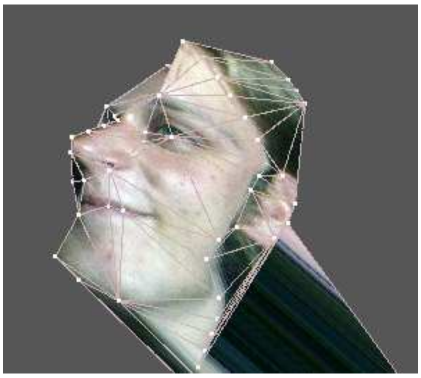
3D-Rekonstruktion und Visualisierung natürlicher Szenen auf der Basis von Polygonmodellen

Norman Feske

10. Januar 2002



Inhaltsverzeichnis

1 Generelle Vorgehensweise **6**

2 Segmentierung **8**

2.1 Was ist eine 'sinnvolle' Segmentierung? 8

2.2 Segmentierung durch vektorbasierte neuronale Netze 8

2.2.1 Wirkungsprinzip vektorbasierter neuronaler Netze . . . 8

2.2.2 Vorverarbeitung - Generieren einer geeigneten Verteilung 9

2.2.3 Räume 11

2.2.4 Bewertung des Lernprozesses 13

2.2.5 Triangulationen 13

2.3 Netzwerkmodelle 15

2.3.1 Learning Vector Quantisation 16

2.3.2 k-means 19

2.3.3 Simulated Annealing 22

2.3.4 Neuronales Gas 22

2.4 Bewertung 24

3 Tiefenschätzung durch Blockmatching **27**

3.1 Korrespondenzanalyse 29

3.2 Untersuchungen im Kantenbild 30

3.3 Untersuchungen im RGB-Farbraum 30

3.4 Untersuchungen HSV-Farbraum 31

3.5 Ausblenden unmöglicher Suchbereiche 34

3.6 Lokale Minima 34

3.7 FFT Block Matching 35

3.8 Bewertung 35

4 Visualisierung **37**

4.1 Polygonnetz 37

4.2 3D Polygonnetz 37

4.3 Texture Mapping 38

5 Implementierung **40**

5.1 Lernbeispiel 41

5.2 OpenGL 41

5.3 Tcl/Tk 41

5.4 Posix-Threads 42

5.5 Shared Objects 43

6 Zusammenfassung und Ausblick **44**

Einführung

Oft sind Verfahren zum automatischen 'Scannen' von dreidimensionalen Objekten mit einem hohen Hardwareaufwand verbunden. So kommen beispielsweise Laser in Verbindung mit spezieller Aufnahmehardware zum Einsatz. Bei einigen Aufnahmetechniken werden Objekte auf einer Drehscheibe positioniert, was von vornherein die Art der Objekte einschränkt. Im Rahmen meines großen Beleges sollte die Gewinnung von Polygonmodellen aus Stereo-Aufnahmen mit Hilfe von vektorbasierten neuronalen Netzen untersucht werden. Dabei kommen lediglich zwei handelsübliche Kameras in Verbindung mit einem PC zum Einsatz. Ziel dieser Arbeit ist die Gewinnung eines $2\frac{1}{2}$D-Modells der aufgenommenen Szene. Das bedeutet, daß einem Pixel des aufgenommenen Bildes nur maximal ein Tiefenwert zugeordnet wird (Tiefenkarte). Als Teilaufgaben werden in der Aufgabenstellung

- Bildsegmentierung durch vektorbasierte neuronale Netze,
- Tiefenschätzung durch Blockmatching,
- und Visualisierung

genannt. Es sollte hierbei ein Programmsystem entstehen, welches als Lehrbeispiel für die Anwendung eines Stereokopfes Verwendung finden kann. Der Bezug zum entwickelten Programmsystem wird innerhalb des folgenden Textes durch entsprechende Abbildungen von Teilen der Benutzeroberfläche hergestellt.

Polygonmodelle natürlicher Objekte haben in vielen Bereichen zahlreiche Einsatzmöglichkeiten. Die Modelle beschreiben lediglich die Oberfläche des betroffenen Objektes. Hierbei wird versucht, die tatsächliche Objektoberfläche durch ein Polygonnetz zu approximieren. Die entstehenden Daten sind sehr kompakt und lassen vielfältige Visualisierungs- und Weiterverarbeitungsmöglichkeiten zu.

Auch aus einem anderen Grund macht es Sinn, die Oberfläche natürlicher Objekte durch Polygone zu modellieren: Die Tiefeninformationen des Polygonmodells sollen aus einem Paar von Stereoaufnahmen gewonnen werden. Aus der horizontalen Verschiebung eines Objektpunktes zwischen den Aufnahmen kann die Tiefeninformation dieses Punktes ermittelt werden. Die Schwierigkeit besteht darin, korrespondierende Punkte auf den beiden Aufnahmen zuzuordnen. Meist besteht die Aufnahme einer natürlichen Szene zu großen Teilen aus Flächen. Punkte innerhalb der Flächen liefern keine Informationen über deren Tiefe im Raum. Dagegen bieten Regionen mit großen Kontrasten die Möglichkeit der Zuordnung. Es erscheint sinnvoll, die Punkte des Polygonnetzes genau an die 'interessanten' Stellen des Bildes zu platzieren.

1 Generelle Vorgehensweise

Ausgangspunkt ist ein Paar von Stereoaufnahmen. Zur Aufnahme des Stereobildpaares werden zwei Kameras in Standardstereogeometrie-Anordung verwendet. D.h. die Kameras sind parallel ausgerichtet, haben die gleichen internen Kameraparameter und die gleiche Orientierung des Kamerakoordinatensystems. Die beiden Aufnahmen werden im Folgenden als linke und rechte Aufnahme bezeichnet. Aus den Stereoaufnahmen ist es prinzipiell möglich, Tiefeninformationen über die dargestelle Szene zu gewinnen. Genutzt werden dabei die sogenannten Disparitäten korrespondierender Objektpunkte. Durch die Verwendung der standardstereogemetrischen Anordnung der Kameras kann der Suchbereich eines korrespondierenden Punktepaares auf eine Dimension beschränkt werden. Es besteht ein einfacher Zusammenhang zwischen der Disparität eines Punktes und dessen Tiefe im Raum. Das Problem für einen Algorithmus besteht im Zuordnen korrespondierender Punkte der beiden Aufnahmen zueinander. Einfache Algorithmen, wie der Blockmatching-Algorithmus, suchen entlang einer horizontalen Geraden nach ähnlichen Blöcken. Solche Verfahren haben allerdings den Nachteil einer sehr langen Laufzeit. Außerdem versagt Blockmatching bei größeren einfarbigen Flächen.

Es wird zunächst versucht die rechte Aufnahme in 'sinnvolle' Bereiche zu segmentieren. Die Segmentierung soll anhand von Polygonen (meist Dreiecken) geschehen. Nachdem die Aufnahme komplett segmentiert wurde, d.h. mit einem Polygonnetz überdeckt wurde, sollen den einzelnen Polygonnetzpunkten Tiefeninformationen zugeordnet werden.

Es stellen sich folgende Probleme:

1. Was ist eine 'sinnvolle' Segmentierung?
2. Wie lautet der Algorithmus für eine solche Segmentierung?
3. Wie erfolgt die Zuordnung der Tiefe zu den Polygonpunkten?

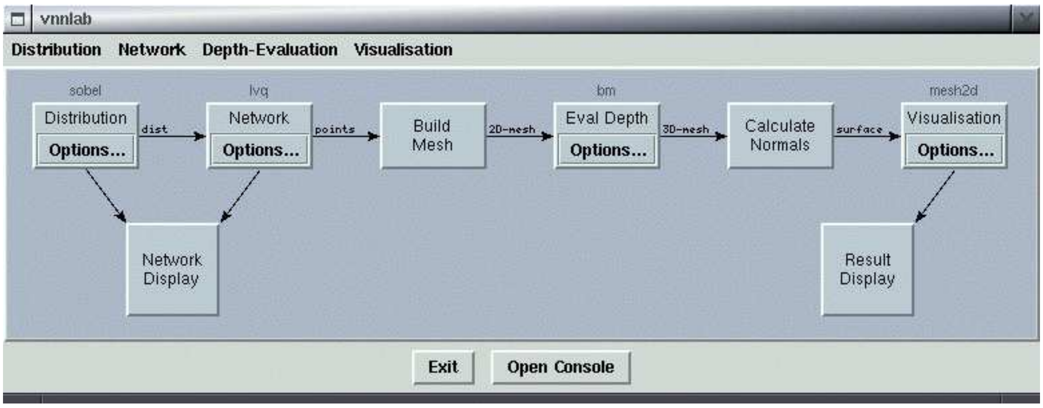


Abbildung 1: Darstellung des Datenflusses im Hauptfenster des Programmsystems

Die hier beschriebene Vorgehensweise spiegelt sich im Aufbau der Benutzeroberfläche des entwickelten Programmsystems wieder. In Abbildung 1 ist das Hauptfenster des Programmsystems dargestellt. Der Datenflußgraph bietet gleichzeitig die Grundlage der Bedienung des Programmes. Die einzelnen Verarbeitungsschritte sind als Kästchen dargestellt. Der Parameterdialog eines Verarbeitungsschrittes kann durch einen Mausklick auf den 'Options'-Button des entsprechenden Kästchens aufgerufen werden. Die Pfeile zwischen den Kästchen stellen den Fluß der Daten zwischen den Verarbeitungsschritten dar. Dabei erscheint der Typ der Daten als Beschriftung an den Pfeilen.

Zunächst wird eine diskrete Verteilung erzeugt ('Distribution'). Diese wird als Datenmenge zum Training eines Netzes ('Network') verwendet. Das Ergebnis des Lernprozesses ist eine Menge von Punkten. Aus dieser Punktemenge wird ein Polygonnetz erzeugt ('Build Mesh'). Anschließend wird den Polygonnetzpunkten deren Tiefeninformationen zugewiesen und dem Polygonnetz hiermit eine dritte Dimension verliehen. Um das Polygonnetz visualisieren zu können, werden die Normalenvektoren der Polygone und deren Punkte berechnet ('Calculate Normals'). Das Ergebnis ('Surface') wird anschließend dargestellt ('Visualisation').

2 Segmentierung

2.1 Was ist eine 'sinnvolle' Segmentierung?

Ein Algorithmus hat natürlich kein Wissen über den dargestellten Kontext der Szene. Es ist nötig, das Wort 'Sinn' in unserem Zusammenhang näher zu beleuchten. Hierbei spielt die Interpretation des Szeneninhaltes keine Rolle. Vielmehr sind geometrische Eigenschaften der Szene von Bedeutung.

Die Aufnahmen beinhalten charakteristische Elemente, wie Kanten oder einzelne Punkte, die auf beiden Aufnahmen gut zu erkennen sind. Für solche Elemente ist es prinzipiell möglich, ihre Disparitäten und damit ihre Tiefe zu ermitteln. Dagegen bieten große Flächen keine Möglichkeiten Tiefeninformationen zu gewinnen. Eine 'sinnvolle' Segmentierung ist also eine solche, in der die Polygonnetzpunkte an den aussagekräftigsten Bereichen angeordnet sind. Dagegen sollen zusammenhängende Flächen keine Polygonpunkte enthalten. Ein anderer Gesichtspunkt ist der Grad der Segmentierung. Natürlich bietet eine sehr feine Segmentierung den Vorteil, eine Szene sehr genau zu beschreiben. Allerdings besteht hierbei die Gefahr, daß sich der Algorithmus in Details verliert und das Bild in winzige (im Grenzfall pixelgroße) Segmente unterteilt. Eine sehr feine Segmentierung hat sehr komplexe Polygonmodelle zur Folge. Sehr einfache Segmentierungen erlauben keine aussagekräftigen Polygonmodelle.

2.2 Segmentierung durch vektorbasierte neuronale Netze

2.2.1 Wirkungsprinzip vektorbasierter neuronaler Netze

Vektorbasierte neuronale Netze bestehen aus einer Menge von Netzknoten und einer Menge von Verbindungen dieser Netzknoten. Die Netzknoten befinden sich im n-dimensionalen Raum. Bei der hier vorliegenden Aufgabenstellung wird die Dimension des Raumes 2 sein. Ein solches Netz hat die Fähigkeit, sich an eine vorgegebene Verteilung anzupassen. Dabei hat das Netz keinerlei Informationen über die Ursache der Verteilung und deren Gesetzmäßigkeiten. Nach einem Lernvorgang sollte das Netz die zugrundeliegende Verteilungsfunktion so gut wie möglich nachgebildet haben. Dem Netz werden der Reihe nach (iterativ) zufällige Beispielvektoren aus der gegebenen Verteilung präsentiert. Ein solches Beispiel wird Datenvektor genannt. Der Abstand zwischen dem Datenvektor und dessen nächstem Netzknoten wird Quantisierungsfehler genannt. Das Ziel der Verfahren ist es, die Netzknoten (Referenzvektor) so zu bewegen, daß die Quantisierungsfehler minimiert werden.

2.2.2 Vorverarbeitung - Generieren einer geeigneten Verteilung

Die vektorbasierten neuronalen Netze benötigen also eine Verteilung, aus dieser ihnen zufällige Datenpunkte präsentiert werden. Wie sieht nun eine solche Verteilung im vorliegenden Fall aus?

Ziel soll es sein, daß die Netzknoten nach dem Lernprozeß genau solche Positionen einnehmen, die große Kontraste aufweisen. Dies hat zweierlei Gründe. Zum einen repräsentieren Kontraste (Kanten, Konturen, einzelne Punkte) den Inhalt des Bildes. Das Netz soll im Endzustand natürlich in gewisser Weise den Inhalt des Bildes repräsentieren.

Zum anderen bieten solche charakteristischen Punkte die Möglichkeit, Tiefeninformation mit maximaler Zuverlässigkeit (bei der Verwendung von Blockmatching) zu ermitteln. Offensichtlich bietet sich eine Kantenfilterung an, um genau diese 'interessanten' Stellen im Bild zu erhalten.

In praktischen Versuchen wurde der Sobel-Filter mit nachgeschaltetem Schwellwert verwendet. Dieser bietet den Vorteil einer effizienten Implementierungsmöglichkeit und liefert zudem Resultate, die sich gut als Basis einer diskreten Verteilung eignen. Andere Kantenfilter (wie z.B. der in [1] beschriebene LoG-Operator) sind sehr anfällig gegenüber Rauschen, wohingegen der Sobel-Filter eine glättende Wirkung hat. Desweiteren liefert der Sobel-Filter

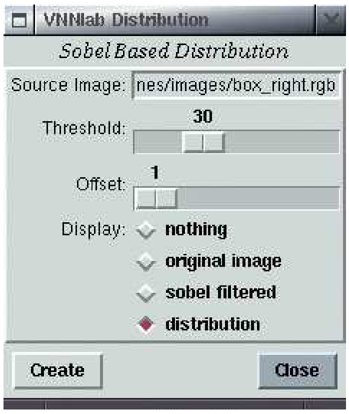


Abbildung 2: Parameter für die Generierung einer Verteilung nach der Sobel-Filter Methode

anschauliche Informationen über die 'Stärke' von Kanten und macht es durch den nachgeschalteten Schwellwert möglich, gezielt Kanten ab einer gewissen Stärke in die Verteilung einfließen zu lassen.

Die Antwort des Sobel-Filters ist der Betrag des Gradientenvektors an der entsprechenden Bildposition. Der Gradientenvektor besteht aus zwei Komponenten, die mittels der folgenden Filtermatrizen ermittelt werden.

$$\text{horizontale Komponente des Gradienten: } SM_{hor} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$
$$\text{vertikale Komponente des Gradienten: } SM_{ver} = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Die 'Gradientenbilder' ergeben sich aus der Faltung des Ausgangsbildes B mit der entsprechenden Filtermatrix:

$$grad_{hor}(x,y) = \sum_{i=-1}^{1} \sum_{j=-1}^{1} SM_{hor}(i+1, j+1) * B(x+i, y+j)$$
$$grad_{ver}(x,y) = \sum_{i=-1}^{1} \sum_{j=-1}^{1} SM_{ver}(i+1, j+1) * B(x+i, y+j)$$

Die eigentliche Antwort des Sobel-Filters ist der Betrag der Gradienten:

$$sobel(x,y) = \sqrt{grad_{hor}(x,y)^2 + grad_{ver}(x,y)^2}$$

Das gefilterte Bild wird nun als diskrete Verteilung von Datenpunkten aufgefasst. Durch den Schwellwert ist es gut möglich, die Größe der Verteilung zu regulieren. Abbildung 2 zeigt den Programmdialog zur Generierung einer diskreten Verteilung nach der hier beschriebenen Methode. Der Parameter 'Offset' gibt an, in welcher Schrittweite auf die Bildpunkte zugegriffen wird. Ein hoher Wert für diesen Parameter bewirkt, daß der Filter nicht auf sehr hohe Frequenzen im Bild anspricht (z.B. einzelne Haare), sondern starke Kanten betont werden.

Dies ist sicher eine sehr einfache Methode der Gewinnung einer diskreten Verteilung aus einem Quellbild. Denkbar wäre anstelle des Schwellwertes eine Gewichtung der verschiedenen Filterantworten zu verwenden.

Die erhaltene diskrete Verteilung wird als eine Menge von Datenvektoren aufgefaßt und kann zum Training vektorbasierter neuronaler Netze verwendet werden.

2.2.3 Räume

Der betrachtete Problembereich des Segmentierens von Bildern hat bestimmte Eigenschaften der Räume zur Folge, in dem das Netz arbeitet:

1. Der Datenraum ist endlich. Da die zugrunde liegenden Bilder eine endliche Größe haben, ist natürlich auch der Raum der resultierenden Datenpunkte begrenzt. Dies ermöglicht beispielsweise eine gleichmäßige Verteilung von Referenzvektoren über den gesamten Datenraum. Dies ist insbesondere bei der Initialisierung der Netzwerke von großem Vorteil.

2. Der Datenraum ist zweidimensional. Diese Eigenschaft macht den Problembereich gut visualisierbar.

Abbildung 3: Ausgangsbild zur Generierung einer diskreten Verteilung

Abbildung 4: Ergebnis der Anwendung des Sobel Operators

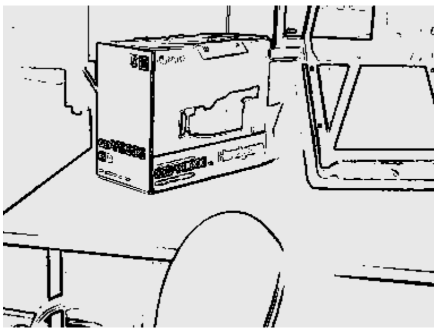
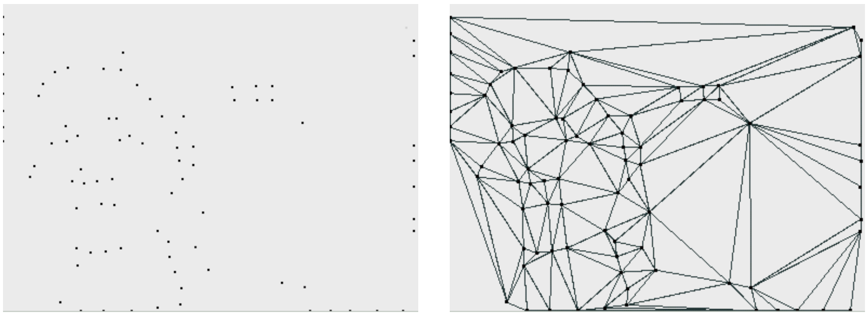


Abbildung 5: Aus dem Sobel gefilterten Bild mit Hilfe eines Schwellwertes gewonnene diskrete Verteilung



a) b)

Abbildung 6: Beispiel für das Bilden einer (fast) Delaunay Triangulation b) der Punktemenge aus Teilbild a) nach dem vorgestellten Algorithmus

2.2.4 Bewertung des Lernprozesses

Oft ist es für die Bewertung des Lernprozesses sinnvoll, ein Maß für dessen Erfolg zu haben. Eine mögliche Bewertungsgröße hierfür ist die Summe der Quantisierungsfehler beim Durchlauf von einer vorgegebenen Anzahl von Lernschritten. Der Quantisierungsfehler ist der euklidische Abstand zwischen einem gegebenen Eingabevektor und dessen nächstem Referenzvektor.

2.2.5 Triangulationen

Häufig ist das Ergebnis des Lernalgorithmus noch nicht sofort als Segmentierung verwendbar. Die Referenzvektoren müssen derart durch Polygone verbunden werden, daß ein vollständiges Netz von Polygonen entsteht.

Dabei erweist es sich als notwendig, ausschließlich mit Dreiecken als Polygonen zu arbeiten. Schließlich sollen den einzelnen Polygonpunkten später Tiefeninformationen zugewiesen werden. Polygone mit mehr als drei Ecken führen hierbei zu undefinierten Flächen. Ein einfaches Beispiel ist ein Quadrat, dessen diagonal gegenüberliegende Punkte jeweils den gleichen Tiefenwert haben. Eine Fläche durch diese Punkte ist in jedem Fall gekrümmt. Aber die Art der Krümmung ist nicht eindeutig.

Dreiecke haben den Vorteil, daß es immer eine ebene Fläche gibt, die alle drei Punkte beinhaltet. Eine gute Möglichkeit, eine Menge von gegebenen Punkten im zweidimensionalen Raum durch ein geschlossenes Netz von Dreiecken zu verbinden, ist die Delaunay-Triangulation.

Die Delaunay-Triangulation hat die Eigenschaft, daß der Umkreis jedes Dreieckes der Triangulation keinen Punkt außer den drei Dreieckspunkten beinhalten darf.

Es gibt eine Reihe von Algorithmen zum Bestimmen von Delaunay Triangulationen. Die meisten Algorithmen sind kantenbasiert. D.h. das Ergebnis des Algorithmus sind die Kanten, die zu den Delaunay-Dreiecken gehören. Aus diesen Informationen müßte in einem nachfolgenden Schritt ein Netz von Dreiecken gebildet werden. Die effizientesten Algorithmen haben ein Zeitverhalten von n*log(n), erfordern allerdings eine spezielle Organisation der Daten.

Aus diesen Gründen wurde ein eigener Algorithmus entwickelt, der ein etwas schlechteres Zeitverhalten aufweist, aber mit einfachen Datenstrukturen arbeitet und als Ergebnis Dreiecke liefert, deren Punkte im Urzeigersinn liegen. Dies ist für die spätere Visualisierung von Vorteil (backface culling). Desweiteren ist der Algorithmus sehr kompakt und verfolgt eine einfache und anschauliche Strategie. Es wird zunächst ein beliebiger Punkt 'P1' der Punktemenge und sein nächster Nachbar 'P2' bestimmt. Danach wird unter all den Punkten, die 'rechts vom Weg' von P1 nach P2 liegen, derjenige Punkt 'P3' ausgewählt, der ein Optimalitätskriterium erfüllt. Im Falle der Delaunay-Triangulation ist der Punkt 'P3' optimal, für den die Quadrate der Innenwinkel des entstehenden Dreieckes minimal sind. In der konkreten Implementation wurde ein vereinfachtes Kriterium gewählt, welches nicht immer zur Delaunay-Triangulation führt, aber viel schneller zu berechnen ist: Der Winkel P1-P3-P2 soll maximiert werden. Anschließend wird der Algorithmus jeweils mit den Argumenten P3, P2 und P1, P3 (rekursiv) ausgeführt. Es zeigt sich, daß selbst unter diesem abgeschwächten Kriterium sehr gute Triangulationen erreicht werden.

In Abbildung 6 ist eine Triangulation nach dem hier beschriebenen Algorithmus für eine gegebene Punktemenge dargestellt.

2.3 Netzwerkmodelle

Wesentlich bei der Betrachtung der verschiedenen Netzwerkmodelle sind:

- deren Initialisierung,
- deren Wirkungsweise (auch z.B. Entwicklung des Quantisierungsfehlers),
- das Abbruchkriterium und das Endresultat.

Viele Netzwerkmodelle besitzen Parameter, die die Arbeit der Netze maßgeblich beeinflussen. Generell ist es ungünstig, wenn Algorithmen einen sehr großen Parameterraum haben, die deren Wirkungsweise stark beeinflussen. Auch sollte die Interpretierbarkeit von Parametern nicht außer Acht gelassen werden.

Weiterhin ist es sinnvoll, Parameter dahingehend zu unterscheiden, ob eine einmal gewählte Parameterbelegung nur für einen speziellen Fall geeignet oder günstig für eine große Anzahl von Problemen ist. Im zweiten Fall ist der Algorithmus mit dieser Parameterbelegung sehr gut allgemein einsetzbar. Es muß lediglich diese Parameterbelegung gefunden werden.

Beim Einsatz von vektorbasierten neuronalen Netzen trifft man häufig auf folgende Parameter:

- die Netzwerkgröße bei der Initialisierung und
- die Lernrate.

Letztere gibt an, wie stark sich das Netz in einem Interationszyklus verändern darf. Eine klein gewählte Lernrate hat zur Folge, daß viele Iterationen nötig sind, um den Lernprozeß zufriedenstellend zu absolvieren. Wird die Lernrate zu groß gewählt, gewinnen 'Ausreißer' in der Datenmenge ein großes Gewicht. Die Netze neigen dann dazu, stark zu 'zittern' anstatt zu konvergieren.

Die Wahl dieser Parameter hängt vom Anwendungsbereich ab und sollte durch entsprechende Untersuchungen getroffen werden.

2.3.1 Learning Vector Quantisation

Die Lernmethode 'learning vector quantisation' stellt eine Vorstufe zu den eigentlichen vektorbasierten Netzen dar. Hierbei wird die Topologie des Netzes, also dessen Verbindungen, vollkommen außer Acht gelassen. Die Wirkungsweise der Methode wird als hartes Wettbewerbslernen bezeichnet. Das Netz wird mit einer vorgegebenen Anzahl von Knoten so initialisiert, daß der gesamte Datenraum so gut wie möglich abgedeckt ist. Während des Lernprozesses werden den Knoten Datenpunkte aus der vorliegenden Verteilung präsentiert. Das Netz findet nun denjenigen Referenzvektor, dessen Abstand zum gegebenen Datenvektor minimal ist. Dieser Referenzvektor wird als Gewinner bezeichnet. Die Position des Gewinners wird nun so verändert, daß sich der Gewinner-Referenzvektor in die Richtung des Eingabevektors bewegt. Wie stark die Bewegung ist, hängt von der vorgegebenen Lernrate ab. Dies ist ein Faktor im Intervall [0..1]. Eine Lernrate von 0 hat keine Bewegung des Gewinners zur Folge. Ein Faktor von 1 würde den Gewinner veranlassen, direkt auf die Position des Eingabevektors zu springen.

Es wird schnell deutlich, daß es vor allem in Bereichen des Datenraumes ohne Datenvektoren viele Referenzvektoren gibt, die ihre Position nie verändern, weil sie niemals die Chance haben, zu gewinnen.

Solche Referenzvektoren besitzen für das Segmentierungsproblem keinerlei Wert. Es ist also sinnvoll, solche Referenzvektoren aus dem Endresultat zu entfernen. Um festzustellen, wie wertvoll ein Referenzvektor ist, kann jedem Referenzvektor ein zusätzliches Attribut 'Wichtigkeit' gegeben werden. Dieses Attribut wird mit 0 initialisiert. Ein Gewinner ändert nun nicht nur seine Position, sondern modifiziert auch seinen Wichtigkeitswert w. Wichtig-

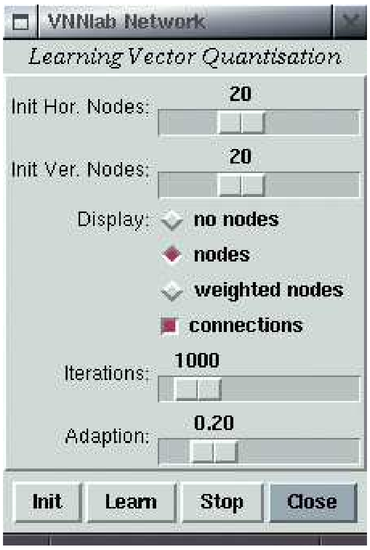


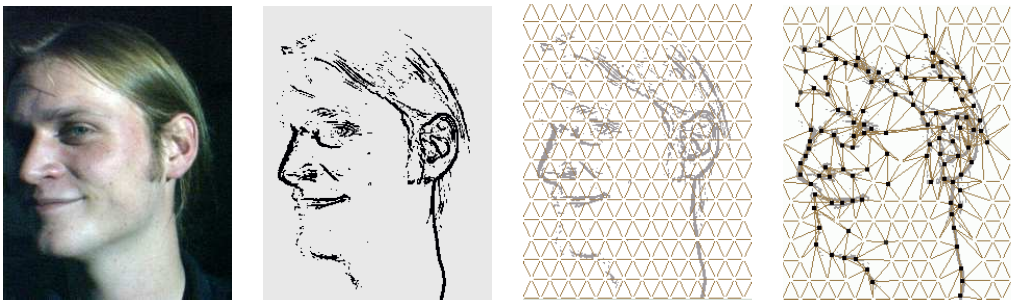
Abbildung 7: Die Parameter des 'learning vector quantisation' Algorithmus.

keit wird im Folgenden mit Gewicht gleichgesetzt. Das Gewicht eines Netz-Knotens ist ein Maß für dessen Güte. Im praktischen Fall wurde die Modifikation der Wichtigkeit wie folgt vorgenommen:

$$w := w + s * (1.0 - w),$$

$s \in (0, 1)$ gibt dabei an, wie stark das Gewicht des Gewinners vergrößert wird. Diese Art der Modifikation wurde gewählt, da sie zum Wert 1.0 konvergiert. Zum Experimentieren wurde $s = 0.5$ verwendet. Dies hat zur Folge, daß ein nie benutzter Referenzvektor stets den Wert 0 behält. Sobald ein Referenzvektor nur einmal Gewinner ist, steigt seine Wichtigkeit stark an. Ein Referenzvektor kann nie wichtiger als 1.0 sein. Dies ist sehr praktisch bei der späteren Visualisierung.

In Abbildung 7 sind sind die Parameter des Algorithmus dargestellt. Die Anzahl der Netzknoten wird durch die Parameter 'Init Hor. Nodes' und 'Init Ver. Nodes' bestimmt. Die Lernrate ist als 'Adaption' bezeichnet. Der Parameter 'Iterations' bestimmt, wie viele Iterationen in einem Darstellungs-



a) b) c) d)

Abbildung 8: Beispiel für die Anwendung der 'learning vector quantisation': Aus dem Ausgangsbild a) wurde mit der Sobel-Filter Methode eine Verteilung b) generiert. Ein Netz der Größe 13x16 ist in c) vor und in d) nach der Anwendung des 'learning vector quantisation' Algorithmus dargestellt.

schritt durchlaufen werden sollen. Er hat keinen Einfluß auf die Wirkungsweise des Netzes, sondern nur auf dessen Visualisierung.

Der Algorithmus in seiner ursprünglichen Form terminiert nicht. Anhand der Entwicklung des durchschnittlichen Quantisierungsfehlers läßt sich jedoch erkennen, daß sich dieser ab einem gewissen Zeitpunkt nicht mehr stark verändert. Unter Zuhilfenahme dieser Eigenschaft läßt sich leicht ein geeignetes Kriterium zur Terminierung definieren.

Abbildung 8 zeigt die Anwendung des Algorithmus auf ein Beispielbild. Zunächst wurde aus dem Ausgangsbild eine diskrete Verteilung mit der in Kapitel 2.2.2 beschriebenden Methode generiert. Diese Verteilung wurde zum Training eines 16x13 Knoten umfassenden Netzes verwendet. In der Darstellung des Ergebnisses (Teilbild d) entspricht die Helligkeit der Netzknoten der Wichtigkeit dieser entsprechenden Knoten. Man sieht deutlich, daß ein Großteil der Knoten überhaupt nicht genutzt wird. Diese werden in [2] als 'dead units' (tote Einheiten) bezeichnet. Die dargestellten Verbindungen zwischen den Netzknoten haben keine Bedeutung für den Algorithmus, bieten aber eine gute Möglichkeit, Aufschluß über die Lageveränderungen der einzelnen Netzknoten zu gewinnen.

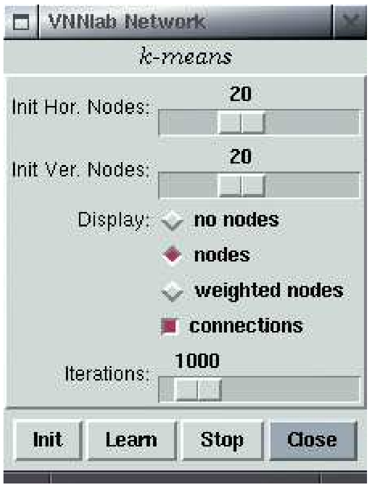


Abbildung 9: Parameter von 'k-means'

Das Ergebnis des Algorithmus ist noch keine Segmentierung. Bislang wurde lediglich mit Referenzvektoren gearbeitet. Eine Segmentierung entsteht durch die Bildung einer Triangulation (das Verbinden der Referenzvektoren durch Dreiecke, sodaß eine geschlossene Fläche entsteht) über den Referenzvektoren. Dabei kann auf all die Referenzvektoren verzichtet werden, deren Wichtigkeit 0 ist (tote Einheiten).

2.3.2 k-means

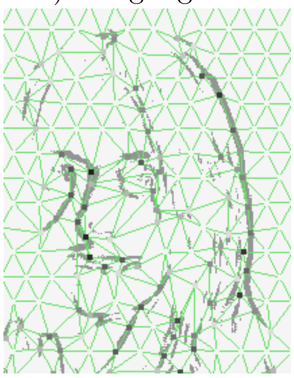
Eine feste Lernrate hat die Nachteile, daß sie von vornherein festgelegt werden muß und daß das Verfahren nicht konvergiert. Es ist vor allem im hier untersuchten Anwendungsbereich der Segmentierung sehr wichtig, daß die Einheiten möglichst exakt auf den Punkten der zugrundeliegenden Verteilung (z.B. dünnen Kanten) liegen. Bei der Anwendung eines Verfahrens mit konstanter Lernrate kommt es zu einem 'Zittern' der Einheiten auf den Kanten. Der Grundgedanke des k-means Algorithmus basiert auf dem langsamen Absenken der Lernrate. Hierfür wird pro Einheit eine eigene Lernrate verwendet, die immer dann verringert wird, wenn die Einheit Gewinner ist. Die

Lernrate einer Einheit c wird definiert als $\epsilon_c(t) = \frac{1}{t}$, wobei t für die Anzahl der Signale steht, für die c der Gewinner gewesen ist. In [2] wird gezeigt, daß ein Referenzvektor stets der Mittelwert aller Eingabesignale ist, für die er der Gewinner war.

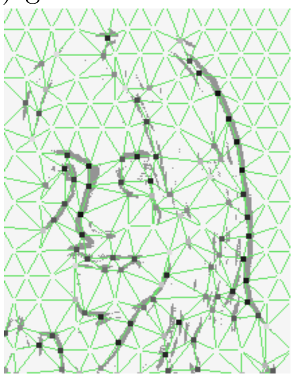
Wie in Abbildung 9 dargestellt, werden für dieses Verfahren außer der Größe des Netzwerkes keinerlei Parameter benötigt. (Die anderen Parameter in der Abbildung beziehen sich auf die Darstellung des Netzes) Die Wirkungsweise des 'k-means' Algorithmus ist für das Beispielbild 'Christin' in Abbildung 10 dargestellt. Dabei erscheinen die 'wichtigen' Netzknoten dunkel. Nach 10000 Iterationen verändert sich das Netz nicht mehr stark. In Teilbild f) ist deutlich zu erkennen, daß die Netzknoten auf Punkten der zugrundeliegenden Verteilung zur Ruhe kommen.

a) Ausgangsbild

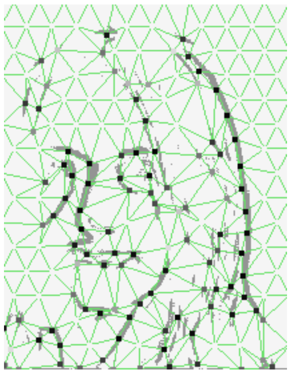
b) generierte Verteilung



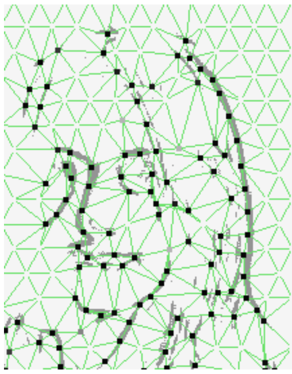
c) 'k-means' nach 500 Iterationen



d) 'k-means' nach 1000 Iterationen



e) 'k-means' nach 2000 Iterationen



f) 'k-means' nach 10000 Iterationen

Abbildung 10: Entwicklung eines Netzes unter der Lernmethode 'k-means'

2.3.3 Simulated Annealing

Eine andere Möglichkeit, die Lernrate abzusenken, wurde in [2] mit

$$\epsilon(t) = \epsilon_i * (\frac{\epsilon_f}{\epsilon_i})^{\frac{t}{t_{max}}}$$

vorgeschlagen. Dabei sind ϵ_i und ϵ_f die Start- und Endwerte der Lernrate, t die aktuelle Iteration des Verfahrens und t_{max} die Anzahl der gesamten Iterationen. Durch das Absenken der Lernrate im Laufe der Zeit konvergiert das Verfahren. Dabei kann $\epsilon(t)$ als Temperatur interpretiert werden, die im Laufe der Zeit abnimmt. Unter hoher Temperatur haben die Referenzvektoren die Möglichkeit, große Sprünge zu machen und damit lokale Minima zu verlassen. Durch die Abkühlung werden solche großen Bewegungen immer weiter eingeschränkt und schließlich nur noch die eigentliche Optimierungsstrategie (Gradientenabstieg) verfolgt. Das Resultat des Verfahrens ist, daß die Referenzvektoren nach dem Lernvorgang genau auf Datenpunkten (also z.B. dünnen Kanten) liegen, für die eine Tiefenschätzung (durch die in Kapitel 3 beschriebenen Verfahren) möglich ist.

2.3.4 Neuronales Gas

Die bisher vorgestellten Verfahren haben die Gemeinsamkeit des harten Wettbewerbslernens. Es wird immer nur die Einheit adaptiert, die für einen gegebenen Datenpunkt der Gewinner ist. Das neuronale Gas ist eine Methode des weichen Wettbewerbslernens. Bei diesem Verfahren werden alle Einheiten nach deren Abstand zum gegebenen Datenvektor ξ sortiert. Die Adaptionsstärke (Lernrate) einer Einheit hängt nun von deren Position in dieser Rangliste ab. Der Gewinner hat den Ranglistenplatz 1. Die Adaption einer Einheit w_{s_k}mit dem Ranglistenplatz k erfolgt durch:

$$\triangle w_{s_k} := \epsilon(t) * h(t, k) * (\xi - w_{s_k})$$

Dabei ist $\epsilon(t)$ wie in Kapitel 2.3.3 definiert. Die Funktion $h(t, k)$ drückt den Einfluß des Ranglistenplatzes auf die Adaptionsstärke aus und wird in [2] mit

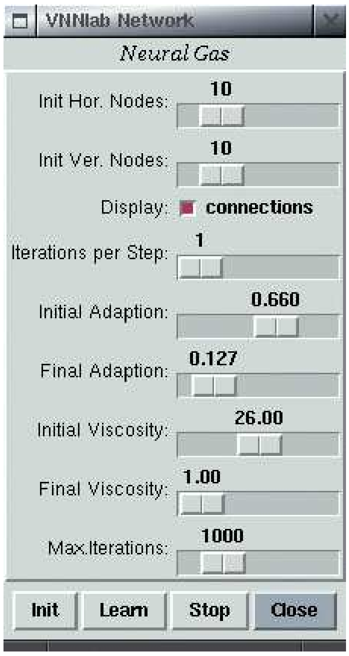


Abbildung 11: Parameter für das 'neuronale Gas'

$$h(t,k) := exp(\frac{1-k}{\lambda(t)})$$

angegeben. Dabei beschreibt $\lambda(t)$ die zeitliche Veränderung der Adaptionsstärke und ist durch

$$\lambda(t) = \lambda_i * (\frac{\lambda_f}{\lambda_i})^{\frac{t}{t_{max}}}$$

(ähnlich wie $\epsilon(t)$) definiert. Durch eine Wahl von $\lambda_i > \lambda_f$ wird bewirkt, daß am Anfang des Lernverganges viele Einheiten zusammen mit dem Gewinner adaptiert werden und sich gegen Ende lediglich der Gewinner selbst an gegebene Datenpunkte anpaßt.

In Abbildung 11 ist der Parameterdialog des Verfahrens dargestellt. Dabei entsprechen 'Adaption' jeweils den Parametern ϵ und 'Viscositiy' (Zähigkeit) jeweils den Parametern λ.

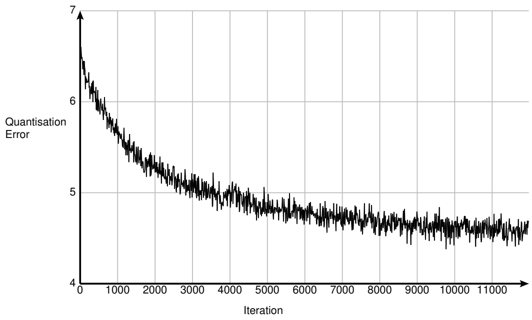
Es fällt auf, daß für dieses Verfahren schon 5 Parameter ($\lambda_i, \lambda_f, \epsilon_i, \epsilon_f$ und t_{max}) existieren. Über die Wahl dieser Parameter werden in [2] keine Aussagen getroffen. Weiterhin hat dieses Verfahren (wie alle Verfahren des weichen Wettbewerbslernens) die Eigenschaft, daß die Referenzvektoren nicht zwangsweise auf den Datenvektoren 'zur Ruhe kommen'. Dies ist aber eine wichtige Vorraussetzung für eine zuverlässige Tiefenbestimmung mittels der hier untersuchten Verfahren. Das neuronale Gas muß durch eine geeignete Wahl der Parameter λ_i und λ_f so konfiguriert werden, daß das Verfahren in den ersten Iterationen sehr 'weich' lernt und schließlich gegen hartes Wettbewerbslernen konvergiert. Somit ist das Netz in der Lage, sich zu Beginn fast vollständig zu einem 'interessanten' Ort im Merkmalsraum zu bewegen. An diesem Ort angekommen, beschränkt sich das Lernen eines Netzknotens nur noch auf einen lokalen Bereich. Die Konsequenz ist, daß unabhängig vom Bild alle Referenzvektoren zur Repräsentation der Daten dienen und es somit keine toten Einheiten (dead units) mehr gibt.

Leider wird dieser Vorteil mit einem großen Parameterraum erkauft. Im hier untersuchten Bereich der Bildsegmentierung können gute Parameter experimentell ermittelt werden, da sich der Einfluß der Parameter unmittelbar visuell bemerkbar macht. Durch systematisches Probieren wurde eine Parameterkonfiguration ermittelt, die sich zur Segmentierung unterschiedlicher Bilder als geeignet erwiesen hat: $\epsilon_i = 0.7, \epsilon_f = 0.15, \lambda_i = 25.0, \lambda_f = 0.1, t_{max} = 1000$

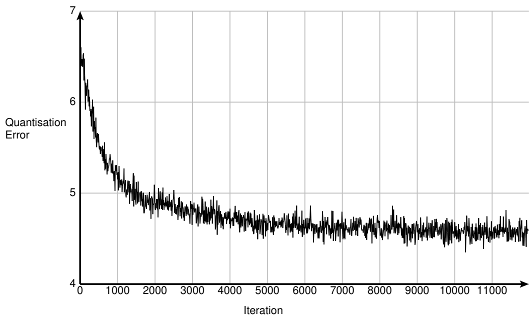
2.4 Bewertung

Die Algorithmen 'Learning Vector Quantisation', 'k-means' und 'Simulated Annealing' sind gut miteinander vergleichbar. Dabei erreicht schon der einfachste Algorithmus 'Learning Vector Quantisation' (LVQ) bei der Benutzung einer Lernrate von 0.2 verwendbare Ergebnisse. Ein Nachteil von LVQ beim Einsatz zur Segmentierung von Einzelbildern ist, daß sich das Netz nicht beruhigt und die Netzknoten oft nicht auf den Punkten der Datenmenge liegen.

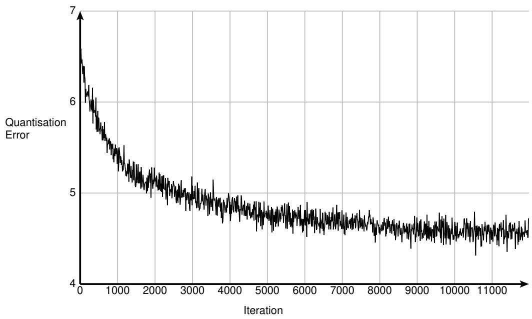
Besser verhält sich 'k-means'. In Abbildung 12 ist deutlich erkennbar,



a) 'Learning Vector Quantisation'



b) 'k-means'



c) 'Simulated Annealing'

Abbildung 12: zeitliche Entwicklung des mittleren Quantisierungsfehlers beim Lernen eines Netzes mit den Algorithmen: a) 'Learning Vector Quantisation', b) 'k-means' und c) 'Simulated Annealing' daß sich 'k-means' am schnellsten an eine gegebene Datenmenge anpaßt. Desweiteren hat 'k-means' den Vorteil, daß keinerlei Parameter außer der Netzgröße benötigt werden.

'Simulated Annealing' weist ein sehr ähnliches Verhalten wie 'k-means' auf. Das Verfahren läßt sich stark durch seine Parameter beeinflussen. In [2] wird gezeigt, daß das Verfahren konvergiert. Somit ist es gut möglich, ein geeignetes Abbruchskriterium für den Algorithmus zu definieren.

Das 'neuronale Gas' unterscheidet sich maßgeblich von den bisher genannten Verfahren, da es in einer Iteration sämtliche Netzknoten adaptiert. Die Ergebnisse sind sehr von den Parametern des Verfahrens abhängig. Dabei kann sich das Verfahren relativ gut bei Szenen ohne komplizierten Bildhintergrund verhalten. Das 'neuronale Gas' hat die Eigenschaft, daß sich lange 'Ketten' von Einheiten bilden, die sich entlang von Kanten anordnen. Der größte Vorteil des 'neuronalen Gases' gegenüber den anderen untersuchten Verfahren ist, daß sämtliche Netzknoten zum Lernen der Datenmenge genutzt werden und es somit keine 'toten Einheiten' gibt. Insgesamt konnte das Verfahren für das hier behandelte Problem der Bildsegmentierung bei einer großen Anzahl von Testbildern jedoch nicht überzeugen.

Der mittlere Quantisierungsfehler ist nicht die einzige Größe, die zur Bewertung der verschiedenen Algorithmen herangezogen werden sollte. Eine qualitative Beurteilung der erzeugten Segmentierung ist sehr wichtig. Hier hat sich bei den durchgeführten Experimenten das Verfahren 'Simulated Annealing' bei geeigneter Wahl seiner Parameter sehr bewährt. Die einmal gefundenen Parameter ($\epsilon_i = 0.4, \epsilon_f = 0.05, t_{max} = 10000$) haben sich für verschiedenste Bilder als geeignet herausgestellt. Da alle Netzeinheiten auf Punkten der Datenmenge (die den 'interessanten' Punkten im Ausgangsbild entsprechen) zur Ruhe kommen, konnte auch die Tiefenschätzung erfolgreicher durchgeführt werden als bei den anderen Verfahren.

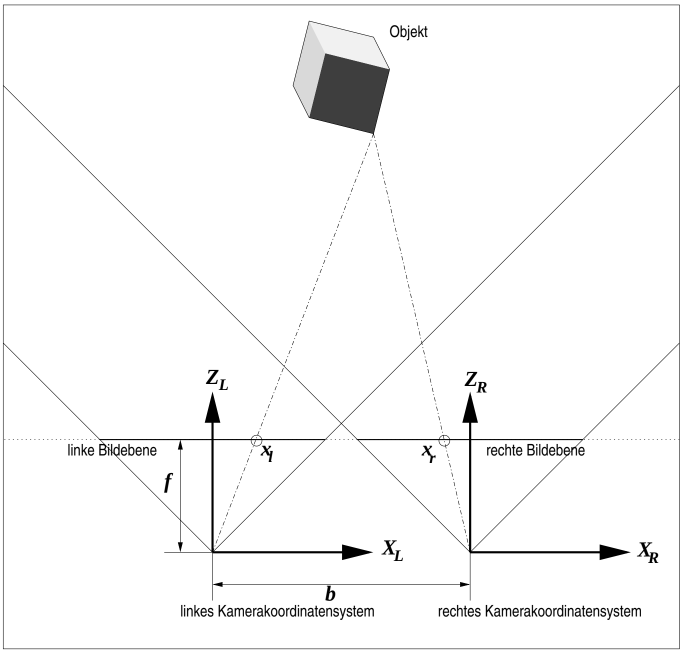


Abbildung 13: Standardstereogeometrie unter der Verwendung des Lochkameramodells

3 Tiefenschätzung durch Blockmatching

Alle hier untersuchten Verfahren basieren auf lokaler Korrespondenzanalyse. Es wird versucht, zu einem Punkt im rechten Bild den entsprechenden Punkt im linken Bild zu finden. Die Verschiebung der Projektionen eines Punktes auf die Bildebenen der linken und rechten Kamera wird Disparität genannt. Es besteht ein einfacher Zusammenhang zwischen der Disparität eines Punktes und dessen Entfernung vom Ursprung des Kamerakoordinatensystems:

$$Z = \frac{f * b}{d}$$

f ist die Brennweite der Kameras, b ist der Abstand zwischen den Ursprüngen der Kamerakoordinatensysteme und d ist die Disparität eines Objektpunktes.

Dabei ist $f * b$ eine Konstante, die lediglich eine Skalierung der gewonnenen Tiefenwerte bewirkt.

Das Problem, die Tiefeninformation eines Objektpunktes zu bestimmen, reduziert sich also auf das Ermitteln korrespondierender Punkte in der linken und rechten Aufnahme (Korrespondenzanalyse). Dabei können verschiedene Einschränkungen gemacht werden, die zum Teil aus der Verwendung der Standardstereogeometrie folgen.

Abbildung 13 illustriert die Standardstereogeometrie anhand des Lochkameramodells. Die beiden Kameras sind lediglich horizontal verschoben (Verschiebung b). Die inneren Kameraparameter (Öffnungswinkel, Brennweite f, Hauptpunkt) der beiden Kameras sind identisch. In der Abbildung ist beispielhaft die Projektion eines Objektpunktes auf die Bildebenen gekennzeichnet. Dabei ist x_l die x-Koordinate der Abbildung des Objektpunktes auf die linke Bildebene bezüglich des linken Kamerakoordinatensystems und x_r die x-Koordinate der Abbildung des Objektpunktes auf die rechte Bildebene bezüglich des rechten Kamerakoordinatensystems.

Ein zu einem gegebenen Punkt der rechten Aufnahme korrespondierender Punkt in der linken Aufnahme liegt auf einer (durch die Anordnung der Kameras) definierten Geraden (Epipolarlinie) im linken Bild. Durch die parallele und nur horizontal verschobene Anordnung der Kameras stimmen die Epipolarlinien der beiden Bildebenen mit den Bildzeilen überein. Daraus ergibt sich:

$$Z = \frac{f*b}{x_l - x_r}$$

Somit muß die Suche eines zu einem Punkt im rechten Bild korrespondierenden Punktes im linken Bild nur entlang einer Bildzeile des linken Bildes geschehen.

Weitere Informationen hierzu findet man in [1].

3.1 Korrespondenzanalyse

Die Korrespondenzanalyse auf der Basis von Blockmatching wird durch den Vergleich der Ähnlichkeit von Pixelblöcken realisiert. Zu einem gegebenen Pixelblock im rechten Bild (E_r) wird der Pixelblock im linken Bild (E_l) gesucht, der ihm am ähnlichsten ist. Hierzu wird ein Ähnlichkeitsmaß benötigt. Ein naheliegendes Ähnlichkeitsmaß ist der quadratische Fehler (square error). Dieser ist für die beiden Pixelblöcke E_l und E_r wie folgt definiert:

$$dissim_{SE}(E_l, E_r, x_l, x_r, y) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} (E_l(x_l+i, y+j) - E_r(x_r+i, y+j))^2,$$

wobei die Blöckgröße $2 * k + 1$ vorgegeben wird.

Dieses Ähnlichkeitsmaß ist eigentlich ein Unähnlichkeitsmaß, da es minimal ist, wenn beide Pixelblöcke exakt gleich sind. Leider zeigt sich, daß dieses Maß sehr anfällig gegenüber Rauschen und Shading-Störungen ist. Die mittlere Helligkeit der Pixelblöcke geht maßgeblich in die Berechnung ein, was sich bei der Benutzung zweier (nicht exakt gleich eingestellter) Kameras und Shading-Störungen sehr negativ auswirkt. Desweiteren kann bei natürlichen Aufnahmen nicht davon ausgegangen werden, daß y_l exakt mit y_r übereinstimmt. Besonders negativ macht sich dieser Umstand bei hochfrequenten Texturen bemerkbar. Der Extremfall ist hier ein Schachbrettmuster. Eine Differenz von nur einem Pixel zwischen $y_l$und y_r führt zu einer maximalen Unähnlichkeit der Pixelblöcke. Bei starkem Shading konvergiert das Ähnlichkeitsmaß lediglich zu einem Vergleich der mittleren Blockhelligkeiten.

Im vorliegenden Verfahren wird direkt mit den Bildwerten (Irradianzen) gearbeitet. Es wird vorrausgesetzt, daß ein Objektpunkt auf der linken Aufnahme eine ähnliche Helligkeit aufweist, wie auf der rechten Aufnahme. Die Helligkeit des Objektpunktes muß also unabhängig vom Betrachterwinkel sein. Diese Forderung wird nur von diffus reflektierenden Oberflächen erfüllt. Hier wird das einstrahlende Licht in alle Richtungen gleichmäßig reflektiert. Diese Oberflächen werden Lambertsche Oberflächen genannt. Bei realen Objekten ist dies oft nicht der Fall. Zusätzlich zur diffusen Reflektion tritt meist auch spiegelnde (specular) Reflektion auf, welche abhängig vom Betrachterwinkel (Winkel zwischen der Oberflächennormalen im Objektpunkt und dem Vektor vom Betrachter zum Objektpunkt) ist. Beispielsweise treten im 'Phong' Beleuchtungsmodell diese beiden Komponenten additiv auf. In den folgenden Verfahren wird eine Lambertsche Oberfläche des Objektes vorausgesetzt.

Die Untersuchungen des Einsatzes des hier beschriebenen Verfahrens bei der Anwendung auf Aufnahmen natürlicher Objekte führte erwartungsgemäß zu keinen verwertbaren Ergebnissen. Im folgenden wird versucht, die Störanfälligkeit des Verfahrens zu verringern.

3.2 Untersuchungen im Kantenbild

Shading-Störungen sind niederfrequente Störungen. Um das Verfahren invariant gegenüber solchen Störungen zu machen, liegt es nahe, die Bilder vor der Korrespondenzanalyse mit einem Hochpaßfilter zu filtern. Hierzu wurden Experimente mit dem Sobel-Filter durchgeführt. Der Sobel-Filter hat die angenehme Eigenschaft, das Bild zusätzlich zu glätten. Hierdurch kann der Einfluß von eventuellem Rauschen maßgeblich verringert werden.

3.3 Untersuchungen im RGB-Farbraum

Bei der Verwendung von Farbkameras bietet sich natürlich die Nutzung der Farbinformation zur Korrespondenzanalyse an. Die Bilder liegen nach der Aufnahme im RGB-Farbformat vor. Ein Pixel besteht aus den Komponenten R (rot), G (grün) und B (blau). Daraus lassen sich die Intensitäten der normierten Farbwertanteile gewinnen:

$$r = \frac{R}{R+G+B},\ g = \frac{G}{R+G+B},\ b = \frac{B}{R+G+B}$$

In [1] wurden vier verschiedene Ähnlichkeitsmaße für Farben angegeben. Gegeben sind zwei Farben $F_1 = (r_1, g_1, b_1)$ und $F_2 = (r_2, g_2, b_2)$. Die Differenz zwischen zwei Farben beträgt nach diesen Ähnlichkeitsmaßen:

$$
\begin{aligned}
D_1(F_1, F_2) &= \sqrt{(r_1 - r_2)^2 + (g_1 - g_2)^2 + (b_1 - b_2)^2}, \\
D_2(F_1, F_2) &= \mid r_1 - r_2 \mid + \mid g_1 - g_2 \mid + \mid b_1 - b_2 \mid, \\
D_3(F_1, F_2) &= \mid r_1 - r_2 \mid^2 + \mid g_1 - g_2 \mid^2 + \mid b_1 - b_2 \mid^2, \\
D_4(F_1, F_2) &= \max\{\mid r_1 - r_2 \mid, \mid g_1 - g_2 \mid, \mid b_1 - b_2 \mid\}
\end{aligned}
$$

Diese vier Ähnlichkeitsmaße wurden implementiert und für den Einsatz im Blockmatching-Algorithmus untersucht. Leider konnte keines der angegebenen Ähnlichkeitsmaße überzeugende Ergebnisse liefern. Durch das Arbeiten mit normierten Farbanteilen wird die Helligkeitsinformation des Bildes nicht berücksichtigt. Es zeigt sich aber, daß vor allem die Helligkeitsinformationen für die lokale Korrespondenzanalyse besonders wichtig ist. Dies wurde auch in den nachfolgenden Untersuchungen im HSV-Farbraum (siehe Kapitel 3.4) festgestellt.

Es stellt sich heraus, daß die hier untersuchte RGB-Korrespondenzanalyse besonders empfindlich auf Shading-Störungen reagiert. Da Shading prinzipiell in den drei Farbanteilen unterschiedlich vorkommen kann, ergibt sich dadurch eine große Vielfalt an möglichen Shading-Störungen. Desweiteren sind die in [1] vorgestellten Ähnlichkeitsmaße nur für sehr farbintensive Bereiche im Bild geeignet. In grauen Bildregionen versagen diese völlig.

3.4 Untersuchungen HSV-Farbraum

Der HSV-Farbraum bietet eine wesentlich bessere Interpretierbarkeit der verwendeten Komponenten. Die H (hue) Komponente ist der Buntton und charakterisiert die dominante Farbe. Dieser ist als Winkel auf dem Farbkreis definiert. Der Winkel 0° entspricht der Farbe Rot. Die S (saturation) Komponente ist ein Maß für die Sättigung der Farbe und liegt im Intervall $[0, 1]$.

Eine Sättigung von 1.0 kennzeichnet eine Farbe mit maximaler Reinheit. Die V (value) Komponente charakterisiert die Helligkeit einer Farbe.

Der HSV-Farbraum wird in der Literatur auch als 'HSI' (hue, saturation, intensity) oder 'FSI' (Farbwert, Sättigung, Intensität) Farbraum bezeichnet.

Um mögliche Kriterien zur Korrespondenzanalyse im HSV-Farbraum zu finden, wurden die untersuchten Bilder, die im RGB-Farbraum vorlagen, zunächst in den HSV-Farbraum transformiert. Die Transformation wird in verschiedenen Büchern unterschiedlich angegeben. Hier wurde der Algorithmus verwendet, der in [6] angegeben wurde. Anschließend wurden je 2 Komponenten des HSV-Bildes ausgeblendet, um die verbliebene Komponente zu visualisieren. Um die Bunttonkomponente (hue) darzustellen, wurde die Sättigung aller Pixel auf 1.0 (Maximalwert) und die Helligkeit auf 0.5 gesetzt. Zur Darstellung der Sättigungskomponente (saturation) wurde der Buntton aller Pixel auf 0.0 (rot) und die Helligkeit auf 0.5 gesetzt. Die Helligkeitskomponente wurde dargestellt, indem Sättigung und Buntton aller Pixel auf 0.0 gesetzt wurden. Anschließend wurde das HSV-Bild zur Visualisierung wieder in den RGB-Farbraum transformiert. Die Ergebnisse sind in den Abbildungen 14 bis 17 für ein Stereobildpaar dargestellt. Das Stereobildpaar ist Shading gestört. Der Einfluß dieser Störung auf die einzelnen Komponenten des HSV-Farbraumes ist durch die Gegenüberstellung der linken und rechten Aufnahme gut erkennbar. Besonders auffällig ist der Einfluß des (Farb-) Shadings auf den Buntton (hue).

Aus den Experimenten läßt sich vermuten, daß die Sättigung einer Farbe besonders abhängig von der Betrachterrichtung ist. Somit stellt die Sättigung einer Farbe kein gutes Merkmal dar, um korrespondierende Objektpunkte zu finden. Offensichtich hat der spiegelnde Reflektionsanteil des Lichtes besonderen Einfluß auf die Sättigung der empfundenen Farbe. Als besonders aussagekräftig erweist sich, wie zu erwarten, vor allem die Helligkeit. Aufgrund dieser Untersuchungen wurde im Rahmen dieser Arbeit darauf verzichtet, die Farbe als Merkmal direkt in das Ähnlichkeitsmaß von Bildblöcken einfließen s

Abbildung 14: Shading gestörtes Stereobildpaar

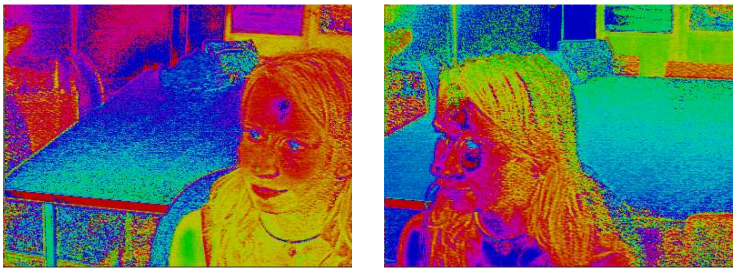


Abbildung 15: Die H (hue) Komponente des Stereobildpaares

Abbildung 16: Die S (saturation) Komponente des Stereobildpaares

Abbildung 17: Die V (value) Komponente des Stereobildpaares.

zu lassen.

3.5 Ausblenden unmöglicher Suchbereiche

Trotz der ernüchternden Untersuchungen der Farbe als Merkmal zur Korrespondenzanalyse läßt sie sich dennoch sinnvoll verwenden, um den Suchbereich einzuschränken. Von zwei korrespondierenden Punkten wird zusätzlich gefordert, daß sich ihre Bunttöne nicht weiter als einen bestimmten Winkel unterscheiden. Diese Einschränkung kann natürlich nur gemacht werden, wenn es sich bei den Farben um 'bunte' Farben handelt. Eine Farbe ist dann 'bunt', wenn ihre Sättigungs-Komponente einen bestimmten Mindestwert überschreitet.

Dies ist bislang nur eine Idee zur Nutzung der Farbinformationen. Es wurden hierzu im Rahmen der Arbeit keine Untersuchungen angestellt.

3.6 Lokale Minima

In flächigen Bildbereichen kann keine zuverlässige Bestimmung der Tiefe eines Objektpunktes erfolgen. Aber selbst, wenn man sich bei der Tiefensuche auf 'zuverlässige' Punkte beschränkt, führen Flächen im Bild zu Problemen. Oft treten Flächen schattiert auf. Ähnlichkeitsmaße, wie der quadratische Fehler, sind sehr abhängig von der durchschnittlichen Helligkeit der Pixelblöcke. Für einen gegebenen Pixelblock im rechten Bild findet sich in der entsprechenden linken Bildzeile leicht ein Pixelblock entlang einer Schattierung einer Fläche, der vom Ähnlichkeitsmaß als 'ähnlicher' empfunden wird, als der tatsächlich korrespondierende Punkt.

Allerdings haben solche falschen 'Unähnlichkeitsminima' charakteristische Eigenschaften. Um diese zu finden, ist es nützlich, die 'Unähnlichkeitskurve' entlang einer Bildzeile zu betrachten. Bei einem gegebenen (Un-)Ähnlichkeitsmaß *dissim* wird diese Unähnlichkeitskurve wie folgt definiert:

$$dissimcurve_{E_l,E_r,y,x_r}(x_l) = dissim(E_l, E_r, x_l, x_r, y)$$

Beim Betrachten solcher Kurven, bei denen falsche 'Unähnlichkeitsminima' auftreten, fällt auf, daß die gefundenen globalen Minima in sehr großen Tälern, oft auch am Rand des Suchbereiches liegen. Dagegen liegt der eigentlich richtige Wert in einem charakteristischen lokalen Minimum.

3.7 FFT Block Matching

Alle bisher vorgestellten Verfahren basieren auf der Analyse von Pixelintensitäten. Damit sind typische Probleme bei rausch- oder shading-gestörten Bildern verbunden, die schon im Kapitel 3.1 angesprochen wurden. Leistungsfähigere Abstandsmaße können auf texturbeschreibenden Merkmalen basieren. Das Fourierspektrum eines Pixelblockes kann als texturbeschreibendes Merkmal aufgefaßt werden. Auf eine genaue Untersuchung und Darstellung eines solchen Verfahrens wird hier im Bezug zur Aufgabenstellung der Belegarbeit verzichtet. Dennoch wurde ein einfaches, auf dem Fourierspektrum basierendes, Verfahren implementiert und liefert stabilere Ergebnisse als die intensitätsbasierten Verfahren. Dabei werden die zwei zu vergleichenden Pixelblöcke (z.B. 8x8 Pixel) in den Fourierraum transformiert und dessen Powerspektren berechnet. Die Unähnlichkeit der Blöcke ist der absolute auftretende Fehler zwischen den Powerspektren.

3.8 Bewertung

Als besonders wichtig hat sich bei den durchgeführten Versuchen die richtige Wahl des Disparitätslimits (maximal mögliche Disparität) herausgestellt. Bei einer zu großzügigen Wahl dieses Parameters kommt es bei allen hier untersuchten Verfahren häufig zu Fehlentscheidungen, insbesondere, wenn die Szenen horizontal ähnlich wiederholte Elemente aufweisen.

Die Wahl der Blockgröße hat ebenfalls bei allen Verfahren maßgeblichen Einfluß auf deren Güte. Gute Resultate wurden mit 7x7 Blöcken erzielt. Allerdings muß hierbei die Relation zur Gesamtgröße des Bildes beachtet werden. Sehr kleine Blöcke führen zu sehr verrauschten 'Unähnlichkeitskurven' mit vielen lokalen Minima und (leider) oft falschem globalen Minimum. Im Extremfall der Blockgröße 1x1 geht nur noch die Intensität einzelner Pixel in das Abstandsmaß ein. Sehr groß gewählte Blockgrößen führen zu einer ungenügenden Berücksichtigung kleiner Bildstrukturen.

Rein intensitätsbasierte Verfahren liefern selbst für markante Bildpunkte natürlicher Bilder unstabile Ergebnisse. Generell verbessert eine Tiefpaßfilterung natürlicher Ausgangsbilder die Stabilität der Ergebnisse. Das Fourier Blockmatching ist weniger anfällig für Sörungen der Ausgangsbilder. Insbesondere ist hier das Ähnlichkeitsmaß invariant gegenüber kleinen vertikalen Verschiebungen der Ausgangsbilder.

Die Untersuchungen der lokalen Minima deuten an, daß sowohl bei den intensitätsbasierten als auch texturbasierten Verfahren lokale Minima an den richtigen Stellen auftreten. Leider besteht die Frage für einen Algorithmus, welches lokale Minimum das Richtige ist.

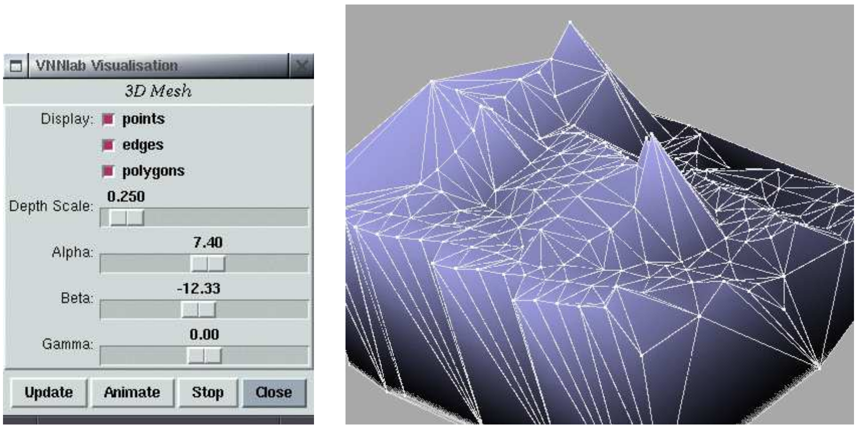


Abbildung 18: 3D Darstellung eines Polygonnetzes

4 Visualisierung

Als Visualisierungmöglichkeiten wurden exemplarisch drei Module implementiert. Weiter Visualisierungsmöglichkeiten können dank dem modularen Aufbau des Programmsystems leicht hinzugefügt werden.

4.1 Polygonnetz

Das durch die Triangulation entstandene Polygonnetz kann direkt sichtbar gemacht werden. Es können wahlweise die Netzknoten, Kanten und Polygone angezeigt werden. Die Tiefeninformation der Netzknoten wird als Helligkeit auf den Polygonen dargestellt, wobei innerhalb der Polygone linear zwischen den Netzknoten interpoliert wird.

4.2 3D Polygonnetz

Diese Darstellungsvariante ermöglicht das dreidimensionale Betrachten des Polygonnetzes. Die drei Betrachtungswinkel (Rotation um X-, Y- und Z-Achse) können interaktiv eingestellt werden. Weiterhin kann die Tiefeninformation skaliert werden. Der Skalierungsfaktor kann als die Konstante $f * b$

(siehe Kapitel 3) aufgefaßt werden und spiegelt die Kamerakonfigurationsparameter wieder. Auch hierbei können wahlweise die Netzknoten, Kanten und Polygone sichtbar gemacht werden.

4.3 Texture Mapping

Ein sehr realistischer Eindruck wird durch Texture Mapping mit dem rechten Ausgangsbild als Textur erzielt. Basierend auf dem 3D Polygonnetz wird hier jedem Netzknoten seine entsprechende Texturkoordinate im Ausgangsbild zugeordnet und somit das Ausgangsbild auf das Polygonnetz abgebildet. In Abbildung 19 ist diese Art der Visualisierung für das Stereobildpaar 'Nanraku' dargestellt. Im entwickelten Programmsystem lassen sich die Betrachtungswinkel interaktiv wählen und beeinflussen die Darstellung in Echtzeit. Hier kommt der Vorteil der Polygonnetze zur Darstellung voll zur Geltung. Ein weiteres Beispiel für die Darstellung mit Texture Mapping befindet sich auf der Titelseite.

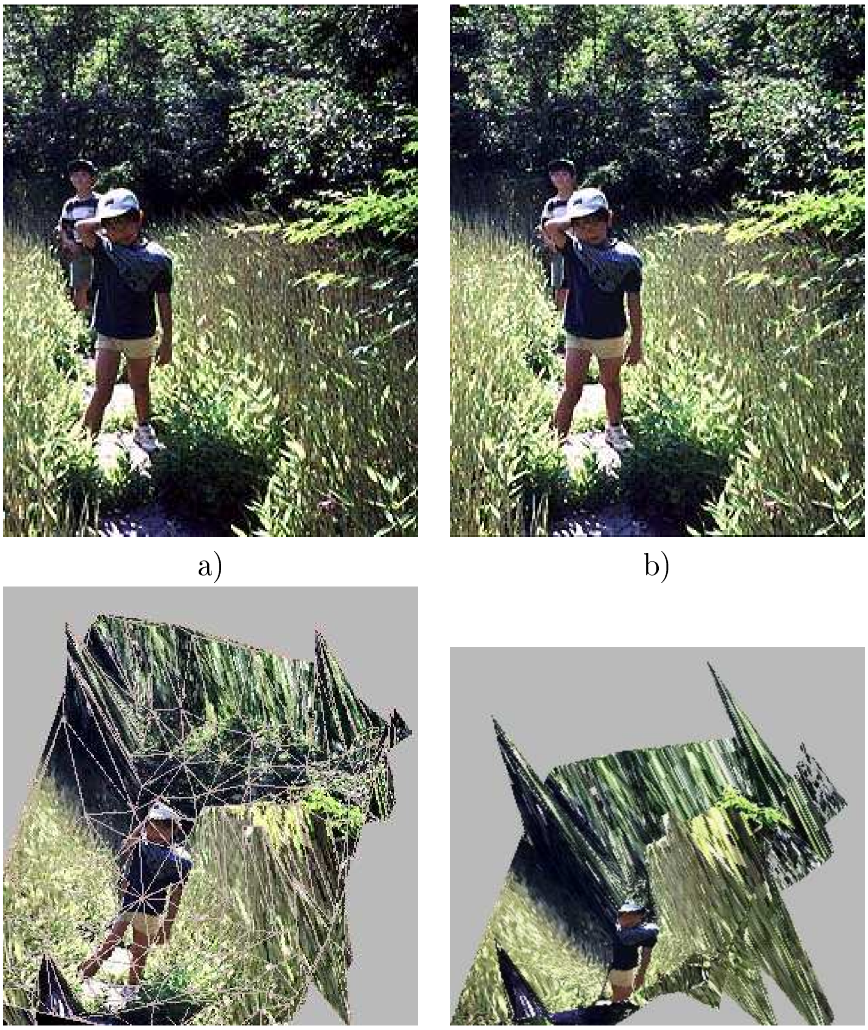


Abbildung 19: Beispiel für die Visualisierung durch Texture Mapping: Für das rechte Bild b) des Stereobildpaars 'Nanraku' (Teilabbildungen a und b) wurde mit 'Simulated Annealing' ein Polygonnetz erzeugt. Die Tiefeninformationen der Punkte wurde mit dem Fourier Matching Verfahren ermittelt. Die Teilabbildungen c) und d) zeigen das Ergebnis mit Bild b) als Textur mit und ohne Darstellung der Delaunay Kanten. Die Benutzung des Stereobildpaares 'Nanraku' erfolgte mit freundlicher Genehmigung von Hidehisa Kohno.

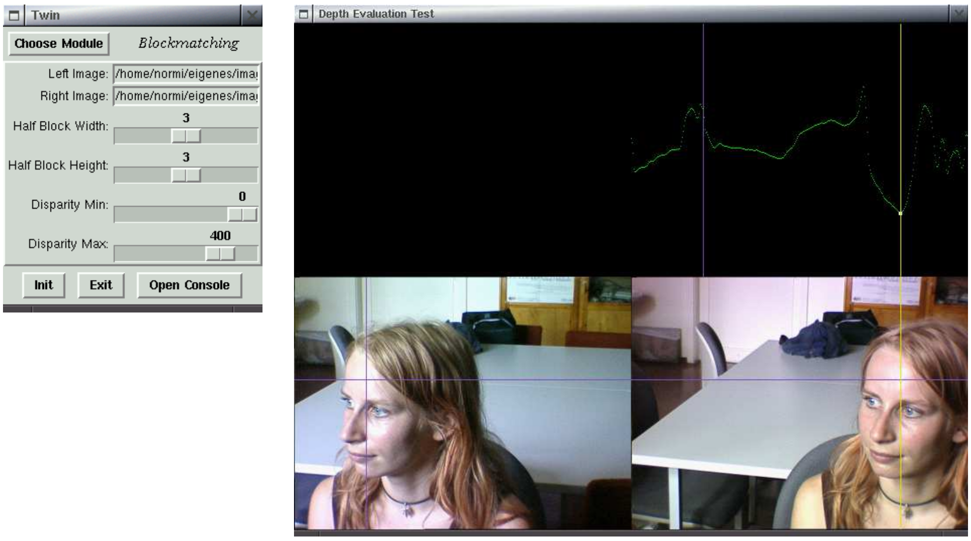


Abbildung 20: Programm zur Untersuchung von Algorithmen zur Korrespondenzanalyse

5 Implementierung

Es entstanden zwei Programme, mit denen die hier vorgestellten Verfahren untersucht wurden. Das erste Programm 'Twin' wurde speziell für die Untersuchungen der Verfahren zur Korrespondenzanalyse entwickelt. Dabei wurden die verschiedenen Algorithmen der Korrespondenzanalyse in einzelne Programm-Module umgesetzt. Diese Module sind gleichzeitig auch vom zweiten Programm 'VNNlab' benutzbar, welches die Korrespondenz-Analyse mit der Segmentierung von Bildern durch vektorbasierte neuronale Netze verbindet. 'VNNlab' ist der eigentliche Gegenstand des großen Belegs. Es bietet eine Plattform zum Testen und Experimentieren mit den verschiedenen Verfahren. Es stehen vielfältige Visualisierungsmöglichkeiten zur Verfügung, um die Wirkungsweise der vektorbasierten neuronalen Netze zu erkunden.

5.1 Lernbeispiel

Im Hinblick auf die Aufgabenstellung wurde bei der Entwicklung des Lernbeispiels vor allem auf eine übersichtliche und transparent zu bedienende Oberfläche Wert gelegt. Dabei sollen dem Anwender Grundprinzipien der vektorbasierten neuronalen Netze und der Stereoanalyse experimentell nahe gebracht werden. Weiterhin bietet das Programm durch seine Struktur eine Plattform für Experimente mit eigenen Algorithmen.

5.2 OpenGL

Für sämtliche grafische Ausgaben des Programms wurde die plattformübergreifende Standardbibliothek "OpenGL" verwendet. Diese Bibliothek wurde ursprünglich von Silicon Graphics entwickelt, existiert heute aber nahezu auf allen gebräuchlichen Plattformen. Hauptsächlich bietet sie eine einheitliche Schnittstelle zur Programmierung von 3D-Grafik. Dabei wird, wenn möglich, Gebrauch von vorhandener 3D-Hardware gemacht. OpenGL bietet neben 2D- und 3D-Primitiven auch vielfältige Möglichkeiten zum Darstellen von Bildern.

Die zusätzliche Bibliothek 'glut' ermöglicht sogar die Vereinheitlichung von OpenGL - Initialisierungscode. So kümmert sie sich unter Unix um die Kommunikation zum X-Server. Der Einsatz der OpenGL- und glut-Bibliotheken macht Programme gut portierbar und ermöglicht zusätzlich die Ausnutzung spezifischer 3D- und 2D-Hardware.

5.3 Tcl/Tk

Tcl/Tk ist eine sehr verbreitete Scriptsprache zur Programmierung von grafischen Benutzeroberflächen. Dabei bietet Tcl/Tk den großen Vorteil, auf vielen Plattformen verfügbar zu sein. Ein Tcl/Tk Programm wird auf den verschiedenen Plattformen so dargestellt, wie es auf der jeweiligen Plattform üblich ist. Der Programmierer liefert im wesentlichen nur eine Beschreibung der Oberfläche in Form eines Scripts. Eine komplette Tcl/Tk-Shell läßt sich als C-Bibliothek in jedes beliebige Programm einbinden. Die Shell kann leicht um neue, in C geschriebene, Befehle erweitert werden. Es existieren sehr einfache Wege der Kommunikation zwischen dem Tcl/Tk Script und dem C-Programm.

Die gesamte Benutzeroberfläche des Programms wurde mit Tcl/Tk erstellt. Dabei wurde die Shell durch neue C-Funktionen ergänzt. Eine genaue Beschreibung der C-Funktionen findet man im Anhang.

5.4 Posix-Threads

Mitunter kommt es im Programm zu länger andauernden Berechnungen. Diese Berechnungen würden dazu führen, daß in der Zeit der Berechnungen die Benutzeroberfläche 'einschläft', d.h. nicht mehr auf den Benutzer reagiert. Dies macht einen mehrpfädigen Programmablauf notwendig. Das Programm besteht somit nicht nur aus einem Prozeß, sondern aus einer Reihe parallel laufender Threads. Diese Threads sind folgende:

- Tcl/Tk-Hauptschleife: Diese nimmt Benutzeraktionen entgegen und kümmert sich um die Aktualisierung der Fenster der grafischen Benutzeroberfläche.

- Glut-Hauptschleife: Hier erfolgt die Verwaltung der OpenGL Ausgabefenster.

- Netzwerk-Berechnung: Der eventuell lang andauernde Adaptionsprozeß eines Netzes an eine Verteilung

- 3D-Berechnungen: Dieser Thread berechnet aus einem aktuellen Zustand des Netzwerkes eine 3D-Oberfläche. Dabei werden die Netzwerk-Knoten durch Dreiecke verbunden (Triangulation) und den Knoten Tieferinformationen zugeordnet. Dieser Prozeß kann bei einer großen Anzahl von Netzwerk-Knoten auch länger andauern. Deshalb ist es sinnvoll, ihn asynchron zum Netzwerk-Adaptions-Prozeß zu betreiben.

5.5 Shared Objects

Das Programm ist durchweg modular aufgebaut. Sämtliche Netzwerk-Modelle, Darstellungsvarianten, Verteilungsgeneratoren und Tiefenschätzungs-Algorithmen sind separate Module, die in Laufzeit hinzugeladen werden (dynamic linking). Die einzelnen Komponenten des Programmes sind somit ständig frei wählbar. Dies macht ein Experimentieren mit verschiedenen Algorithmen sehr leicht und anschaulich möglich. Weiterhin ist das Programm durch seinen modularen Aufbau sehr einfach durch neue Algorithmen erweiterbar. Bei der Entwicklung der Modulschnittstellen wurde darauf geachtet, daß sich der Modulprogrammierer lediglich um sein Problem (Algorithmus) und nicht um Verwaltungsaufwand kümmern muß. Jedes Modul kann auch ein Tcl/Tk-Script zur Beschreibung eines Einstellungsdialoges beinhalten. Um die Synchronisation der Tcl/Tk- und C-Variablen braucht sich der Modulprogrammierer nicht zu kümmern. Er gibt lediglich die Namen der gemeinsam genutzten (Tcl/Tk und C) Variablen an.

6 Zusammenfassung und Ausblick

In dieser Arbeit wurde die Segmentierung von Bildern durch die Nutzung von vektorbasierten neuronalen Netzen und die Tiefenschätzung durch verschiedene lokale Korrespondenzanalyseverfahren untersucht. Dabei entstand ein Verfahren zur Gewinnung von Polygonmodellen natürlicher Objekte durch die Nutzung von Stereobildpaaren. Das Verfahren wurde in Form eines Lernbeispiels implementiert. Das Lernbeispiel umfaßt verschiedene Algorithmen zur Bildsegmentierung in Polygonnetze, verschiedene Tiefenschätzungsalgorithmen und Visualisierungsmöglichkeiten.

Die in dieser Arbeit untersuchten Netzwerke haben die Gemeinsamkeit, daß deren Größe als Parameter vorgegeben wird. Günstige Parameter für einzelne Bilder lassen sich leicht (z.B. durch Probieren) finden. Dennoch wären Netze ohne vorgegebene Größe von Vorteil (z.B. für einen vollautomatischen Einsatz der Algorithmen). Hierzu ließen sich inkrementelle Netzwerke einsetzen. Diese hätten auch den Vorteil, Rücksicht auf lokale Unterschiede der Bilddetails zu nehmen. So würden diese Netze vorzugsweise in Regionen 'wachsen', in denen viele 'interessante' Punkte vorkommen. Interessant für den Problembereich der Bildsegmentierung wären auch Netze mit Netztopologien (Kanten zwischen den Netzknoten und die daraus resultierenden Nachbarschaftsbeziehungen), die eventuell gleich als Grundlage für die Triangulation dienen könnten. Denkbar wären auch kantenbasierte Netze. D.h. anstelle von Referenzvektoren werden 'Referenzkanten' adaptiert. Dieser Ansatz könnte in Verbindung mit einem entsprechenen Lernalgorithmus gute Segmentierungsergebnisse kantenreicher Szenen (z.B. Gebäude) erzielen.

Zur Tiefenschätzung wurden in dieser Arbeit nur Verfahren zur lokalen Korrespondenzanalyse eingesetzt. Diese lassen sich vielfältig verbessern. Denkbar wäre ein hybrides Verfahren, welches die Vorzüge verschiedener Ähnlichkeitsmaße kombiniert. Durch die Nutzung des Fourier Blockmatching und deren Phaseninformationen wäre es eventuell möglich, subpixelgenaue Ergebnisse zu erzielen.

Die Analyse der lokalen Minima der 'Unähnlichkeitskurven' erscheint ebenfalls vielversprechend. Eine Idee hierzu ist, daß zunächst eine Liste von möglichen Kandidaten im linken Bild gewonnen wird, die zu einem Ausgangspunkt im rechten Bild passen könnten. Für einen Kandidat gilt, daß die 'Unähnlichkeitskurve' an seiner Position ein lokales Minimum besitzt. Für den richtigen Kandidaten muß zudem gelten, daß deren 'Unähnlichkeitskurve' im rechten Bild ein lokales Minimum beim Ausgangspunkt aufweist. D.h. zwei Punkte P1 und P2 sind sich nur dann ähnlich, wenn P1 P2 ähnelt und P2 P1 ähnelt.

Anmerkungen

Vielen Dank an die Models Christin und Nils für ihre bereitwillige Unterstützung, sowie Hidehisa Kohno für die Genehmigung der Nutzung seiner Stereobilder (http://www.uwajima-mh.go.jp/~kohno/photoE.htm).

Literatur

[1] Reinhard Klette, Andreas Koschan, Karsten Schlüns: "Computer Vision" Vieweg Technik

[2] Bernd Fritzke : "Vektorbasierte Neuronale Netze" Shaker Verlag

[3] Brent B. Welch : "Practical Programming in Tcl and Tk" Prentice Hall

[4] Alan Watt, Mark Watt : "Advanced Rendering and Animation Techniques" Addison Wesley

[5] "The OpenGL Programming Guide" ISBN 0201604582

[6] Foley/Dam/Feiner/Hughes/Phillips: "Grundlagen der Computergrafik" Addison Wesley

[7] B.Radig : "Verarbeiten und Verstehen von Bildern" R.Oldenburg Verlag, ISBN 3-486-20787-3

Abbildungsverzeichnis

1 Darstellung des Datenflusses im Hauptfenster des Programmsystems . . . 7
2 Parameter für die Generierung einer Verteilung nach der Sobel-Filter Methode . . . 10
3 Ausgangsbild zur Generierung einer diskreten Verteilung . . . 12
4 Ergebnis der Anwendung des Sobel Operators . . . 12
5 Aus dem Sobel gefilterten Bild mit Hilfe eines Schwellwertes gewonnene diskrete Verteilung . . . 12
6 Beispiel für das Bilden einer (fast) Delaunay Triangulation b) der Punktemenge aus Teilbild a) nach dem vorgestellten Algorithmus . . . 13
7 Die Parameter des 'learning vector quantisation' Algorithmus. 17
8 Beispiel für die Anwendung der 'learning vector quantisation': Aus dem Ausgangsbild a) wurde mit der Sobel-Filter Methode eine Verteilung b) generiert. Ein Netz der Größe 13x16 ist in c) vor und in d) nach der Anwendung des 'learning vector quantisation' Algorithmus dargestellt. . . . 18
9 Parameter von 'k-means' . . . 19
10 Entwicklung eines Netzes unter der Lernmethode 'k-means' . . 21
11 Parameter für das 'neuronale Gas' . . . 23
12 zeitliche Entwicklung des mittleren Quantisierungsfehlers beim Lernen eines Netzes mit den Algorithmen: a) 'Learning Vector Quantisation', b) 'k-means' und c) 'Simulated Annealing' . . . 25
13 Standardstereogeometrie unter der Verwendung des Lochkameramodells . . . 27
14 Shading gestörtes Stereobildpaar . . . 33
15 Die H (hue) Komponente des Stereobildpaares . . . 33
16 Die S (saturation) Komponente des Stereobildpaares . . . 33
17 Die V (value) Komponente des Stereobildpaares. . . . 33

18 3D Darstellung eines Polygonnetzes 37
19 Beispiel für die Visualisierung durch Texture Mapping: Für das rechte Bild b) des Stereobildpaars 'Nanraku' (Teilabbildungen a und b) wurde mit 'Simulated Annealing' ein Polygonnetz erzeugt. Die Tiefeninformationen der Punkte wurde mit dem Fourier Matching Verfahren ermittelt. Die Teilabbildungen c) und d) zeigen das Ergebnis mit Bild b) als Textur mit und ohne Darstellung der Delaunay Kanten. Die Benutzung des Stereobildpaares 'Nanraku' erfolgte mit freundlicher Genehmigung von Hidehisa Kohno. 39
20 Programm zur Untersuchung von Algorithmen zur Korrespondenzanalyse 40