672520

Betreuer:
Dr. Michael Roth

Einleitung

Als ich das Thema „Interaktive, modellbasierte Segmentierung“ zugeteilt bekam, wusste ich damit wenig anzufangen. Wie sich aber nach einiger Einarbeitungszeit herausstellte, war ich mit dieser Technik schon längst vertrauter als ich dachte, allerdings ohne zu wissen, was eigentlich dahintersteckt. Deswegen möchte ich meine Ausarbeitung mit einer kleinen Geschichte beginnen. Vor 2 Jahren hatten eine Freundin und ich die Idee, wir könnten selbst produzierte Video-CDs zu Weihnachten verschenken. Das Cover der CD sollte ein Bild von uns beiden vor dem Hintergrund einer Schneelandschaft sein. Allerdings wollten wir nicht mit dicken Jacken auf diesem Bild erscheinen, andererseits waren wir aber auch zu verfroren, um uns bei dieser Kälte ohne Winterbekleidung draußen fotografieren zu lassen. Meine Idee war es, das Foto einfach im warmen Wohnzimmer zu machen und uns dann per Bildbearbeitung in ein geeignetes winterliches Hintergrundbild zu kopieren. Bei seinerzeit aktuellen Version von Jascs Paint Shop Pro gibt es eine Funktion, in der man die Objekte aus einem Bild ausschneiden kann, ohne jedes Detail umständlich und zeitaufwändig mit der Maus nachfahren zu müssen. Ich musste also nur uns beide grob mit der Maus umfahren und das intelligente Auswahl-Werkzeug der Software suchte sich dann automatisch die Objekte aus dem Bild, die sich am stärksten vom Hintergrund unterscheiden (nämlich uns beide) und markierte diese. So konnte ich also bequem und ohne frieren zu müssen ein schönes CD-Cover erzeugen. Was ich damals nicht wusste, war, dass sich hinter diesem Verfahren das Prinzip der interaktiven Segmentierung mit „intelligent scissors“ verbarg, ein Teilbereich des Themas, das ich nun behandeln werde.

Vorstellung der verschiedenen Verfahren

Es gibt verschiedene Arten, ein Bild mehr oder weniger vollständig in Regionen zu zerlegen (= segmentieren).
Ziel der interaktiven Segmentierung ist es, im Gegensatz zum Template Matching oder anderen automatischen Segmentierungsmethoden, diese Zerlegung benutzergesteuert und nach Möglichkeit in Real Time auszuführen. Der Benutzer wählt also die für ihn interessante Region per Maus und Tastatur aus. Hierzu werden dem Benutzer in diversen Bildverarbeitungsprogrammen einige Werkzeuge in die Hand gegeben, die je nach Bedarf das Bild auf unterschiedliche Art und Weise in Regionen unterteilen, sei es bezüglich der Farbe in der Nähe eines Pixels (Region Growing), bezüglich von Kanten (interaktive Kantenverfolgung) oder aber mittels Unterteilung nach allen im Bild vorkommenden Farben (Wasserscheidentransformation).
Des Weiteren kann jede dieser Ausprägungen der interaktiven Segmentierung noch weiter vom Benutzer justiert werden, um ein optimales Ergebnis zu erzielen. In der Folge werden die verschiedenen Möglichkeiten vorgestellt, die Funktionsweise erklärt und ihre Vor- und Nachteile aufgezeigt.

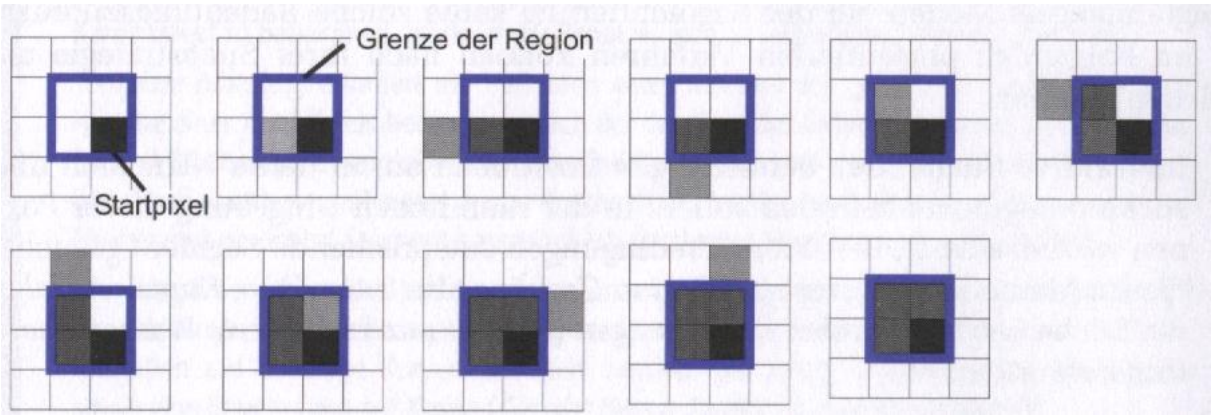
a) Region Growing

1. Anwendung

Sinn des Region Growings ist es, eine zusammenhängende Region, die farblich mehr oder weniger homogen ist, mit einer einheitlichen Farbe oder Textur zu füllen. Des Weiteren ist es auch möglich, diese Fläche automatisch mit einer Auswahl markieren zu lassen, um sie dann zu kopieren oder separat weiterzubearbeiten. In Gimp zum Bespiel existiert das Tool „Magic Wand“ (= magischer Zauberstab) das diese Funktion bietet.

2. Funktionsweise

Beim Region Growing wird der Startpunkt für die gewünschte Region vom Benutzer vorgegeben. Welcher Punkt als Startpunkt dient, ist innerhalb der Region egal, jeder Startpunkt in einer Region liefert das gleiche Resultat. Von diesem Punkt aus wird rekursiv oder iterativ in horizontale und vertikale Richtung gesucht, wie weit sich die Region ausdehnt bzw. an welchem Pixel sie ihre Grenze hat. Zu diesem Zweck wird die Farbe eines Nachbarpixels (im Folgenden wird so ein Pixel bezeichnet, das sich entweder links, rechts, oben oder unten neben dem gerade betrachteten Pixel befindet) mit dem aktuellen Pixel verglichen. Ist die Farbe gleich oder ähnlich (innerhalb eines vom Benutzer definierten Thresholds, dazu gleich), wird das Pixel zur Region hinzugenommen. Ansonsten bricht die Suche in diese Richtung ab. Wurde ein Pixel hinzugenommen, so wird dieses das neue Startpixel und das Verfahren beginnt von vorne. Es wird so lange gesucht, bis keine weiteren Pixel mehr zur Region hinzugenommen werden können, weil man überall am Rand der Region oder am Rand des Bildes angestoßen ist. Hier terminiert das Verfahren. Alle Pixel, die durch das Verfahren als zur Region gehörig klassifiziert wurden, gehören nun zur gesuchten Fläche (siehe Grafik).



Quelle: Klaus D. Tönnies, Grundlagen der Bildverarbeitung, Pearson Studium, 2005

Da das Absuchen der Fläche von einem Startpunkt aus an eine Flutung eines Terrains mit Wasser erinnert, wird dieser Algorithmus auch als „Flood-Fill“ Algorithmus bezeichnet.

Nun noch zur Erklärung des Farb-Thresholds: Es ist erforderlich, dass nicht nur exakt gleichfarbige Pixel als zur Region gehörig erkannt werden. Deswegen hat der Nutzer die Möglichkeit, einen gewissen Spielraum anzugeben, um den sich das gerade überprüfte Pixel vom Startpixel in der Farbe unterscheiden darf. Dieser Spielraum wird Threshold bezeichnet. Ist der Threshold auf null gesetzt, so müssen die Pixel, die zur Region hinzugenommen werden, exakt den gleichen Farbwert haben wie das Startpixel. Je größer der Wert des Thresholds ist, umso mehr darf sich die Farbe des

neuen Pixels auch unterscheiden. Für gute Ergebnisse ist es wichtig, einen geeigneten Threshold-Wert zu wählen. Die Probleme, die dabei auftauchen, werden im Folgenden noch aufgezeigt.

3. Vor- und Nachteile

Zu den Stärken von Region Growing zählt auf jeden Fall seine leicht zu implementierende Eigenschaft. Außerdem kann es wegen seines im Vergleich zum Live Wire relativ geringen Rechenaufwands schon seit verhältnismäßig langer Zeit im Heimcomputerbereich eingesetzt werden ohne dass Verzögerungen auftreten. Sind die Threshold Parameter richtig eingestellt und das Bildmaterial für dieses Verfahren geeignet so kann mit sehr wenig Aufwand ein Bild effizient Segmentiert werden.

Die Nachteile des Verfahrens liegen somit auf der Hand: da wie gerade erwähnt das Bildmaterial „geeignet" sein muss, kann nicht problemlos jedes Bild mit diesem Verfahren bearbeitet werde. Region Growing stellt folgende Anorderungen an das zu bearbeitende Bild: die Regionen müssen von klaren Kanten abgeschlossen werden, fließende Übergänge machen Probleme weil die Region an dieser Stelle auslaufen kann. Des Weiteren müssen die Regionen vollkommen abgeschlossen sein weil auch sonst die segmentierte Fläche durch die Lücke in der Kante ausläuft.
Das andere Problem das sich dem Benutzer bei der Verwendung des Region Growing Verfahrens aufdrängt ist die richtige Wahl des Thresholds: dieser kann zu eng, genau richtig oder zu groß gesetzt sein. Leider ist es für den Benutzer nicht ohne weiteres erkennbar welchen Wert er angeben soll, er muss also so lange probieren bis er zufällig einen passenden Wert gefunden hat. Im weiteren werden an einem Beispielbild die Fälle behandelt die auftreten können wenn der Threshold zu groß, zu klein oder genau richtig gewählt wird:

	Beispielbild ohne Anwendung von Region Growing. Der blaue Punkt zeigt das Startpixel an.
	Bei zu kleinem Threshold wird die gesuchte Region nicht vollständig gefunden. Es entstehen „Inseln" in dem von Region Growing gefluteten Bereich)

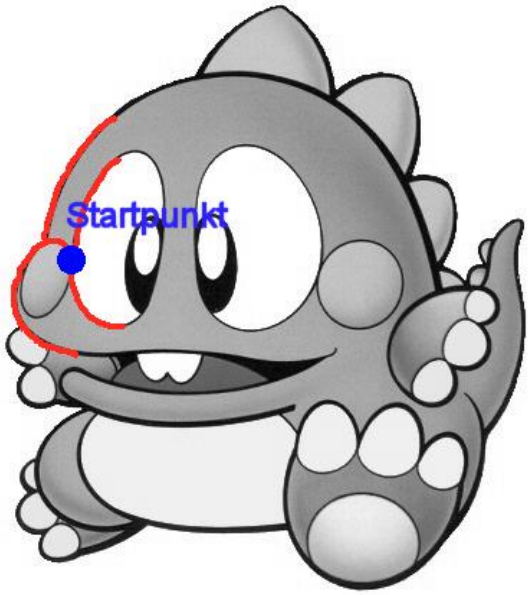
	Bei genau richtig gesetztem Threshold wird die Region exakt erkannt.
	Bei zu großem Threshold läuft die Region aus

Wegen dieser Probleme ist Region Growing nicht ohne weiteres auf jede Art von Bildmaterial anwendbar. Wo dieser Flood-Fill Algorithmus fehlschlägt, setzt die interaktive Kantenverfolgung an.

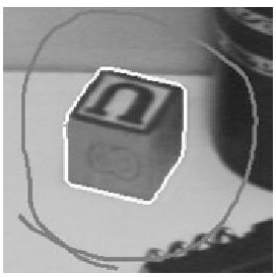
b) Interaktive Kantenverfolgung

1. Anwendung

Bei der interaktiven Kantenverfolgung werden gerade und gebogene Linien (im weiteren als Kanten bezeichnet) benutzt um das Bild zu unterteilen. Eine in Grafikprogrammen oft benutzte und weit entwickelte Methode Kanten bequem und präzise zu Verfolgen ist das Mausgesteuerte Live Wire Verfahren. Es existieren noch eine Menge andere Ausprägungen der benutzergesteuerten Kantenverfolgung, Live Wire vereint aber die Ansätze mehrerer anderer Verfahren in sich und kombiniert sie. Diese Vorgehensweise resultiert in hoher Effizienz und wenig Zeitaufwand seitens des Benutzer auch bei komplexeren Objekten. Der Name des Live-Wire Verfahrens erklärt sich fast von selbst: vom letzten gesetzten Wegpunkt wird immer sie beste Verbindung zur aktuellen Mausposition berechnet. Je nachdem in welcher Stelle sich die Maus gerade befindet springt der momentan noch freie Pfad immer an die passendste Kante im Umfeld und erinnert somit an ein unter Strom stehendes Kabel das den Boden berührt und dabei auf dem Boden tanzt (siehe rote Linien im Bild)



Die Kanten die die Begrenzung der Fläche darstellen sollen können vom Benutzer interaktiv ausgewählt werden. Um die Eigenschaft der gewählten Kante weiter spezifizieren zu können hat der Benutzer die Möglichkeit das Tool auf eine spezielle Art von Kante zu trainieren. Ohne Training übernimmt Live Wire immer die stärkste Kante in der nähe des neu gesetzten Wegpunktes. Folglich kann man, wenn sich Objekt recht deutlich vom Hintergrund abheben einfach mit der Maus ungefähr das Objekt umfahren und der gezogene Pfad „snappt“ dann automatisch an das auffallendste Objekt in der Nähe der Markierung (siehe Grafik):



(Grau: vom Benutzer aufgezogene Markierung, Weiß: automatisch an das Objekt "gesnappte" Kanten)

(Quelle: Interactive Segmentation with Intelligent Scissors - Mortensen Barrett GMaIP 1998)

Besteht das zu selektierende Objekt allerdings aus schwächeren Kanten, so ist es sinnvoll, das Live Wire Toll vor seiner Benutzung erst an einer Beispielkante zu trainieren, da sonst bei zu großem Abstand zwischen 2 Wegpunkten die Markierung immer wieder an die stärkeren Kanten zurückschnappt. Trainieren des Tools heißt, dass vor dem Setzen des Startpunktes eine Bespielkante erstmal mit der Maus ein kurzes Stück abgefahren werden muss. Das Werkzeug analysiert die Werte in der Umgebung der Testkante und verwendet diese dann bei der Berechnung der „passendsten“ Kante während der interaktiven Segmentierungsphase.
Ein Live Wire ähnliches Werkzeug wird auch in Gimp angeboten: es nennt sich „intelligent scissors“. Es lohnt sich, damit ein bisschen zu spielen. Sobald man sich daran gewöhnt hat, spart man eine Menge Zeit beim Markieren von Bildobjekten (z.B. um rote Pupillen zu entfernen, eine Jacke in schöner Farbe einzufärben ohne dafür Geld ausgeben zu müssen oder besonders schöne Menschen aus dem einen Bild in ein anderes zu kopieren). Im Weiteren sollen jedoch nicht die schier endlosen Anwendungsmöglichkeiten dieses Werkzeugs aufgezeigt werden, sondern es wird auf dessen Funktionsweise näher eingegangen.

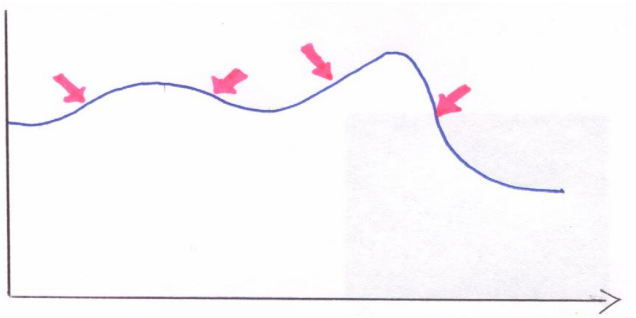
2. Funktionsweise

2.1 Berechnung des Gewichts

Live Wire macht die Wahl des optimalen Pfades von einigen Faktoren abhängig, die alle erst einzeln berechnet werden müssen. Diese teilen sich auf in statische Einflussgrößen (zu ihnen zählen die Berechnung der Nulldurchgänge und die Gradientenrichtung), deren Wert am Anfang der Segmentierung für das komplette Bild berechnet wird und dann während der Segmentierung konstant bleibt und dynamische Einflussgrößen die fortwährend ändern und somit immer wieder neu berechnet werden müssen. Zu ihnen zählen die trainierbaren Größen: der Kantenpixelwert und der Innen- bzw. Außenpixelwert. Bei dieser Aufteilung nimmt die Gradientenänderung eine Sonderposition ein weil diese sowohl dynamisch Elemente enthält.
Nun wird auf die Berechnung und Funktion der einzelnen Größen, aus denen sich später das Gewicht einer Kante errechnet, eingegangen:

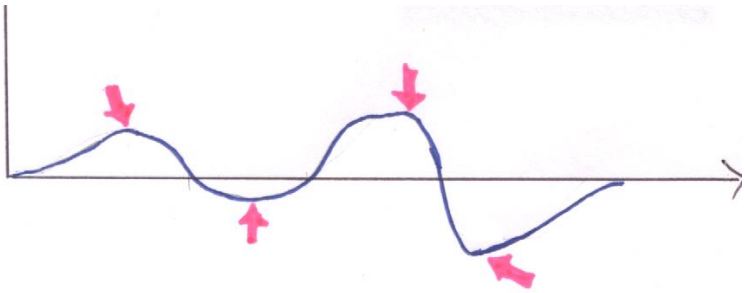
2.2.1 Nulldurchgänge

Die Berechnung der Nulldurchgänge dient der Kantenerkennung. Ein Grauwertbild kann als eine 3-dimensionale Funktion dargestellt werden. Die x und y Werte der Funktion geben die Position des Pixels an, auf der z-Achse ist der jeweilige Grauwert des Pixels an der Stelle (x,y) angetragen. Handelt es sich um ein Farbbild so wird das Verfahren für den roten grünen und blauen Anteil separat durchgeführt und danach entsprechend kombiniert, vom Prinzip ändert sich aber nichts im Vergleich zu Graustufenbildern. Nun wird wahlweise entlang der x-Achse oder entlang der y-Achse für alle y bzw. x Werte die 2dimensionale Funktion betrachtet und ihre Ableitung gebildet.



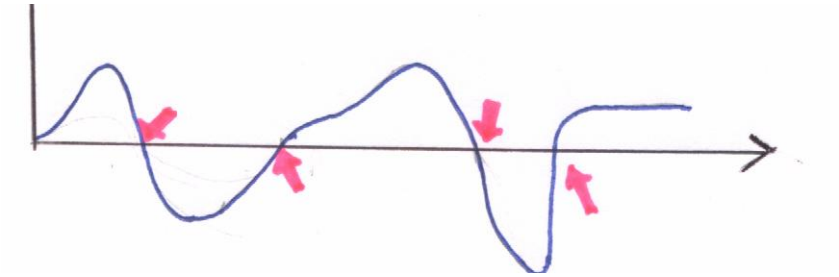
(ein Teil der Bildfunktion im Querschnitt, die Pfeile kennzeichnen die Kanten)

Die Ableitung der Funktion (im Weiteren nur noch als 1. Ableitung bezeichnet) liefert die Steigung der Funktion zurück. Nun gilt es herauszufinden an welchen Stellen die Funktion selbst sehr schnell steigt oder fällt, den gerade an diesen Stellen haben wir auffallende Farb- / Grauwertänderungen zu verzeichnen und damit eine Stelle die als Kante gilt.



(ein Teil der 1. Ableitung im Querschnitt, die Pfeile kennzeichnen die Kanten)

Leitet man nun die Ableitung ein weiteres Mal ab (im Folgenden als 2. Ableitung bezeichnet), so erhalten wir eine Funktion, die uns die Steigung der 1. Ableitung angibt. Die Steigung der ersten Ableitung (die ja die Farb- / Grauwertänderung im Bild angibt) sehr groß oder sehr klein sein, denn genau an diesen Stellen sind starke Farbänderungen im Bild verzeichnen. Die Maximas bzw. Minimas der 1. Ableitung bekommt man, wenn alle Nullstellen der 2. Ableitung berechnet sind(denn die erste Ableitung hat die Steigung 0 an ihren Maximalen bzw. minimalen Stellen).



(Ein Teil der 2. Ableitung im Querschnitt. Die Pfeile kennzeichnen die Kanten, die jetzt genau die Schnittpunke der Funktion mit der x-Achse darstellen)

Wenn man also die dreidimensionale Funktion an jeder x bzw. y Stelle im Querschnitt betrachtet und von der erhaltenen Funktion die Nullstellen der 2. Ableitung berechnet, so bekommt man gerade die Stellen, an denen eine Kante im Bild Ursprungsbild zu sehen ist. Diese Stellen werden in dem Resultatsbild das der „zero-crossing" Filter zurückliefert auf „0“ gesetzt, alle anderen auf „1“.

So bekommt man also ein Bild, bei dem alle Stellen, an denen eine Kante ist, weiß sind (man interpretiere „0“ als Farbe weiß) und alle Stellen, an denen keine Kante auftritt, schwarz (man interpretiere analog die „1“ als Farbe schwarz).
Das Verfahren lässt sich sehr schön an dieser Grafik nachvollziehen:

Originalbild

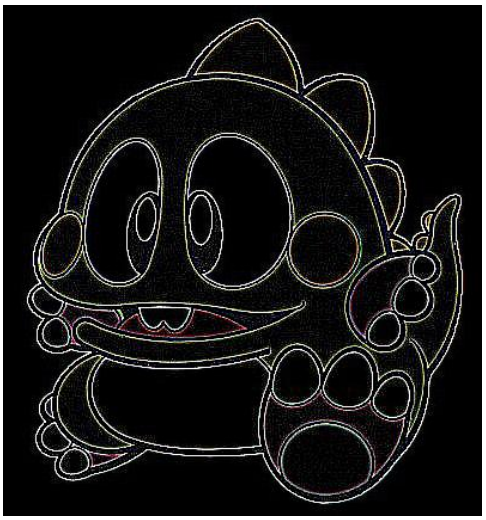
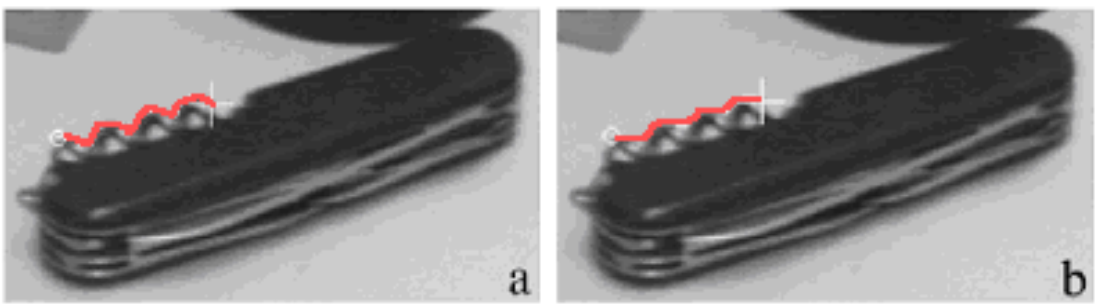


Bild nach Anwendung des Nulldurchgang Filters

In der Realität wird bei Live Wire dieses Verfahren wiederholt ausgeführt und zwar mit verschiedenen Rasterauflösungen (z.B. 3*3 Pixel oder 5*5 Pixel). Je nachdem wie der Kontrast des Bildes ausgeprägt ist, eignet sich eine bestimmte Auflösung besser oder schlechter für das zu bearbeitende Bild. Um möglichst für jede Bildbeschaffenheit gute Ergebnisse zu erzielen werden die Ergebnisse kombiniert (aufsummiert und normalisiert). Das Resultat ist kein 2bit schwarz-weiß Bild mehr sondern ein Graustufen Bild: ein Pixel im Resultatbild ist weiß falls jede Auflösung eine Kante zurückgeliefert hat (sehr stark ausgeprägte Kante im Ursprungsbild) und schwarz falls keine Auflösung eine Kante erkannt hat (Stelle ohne Farbänderung im Ursprungsbild). Weiterhin differenzieren Grauwerte die Ausprägung der Kante (sehr dunkel bedeutet: im Verfahren wurde bei vielen Auflösungen die 1 zurückgegeben, also eine sehr schwache Kante; sehr hell heißt: es wurde oft die 0 zurückgegeben also recht deutliche Kante). Da bei sehr starken Kanten genau 0 zurückgeliefert wird, hat das zur Folge, dass diese Stelle sehr geringe Kosten verursacht. Dies erzeugt eine „Kostenschlucht“ die die vom Benutzer erzeugte Kante nahe an das relevante Objekt zwingt. Dieser Vergleich zeigt auf, wie sehr sich das Ergebnis durch Einbeziehung der Nulldurchgänge verbessert:



(Quelle: Interactive Segmentation with Intelligent Scissors - Mortensen Barrett GMaIP 1998)
Die rote Linie kennzeichnet jeweils den Verlauf der Kante
a: Berechnung mit Nulldurchgängen, **b:** Berechnung ohne Nulldurchgänge

2.2.2 Gradientenänderung

Die genauere Unterscheidung mit mehreren Graustufen bei der Berechnung der Nulldurchgänge dient hauptsächlich dazu, möglichst viele qualitativ unterschiedliche Bilder bedienen zu können, lässt aber keinen ausreichend genauen Rückschluss auf die Stärke der Kante zu. Die Werte aus denen sich das Endresultat zusammensetzt sind schließlich nur binär: Kante oder nicht Kante. Das ist zwar ein ausreichend genauer Wert um eine Kante zu erkennen, die Kantenstärke kann aber mit einer anderen Größe sehr viel genauer differenziert werden. Zu diesem Zweck wird die Gradientenänderung eingeführt. Sie ist direkt proportional zum Bildgradienten und berechnet sich aus den partiellen Ableitungen des Bildgraphen (wie unter 2.2.1 definiert) an der Stelle (x,y). Es wird sowohl die partielle Ableitung nach x (I_x), als auch die partielle Ableitung nach y (I_y) gebildet. Die Gesamtableitung G an der Stelle (x,y) berechnet sich dann folgendermaßen:

$$G = \sqrt{I_x^2 + I_y^2}$$

Wenn der Wert G für jede (x,y) Kombination berechnet ist, werden alle Werte so angeglichen dass der kleinste vorkommende Wert von G mit dem Wert „0“ belegt ist. Diese verschobene Größe wird im Folgenden mit G’ bezeichnet:

$$G' = G - \min(G)$$

Darauf wird G’ so skaliert, dass der größte vorkommende Wert „1“ beträgt (sehr starke Kante), der kleinste vorkommende Wert ist „0“ (keine Kante). Diese Definition macht wenig Sinn, da die Einheit, in der gerechnet wird, nicht die Kantenstärke ist, sondern die Kosten für eine dem Pfad hinzugefügte Kante. Diese Kosten sollten gerade niedrig (also nahe bei „0“) sein, wenn die Kante stark ausgeprägt ist, während eine schwache Kante entsprechend auch hohe Kosten verursachen soll (nahe bei „1“). Dieses Problem ist einfach zu beheben, indem man die komplette Formel von „1“ subtrahiert. Die fertige Formel zur Berechnung der Gradientenänderung (f_g) sieht damit so aus:

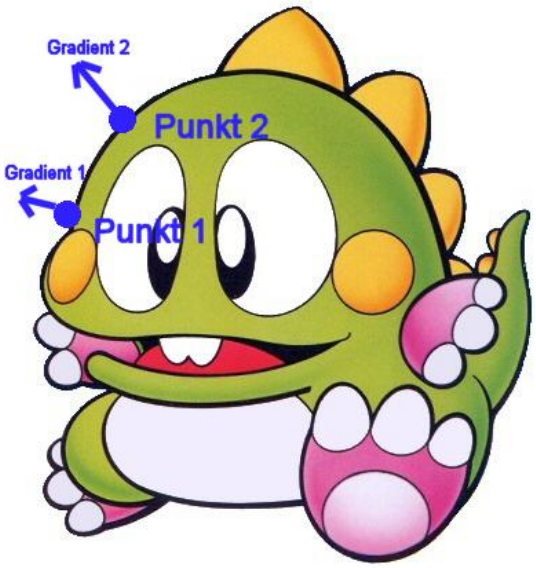
$$f_{\mathrm{G}} = 1 - \frac{G'}{\max(G')},$$

Auch bei dieser Größe wird das Verfahren mit verschiedenen Rasterauflösungen durchgeführt. Auch hier wird dieser Aufwand deswegen getrieben, um verschiedene Bildqualitäten gleichermaßen gut bedienen zu können. Allerdings ergibt sich bei der Berechnung der Gradientenänderung das Endresultat nicht aus der gewichteten Summe aller Einzelergebnisse, sondern es wird das Ergebnis der Rasterauflösung benutzt, welche die höchste Gradientenänderung erzeugt hat.

2.2.3 Gradientenrichtung

Die bisher vorgestellten Verfahren reichen immer noch nicht aus, um ein optimales Ergebnis bei der interaktiven Kantenverfolgung zu erzielen. Um das Ergebnis weiter zu verbessern, wird die Richtung des Gradienten überprüft. Da die gesuchte Kante im Normalfall zum großen Teil aus Strecken besteht, bei denen sich die Richtung nicht oder nur wenig ändert, wird versucht, die Kante so glatt wie möglich zu führen.

Die Betrachtung der Gradientenrichtung versucht dies umzusetzen und abrupte Richtungsänderungen zu reduzieren.
Das Verfahren gestaltet sich vom Prinzip her recht einfach: Man berechnet die Richtung des Einheitsvektors an der Stelle (x,y) und an der Stelle (a,b), während Punkt 1 (x,y) und Punkt 2 (a,b) auf einer gemeinsamen Kante liegen (siehe Grafik).



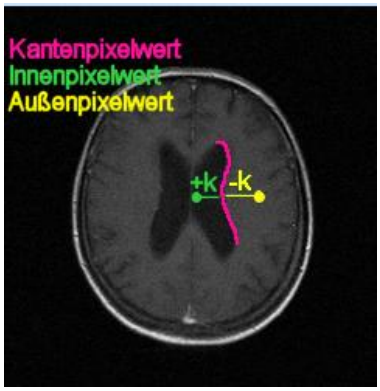
Nun werden die beiden Einheitsvektoren miteinander verglichen: Je geringer der Unterschied ist, desto geringer fallen die Ergebniskosten aus und umgekehrt. Dieses Verfahren wird für alle Pixel auf dem Pfad, dessen Kosten gerade berechnet werden sollen, durchgeführt. Somit werden Verbindungen mit geringen Richtungsänderungen bei den Gesamtkosten gewissermaßen bevorzugt und Kanten mit abgehacktem Verlauf vermieden.

2.2.4 Training

Die eben vorgestellten Verfahren sind ausreichend für Live Wire, um eine Kante zuverlässig zu erkennen und auszuwählen. Würden allerdings nur diese Kriterien beachtet werden, kämen als Grenze für die Segmentierung nur relativ stark hervortretende Kanten, Kanten mit geringer Richtungsänderung oder Kompromisskanten aus diesen beiden Einflussgrößen in Frage. Bei bestimmten Aufgaben können weniger auffallende Kanten allerdings viel interessanter sein. Könnte man das Live Wire Verfahren also nicht trainieren, so wäre es in diesem Anwendungsgebiet nutzlos. Ein Arzt, zum Beispiel, der einen Tumor im Gehirn markieren will, ist wohl wenig erfreut, wenn die Segmentierungsgrenze automatisch immer wieder an die Gehirnrinde zurückschnappt und somit viel mehr auswählt, als eigentlich von Bedeutung ist. Zu diesem Zweck ist in Live Wire die Möglichkeit integriert, das Verfahren darauf zu trainieren, welche Art von Kanten von Bedeutung sind. Das Training wird entweder in einer separaten Phase durchgeführt (der Benutzer trainiert des Verfahren an einer Testkante und beginnt dann erst mit dem eigentlichen Segmentierungsvorgang) oder es lernt „on-the-fly“. Das heißt, das Verfahren versucht während der Segmentierung einer Region aus den schon gesetzten Wegpunkten Schlüsse über die Art der gewünschten Kante zu ziehen und diese konsequent fortzusetzen.
Aus der trainierten Kante können drei Parameter geschlossen werden:

- Wert des Pixels, das auf der Kante liegt (Kantenpixelwert = f_p),
- Wert des Pixels, das innerhalb der trainierten Region liegt (Innenpixelwert = f_i)
- Wert des Pixels, das außerhalb der trainierten Region liegt (Außenpixelwert = f_o)

Um f_i und f_o zu bestimmen, werden beliebig viele Punkte auf der Trainingskante betrachtet und von dieser Stelle aus ein bestimmter Offset (k) aufaddiert bzw. subtrahiert. Die erhaltenen Farbwerte dienen dann zur Bestimmung von f_i und f_o (siehe Grafik).



Die erhaltenen Werte werden wieder entsprechend skaliert, so dass sie nur Werte zwischen „0“ und „1“ annehmen. Wenn es sich bei der Segmentierung um ein Graustufenbild handelt, gelten folgende Formeln:

$$f_I(\boldsymbol{p}) = \frac{1}{255} I(\boldsymbol{p} + k \cdot \boldsymbol{D}(\boldsymbol{p}))$$

$$f_O(\boldsymbol{p}) = \frac{1}{255} I(\boldsymbol{p} - k \cdot \boldsymbol{D}(\boldsymbol{p}))$$

- p ist das betrachtete Kantenpixel,
- I(p) ist der Graustufenwert des betrachteten Pixels
- D(p) ist der Einheitsvektor des Gradienten

Auch die Berechnung dieser Größe kann unschwer auf RGB Bilder übertragen werden, indem die Werte für jeden Farbkanal einzeln berechnet und entsprechend kombiniert werden.
Auch diese drei Einflussgrößen werden gewichtet in die Berechnung der Gesamtkosten eines Pfades mit einbezogen. Ist kein Training durchgeführt worden oder ist das dynamische Trainingsfeature deaktiviert worden, so werden sie mit dem Faktor „0“ gewichtet und haben somit keinen weiteren Einfluss auf die Gesamtkosten der Kante.

2.2.5 Aufsummieren der Teilgrößen

Sind sowohl die statischen, zu Beginn des Segmentierungsprozesses bestimmten, als auch die dynamischen, bei jeder Änderung des aktiven Pfads von neuem ausgewerteten, Teilwerte für einen Pfad berechnet, so können die Gesamtkosten bestimmt werden.
Die Gesamtkosten berechnen sich einfach aus der gewichteten Summe aller Teilwerte. Es werden für die Berechnung folgende Variablen eingeführt:

- f_z : Nulldurchgänge
- f_g : Gradientenänderung
- f_d : Gradientenrichtung
- f_p : Wert des Kantenpixels
- f_i : Wert des Pixels innerhalb der segmentierten Region
- f_o : Wert des Pixels außerhalb der segmentierten Region

Weiter gilt für die Berechnung des Gesamtgewichts *l (p,q)* für den Pfad von Pixel p zu Pixel q:

$$l(\boldsymbol{p},\boldsymbol{q}) = \omega_{\mathrm{Z}} \cdot f_{\mathrm{Z}}(\boldsymbol{q}) + \omega_{\mathrm{G}} \cdot f_{\mathrm{G}}(\boldsymbol{q}) + \omega_{\mathrm{D}} \cdot f_{\mathrm{D}}(\boldsymbol{p},\boldsymbol{q}) + \omega_{\mathrm{P}} \cdot f_{\mathrm{P}}(\boldsymbol{q}) + \omega_{\mathrm{I}} \cdot f_{\mathrm{I}}(\boldsymbol{q}) + \omega_{\mathrm{O}} \cdot f_{\mathrm{O}}(\boldsymbol{q}),$$

Jede Größe wird mit einem Faktor gewichtet. Die Gewichte sind auf sinnvolle Werte voreingestellt, können aber bei Bedarf angepasst werden.
Nachdem man nun die Kosten für jeden Pfad bestimmen kann, stellt sich die Frage, wie man schnell und effizient diese Werte miteinander vergleichen kann und entsprechend die beste Verbindungskante in real Time ausgeben kann. Zu diesem Zweck wird der Dijkstra Algorithmus benutzt, der im nächsten Kapitel eingeführt wird.

2.2 Berechnung der besten Verbindung

Der Algorithmus von Dijkstra (nach seinem Erfinder Edsger W. Dijkstra) dient der Berechnung eines kürzesten Pfades zwischen einem Startknoten und einem beliebigen Knoten in einem kantengewichteten Graphen.
Hierzu wird zunächst der direkte Weg vom Startpixel zu den einzelnen Pixeln als kürzester Weg eingetragen. In den Folgeschritten wird immer der am billigsten zu erreichende, bislang noch nicht besuchte Pixel gewählt und getestet, ob von diesem aus andere Pixel günstiger erreicht werden können als bislang der Fall.

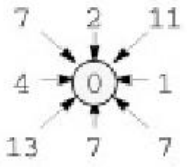
Am Ende entstehen die billigsten Kosten für die Wege vom Startknoten zu allen anderen Knoten. Speziell in diesem Fall ist noch eine Besonderheit anzumerken: Nachbarpixel, die horizontal oder vertikal bezüglich des Ausgangspixels angeordnet sind, werden mit dem Faktor 1 gewichtet (wie üblich). Diagonal liegende Pixel allerdings werden mit dem Faktor $\sqrt{2}$ gewichtet, da deren Abstand vom Ursprungspixel entsprechend größer ist (wie sich mit dem Satz von Pythagoras nachrechnen lässt).
Folgende Abbildungen soll diese Vorgehensweise verdeutlichen, da sie meiner Meinung nach in Worten beschrieben schwer zu verstehen ist, in Bildern aber transparent dargestellt werden kann:

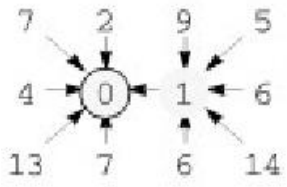
Schritt 1: Alle nötigen Gewichte sind berechnet, das Startpixel ist mit einem Kreis gekennzeichnet:

11	13	12	9	5	8	3	1	2	4	10
14	11	7	4	2	5	8	4	6	3	8
11	6	3	5	7	9	12	11	10	7	4
7	4	6	11	13	18	17	14	8	5	2
6	2	7	10	15	15	21	19	8	3	5
8	3	4	7	9	13	14	15	9	5	6
11	5	2	8	3	4	5	7	2	5	9
12	4	(2)	1	5	6	3	2	4	8	12
10	9	7	5	9	8	5	3	7	8	15

(Quelle: Interactive Segmentation with Intelligent Scissors - Mortensen Barrett GMaIP 1998)

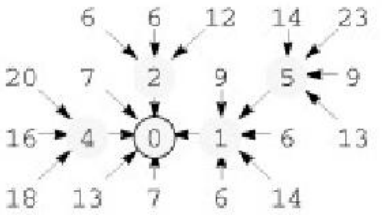
Schritt 2: Da das Startpixel keine Verbindungskosten zu sich selbst verursacht, wird sein eigenes Gewicht auf „0“ gesetzt. Das Gewicht der diagonal liegenden Pixel wird mit dem Faktor √2 multipliziert und nach gängigen Regeln auf eine Ganzzahl gerundet.



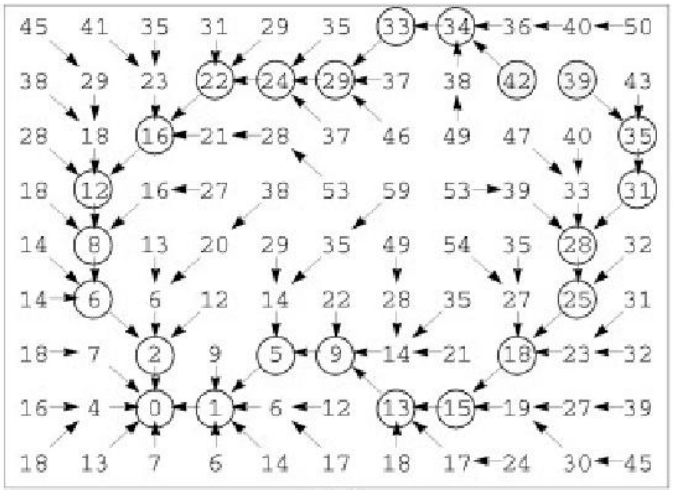


Schritt 3: Ein weiteres Nachbarpixel wird aufgenommen. Von hier aus werden wieder die neuen Verbindungskosten (evtl. multiplizieren mit √2!) berechnet. Im Inneren des neu aufgenommenen Pixels stehen die Verbindungskosten zu seinem Vorgänger. Sie müssen zum Kantengewicht der Nachbarpixel addiert werden. In diesem Fall wird also das Gewicht jedes Nachbarn um 1 erhöht, da die Verbindung vom Startpixel zum neu hinzugefügten Pixel Kosten in Höhe von „1“ verursacht. Werden kostengünstigere neue Verbindungen gefunden, so werden die teureren alten durch diese ersetzt *Beispiel:* es ist günstiger, erst vom Startknoten nach rechts und dann nach oben zu gehen (neue Kosten: 9) als direkt vom Startknoten diagonal nach rechts oben (alte Kosten: 11)

Schritt 4: Gleiches Verfahren wie bei Schritt 3 mit weiteren hinzugefügten Pixeln. Dieses Schema wird nun durchgeführt bis alle Pixel vom Startpixel aus erreichbar sind.



Schritt 5: Alle Pixel wurden betrachtet und die günstigste Verbindung zu jedem anderen Pixel berechnet. Das Verfahren ist beendet:

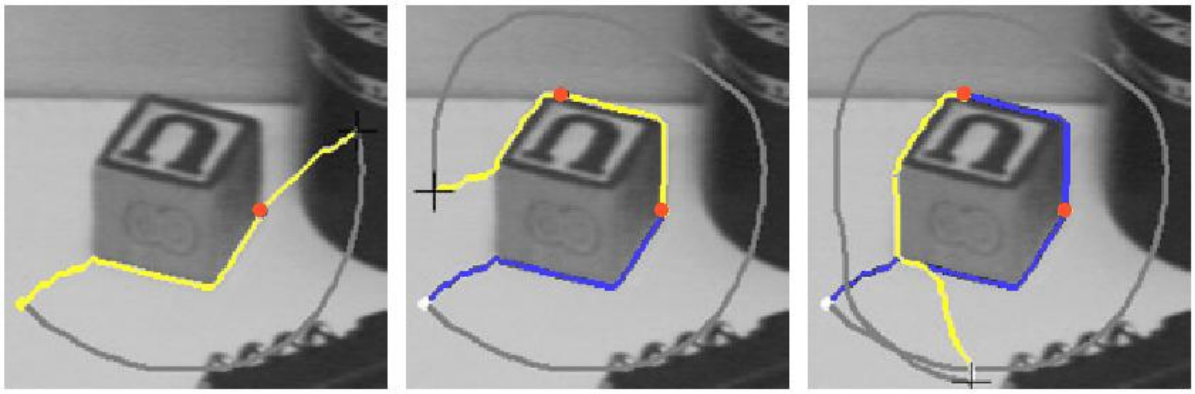


(Quelle: Interactive Segmentation with Intelligent Scissors - Mortensen Barrett GMaIP 1998)

Nun kann die günstigste Verbindung vom Startpixel (dem letzten gesetzten Wegpunkt) und der aktuellen Position einfach ausgelesen werden und die Kante entsprechend dem Benutzer als mögliche Segmentierungskante auf dem Bildschirm ausgegeben werden.
Es liegt auf der Hand, dass es durchaus einiges an Rechenleistung kostet, all diese Informationen ständig neu zu berechnen. Da Live Wire ein interaktives Verfahren ist, sind Verzögerungen länger als eine Sekunde, bis der neue Pfad gezeichnet wird, nicht in Kauf zu nehmen.
Um die Geschwindigkeit (insbesondere auf langsameren Rechnern) zu verbessern, sollten weitere Optimierungen am Verfahren vorgenommen werden die kurz angesprochen werden sollen.

2.3 Path cooling

Je länger die Kante wird, die der Benutzer aufzieht ohne dazwischen weitere Wegpunkte zu setzen, desto größer wird der Rechenaufwand, der erbracht werden muss, um die optimale Verbindung zu berechnen. Beobachtet man, in welchem Bereich der freien Kante (der Teil der Kante zwischen dem Startpunkt oder dem zuletzt gesetzten Wegpunkt und der aktuellen Mausposition) sich der Verlauf der Kante ändert, so beschränkt sich die Änderung meistens auf den aktuellsten Teil der Kante (live Lane). Um nicht jedes Mal die beste Kante vollständig neu errechnen zu müssen (sie ist in den meisten Fällen sowieso zu einem Großteil die gleiche wie die, die dem Benutzer schon angezeigt wird), werden von Live Wire Wegpunkte automatisch generiert, wenn sich an einem Teil der aktiven Kante längere Zeit nichts ändert. So muss nur der neueste Teil der aktiven Kante neu berechnet werden, was sich positiv auf die Geschwindigkeit des Verfahrens auswirkt. Auch hier ist es wohl besser, die Funktionsweise von path cooling an einem Bild nachzuvollziehen, als sie kompliziert mit vielen Worten zu beschreiben. Die gelben Linien stellen den aktiven Pfad (live Lane) dar, blaue Linien den gekühlten Pfad (cooled path). Die roten Punkte sind die automatisch gesetzten Wegpunkte (seed points):



(Quelle: Interactive Segmentation with Intelligent Scissors - Mortensen Barrett GMaIP 1998, leicht verändert)

Nachdem das Live Wire Verfahren nun eingehend beschrieben worden ist, sollen noch kurz die Vor- und Nachteile erwähnt werden, die dieses Verfahren mit sich bringt.

3. Vor- und Nachteile

Im Vergleich zu den anderen Techniken der interaktiven Segmentierung ist die kantenbasierte Variante wohl die flexibelste. Ihr Vorteil liegt darin, dass sehr verschiedene Arten von Objekten ausgewählt werden und eventuelle Fehler beim Erkennen der gewünschten Region sofort und interaktiv beseitigt werden können. Diese beiden Eigenschaften machen das Verfahren schnell und effizient.

Allerdings bringt die kantenbasierte interaktive Segmentierung auch seine Nachteile mit sich: der hohe Rechenaufwand macht das Verfahren auf älteren Computern träge, insbesondere dann, wenn die Umsetzung aus programmiertechnischer Sicht nicht genügend optimiert wurde. Daraus folgt, dass das Verfahren kompliziert und verhältnismäßig schwer umzusetzen ist. Der Programmieraufwand liegt also bei weitem höher als er für die Programmierung von Region Growing ansetzt werden müsste. Zuletzt bietet das Verfahren viele Parameter, die an die individuellen Bedürfnisse angepasst werden müssen. Diese funktionieren zwar recht gut für eine Vielzahl von Bildern, für spezielle Aufgaben bedeutet dies aber erhöhten Konfigurationsaufwand. Des Weiteren sind erfahrende Nutzer erforderlich, um das Verfahren auch ausreizen und an das Einsatzgebiet anpassen zu können.
Nachfolgend soll noch eine weitere Möglichkeit behandelt werden, Bilder in Regionen zu unterteilen, die weniger kompliziert und rechenaufwändig ist.

c) Wasserscheidentransformation

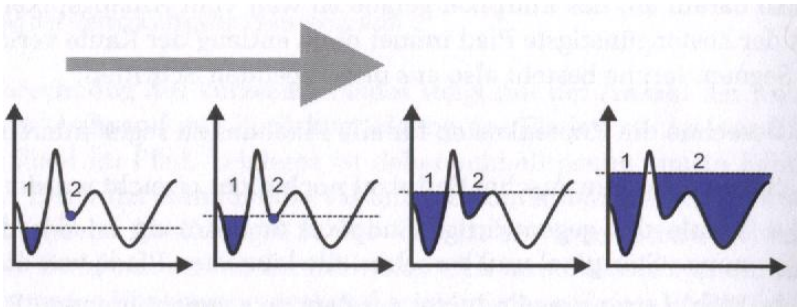
1. Anwendung

Aufgabe der Wasserscheidentransformation ist es (ähnlich wie beim Region Growing), Flächen gleicher oder ähnlicher Farbe zu markieren oder zu füllen. Im Gegensatz zum Region Growing arbeitet sich die Wasserscheidentransformation aber nicht von einem Pixel zum Rand hin nach außen, um die Region abzugrenzen, sondern bezieht sich auf das ganze Bild (bzw. auf vorher ausgewählte Teilbereiche des Bildes). Will man zum Beispiel im Ursprungsbild alle Flächen, die mit der Farbe rot gefüllt sind, durch blaue Flächen ersetzen, so müsste beim Region Growing jede Region einzeln mit Farbe gefüllt werden. Die Wasserscheidentransformation führt diesen Durchgang in einem Schritt für alle Regionen aus, die dem Auswahlkriterium entsprechen, unabhängig davon, ob sie zusammenhängen oder nicht. Der Sinn des Verfahrens kann an folgendem Bildbeispiel nachvollzogen werden:

Durch die Anwendung der Wasserscheidentransformation wurden in einem Zug alle Flächen im Bild, die die Farbe beige enthielten, mit lila gefüllt.
Die Funktionsweise dieses Verfahrens basiert nicht auf der wiederholten Ausführung von Region Growing, wie man vielleicht denken könnte.
Die technischen Details sollen im nächsten Kapitel genauer erläutert werden.

2. Funktionsweise

Man stelle sich die Funktion, die das Bild beschreibt, wieder als dreidimensionale Funktion vor (x-Koordinaten des Bildes auf der x-Achse der Funktion, y-Koordinaten des Bildes auf der y-Achse der Funktion, Grauwert/Farbwert auf der z-Achse der Funktion).
Die Funktion wird in gleicher Art, wie in Kapitel 2.2.1 beschrieben, definiert und an einer Stelle quergeschnitten.



Quelle: Klaus D. Tönnies, Grundlagen der Bildverarbeitung, Pearson Studium, 2005

Nun wird ein Wert festgelegt, wie hoch die Funktion geflutet werden soll (siehe gestrichelte Linie). Je höher dieser Wert liegt, desto mehr der Bildfunktion „steht unter Wasser“ und desto größere Regionen im Bild fallen in das Segmentierungskriterium (also werden mit Farbe gefüllt oder mit der Markierung belegt). Das Verfahren kann an obiger Grafik nachvollzogen werden. Der Wasserstand steigt von links nach rechts, das heißt es werden immer größere Anteile des Bildes „geflutet“.
Auch diese Verfahren sukzessive für alle x bzw. y-Werte (je nachdem, wie die Funktion quergeschnitten wurde) wiederholt.
Gibt es Punkte, die bei einem unmittelbar bestimmten „Wasserstand“ direkt an zwei überflutete Gebiete im Graphen grenzen, so werden diese als Wasserscheide bezeichnet. Wird der „Wasserstand“ zu hoch gewählt (also ist der Schwellwert zu hoch), so steht der gesamte Graph „unter Wasser“ (die Aktion wird also auf das komplette Bild ausgeführt, unabhängig von den im Bild vorkommenden Farben).

3. Vor- und Nachteile

Vorteil dieses Verfahrens ist, dass es recht einfach zu implementieren ist und keine großen Ansprüche an die Rechenleistung des Computers stellt, auf dem es ausgeführt werden soll.

Bei diesem Verfahren sind ähnliche Probleme anzuführen, wie sie auch beim Region Growing aufgetaucht sind: Es ist dem Bild nicht anzusehen, wie hoch der Schwellwert gewählt werden muss, um gerade die Regionen zu segmentieren, die für den Benutzer auch von Interesse sind. Abgesehen davon muss auch für die Wasserscheidentransformation geeignetes Bildmaterial vorhanden sein: Regionen können zwar nicht „auslaufen“ (da man sich nicht auf eine einzelne Region beschränkt, sondern auf dem ganzen Bild arbeitet). Trotzdem können Regionen zum segmentierten Bereich hinzugenommen werden, die der Benutzer gar nicht auswählen wollte, nur weil sie zufälligerweise gerade beim gleichen Schwellwert geflutet werden, aber eigentlich gar nicht von Interesse sind.

Bewertung der Verfahren in der Praxis

Region Growing ist am Besten geeignet, wenn größere zusammenhängende Flächen markiert oder mit einer anderen Farbe gefüllt werden sollen.

Sind die zu verändernden Regionen nicht miteinander verbunden, haben aber alle mehr oder weniger den gleichen Farbwert, so sollte die *Wasserscheidentransformation* gewählt werden.

Die *interaktive Kantenverfolgung* kommt dann zum Einsatz, wenn komplexere Objekte ausgewählt werden sollen, die im Inneren keine homogene Farbgebung aufweisen und deren Kantenverlauf zu filigran ist, um ihn mit der Hand oder einem Standardwerkzeug auswählen zu können.

Für verschiedene Aufgaben muss das geeignete Verfahren ausgewählt werden. Letzten Endes sollte der Benutzer diese drei Möglichkeiten einfach testen und so feststellen, welches Tool das optimalste für die Anforderungen ist, die er stellt.

Quellen:

- http://de.wikipedia.org/wiki/Algorithmus_von_Dijkstra
- Interactive Segmentation with Intelligent Scissors - Mortensen Barrett GMaIP 1998
- Klaus D. Tönnies, Grundlagen der Bildverarbeitung, Pearson Studium, 2005.