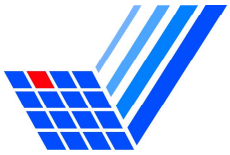
UNIVERSITÄT DORTMUND
FACHBEREICH INFORMATIK



Diplomarbeit

Dynamisches Lernen von Nachbarschaften zwischen Merkmalsgruppen zum Zwecke der Objekterkennung

Jochen Kerdels

August 2006

Gutachter:
Prof. Dr. Heinrich Müller
Dr. Gabriele Peters

Lehrstuhl Informatik 7
Graphische Systeme
der Universität Dortmund

Danksagung

An dieser Stelle möchte ich all jenen danken, die durch ihre fachliche und persönliche Unterstützung zum Gelingen dieser Diplomarbeit beigetragen haben. Mein Dank gilt Mathias Hülsbusch für seinen Rat in mathematischen Fragen und Thomas Kindler für seine Unterstützung bei der Gestaltung der Arbeit in TeX.

Desweiteren danke ich meinen Eltern, die mich immer unterstützt und mein Studium erst ermöglicht haben.

Besonderer Dank gebührt Prof. Dr. Heinrich Müller und Dr. Gabriele Peters für ihre engagierte und aufmerksame Betreuung meiner Diplomarbeit.

Inhaltsverzeichnis

1 Einleitung

Die Objekterkennung in Bildern stellt für einen Menschen in der Regel keine sonderlich schwierige Aufgabe dar. Mit Leichtigkeit erkennt der Mensch Objekte auch unter schlechten Sichtbedingungen, in großer Entfernung oder auch dann, wenn große Teile des Objektes verdeckt sind. Im starken Gegensatz zum Menschen gehört für einen Computer die Objekterkennung in Bildern zu den schwierigsten Aufgabenstellungen. Wenn in diesem Zusammenhang von künstlicher Intelligenz gesprochen wird, so geht es also darum, eine für den Menschen selbstverständliche Fähigkeit auf dem Computer umzusetzen. Ein Verfahren, das diese Aufgabe im Allgemeinen löst, also beliebige Objekte in beliebigen Umgebungen zu erkennen, existiert derzeit nicht.

Die bekannten Verfahren behelfen sich damit, entweder die Anzahl der Objekte und/oder die Komplexität der Umgebung zu reduzieren. Ein Beispiel dafür ist der Robocup [1]. Dies ist eine Veranstaltung, deren Ziel es ist, Robotern das Fußballspielen beizubringen. Dabei gilt das Fußballspiel als neuer Leistungstest für künstliche Intelligenz, so wie es bislang das Schachspiel war. Damit sich die Roboter auf dem Spielfeld zurecht finden, sind alle Objekte des Spiels, wie etwa der Ball und die Tore, farbig kodiert. Zudem gibt es Landmarken am Rande des Spielfeldes, die es den Robotern ermöglichen, sich auf dem Spielfeld zu lokalisieren. Desweiteren ist für eine gleichmäßige und konstante Beleuchtung des Spielfeldes gesorgt. Nur durch diese starke Einschränkung der Umgebung und der Anzahl der zu erkennenden Objekte ist es den Robotern möglich, ein ansehnliches Fußballspiel zustande zu bringen. Auf ähnliche Weise arbeiten industrielle Anlagen, die zum Beispiel in der Materialprüfung oder Produktionsüberwachung eingesetzt werden. Auf diese Weise können einfache und robuste Verfahren eingesetzt werden, die den industriellen Anforderungen von Geschwindigkeit und Präzision gerecht werden.

Sollen Objekte in natürlichen Bildern erkannt werden, so bestehen die Einschränkungen meist darin, daß nur eine Klasse von Objekten erkannt wird und/oder sich die Kamera nicht bewegen darf. Sollen beispielsweise nur Gesichter in Bildern erkannt werden, so besteht eine einfache Methode darin, nach signifikanten Hell/Dunkel-Mustern im Bild zu suchen [1]. Im Allgemeinen arbeiten viele Verfahren zur Objekterkennung in natürlichen Bildern so, daß sie zunächst eine Datenbank mit Informationen über die zu erkennenden Objekte anlegen und dann jeweils diese Daten mit

[1] http://www.robocup.org/

den in einem Bild vorgefundenen Informationen vergleichen.

Unter allen Verfahren zur Objekterkennung in Bildern kann im wesentlichen zwischen farbbasierten (z.B. [2, 3, 4]), formbasierten (z.B. [5, 6]) und merkmalbasierten Verfahren (z.B. [7, 8, 9]) unterschieden werden.

Farbbasierte Verfahren verwenden häufig Farbhistogramme, die von einer oder mehreren Ansichten des jeweiligen Objektes generiert werden. Zudem werden zum Teil Informationen über die räumliche Verteilung der Farben eines Objektes gesammelt. Farbbasierte Verfahren können im Allgemeinen sehr effizient implementiert werden und funktionieren recht zuverlässig, wenn die Anzahl der zu unterscheidenen Objekte nicht allzu groß ist und die Farbe der Objekte in unterschiedlichen Bildern nur wenig variiert (z.B. durch den Weißabgleich der Kamera).

Formbasierte Verfahren verwenden eine Vielzahl unterschiedlicher Methoden, die Silhouetten von Objekten zu beschreiben – beispielsweise über Momente, Fouriertransformationen oder Eigenvektoren. Da im Allgemeinen die Formen der Objekte nicht direkt vorliegen, müssen diese erst in einem Vorverarbeitungsschritt, z.B. über eine Kantenfilterung, erzeugt werden. Bei natürlichen Bildern ist dies nur sehr bedingt möglich, da Strukturen auf der Oberfläche der Objekte oder das Rauschen im Bildsignal Störungen verursachen. So kann zum Beispiel die Form eines einfarbigen Fahrzeuges noch relativ leicht durch eine Kantenfilterung ermittelt werden. Ist das Fahrzeug jedoch mit einem Werbeschriftzug versehen, so erzeugt dieser Kanten, die eigentlich nichts mit der Form des Fahrzeuges zu tun haben.

Merkmalbasierte Verfahren basieren darauf, zunächst „interessante Stellen“ - Merkmale - in einem Bild zu finden und diese dann möglichst transformationsunabhängig zu beschreiben. Dadurch wird gewährleistet, daß sich die Merkmale eines Objektes, dessen Lage in verschiedenen Bildern variieren kann, nur gering verändern. Häufig beschränkt man sich dabei auf Helligkeits-, Skalierungs- und Rotationsinvarianz. Für das Auffinden der Merkmale werden Kantenfilter, Difference of Gaussian oder auch auf Entropie basierende Verfahren verwendet [10, 11, 9, 8]. Für die Beschreibung der Merkmale werden u.a. Richtungshistogramme, Steerable Filters, Gabor Wavelets oder auch, entsprechend normiert, direkt die lokalen Bilddaten eingesetzt [9, 12, 8]. Die daraus resultierenden, meist hochdimensionalen Merkmalvektoren können zum Teil mittels Dimensionsreduktionsverfahren (z.B. PCA oder LLE) in ihrer Größe beschränkt werden, ohne daß ihre diskriminierenden Eigenschaften zu sehr verloren gehen [13, 14]. Eine Untersuchung [15] der häufigsten Beschreibungs-

formen für Merkmale hat ergeben, daß bislang die SIFT-Richtungshistogramme [9] unter diesen die besten Ergebnisse liefern.

Die überwiegende Anzahl der bisherigen Methoden zur Objekterkennung in Bildern erfordert a priori Informationen über die Objekte, die später erkannt werden sollen. In einer eingeschränkten Umgebung ist dieses Vorgehen auch durchaus adäquat und erlaubt es, eine recht effiziente Objekterkennung durchzuführen. So sind z.B., wie oben erwähnt, beim Robocup alle Objekte der Umgebung farbig kodiert. Die Klassifizierung der Farben in den Kamerabildern erfolgt dabei über eine Farbtabelle, die zuvor manuell erstellt werden muß. Solange sich die Lichtverhältnisse der Umgebung nicht stark ändern, funktioniert diese Art der Objekterkennung sehr gut. Treten jedoch Veränderungen in den Lichtverhältnissen auf, so bricht die Objekterkennung fast vollständig zusammen.

Verfahren, wie beispielsweise in [9] beschrieben, die in natürlichen Bildern Objekte erkennen, benötigen für jedes zu erkennende Objekt ein oder mehrere Beispielbilder. Aus diesen Bildern werden zunächst Merkmale extrahiert und in einer Datenbank abgelegt. Für die Objekterkennung werden nun die im untersuchten Bild gefundenen Merkmale mit denen in der Datenbank verglichen. Findet sich eine ausreichend große Übereinstimmung zwischen den Merkmalen im Bild und den Merkmalen in der Datenbank, so wird das entsprechende Objekt im Bild erkannt. Dieses Vorgehen eignet sich nur, wenn die zu erkennenden Objekte im voraus bekannt sind und entsprechende Trainingsdaten zur Verfügung stehen. Desweiteren sollte die Zahl der zu erkennenden Objekte nicht allzu groß sein, da ansonsten der Datenbankabgleich sehr viel Zeit in Anspruch nimmt.

Ist die Art und Anzahl der zu erkennenden Objekte nicht im voraus bekannt, können die oben beschriebenen Verfahren nicht eingesetzt werden. Soll sich z.B. ein autonomer Roboter in einer unbekannten natürlichen Umgebung zurechtfinden, so sollte er in der Lage sein, die Objekte dieser Umgebung dynamisch zu lernen.

2 Ziel der Diplomarbeit

Die Diplomarbeit soll einen Lösungsansatz für die Objekterkennung in Bildern aufzeigen, der ohne a priori Information über die zu erkennenden Objekte auskommt. Durch die statistische Auswertung von Merkmalsgruppen und ihrer Nachbarschaftsbeziehungen soll das System im Laufe der Zeit erlernen, was „Objekt“ und was

„Hintergrund“ in einem Bild ist. Das System soll dabei wie folgt arbeiten: Das Eingabebild wird zunächst segmentiert. Die Segmentierung erfolgt dabei mittels eines Kantendetektors, der relativ unempfindlich gegenüber unruhigen Strukturen ist. Anschließend wird, ausgehend von den Kanten in Richtung der Flächenmitten, eine Art „Höhenbild“ generiert. Auf diesem Höhenbild wird daraufhin ein Watershed-Algorithmus angewandt, der die Segmentierung des Bildes erzeugt. Dieses Vorgehen garantiert, daß das Bild lückenlos segmentiert wird und es keine überlappenden Segmente gibt.

Für jedes Segment werden nun Merkmale innerhalb des Segmentes gesucht. Diese Merkmale werden mittels eines „wachsenden neuronalen Gases“ [16] quantisiert und in einer Merkmalmenge, die dem Segment zugeordnet ist, zusammengefaßt. Diese Menge und die relativen Winkel der Merkmale zueinander definieren das Segment auf eine rotations- und skalierungsunabhängige Weise. Somit gelten zwei Segmente mit gleichen Merkmalmengen und den gleichen relativen Winkeln zwischen den Merkmalen als identisch. Jedes Segment verwaltet zudem, in welchen direkten Nachbarschaftsbeziehungen zu anderen Segmenten das Segment bislang stand.

Die Grundidee ist nun folgende: Wenn das System hinreichend viele Bilder eines Objektes verarbeitet hat, dann werden die Segmente, die das Objekt darstellen, relativ häufig auftreten und die Nachbarschaft zwischen diesen Segmenten sollte relativ stabil sein im Gegensatz zur Umgebung des Objektes, die sich in gewisser Weise von Bild zu Bild verändert. Nach einiger Zeit sollte das System in der Lage sein, ausgehend von einem Segment abzuschätzen, ob benachbarte Segmente noch zu einem konkreten Objekt gehören oder nicht. Da im allgemeinen die Segmentierung der Objekte in einem gewissen Maße von Bild zu Bild verschieden ist, ist es wichtig, nicht nur einzelne Segmente zu betrachten, sondern auch Mengen von ähnlichen Segmenten. Ein brauchbares Maß für die Ähnlichkeit zweier Segmente kann dabei der Bioinformatik entnommen werden. Der Smith-Waterman Algorithmus [17] berechnet die Ähnlichkeit zweier DNA-Sequenzen und kann für den Vergleich der hier beschriebenen Segmente angepaßt werden.

3 Überblick

Im Folgenden soll ein kurzer Überblick über die nächsten Abschnitte gegeben werden. Abschnitt 4 beschreibt verschiedene Verfahren zur Segmentierung von Bildern.

Dabei werden zunächst Methoden vorgestellt, die für ein Bild auf direktem Wege eine Segmentierung erzeugen. Anschließend werden indirekte Verfahren beschrieben, die zunächst ein Kantenbild erzeugen und daraus die Segmentierung ableiten.
Abschnitt 5 behandelt das Auffinden und Beschreiben von Merkmalen innerhalb der Segmente. Es werden zunächst die drei klassischen Merkmaldetektoren von Moravec, Harris und Kanade/Lucas/Tomasi und ein neuerer Ansatz von Kadir/Brady beschrieben und in einem Test miteinander verglichen. Im Weiteren wird die Ermittlung der Hauptrichtung eines Merkmals durch den von Lowe in [9] beschriebenen Algorithmus erläutert und ein alternativer Algorithmus vorgestellt. Auch für die Beschreibung eines Merkmals wird zunächst die von Lowe im gleichen Paper vorgestellte Merkmalbeschreibung erläutert und auch hier eine alternative Merkmalbeschreibung aufgezeigt.
Abschnitt 6 erläutert, wie die einzelnen Segmente über Folgen von Merkmalvektoren und den relativen Winkeln zwischen den Merkmalen beschrieben werden können. Anschließend wird in Anlehnung an Verfahren für den DNA-Sequenzvergleich für diese Segmentbeschreibung ein Ähnlichkeitsmaß vorgestellt. Desweiteren wird in diesem Zusammenhang beschrieben, wie mittels eines wachsenden neuronalen Gases (siehe [16]) die Merkmale eines Segmentes quantisiert werden können.
Abschnitt 7 geht auf die Besonderheiten bei der Erfassung und Auswertung der statistischen Segmentdaten ein und Abschnitt 8 stellt die gewonnenen Ergebnisse vor, zeigt Erweiterungs- und Verbesserungsmöglichkeiten auf und liefert einen Ausblick auf mögliche Anwendungen des hier beschriebenen Verfahrens.

4 Segmentierung

Merkmalbasierte Verfahren zu Objekterkennung in Bildern, wie z.B. SIFT [9], generieren durchschnittlich etwa 1000 Merkmale pro Bild. Eine Suche in den Merkmalmengen der bisher verarbeiteten Bilder nach ähnlichen und häufig auftretenden Teilmengen würde einen enormen Rechenaufwand zur Folge haben. Die Segmentierung der Eingabebilder verringert diesen Rechenaufwand. Die Segmente definieren lokale Bereiche des Bildes und die in diesen Bereichen auftretenden Merkmale bilden jeweils eine Teilmenge der Merkmalmenge des Bildes. Unter der Annahme, daß ein Objekt in verschiedenen Bildern durch gleiche bzw. ähnliche Segmente repräsentiert wird, sind die Teilmengen der Merkmalmenge, die durch die Segmente induziert

werden, genau die Teilmengen, deren Häufigkeit von Interesse ist. Es müssen also nicht mehr alle möglichen Teilmengen der Merkmalmengen der Bilder und ihre Häufigkeit untersucht werden, sondern nur noch die durch die Segmente vorgegebenen Teilmengen. Da Objekte in aller Regel nicht nur durch ein Segment, sondern durch mehrere Segmente beschrieben werden, müssen auch die Nachbarschaftsbeziehungen der Segmente Berücksichtigung finden. Hierbei reicht es aus, sich auf die direkten Nachbarn eines Segmentes zu beschränken, wenn auch mitgespeichert wird, zu welchen Zeitpunkten eine bestimmte Nachbarschaftsbeziehung aufgetreten ist. Mit diesen Informationen kann für jedes bisher verarbeitete Bild ein vollständiger Nachbarschaftsgraph rekonstruiert werden.

Eine Segmentierung natürlicher Bilder, so daß die erzeugten Segmente der Wahrnehmung des Menschen entsprechen oder auch nur gleiche Objekte in immer gleicher Weise segmentiert werden, ist ein noch ungelöstes Problem der Bildverarbeitung. Jedoch existieren für eingeschränke Umgebungen, wie oben bereits erwähnt, eine Vielzahl an brauchbaren Segmentierungsverfahren. Im Folgenden soll ein kleiner Überblick über eine Auswahl dieser Verfahren gegeben werden.

4.1 Segmentierung durch Klassifizierung

Erlaubt die eingeschränkte Umgebung eine direkte Klassifizierung der Bildpunkte, z.B. über eine Farbtabelle, kann mit dieser Klassifizierung eine Segmentierung erzeugt werden. Das Bild P wird dazu zeilenweise abgearbeitet:

```
Überprüfe für jeden Punkt P[x,y] {
  Wenn P[x-1,y] zur gleichen Klasse wie P[x,y] gehört, dann {
    Füge P[x,y] dem Segment, zu dem auch P[x-1,y] gehört,
      hinzu;
    Wenn P[x,y-1] zur gleichen Klasse wie P[x,y] gehört, jedoch
      zu einem anderen Segment, dann {
        Vereinige die beiden Segmente;
      }
  } ansonsten {
    Wenn P[x,y-1] zur gleichen Klasse wie P[x,y] gehört, dann {
      Füge P[x,y] dem Segment, zu dem auch P[x,y-1] gehört,
        hinzu;
```

```
      } ansonsten {
        Erstelle ein neues Segment und füge ihm P[x,y] hinzu;
      }
    }
  }
```

Wie bereits oben erwähnt, wird die Segmentierung mittels Farbklassifizierung z.B. im Robocup eingesetzt. Die Erfahrung zeigt hierbei, daß auf der einen Seite diese Art der Segmentierung effizient zu berechnen ist und gute Ergebnisse liefert, auf der anderen Seite jedoch äußerst störanfällig ist. Schon relativ kleine Veränderungen der Umgebung, beispielsweise plötzlich auftretender Sonnenschein oder die zusätzlichen Strahler einer Fernsehkamera, verändern die wahrgenommenen Farben derart stark, daß sie nicht mehr korrekt durch die verwendete Farbtabelle klassifiziert werden können. Jedoch kann die Wahl eines geeigneten Farbraums [18, 4] dazu beitragen, die Störanfälligkeit zu reduzieren.

4.2 Bereichswachstumsverfahren

Bereichswachstumsverfahren erzeugen die Segmentierung eines Bildes über das simultane Vergrößern von einzelnen Bildbereichen ausgehend von zuvor gewählten Startpunkten. Ein einzelner Bildpunkt kann genau dann zu einem Bildbereich hinzugefügt werden, wenn ein bestimmtes Ähnlichkeitskriterium erfüllt ist. Häufig wird die Intensität oder die Zugehörigkeit zu einer Farbklasse als Kriterium verwendet. Desweiteren eignen sich auch solche Kriterien für das Bereichswachstum, die über ein lokales Fenster um den Bildpunkt herum erzeugt werden, wie z.B. Entropie, Varianz oder Durchschnitt der Intensitätswerte oder ein Histogramm über die Richtungen der Gradienten innerhalb des lokalen Fensters.

Für das Verfahren müssen zunächst geeignete Startpunkte für das Bereichswachstum im Bild gefunden werden. Es empfiehlt sich, die Startpunkte so zu wählen, daß sie möglichst in gleichmäßigen Bildbereichen bzgl. des verwendeten Ähnlichkeitskriteriums liegen. In einem weiteren Vorverarbeitungsschritt werden die Startpunkte entfernt, die von einem anderen Startpunkt aus erreicht werden können, ohne das Ähnlichkeitskriterium zu verletzen. Während des „simultanen“ Bereichswachstums werden reihum zu den einzelnen Bildbereichen Bildpunkte hinzugefügt. Dabei ist die maximale Anzahl an Bildpunkten, die in einem Schritt hinzugefügt werden

können, durch eine Wachstumsschranke begrenzt. Können in einem Durchlauf zu keinem Bildbereich mehr Bildpunkte hinzugefügt werden, so wird das verwendete Ähnlichkeitskriterium gelockert und ein neuer Durchlauf gestartet. Das Verfahren terminiert, wenn jeder Bildpunkt einem Bildbereich zugeordnet ist. In einem Nachberarbeitungsschritt kann eine mögliche Übersegmentierung durch das Verschmelzen von Bildbereichen ausgeglichen werden.

4.3 Bereichsunterteilungsverfahren

Während das oben beschriebene Bereichswachstumsverfahren eine bottom-up Methode ist, ist das Bereichsunterteilungsverfahren die entsprechende top-down Methode. Ein Bildbereich wird solange in kleinere Bildbereiche unterteilt, bis ein bestimmtes Einheitlichkeitskriterium erfüllt ist. Anschließend werden benachbarte Bildbereiche, deren Vereinigung das Einheitlichkeitskriterium erfüllt, miteinander verschmolzen.

Bereichswachstums- und Bereichsunterteilungsverfahren segmentieren Bilder mit klaren Strukturen relativ gut. Bilder mit komplexen Strukturen, z.B. von Pflanzen, werden nur mäßig gut segmentiert, da es recht schwierig ist, die verwendeten Kriterien so auszuwählen und zu parametrisieren, daß sie einerseits gegenüber den komplexen Strukturen tolerant genug sind und keine Übersegmentierung erzeugen, andererseits aber so strikt sind, das keine Untersegmentierung entsteht. Das Hauptproblem liegt hierbei darin, daß die verwendeten Kriterien im Falle des Bereichswachstums häufig nur einen einzelnen Bildpunkt oder nur einen kleinen lokalen Bereich um den Bildpunkt herum betrachten und somit gröbere Strukturen nicht erfassen, bzw. im Falle der Bereichsunterteilung nicht zwischen groben Strukturen und eigenständigen Bildbereichen unterscheiden können. Entsprechend zeigt das Bereichswachstum eine Tendenz zur Übersegmentierung, die Bereichsunterteilung eine Tendenz zur Untersegmentierung. Die in den nächsten beiden Unterabschnitten vorgestellten Segmentierungsverfahren liefern Lösungsansätze, wie das Problem der Unterscheidung zwischen Textur und eigenständigem Bildbereich angegangen werden kann.

4.4 Segmentierung mittels Competitive Layer Model

In [19] beschreiben Ontrup und Ritter ein System, das die Art und Weise der menschlichen Wahrnehmung von Strukturen nachbildet und auf dieser Basis Bilder segmentiert. Die Ideen, die dem System zugrunde liegen, haben ihren Ursprung in der Ge-

staltpsychologie, die Anfang des 20. Jahrhunderts durch die Psychologen um Wertheimer, Koffka und Köhler geprägt und propagiert wurde. Unter Verweis auf die Thermodynamik nahm man an, das Wahrnehmungen derart vom Geist organisiert werden, daß ihre internen Repräsentationen energieminimal sind. Auf welche Weise demnach die Welt wahrgenommen wird, beschreibt das *Gesetz der Prägnanz*: Die wahrgenommene Welt wird derart vom Geist geordnet, daß die Simplizität maximiert wird. Größere Simplizität bezieht sich hierbei auf größere Gleichförmigkeit und wenige ausgeprägte Einheiten, den *Gestalten*. Um diesen Minimierungsprozess konkreter zu beschreiben, versuchten die Gestaltpsychologen ein Rahmenwerk zu formulieren. Verschiedene Gesetze dienen dazu, die meist auf Introspektion basierenden Phenomena zu beschreiben. Zu den wichtigsten Gesetzen zählen die folgenden:

- **Gesetz der Nähe**: Reize werden abhängig von ihrer Nähe zueinander zu übergeordneten Einheiten zusammengefaßt.
- **Gesetz der Ähnlichkeit**: Ähnliche Merkmale neigen dazu, gleichförmige Gruppen zu bilden. Bei den Merkmalen kann es sich um Eigenschaften wie etwa Helligkeit, Farbe, Form und Ausrichtung handeln.
- **Gesetz der Kontinuität**: Teile eines Umrisses erscheinen zusammengehörig, wenn diese Teile eine eindeutige Richtung aufweisen. Insbesondere gerade Elemente, die auf einer Linie liegen, fördern die Bildung einer übergeordneten Einheit.
- **Gesetz der Geschlossenheit**: Reize, die eine geschlosse Kontur ausbilden, werden zusammengefaßt.

Auch wenn diese Gesetze nicht präzise genug sind, um daraus direkt einen Algorithmus ableiten zu können, so liefern sie doch gute Anhaltspunkte für eine Analyse der menschliche Wahrnehmung. Das von Ontrup und Ritter eingesetze Competitive Layer Model (CLM, s. Abbildung 1), ein mehrschichtiges neuronales Netz, wurde erstmals von Ritter [20] zur räumlichen Gruppierung von Merkmalen vorgestellt. Als Eingabe bekommt das CLM eine Menge I von Merkmalvektoren $m_r \in M$, jeder gewichtet mit einer skalaren Intensität h_r

$$I = \{(m_r, h_r) \mid r \in \{1, \ldots, N\}\}$$

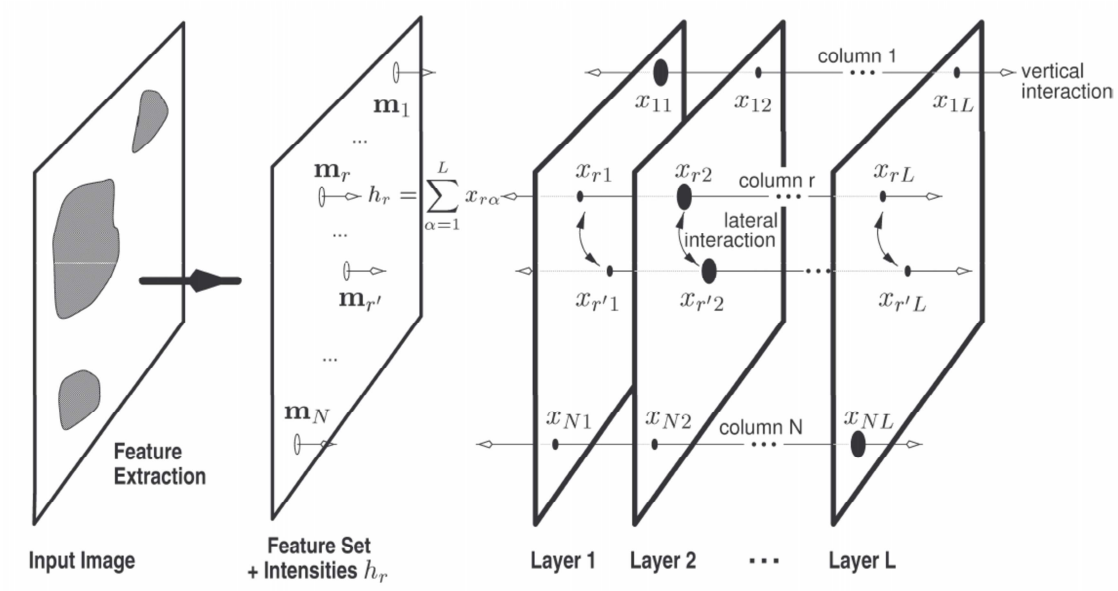


Abbildung 1: *Das von Ontrup und Ritter verwendete CLM. Quelle [19]*

Als Merkmale können z.B. Position, Farbe, Textur- oder Kanteninformationen dienen, oder auch Kombinationen solcher Eigenschaften. Die Idee der Gestaltpsychologen, daß Reize in übergeordneten *Gestalten* energieminimal zusammengefaßt werden, kann nun auf folgende Weise formuliert werden: Finde eine Partition von M in L disjunkte Teilmengen $M_\alpha, \alpha \in \{1, \ldots, L\}$, so daß die Summe der Energien der Teilmengen $\sum_\alpha E(M_\alpha)$ minimal wird. Jede Teilmenge entspricht dann einer *Gestalt*.

Das von Ontrup und Ritter für diese Aufgabe verwendete CLM (s. Abbildung 1) hat L Schichten, die jeweils N Neuronen mit nicht-negativer Aktivität $x_{r\alpha} \geq 0$ besitzen. Vertikal sind die einzelnen Schichten so verbunden, daß für jede Position r eine durchgehende Spalte alle Neuronen an dieser Position miteinander verbindet. Es gibt also N Spalten mit jeweils L Neuronen. An diesen Spalten wird nun die Eingabe I „angelegt".

Für die zu minimierende Energiefunktion müssen zwei Dinge berücksichtigt werden. Zum einen müssen die Aktivitäten der Neuronen einer Spalte in ihrer Summe der Intensität h_r der Eingabe entsprechen, zum anderen muß die benötigte Energie der einzelnen Schichten beschrieben werden. Dies geschieht durch eine Funktion $f(m_r, m_{r'})$, die paarweise die Merkmale des Eingabebildes vergleicht. Aktive Neuronen, die mit ähnlichen Merkmalen assoziiert sind, werden durch eine positive Kopplung aneinander gebunden. Neuronen, die mit unähnliche Merkmalen assoziiert

sind, werden durch eine negative Kopplung voneinander getrennt. Für jedes Eingabebild werden die Werte der Funktion $f(m_r, m_{r'}) = f_{rr'} = f_{r'r}$ berechnet und in einer $N \times N$ großen Tabelle gespeichert. Zusammengenommen erhält man folgende zu minimierende Energiefunktion für das CLM:

$$E = \frac{J_1}{2} \sum_{r=1}^{N} \left(\sum_{\beta=1}^{L} x_{r\beta} - h_r \right)^2 - \frac{1}{2} \sum_{\alpha=1}^{L} \sum_{(r,r') \in N \times N} f_{rr'} x_{r\alpha} x_{r'\alpha}$$

Der Parameter J_1 steuert das Verhältnis zwischen dem Energieerhaltungskriterium der Spalten und der Energie der einzelnen Schichten.

Es ist leider nicht klar, wie die Funktion f gewählt werden muß, um das gewünschte Gruppierungsverhalten zu erzeugen. Selbst dann, wenn eine passende Funktion gefunden ist, bleibt es, die Energiefunktion zu minimieren. Ontrup und Ritter fassen in [19] die Funktion f als eine Art Metrik auf den Merkmalvektoren auf und ermitteln diese mittels PCA. Für die Lösung des Minimierungsproblems greifen sie auf die „Heat Bath Update"-Methode zurück, welche ihre Ursprünge in der theoretischen Physik hat und dort für die Berechnung großer dynamischer Systeme eingesetzt wird.

4.5 Normalized Cuts

Shi und Malik beschreiben in [21] ein auf Graphentheorie basierendes Segmentierungsverfahren. Allgemein kann eine Menge von Punkten eines Merkmalraumes, z.B. die Pixel eines Bildes, duch einen gewichteten ungerichteten und vollständigen Graphen $G = (V, E)$ dargestellt werden. Dabei entsprechen die Knoten des Graphen den Punkten der Menge und das Gewicht jeder Kante $w(i, j)$ beschreibt die Ähnlichkeit zwischen den Knoten i und j. Eine Segmentierung entspricht also einer Partition der Knotenmenge V in disjunkte Mengen $V_1, V_2, \ldots, V_m$. Die Ähnlichkeit der Knoten innerhalb einer Menge sollte hierbei möglichst hoch sein und die Ähnlichkeit der Knoten zwischen den Mengen sollte möglichst gering sein.

Ein wohlbekanntes Problem der Graphentheorie ist der Min-Cut. Ein Cut teilt die Knotenmenge V in zwei Knotenmengen A und B auf und der Wert dieses Cuts ist die Summe der Kantengewichte, die zwischen den Mengen A und B verlaufen:

$$cut(A,B) = \sum_{u \in A, v \in B} w(u,v)$$

.

Der Min-Cut ist nun der Cut mit dem kleinsten Wert. Das Problem bei dieser Definition des Cuts liegt darin, daß eine der erzeugten Knotenmengen, A oder B, dazu tendiert, nur relativ wenige Knoten zu enthalten. Dies ist leicht einzusehen, da sich mit der Anzahl der Knoten einer Menge auch die Anzahl der Kanten zwischen den Mengen und somit auch der Wert des Cuts erhöht. Shi und Malik schlagen aus diesem Grunde eine erweiterte Definition des Cuts vor, den *normalized cut*:

$$Ncut(A,B) = \frac{cut(A,B)}{assoc(A,V)} + \frac{cut(A,B)}{assoc(B,V)}$$

mit

$$assoc(A,V) = \sum_{u \in A, t \in V} w(u,t)$$

Der Vorteil dieser Definition liegt darin, daß kleine Mengen keinen kleinen Ncut-Wert erzeugen, da bei einer kleinen Menge der Wert von cut i.d.R. fast so groß wie der Wert von $assoc$ ist. Auf gleiche Weise wie der NCut ein Maß für die Ähnlichkeit zwischen den Mengen A und B ist, kann ein Maß definiert werden, das die Ähnlichkeit der Knoten zueinander innerhalb der Mengen A und B beschreibt:

$$Nassoc(A,B) = \frac{assoc(A,A)}{assoc(A,V)} + \frac{assoc(B,B)}{assoc(B,V)}$$

NCut und Nassoc liefern demnach ein Maß für die beiden Kriterien, die die Mengen V_i möglichst gut erfüllen sollen. Formt man die Definition von Ncut auf geeignete Weise um, indem man ausnutzt, daß

$$\begin{aligned} cut(A,B) &= assoc(A,V) - assoc(A,A) \\ &= assoc(B,V) - assoc(B,B) \end{aligned}$$

ist, erhält man

$$\begin{aligned} Ncut(A,B) &= \frac{assoc(A,V) - assoc(A,A)}{assoc(A,V)} + \frac{assoc(B,V) - assoc(B,B)}{assoc(B,V)} \\ &= 2 - \left(\frac{assoc(A,A)}{assoc(A,V)} + \frac{assoc(B,B)}{assoc(B,V)} \right) \\ &= 2 - Nassoc(A,B)\,. \end{aligned}$$

Die beiden Kriterien für die Mengen V_i sind demnach identisch und können gleichzeitig erfüllt werden. Shi und Malik zeigen in [21] weiter, wie der Ncut effizent durch Lösen eines generalisierten Eigenwertproblems berechnet werden kann. Für die so erzeugten Mengen A und B wird anschließend geprüft, ob sie rekursiv weiter unterteilt werden oder nicht.

4.6 Kantenbasierte Segmentierung

Kantenbasierte Segmentierung erzeugt die Segmentierung eines Bildes nicht auf direktem Wege aus den Bilddaten. In einem ersten Schritt werden zunächst die „wichtigen" Kanten eines Bildes extrahiert und erst aus diesen Kanten wird in einem zweiten Schritt die Segmentierung des Bildes generiert. Da in der Regel diese beiden Schritte unabhängig voneinander sind, wird im Folgenden zunächst eine Auswahl an Verfahren für den ersten Schritt und anschließend eine Auswahl an Verfahren für den zweiten Schritt vorgestellt.

4.6.1 Kantenerzeugung

Ziel der Kantenerzeugung ist die möglichst vollständige Extraktion der Objektkonturen innerhalb eines Bildes. Die Konturen eines Objektes, so wie sie der Mensch wahrnimmt, werden auf unterschiedliche Weise hervorgerufen. Im einfachsten Fall setzt sich das Objekt vom Hintergund durch einen hohen Farbkontrast ab, der dazu führt, daß an der Grenze zwischen Objekt und Hintergrund der Gradient an dieser Stelle im Bild einen hohen Wert annimmt. Kanten dieser Art können leicht durch einfache Kantenoperatoren bestimmt werden. Komplizierter wird es, wenn

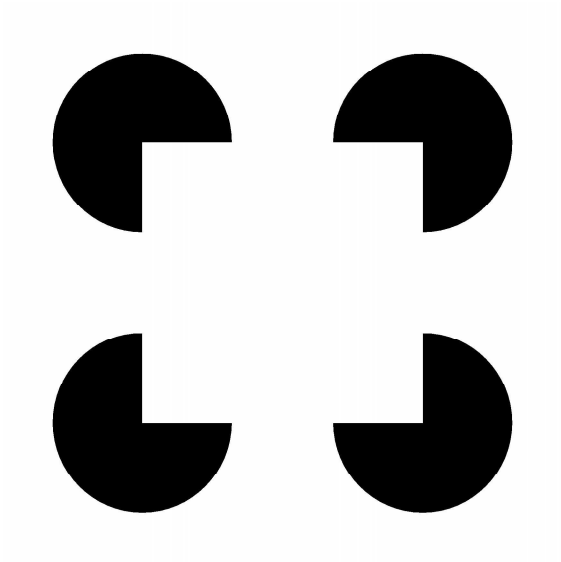


Abbildung 2: *Ergänzung „virtueller" Kanten: Man scheint ein weißes Quadrat auf vier schwarzen Kreisen zu erkennen und nicht nur 4 schwarze Kreissegmente.*

auch die Oberflächenstruktur des Objektes einen hohen Kontrast aufweist und somit auch innerhalb des Objektes Gradienten mit hohem Wert existieren. Die Grenze zwischen Objekt und Hintergrund wird in diesem Fall eher durch das abrupte Ausbleiben der Oberflächenstruktur des Objektes bestimmt. Für die Detektion dieser Art von Grenzen können einfache Kantenoperatoren erweitert oder auch neuronale Ansätze verfolgt werden. Eine weitere Art von Objektkonturen, die der Mensch mühelos wahrnimmt, scheint gänzlich unmöglich auf direktem Wege aus den Bilddaten erzeugt werden zu können. Der Mensch verbindet intuitiv sichtbare Kanten in einem Bild, die in einfachen und klaren geometrischen Beziehungen zueinander liegen. Die auf diese Weise entstehenden „virtuellen" Kanten vervollständigen so die Konturen des jeweiligen Objektes. Ein sehr bekanntes Beispiel für dieses Phänomen zeigt Abbildung 2. Offensichtlich können diese „virtuellen" Kanten nicht direkt aus den Bilddaten gewonnen werden. In einem Nachbearbeitungsschritt kann jedoch versucht werden, die fehlenden Kanten zu ergänzen und so die Objektkonturen zu vervollständigen.

Einfache Kantenoperatoren Einfache Kantenoperatoren ermitteln die Kanten innerhalb eines Bildes durch die Approximation der 1. oder 2. Ableitung mehrerer Richtungen an jeder Stelle des Bildes. Dazu liefert ein Kantenoperator eine Rechen-

vorschrift, wie die Pixel in einem lokalen Bereich um den zu berechnenden Zielpixel herum miteinander verrechnet werden müssen. Eine detailierte Beschreibung einfacher Kantenoperatoren liefert [22] (S. 577ff). Die 1. Ableitung nähert beispielsweise der Sobel-Operator an (hier für vertikale Kanten):

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

Ein Beispiel für einen Operator, der die 2. Ableitung annähert, ist der Laplace-Operator:

-1	-1	-1
-1	8	-1
-1	-1	-1

Normalerweise werden Operatoren, die die 2. Ableitung annähern, nicht direkt für die Kantendetektion eingesetzt, da sie sehr empfindlich gegenüber Rauschen im Bildsignal sind und „Doppelkanten“ erzeugen. Ein Anwendungsgebiet für diese Art von Operatoren wird weiter unten im Abschnitt „Anwendung des Watershed-Algorithmus“ beschrieben.

Ist man nur an den klaren Konturen der Objekte in einem Bild interessiert, besteht ein grundsätzliches Problem einfacher Kantenoperatoren in ihrer starken Lokalität. Klare Objektkanten werden ebenso detektiert wie Kanten innerhalb von Texturen.

Erweiterte Kantenoperatoren Durch geeignete Vor- und Nachbearbeitung kann das Auftreten ungewollter Kanten reduziert werden. Ein Beispiel für einen Detektor, der solches leistet, ist der Canny-Edge-Detektor [23]. In einem Vorverarbeitungsschritt wird das Rauschen im Bild mit einem gaußschen Weichzeichner reduziert. Anschließend wird ein einfacher Kantenoperator für horizontale und vertikale Kanten, z.B. ein Sobel- oder Prewittoperator, angewandt. Für die so erhaltenen Kanten wird an jedem Punkt die Richtung der Kante bestimmt. Die Richtung ergibt sich aus dem Arkustangens des Quotienten aus der Antwort des vertikalen Kantenoperators und der Antwort des horizontalen Kantenoperators am entsprechenden Punkt. Im nächsten Schritt werden die Kanten eliminiert, deren Wert kleiner als einer der orthogonalen Nachbarwerte ist. Abschließend werden die übrigen Kanten mit einem

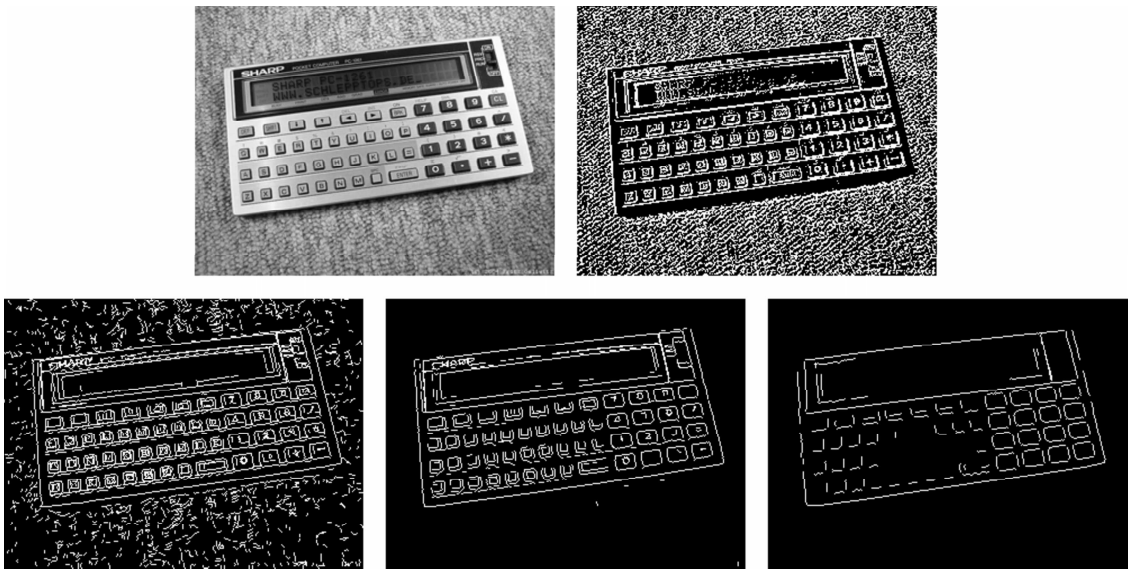


Abbildung 3: *Kantendetektoren im Vergleich: oben links das Originalbild; oben rechts ein einfacher Kantenoperator mit Threshold; unten der hier vorgestellte Kantendetektor mit den Parametersätzen (von links nach rechts)* $[d = 1, b = 1, \gamma = 2.0, \epsilon = 0.01, \lambda = 1.0]$, $[d = 1, b = 1, \gamma = 2.5, \epsilon = 0.01, \lambda = 1.0]$, $[d = 2, b = 2, \gamma = 3.0, \epsilon = 0.01, \lambda = 1.0]$

Schwellwert verglichen. Dabei wird eine Hysterese verwendet, so daß Kanten, die unterhalb eines primären Schwellwerts liegen, eine zweite Chance bekommen. Liegen diese Kanten oberhalb eines sekundären Schwellwerts und in Nachbarschaft zu einer Kante, die oberhalb des primären Schwellwerts liegt, so werden sie nicht eliminiert.

Auch wenn der Canny-Edge-Detektor durchaus brauchbare Ergebnisse liefert, soll im Folgenden ein alternativer Kantendetektor vorgestellt werden. Abbildung 3 zeigt die Ausgabe dieses Kantendetektors auf einem Testbild für 3 verschiedene Parametersätze. Der Kantendetektor erzeugt zunächst die Kanten für jede Richtung und jeden Intensitätsübergang (hell/dunkel und dunkel/hell) separat und fügt diese anschließend zusammen. Abbildung 4 zeigt die einzelnen Schritte des Detektors für horizontale Dunkel/Hell-Übergänge. Im ersten Schritt wendet der Detektor einen einfachen Kantenoperator an:

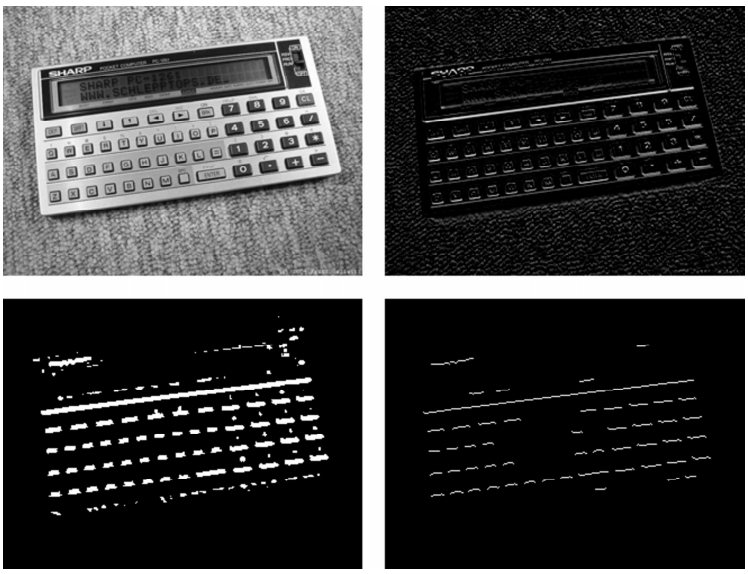
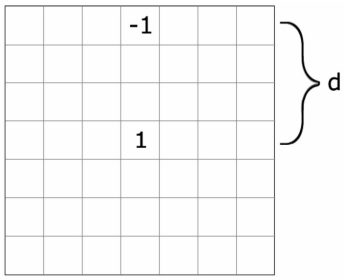


Abbildung 4: *Ansicht der einzelnen Schritte des hier vorgestellten Kantendetektors. (für horizontale Dunkel/Hell-Übergänge und mit den Parametern* $[d = 2, b = 2, \gamma = 3.0, \epsilon = 0.01, \lambda = 1.0]$*)*



Ein Parameter d bestimmt dabei die Distanz zwischen der 1 und der -1 in der Maske des Operators und somit auch die Größe des verwendeten lokalen Fensters. Eine größere Distanz bewirkt, daß „klare“ Kanten im Bild d-breite Kanten in der Antwort des Operators erzeugen. Kanten, die durch Rauschen oder unruhige Strukturen im Bild hervorgerufen werden, ergeben dünnere Kanten. Das so erzeugte Kantenbild wird mit einem Weichzeichner gefiltert, dessen Stärke über den Parameter b bestimmt wird. Während breite Kanten nur an ihren Rändern durch den Weichzeichner geschwächt werden, werden dünne Kanten insgesamt abgeschwächt. In Abbildung 5 kann dieser Effekt beobachtet werden. Die kontrastschwachen, jedoch klaren Kanten des vertikalen Balkens auf der linken Seite bleiben erhalten, während die durch unruhige Struktur auf der rechten Seite hervorgerufenen Kanten unterdrückt werden.

Anstelle eines gaußschen Weichzeichners kann auch die Durchschnittsintensität

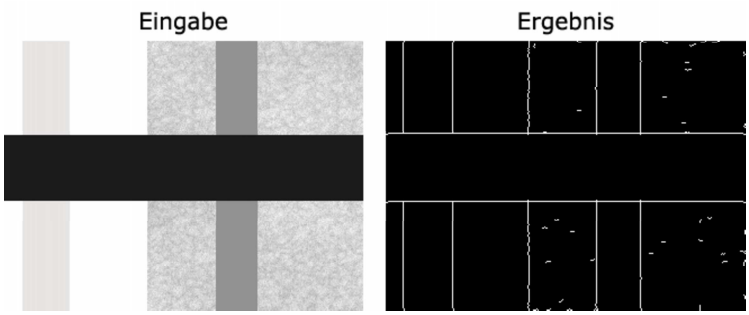


Abbildung 5: *Kontrastschwache, aber klare Kanten bleiben erhalten. Kanten, die durch unruhige Bildstrukturen hervorgerufen werden, werden unterdrückt.*

eines rechteckigen Bereiches um den jeweiligen Bildpunkt herum berechnet werden. Wie Viola und Jones in [1] zeigen, kann die Durchschnittsintensität unabhängig von der Größe des rechteckigen Bereiches nach einem globalen Vorverarbeitungsschritt für jeden Bildpunkt in konstanter Zeit berechnet werden. Dazu wird zunächst ein sogenanntes *Summenbild* S erzeugt. Dieses enthält an jedem Punkt (x, y) die Summe der Intensitätswerte des Originalbildes I aus dem rechteckigen Bereich $(0, 0, x, y)$:

$$S(x, y) = \sum_{i=0}^{x} \sum_{j=0}^{y} I(i, j)$$

Die Durchschnittsintensität eines rechteckigen Bereiches IR_{x_1,y_1,x_2,y_2} mit

$$m = (x_2 - x_1) * (y_2 - y_1)$$

Bildpunkten läßt sich nun durch

$$IR_{x_1,y_1,x_2,y_2} = \frac{S(x_2, y_2) - S(x_2, y_1) - S(x_1, y_2) + S(x_1, y_1)}{m}$$

effizient berechnen.
Im nächsten Schritt des Kantendetektors wird das derart tiefpassgefilterte Kantenbild mit einem Exponenten γ potenziert und es werden anschließend Bildpunkte, deren Intensität geringer als ein Schwellwert ϵ ist, eliminiert. Da der Wertebereich der Punkte des Kantenbildes [0..1] ist, bestimmt der Exponent γ, wie stark die Werte des Kantenbildes gegen null „gedrückt“ werden und der Schwellwert ϵ bestimmt die Grenze, ab wann ein Wert als null interpretiert wird. Auf diese Weise entsteht ein

Schwarzweißbild mit relativ breiten Kanten. In einem Nachbearbeitungsschritt wird die Breite dieser Kanten auf einen Pixel reduziert. Über einen Parameter λ werden zudem die Kanten bestimmt und gelöscht, deren Länge kleiner als λ-mal der Durschnittslänge aller Kanten ist. Abbildung 6 zeigt weitere mit diesem Kantendetektor gefilterte Bilder. Bei allen Bildern wurden dieselben Parameter verwendet.

4.6.2 Segmentierung

Im zweiten Schritt der kantenbasierten Segmentierung muß aus dem im ersten Schritt erzeugten Kantenbild eine Segmentierung erzeugt werden. Ist das erzeugte Kantenbild ohne Fehler, d.h. sind die Konturen der Objekte im Bild vollständig und sind keine zusätzlichen Kanten vorhanden, so kann aus dem Kantenbild sehr leicht eine Segmentierung erzeugt werden. Ein dafür passender Algorithmus kann aus dem in 4.1 vorgestellten Algorithmus mit wenigen Änderungen hergeleitet werden. Ein fehlerfreies Kantenbild ist jedoch allerhöchstens bei sehr eingeschränkten Umgebungen zu erwarten. Üblicherweise sind die Konturen der Objekte unterbrochen und es befinden sich zusätzliche Kanten im Bild, die zu keiner Objektkontur gehören. Für die Segmentbildung müssen also Verfahren gefunden werden, die diese Art von Fehlern im Kantenbild tolerieren und dennoch eine gute Segmentierung erzeugen. Im folgenden sollen für diesen Zweck zwei Verfahren vorgestellt werden. Das erste Verfahren ist ein Scanline-Verfahren, das zweite Verfahren verwendet einen Watershed-Algorithmus.

Scanline-Verfahren Ausgehend von einem binären Kantenbild als Eingabe werden in einem ersten Durchlauf für jede Zeile die Zeilenabschnitte bestimmt, die nicht durch Kanten unterbrochen sind. Anschließend werden die Zeilenabschnitte zusammengefaßt, die sich in der jeweiligen Suchrichtung, z.B. von oben nach unten, überlappen. Die so entstehenden Segmente sind nicht disjunkte Mengen von Zeilenabschnitten. Mit dem Schnitt dieser Mengen können dann disjunkte Mengen bzw. Segmente erzeugt werden. Abbildung 7 zeigt den Ablauf dieses Verfahrens für die Suchrichtung „zeilenweise von oben nach unten“. Führt man das Verfahren auch für die Suchrichtungen „zeilenweise von unten nach oben“, „spaltenweise von links nach rechts“ und „spaltenweise von rechts nach links“ durch, erhält man durch erneuten Schnitt der Mengen eine vollständige und disjunkte Segmentierung des Bildes. In der Regel neigt dieses Verfahren zur Übersegmentierung, der durch

Abbildung 6: *Weitere Beispiele des hier vorgestellten Kantendetektors. Alle Bilder wurden mit dem gleichen Parametereinstellungen gefiltert.*

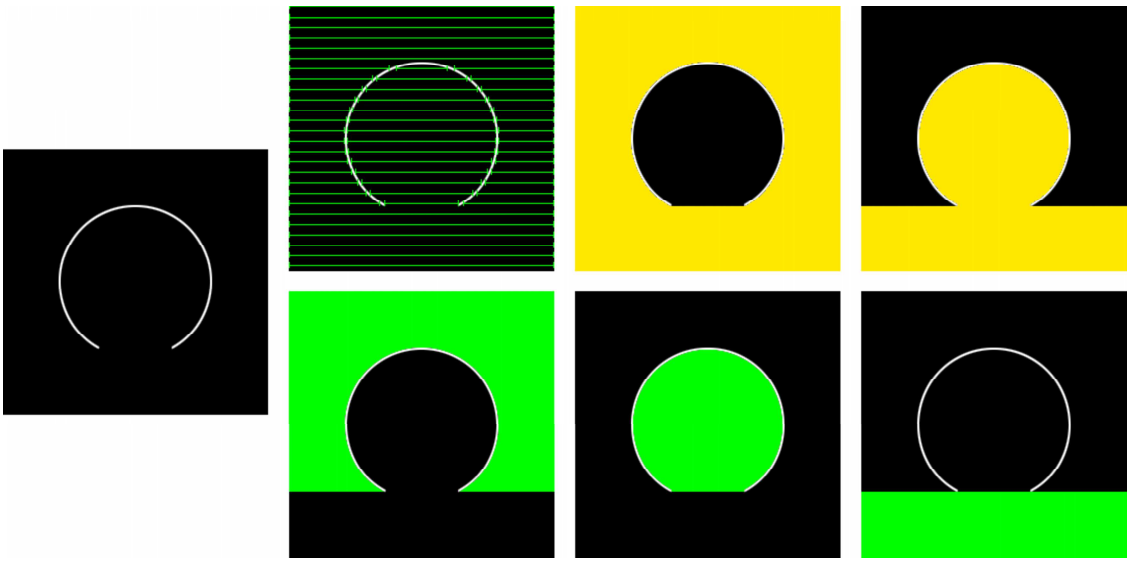


Abbildung 7: *Scanline-Verfahren zur Segmentierung. Links das originale Kantenbild. Obere Reihe: ein Teil der erzeugten Zeilenabschnitte und beide aus diesen Abschnitten erzeugten Segmente. Untere Reihe: Aus den beiden nicht-disjunkten Segmenten entstandene Segmentmenge aus drei disjunkten Segmenten.*

geeignete Segmentverschmelzung begegnet werden kann. Ein mögliches Kriterium für diese Verschmelzung ist das Verhältnis von Randpunkten, die in Nachbarschaft zu gemeinsamen Kanten der zu verschmelzenden Segmente liegen und Randpunkten, die die zu verschmelzenden Segmente direkt gemein haben. Letztere sollten in ihrer Anzahl möglichst groß gegenüber den erstgenannten Randpunkten sein. Dieses Kriterium gewährleistet, das möglichst wenige Kanten innerhalb von Segmenten „eingeschlossen“ werden und aus der Übersegmentierung eine Untersegmentierung hervorgeht. Um eine „unglückliche“ Reihenfolge beim Verschmelzen der Segmente zu vermeiden, kann eine Strategie ähnlich dem in Abschnitt 4.2 beschriebenen simultanen Bereichswachstum angewandt werden. Desweiteren können für ein Segment alle möglichen für eine Verschmelzung in Frage kommenden Segmente bestimmt werden und es kann daraufhin mit dem oben erwähnten Kriterium eines dieser Segmente für die Verschmelzung ausgewählt werden.

Watershed-Verfahren Das Verfahren (siehe [22] S. 617ff) benötigt als Eingabe ein Graustufenbild und beruht darauf, dieses Graustufenbild als Höhenbild zu interpretieren. Die Segmentierung erfolgt durch ein langsames Fluten dieser ima-

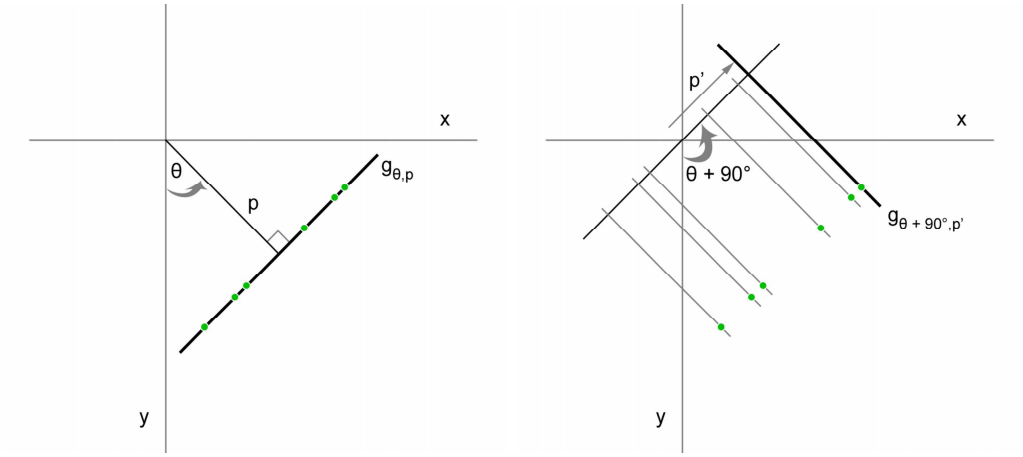


Abbildung 8: *Hough Transformation (links) und Distanzwerte entlang einer Geraden (rechts).*

ginären Landschaft, indem der Grundwasserspiegel immer weiter angehoben wird. Das Wasser erscheint dabei zunächst in den tiefsten Tälern und erfaßt höhergelegene Täler erst zu einem späteren Zeitpunkt. Immer wenn das Wasser droht über einen Bergkamm zu schwappen, wird auf diesem Bergkamm ein Damm errichtet, um das Überlaufen zu verhindern. Der Algorithmus endet, wenn nur noch Dämme aus dem Wasser ragen. Diese bilden dann die Segmentierung des Bildes. In einigen Fällen neigt der Watershed-Algorithmus zur Übersegmentierung. Um dieser entgegenzuwirken, können die Stellen, an denen das Grundwasser aus der Erde kommt, im vorhinein begrenzt werden. Entsprechende Stellen werden über sogenannte Marker definiert. Wo die Marker zu platzieren sind, hängt vom konkreten Anwendungsfall ab. Dadurch bieten die Marker die Möglichkeit, entsprechend anwendungsbezogenes Spezialwissen in den Algorithmus einfließen zu lassen.

Erzeugung eines Höhenbildes Um eine korrekte Eingabe für den Watershed-Algorithmus zu erzeugen, muß zunächst in einem Vorverarbeitungsschritt das binäre Kantenbild in ein Graustufenbild umgewandelt werden. Eine direkte Eingabe des Kantenbildes hätte zur Folge, daß auch kleine Öffnungen in den Konturen der Objekte zum „Auslaufen" der Segmente führen würden. Erzeugt man jedoch aus dem Kantenbild ein Höhenbild, so entsprechen kleine Unterbrechnungen einer Kante nur noch geringen „Dellen" im korrespondierenden Höhenkamm. Um ein entsprechendes Höhenbild aus einem Kantenbild zu erzeugen, muß für jeden Punkt im Kantenbild der minimale Abstand zur nächstliegenden Kante bestimmt werden. Im Folgenden

sollen dafür zwei Methoden vorgestellt werden.

Die erste Methode verwendet eine Erweiterung der Hough-Transformation (siehe [22] S.587ff). Die Hough-Transformation bildet Geraden aus der Bildebene auf Punkte des zweidimensionalen *Parameterraums* ab und bildet Punkte aus der Bildebene auf Sinuskurven des Parameterraums ab. Die linke Seite der Abbildung 8 zeigt eine Gerade in der Bildebene. Diese Gerade $g_{\theta,p}$ läßt sich durch

$$x \cos\theta + y \sin\theta = p$$

beschreiben. Bestimmt man für einen beliebigen Punkt aus der Bildebene für alle möglichen θ die zugehörigen p und trägt diese im Parameterraum ab, so erhält man für diesen Punkt eine sinusförmige Kurve. Erzeugt man auf die gleiche Weise eine weitere Kurve für einen anderen Punkt aus der Bildebene, so liefert der Schnittpunkt dieser beiden Kurven im Parameterraum genau die Werte für θ und p, die die Gerade beschreiben, die in der Bildebene durch die beiden Punkte verläuft. Allgemein schneiden sich in einem beliebigen Punkt im Parameterraum genau die Kurven, deren korrespondierende Punkte in der Bildebene auf derselben Geraden liegen. Unterteilt man den Parameterraum in einzelne Zellen, so kann in jeder Zelle eine Liste der Punkte verwaltet werden, die auf der entsprechenden Geraden[2] liegen. Die grünen Punkte auf der linken Seite in Abbildung 8, die auf der Geraden $g_{\theta,p}$ liegen, sind z.B. in einer solchen Liste zusammengefaßt. Für jeden dieser Punkte kann zudem der Wert

$$p' = x \cos(\theta + \pi/2) + y \sin(\theta + \pi/2)$$

berechnet werden (s. Abbildung 8, rechte Seite). Der Wert p' ist Parameter genau der Geraden, die orthogonal zu der Geraden $g_{\theta,p}$ liegt.

Die Berechnung eines Höhenbildes aus einem binären Kantenbild beginnt nun damit, daß alle Kantenpunkte der Eingabe auf die zuvor beschriebene Art transformiert werden. In jeder Zelle (θ, p) des Parameterraums befindet sich anschließend eine Liste der Kantenpunkte, die auf der jeweiligen Geraden $g_{\theta,p}$ liegen. Für jeden dieser Kantenpunkte wird der p'-Wert der othogonalen Geraden ermittelt und die Liste anhand dieser Werte sortiert.
Um nun den Abstand eines beliebigen Bildpunktes zum nächstliegenden Kanten-

[2] Diese einzelne Gerade ist idealisiert, da der Parameterraum ein Teil des $\Re^2$ ist und jede Zelle dementsprechend die Parameter unendlich vieler Geraden enthält.

punkt zu bestimmen, betrachtet man alle Geraden $g_{\theta,p}$, die durch diesen Bildpunkt gehen. Für jede Gerade $g_{\theta,p}$ wird der p'-Wert der orthogonalen Geraden durch den Bildpunkt bestimmt. Über den Wert p' kann dann durch eine binäre Suche auf der sortierten Liste der Zelle (θ, p) die Position des Bildpunktes auf der Geraden $g_{\theta,p}$ gefunden werden. Vorgänger und Nachfolger in der Liste, sofern existent, sind die nächsten Kantenpunkte bzgl. des Bildpunktes entlang der Geraden $g_{\theta,p}$. Die absolute Differenz der p'-Werte von Vorgänger und Bildpunkt bzw. Nachfolger und Bildpunkt entspricht dem Abstand zwischen den entsprechenden Kantenpunkten und dem Bildpunkt in der Bildebene. Das Minimum der Abstände, die für die einzelnen Geraden $g_{\theta,p}$ berechnet wurden, ist der gesuchte minimale Abstand des Bildpunktes zur nächstliegenden Kante. Diese Erweiterung der Hough-Transformation eignet sich insbesondere für Bilder mit hohen Auflösungen, da der Abstand zu den Kanten berechnet wird und nicht, z.B. durch ein Bereichswachstum, „gesucht" werden muß.

Die zweite Methode approximiert das Höhenbild. Dabei wird über einen Parameter d die maximale Entfernung zur nächstliegenden Kante bestimmt. Punkte, die weiter von einer Kante entfernt liegen, werden alle auf einer Höhe liegend angenommen. Diese Einschränkung hat zur Folge, daß nur noch Lücken der Größe $2d$ im Kantenbild toleriert werden. Größere Lücken führen zum „Auslaufen" der Segmente. Bei Bildern mit üblichen Auflösungen (z.B. 640 x 480) führen maximale Entfernungen von 32 bis 64 Pixeln zu guten Ergebnissen auf den vom oben beschriebenen Kantendetektor erzeugten Kantenbildern. Der Algorithmus erzeugt aus einem Kantenbild ein Höhenbild, indem man von jedem Kantenpunkt ausgehend eine Art Bereichswachstum startet. Für jeden Bildpunkt, der im Rahmen dieses Bereichswachstums untersucht wird, wird geprüft, ob bereits ein Abstandswert vermerkt wurde und ob dieser Abstandswert kleiner ist als der aktuelle Abstand zum Ausgangspunkt des Bereichswachstums. Ist dies nicht der Fall, so wird der aktuelle Abstand zum Ausgangspunkt als neuer Abstandswert vermerkt und es werden die Nachbarpunkte des untersuchten Punktes als spätere Kandidaten im Bereichswachstumsverfahren gespeichert. Das Bereichswachstum endet, wenn keine Kandidaten mehr vorhanden sind, oder die maximale Distanz d erreicht wurde. Insbesondere bei kleinen Auflösungen und nicht zu großen Werten für d ist dieses Verfahren deutlich schneller als die oben vorgestellte erste Methode.

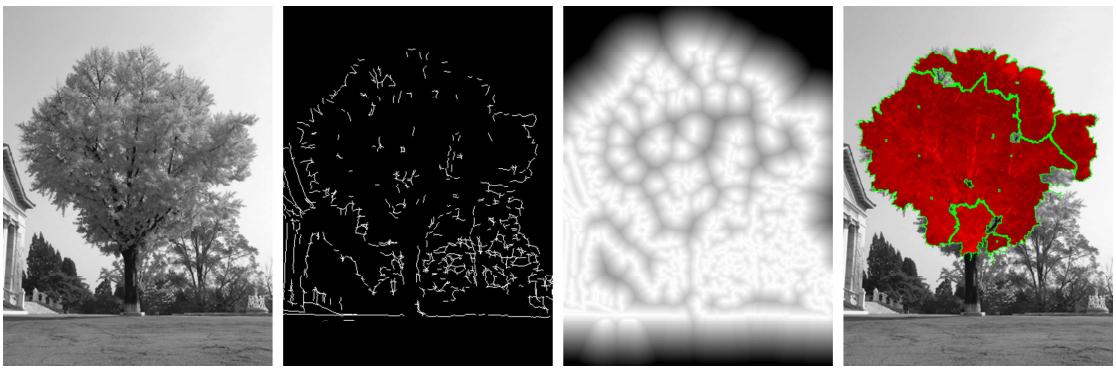


Abbildung 9: *Kantenbasierte Segmentierung im Überblick: Originalbild, Kantenbild, Höhenbild und beispielhaft Teile der Segmentierung der Baumkrone - Auffallend ist die gute Segmentierung der Baumkrone trotz der starken Fehler im Kantenbild.*

Anwendung des Watershed-Algorithmus Angewandt auf das Höhenbild kann nun mit Hilfe des Watershed-Algorithmus (siehe [22] S.617ff) eine Segmentierung erzeugt werden. Die Grundidee des Algorithmus läßt sich mit folgender Analogie beschreiben: Man stelle sich eine Landschaft dem Höhenbild entsprechend vor. Im Laufe der Zeit steigt der Grundwasserspiegel immer weiter an und beginnt, die tiefsten Täler mit Wasser zu füllen. Immer dann, wenn das Wasser droht, von einem Tal in ein benachbartes Tal überzulaufen, wird an der entsprechenden Stelle ein Damm errichtet, um das Überlaufen zu verhindern. Ist die komplette Landmasse mit Wasser bedeckt und ragen nur noch die Dämme aus dem Wasser, terminiert der Algorithmus und die Dämme bilden die Segmentierung des Bildes. Jedes Segment hat dabei seinen Ursprung in einem lokalen Minimum des Höhenbildes. Dies kann u.U. zu einer Übersegmentierung des Bildes führen. Ein geeignetes Mittel, eine solche Übersegmentierung zu verhindern, liegt darin, sogenannte „Marker“ zu verwenden. Im übertragenen Sinne beschreiben Marker die Stellen, an denen Grundwasser zu Tage treten kann und beschränken so im vorhinein die Anzahl der Segmente. Täler, also lokale Minima, die keinen Marker erhalten, werden nicht durch Dämme geschützt. Sie werden irgendwann von einem der benachbarten Täler aus geflutet und dem entsprechenden Segment des Nachbartales zugeordnet. Mit dem in 4.6.1 beschriebenen Laplace Operator, der die 2. Ableitung approximiert, lassen sich passende Marker aus dem Höhenbild generieren. Jedoch kann der Einsatz von Markern eine Übersegmentierung des Bildes nicht in jedem Fall verhindern. Aus diesem Grund werden ähnliche benachbarte Segmente in einem Nachbearbeitungsschritt miteinander verschmolzen.

Für die Bewertung der Ähnlichkeit zweier Segmente werden Durchschnitt, Varianz und Entropie der beiden Segmente berechnet und miteinander verglichen. Nur wenn bei allen drei Kriterien die entsprechenden Grenzwerte unterschritten werden, werden die Segmente miteinander verschmolzen.

Eine Übersicht über die hier vorgestellte kantenbasierte Segmentierung zeigt Abbildung 9. Die Fähigkeit auch große Lücken in der Kontur eines Objektes zu tolerieren zeigt sich im oberen Teil des abgebildeten Baumes. Im Kantenbild ist eine große Lücke in der Kontur der Baumkrone zu erkennen und dennoch bilden die an dieser Stelle erzeugten Segmente die Form der Baumkrone nach.

5 Merkmalbestimmung

Die Bestimmung von Merkmalen in einem Bild läuft in zwei Schritten ab. Zunächst werden mittels eines Keypoint-Detektors interessante bzw. markante Punkte im Bild bestimmt. Anschließend wird an jedem dieser Punkte die lokale Umgebung über einen Merkmalvektor beschrieben. Im folgenden sollen für jeden dieser Schritte mehrere Verfahren beschrieben und miteinander Verglichen werden.

5.1 Keypoint-Detektoren

Ein idealer Keypoint-Detektor sollte unabhängig von Bildstörungen, Helligkeitsschwankungen und beliebigen Transformationen immer die gleichen Punkte eines dargestellten Objektes markieren. Erwartungsgemäß erfüllen die bisher bekannten Keypoint-Detektoren diese Ansprüche nicht in vollem Umfang. Die Wiederholwahrscheinlichkeit der hier vorgestellten Keypoint-Detektoren erreichte bei einem Test (s. Abbildung 13 bzw. Abschnitt 5.1.5) mit einer ausreichenden Anzahl an Punkten und einer geringfügigen Toleranz bzgl. der Position bis zu 80%.

5.1.1 Moravec

Die Idee des Keypoint Detektors von Moravec [10] besteht darin, ein lokales Fenster $W_{x,y}$ geringfügig in verschiedene Richtungen zu bewegen und die dabei entstehenden Veränderungen in der Intensitätsverteilung innerhalb des Fensters zu beobachten. Ist der betrachtete Bildausschnitt gleichförmig, so werden keine oder nur geringe Veränderungen auftreten. Enthält der Bildausschnitt eine Kante, so wird nur die

Bewegung, die orthogonal zur Kante stattfindet eine große Veränderung in der Intensitätsverteilung aufweisen. Befindet sich jedoch ein einzelner Punkt bzw. eine Ecke innerhalb des Bildausschnittes, so wird jede Bewegung eine große Veränderung hervorrufen. Betrachtet man also die kleinste Veränderung, die während der Verschiebungen des Fensters auftritt und erhält man dabei einen großen Wert, so ist dies ein Hinweis auf einen Eckpunkt. Formal läßt sich die Veränderung E, die bei der Verschiebung des lokalen Fensters W am Punkt (x, y) in eine Richtung (φ, ϑ) auf den Intensitätswerten I des Bildes entsteht, folgendermaßen beschreiben:

$$E_{\varphi,\vartheta} := \sum_{u,v \in W_{x,y}} (I_{u+\varphi,v+\vartheta} - I_{u,v})^2.$$

Sei D die Menge der verwendeten Verschiebungen (φ, ϑ), dann werden die Bildpunkte (x, y) als Eckpunkte gewählt, für die

$$E_{min} := \operatorname{argmin} \{E_{\varphi,\vartheta} | \ (\varphi, \vartheta) \in D\}$$

oberhalb eines zu wählenden Schwellwerts liegt.
Wie in [11] näher beschrieben hat der Moravec Detektor einige Schwächen. So wird das lokale Fenster nur in wenige feste Richtungen verschoben. Dies führt zu einem anisotropen Verhalten des Detektors. Kanten im Bild erzeugen also in Abhängigkeit ihrer Ausrichtung unterschiedliche Antworten des Detektors. Desweiteren verursacht das ungewichtete und rechteckige lokale Fenster sprunghafte Antworten und die ausschließliche Betrachtung der minimalen Veränderung der Intensitäten bevorzugt zu sehr einfache Kanten, die somit fälschlicherweise als Ecken identifiziert werden. Dies führt zu einer sehr hohen Gesamtzahl an Eckpunkten und macht es schwierig die Eckpunkte als Ausgangsbasis für merkmalsbeschreibende Verfahren zu nutzen. Andererseits liefert, wie unten näher beschrieben, diese hohe Anzahl an Eckpunkten sehr gute Ergebnisse unter dem Aspekt der Wiederholbarkeit (s. Abbildung 13). Eine Ausgabe des Moravec Detektors auf einem Beispielbild zeigt Abbildung 10, oben links.

5.1.2 Harris

Der Harris Detektor [11] baut auf der Idee von Moravec auf. Anstelle über ein in feste Richtungen verschobenes lokales Fenster am Punkt (x, y), wird die Intensitäts-

veränderung E unter einer kleinen Verschiebung (φ, ϑ) durch

$$E(\varphi, \vartheta) = (\varphi, \vartheta) M (\varphi, \vartheta)^T$$

beschrieben. Die Matrix M beschreibt die Gradienten innerhalb des lokalen Bereiches $W_{x,y}$ um den Punkt (x, y) durch

$$M := \begin{bmatrix} A & C \\ C & B \end{bmatrix}$$

mit

$$A := \sum_{u,v \in W_{x,y}} (I_{u-1,v} - I_{u+1,v})^2 * g_{u,v,x,y}$$

$$B := \sum_{u,v \in W_{x,y}} (I_{u,v-1} - I_{u,v+1})^2 * g_{u,v,x,y}$$

$$C := \sum_{u,v \in W_{x,y}} (I_{u-1,v} - I_{u+1,v}) * (I_{u,v-1} - I_{u,v+1}) * g_{u,v,x,y}$$

und der Gewichtungsfunktion g

$$g_{u,v,x,y} := \exp \frac{-\left((u-x)^2 + (v-y)^2\right)}{2\sigma^2}.$$

Die Matrix M kann als Kovarianzmatrix der aufsummierten Gradienten des lokalen Bereiches in x- und y-Richtung aufgefaßt werden. Seien λ_1 und λ_2 die Eigenwerte der Matrix M. Für die Eigenwerte lassen sich ähnliche Überlegungen wie beim Moravec Detektor anstellen: Sind beide Eigenwerte klein, enthält der lokale Bildbereich keine ausgeprägten Ecken oder Kanten. Ist nur einer der Eigenwerte groß, weist dies auf eine Kante im Bildbereich hin, da die Gradienten überwiegend nur in eine Richtung ausgerichtet sind. Sind beide Eigenwerte groß, sind die Gradienten in unterschiedliche Richtungen ausgerichtet und weisen auf einen Eckpunkt hin. Im Gegensatz zum Moravec Detektor führt die Verwendung der Eigenwerte zu einem isotropen Verhalten des Harris Detektors und die Verwendung der gaußschen Gewichtung verhindert sprunghafte Antworten.
Um die explizite Berechnung der Eigenwerte von M zu vermeiden, können Spur und

Determinante von M verwendet werden:

$$Tr(M) = \lambda_1 + \lambda_2 = A + B,$$

$$Det(M) = \lambda_1 \lambda_2 = AB - C^2.$$

Mit Spur und Determinante läßt sich eine Funktion R definieren, die für Ecken positiv, für Kanten negativ und für Flächen klein wird:

$$R := Det - kTr^2.$$

Der Parameter k bestimmt die Sensitivität des Detektors. Ein größeres k führt zu kleineren Werten für R und zu weniger erkannten Ecken. Ein kleineres k führt zu größeren Werten für R und zu mehr erkannten Ecken. Eine alternative Formulierung von R, die ohne den Parameter k auskommt, wurde von Nobel [24] vorgeschlagen:

$$R = Det/(Tr + \epsilon)$$

mit

$$\epsilon > 0.$$

Die gefundenen Eckpunkte werden so zusammengefaßt, daß in der Nachbarschaft eines Eckpunktes kein weiterer Eckpunkt existiert, der einen größeren R-Wert hat. Die Größe der Nachbarschaft ist ein weiterer Parameter, übliche Werte liegen zwischen 2 und 10 Bildpunkten. Eine Ausgabe des Harris Detektors auf einem Beispielbild zeigt Abbildung 10, oben rechts und unten links (nobel).

5.1.3 KLT

Der KLT (Kanade Lucas Tomasi) Detektor [25] ist dem Harris Detektor sehr ähnlich. Im Gegensatz zum Harris Detektor berechnet der KLT Detektor den kleineren der beiden Eigenwerte λ_1 und λ_2 der Matrix M explizit:

$$\lambda_1 = \frac{A + B + \sqrt{(A-B)^2 + 4C^2}}{2},$$

$$\lambda_2 = \frac{A + B - \sqrt{(A-B)^2 + 4C^2}}{2}.$$

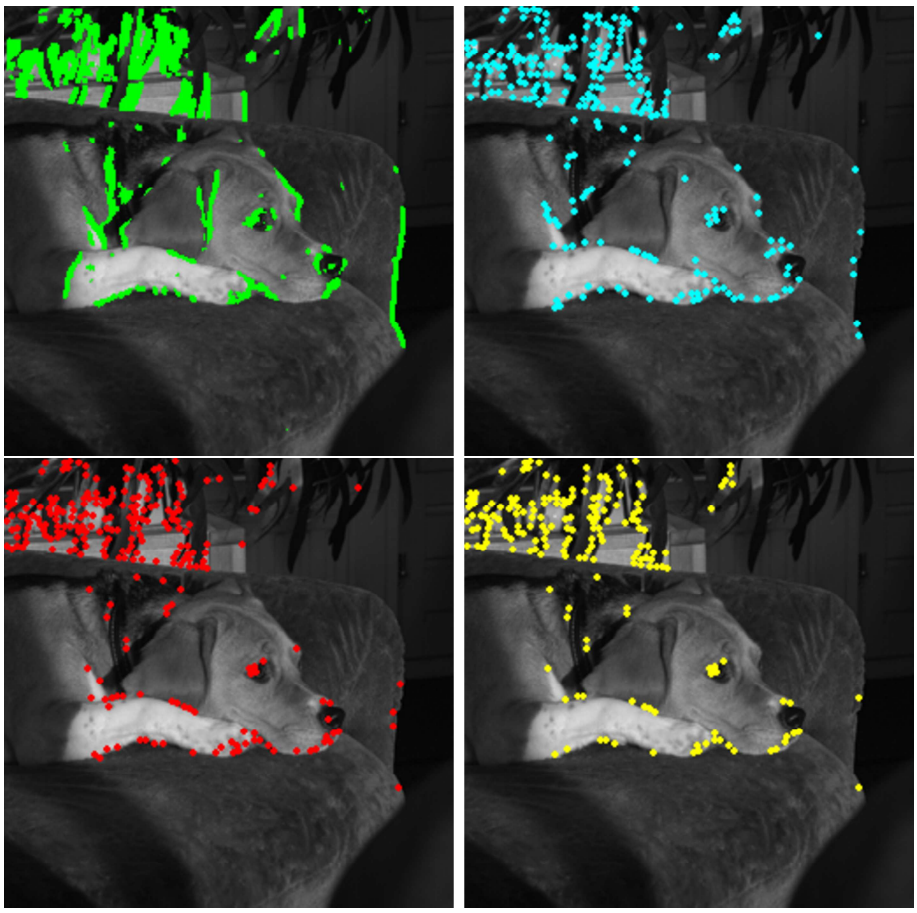


Abbildung 10: *Einfache Keypoint Detektoren von Moravec, Harris-Stephens, Harris-Nobel und Kanade-Lucas-Tomasi*

Für den kleineren Eigenwert λ_2 können die gleichen Überlegungen wie in 5.1.1 getroffen werden. Liegt λ_2 über einem Schwellwert λ_t, so kann der betrachtete lokale Bereich als Ecke angesehen werden. Ein geeigneter Schwellwert λ_t kann z.B. über das Histogramm aller λ_2-Werte des jeweiligen Bildes bestimmt werden. Die gefundenen Eckpunkte werden in gleicher Weise wie beim Harris-Detektor zusammengefaßt, so daß in der Nachbarschaft eines Eckpunktes kein weiterer Eckpunkt existiert, der einen größeren Eigenwert λ_2 besitzt. Abbildung 10 unten rechts zeigt die Ausgabe des KLT Detektors auf einem Beispielbild.

5.1.4 Kadir und Brady

Der Detektor von Kadir und Brady [8] beruht auf der Annahme, daß interessante Stellen in Bildern einen gewissen Grad an Unvorhersehbarkeit bzgl. lokaler Attribute

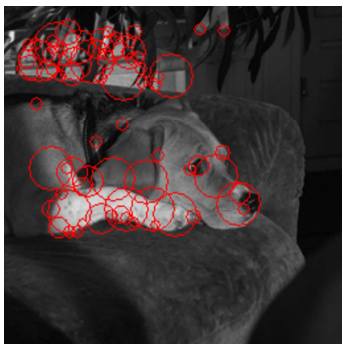


Abbildung 11: *Keypoint Detektor von Kadir und Brady*

und bzgl. ihrer räumlichen Ausdehnung haben. Als Maß für die Unvorhersehbarkeit bzgl. lokaler Attribute wird die Shannon Entropie verwendet. Betrachtet man zum Beispiel einen nahezu konstanten Bildbereich, dann hat die Wahrscheinlichkeitsverteilung der Intensitätswerte dieses Bildbereichs eine schmale und hohe Spitze. Dies bedeutet, das die Intensität der meisten Bildpunkte des Bildbereichs sehr gut vorhersehbar ist. Entsprechend ist die Entropie des Bildbereichs gering. Im Gegensatz dazu ist die Entropie eines Bildbereichs mit einer *ausgedehnten* Wahrscheinlichkeitsverteilung der Intensitätswerte hoch. Die Intensität eines Bildpunktes des Bildbereichs ist demnach weniger gut vorhersehbar bzw. unvorhersehbar.

Die Entropie alleine reicht jedoch nicht als Maß für interessante Bildbereiche aus. Sie trifft keine Aussage über die Anordnung der Bildpunkte. So hat z.B. ein Bildbereich, der ein gleichförmiges Rauschen enthält, eine sehr hohe Entropie, ohne dabei auch zugleich eine interessante Stelle zu sein. Betrachtet man jedoch den Verlauf der Entropiewerte, während man sukzessive den Bildbereich vergrößert, bilden sich, wenn interessante Stellen im Bildbereich vorhanden sind, deutliche lokale Maxima aus. Sind keine interessanten Stellen vorhanden, z.B. im Falle des gleichförmigen Rauschens, so verändert sich die Entropie nicht oder nur gering. Um deutliche lokale Maxima von schwachen lokalen Maxima zu unterscheiden, werden die entsprechenden Entropiewerte zusätzlich durch eine Gewichtungsfunktion bewertet. Als Gewichtungsfunktion könnte direkt die Breite des jeweiligen lokalen Maximums genommen werden. Aus Gründen der Einfachheit wird jedoch die Summe der absoluten Änderung der Wahrscheinlichkeitsverteilungsfunktion über die Größe des lokalen Bildbereichs verwendet. Zudem enthält die Gewichtungsfunktion einen größenabhängigen Faktor, der große Bildbereiche kleinen Bildbereichen vorzieht.

Der Detektor von Kadir und Brady läßt sich folgendermaßen zusammenfassen:

1. Berechne die Shannon Entropie des lokalen Bildbereichs am Punkt x für verschiedene Größen s (Kreise mit Radius s) und den Intensitätswerten D:

$$H(s,x) := -\sum_{d \in D} p(d,s,x) \log p(d,s,x)$$

   mit $p(d,s,x)$, der Wahrscheinlichkeit den Intensitätswert d in einem Umkreis mit Radius s vom Punkt x vorzufinden.

2. Wähle die Größen s_p, an denen die Entropie ein Maximum einnimmt.

3. Berechne die Gewichtungsfunktion an den Stellen s_p

$$W(s,x) := \frac{s^2}{2s-1} \sum_{d \in D} |p(d,s-1,x) - p(d,s+1,x)|$$

4. Bestimme die Interessantheit des Punktes x als Maximum der Produkte aus $H(s,x)$ und $W(s,x)$ für die Größen s_p.

Als Keypoints des Bildes werden schließlich die Punkte ausgewählt, deren Interessantheitswert über einem Threshold liegt, oder es werden alternativ die n-besten Punkte bzgl. ihres Interessantheitswertes genommen. Eine Ausgabe des Detektors auf einem Beispielbild zeigt Abbildung 11.

5.1.5 Test der Wiederholbarkeit

Die Wiederholbarkeit eines Keypoint Detektors bezieht sich darauf, daß ein Detektor auch nach der Transformation eines Bildes (z.B. einer Skalierung) die gleichen Stellen innerhalb des transformierten Bildes markieren sollte, die der Detektor auch vor der Transformation markiert hat. Um dies bei den oben beschriebenen Keypoint Detektoren zu testen, wurden zunächst die Ausgaben der Detektoren auf einer Reihe von Testbildern mit einer Auflösung von 256 x 256 Pixeln (s. Abbildung 14) generiert. Anschließend wurden 9 verschiedene Transformationen (s. Abbildung 12) auf die einzelnen Testbilder angewandt und jeweils die Ausgaben der Detektoren auf den transformierten Bildern erzeugt. Die Ausgaben der Detektoren wurden dann

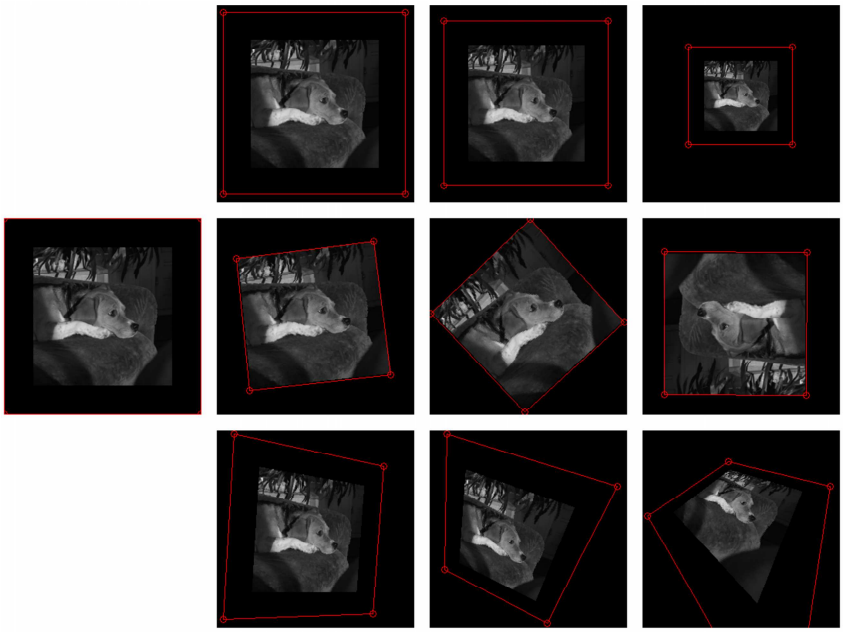


Abbildung 12: *Verwendete Transformationen*

entsprechend rücktransformiert, um sie mit der Ausgabe der nichttransformierten Bilder vergleichen zu können. Abbildung 13 zeigt die Ergebnisse für 250, 100 und 35 markierte Punkte. Die X-Achse stellt die Entfernung dar, in der ein Punkt noch als wiederholt markiert gilt. Also ob sich nach der Rücktransformation eines Punktes in einem bestimmten Umkreis ein markierter Punkt im nichttransformierten Bild finden läßt. Die Y-Achse zeigt die Wahrscheinlichkeit, nach der Rücktransformation einen korrespondierenden Punkt im nichttransformierten Bild zu finden. Beispielsweise zeigt der Detektor von Kadir und Brady bei einer Toleranz von 2 Pixeln nur eine Wiederholbarkeit von etwa 40% (250 Keypoints).

Betrachtet man die Ergebnisse des Tests in Abbildung 13, dann fällt auf, daß der Moravec Detektor im Bereich kleiner Toleranz besser abschneidet als alle anderen getesteten Keypoint Detektoren. Dies liegt daran, daß der Moravec Detektor deutlich mehr Punkte liefert als die übrigen Detektoren. Durch die Beschränkung auf die 250 (100,35) besten Punkte werden vom Moravec Detektor nur wenige, dafür aber dicht markierte Stellen im Bild abgedeckt. Entsprechend hoch ist die Wahrscheinlichkeit, daß nach der Rücktransformation in kurzer Distanz zum jeweils betrachteten Punkt ein korrespondierender Punkt gefunden werden kann. So gesehen liefert der Moravec Detektor die besten Ergebnisse bzgl. der Wiederholbarkeit in diesem Test. Diesen Vorteil erkauft sich der Moravec Detektor jedoch zu einem hohen Preis. Um

möglichst alle interessanten Stellen eines Bildes durch den Detektor abdecken zu können, benötigt man etwa die 10fache Menge an Punkten im Vergleich zu den anderen Detektoren im Test. Im Vergleich zum Detektor von Kadir und Brady sogar etwa die 100fache Menge an Punkten. So enthält beispielsweise die Ausgabe des Moravec Detektors in Abbildung 10 ca. 2500 Punkte.

Der Harris Detektor und seine Variante von Nobel sowie der KLT Detektor liefern in etwa die gleichen Ergebnisse. Geht man von einer Toleranz von 5 Pixel aus, so finden zwischen 60% und 80% aller rücktransformierten Punkte einen korrespondierenden Punkt im Ausgangsbild. Die Zahl der *direkten* Treffer, also bei einer Toleranz von 0 Pixeln, liegt etwa bei 30%. Werte jenseits der 80% ergeben sich bei größeren Toleranzen, was jedoch bei hinreichend vielen und relativ gleichmäßig über das Bild verteilten Punkten keine wirkliche Aussage über die Qualität des jeweiligen Keypoint Detektors zuläßt.

Das im Vergleich zu den *klassischen* Detektoren schlechte Abschneiden des Detektors von Kadir und Brady läßt sich auf zwei Ursachen zurückführen. Zum einen liefert der Detektor nur relativ wenige hoch bewertete Punkte, die im wesentlichen diejenigen sind, die auch nach den Transformationen erneut gefunden werden. Entsprechend erkennt man in Abbildung 13, daß die Kurve des Kadir und Brady Detektor sich den Kurven der *klassischen* Detektoren annähert, je weniger Punkte betrachtet werden. Zum zweiten beziehen sich die vom Kadir und Brady Detektor gelieferten Punkte auf die Mittelpunkte von z.T. relativ großen lokalen Bereichen. Unter perspektivischen Transformationen verändern sich diese Bereiche und entsprechend mit ihnen ihre Mittelpunkte.

Unter Einbeziehung dieser Testergebnisse und aufgrund der Möglichkeit einer effizienten Implementierung verwendet der in dieser Arbeit vorgestellte Algorithmus den KLT Detektor zur Identifikation interessanter Punkte innerhalb der Segmente.

5.2 Merkmalbeschreibung

Für jeden der zuvor berechneten interessanten Punkte wird ein lokaler Bereich um den jeweiligen Punkt herum als Merkmalvektor beschrieben. Wie schon bei den Keypoint-Detektoren sollte eine ideale Merkmalbeschreibung invariant gegenüber Störungen, Helligkeitsunterschieden und Transformationen jeglicher Art sein und zugleich in hohem Maße diskriminierend sein. Diese Anforderungen werden in der Regel

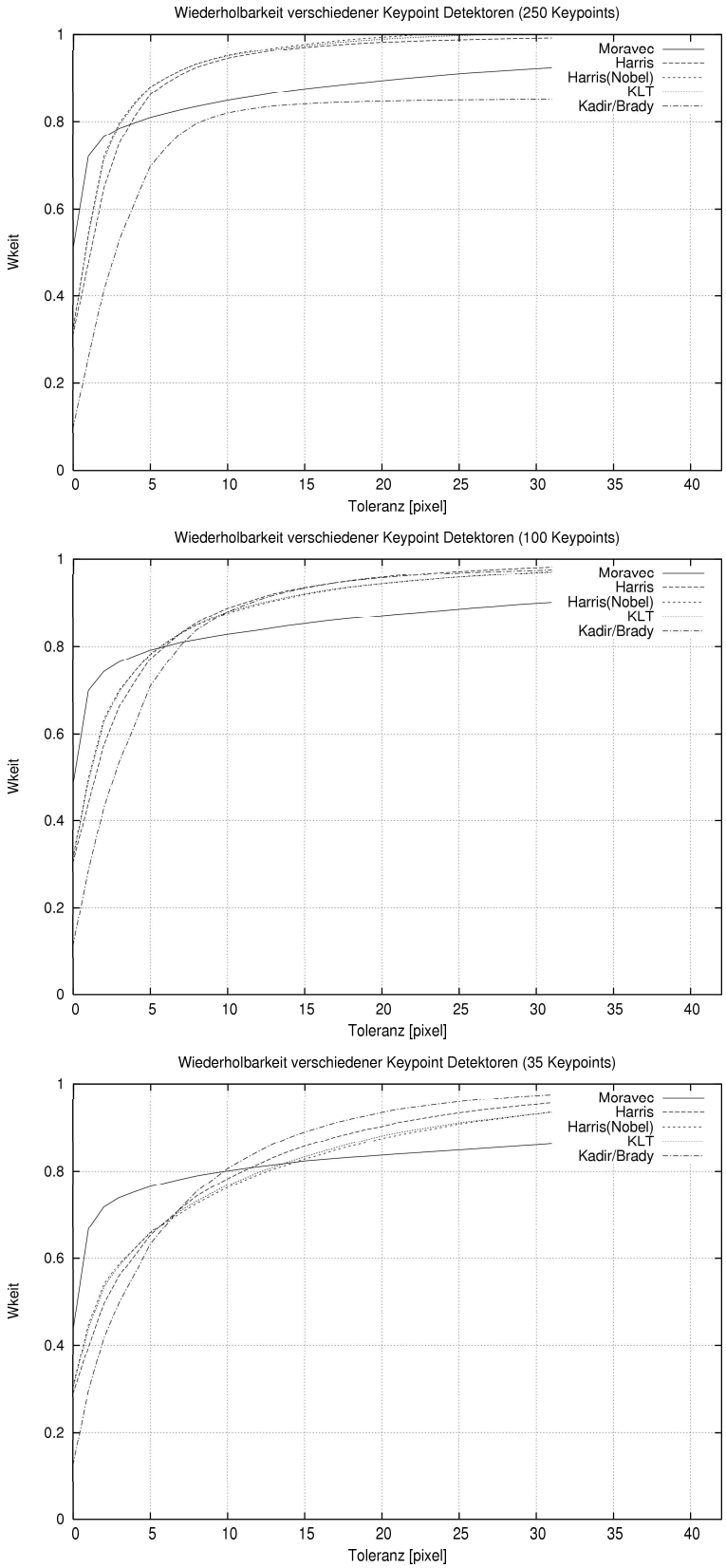


Abbildung 13: *Wiederholbarkeit der Keypoint Detektoren (250, 100 und 35 Keypoints) auf Bildern mit einer Auflösung von 256 x 256 Pixeln.*

Abbildung 14: *Verwendete Testbilder*

nicht vollständig durch die Merkmalbeschreibung erfüllt. Bezüglich der Invarianz gegenüber jeglichen Transformationen beschränkt man sich vielmals auf Translations-, Skalierungs- und Rotationsinvarianz, wobei die letzten beiden nicht durch die Merkmalbeschreibung selbst implementiert werden, sondern durch geeignete Vorverarbeitungsschritte erreicht werden.

5.2.1 Skalierungsinvarianz

Eine Skalierungsinvarianz kann durch die Bestimmung der Größe der lokalen Umgebung eines Keypoints und anschließender Skalierung dieser Umgebung auf eine feste Umgebungsgröße (z.B. 16 x 16 Pixel) erzeugt werden. Die Größe einer lokalen Umgebung kann durch das oben beschriebene Verfahren von Kadir und Brady bestimmt werden. Sollte sich dabei kein lokales Maximum in den Entropiewerten einstellen, so kann alternativ die Größe s_p gewählt werden, bei der der Gradient der Entropiewerte ein Maximum einnimmt. Ist auch in diesem Fall kein Maximum vorhanden, so wird einfach eine feste Umgebungsgröße angenommen.

5.2.2 Rotationsinvarianz

Eine Rotationsinvarianz kann durch die Bestimmung der Hauptrichtung der jeweiligen lokalen Umgebung erzeugt werden. Ist die Hauptrichtung einmal bekannt, so kann der entsprechende Bildausschnitt um genau diesen Anteil auf eine Nullposition gedreht werden. Im Folgenden wird zunächst die von Lowe für den SIFT-Descriptor vorgeschlagene Methode zur Bestimmung der Hauptrichtung beschrieben. Im Anschluß wird eine Alternative vorgestellt, die in erster Linie eine unschöne Eigenschaft des SIFT-Verfahrens vermeidet.

SIFT Methode Die von Lowe in [9] beschriebene Methode erstellt für die Bestimmung der Hauptrichtung des lokalen Bereiches $W_{x,y}$ am Punkt (x, y) ein Histogramm H mit $k \in N$ Bins über die Richtungen $\{d \in N | \ 1 \leq d \leq k\}$ der Gradienten innerhalb des lokalen Bereiches. Jede Richtung d umfaßt dabei $360/k$ Grad und geht in das Histogramm mit der durch eine Gaußfunktion g gewichteten Stärke m ein:

$$H := \{(d, h(d)) \mid 1 \leq d \leq k, \quad d, k \in N\}$$

mit

$$h(d) := \sum_{u,v \in W_{x,y}} \delta\left(\left\lfloor \frac{\theta(u,v) * k}{2\pi} \right\rfloor - d\right) * m(u,v) * g_{x,y}(u,v)$$

und

$$\delta(\xi) := \begin{cases} 1 & \text{für } \xi = 0 \\ 0 & \text{sonst} \end{cases},$$

$$\theta(u,v) := \arctan2\left(I(u,v+1) - I(u,v-1)\right), \left(I(u+1,v) - \left(I(u-1,v)\right)\right) + \pi,$$

$$m(u,v) := \sqrt{\left(I(u+1,v) - I(u-1,v)\right)^2 + \left(I(u,v+1) - I(u,v-1)\right)^2},$$

$$g_{x,y}(u,v) := \exp \frac{-\left((u-x)^2 + (v-y)^2\right)}{2\sigma^2}.$$

Die Hauptrichtung für einen untersuchten lokalen Bereich ergibt sich aus der Extremstelle der Parabel, die über das Maximum des Histogramms und seine beiden benachbarten Werte gelegt wird. Existieren noch weitere lokale Maxima im Histogramm, deren Wert 80% des globalen Maximums übersteigt, so werden auch für diese Maxima Hauptrichtungen bestimmt. Der Ansatz von Lowe sieht vor, daß für alle auf diese Weise erzeugten Richtungen eine Merkmalsbeschreibung erzeugt wird.

Alternative Die zuvor beschriebene Methode liefert leidlich brauchbare Ergebnisse, jedoch stört das Auftreten mehrfacher Richtungen doch sehr. Das hier vorgestellte Verfahren vergleicht Mengen von Merkmalsbeschreibungen und mehrere, unterschiedlich oft auftredende Varianten ein und desselben Merkmals innerhalb der Mengen würde die Bildung einer zuverlässigen Metrik erschweren.

Eine alternative Methode zur Bestimmung der Hauptrichtung eines lokalen Bereiches basiert auf folgender Idee: Betrachtet man den lokalen Bereich in Polarkoordinatendarstellung (Winkel gegen Radius, s. Abbildung 15), dann wird aus jeder Drehung des lokalen Bereiches eine Translation in Richtung der Achse des Winkels. Projiziert man zudem die Radiuswerte auf die Koordinatenachse des Winkels,

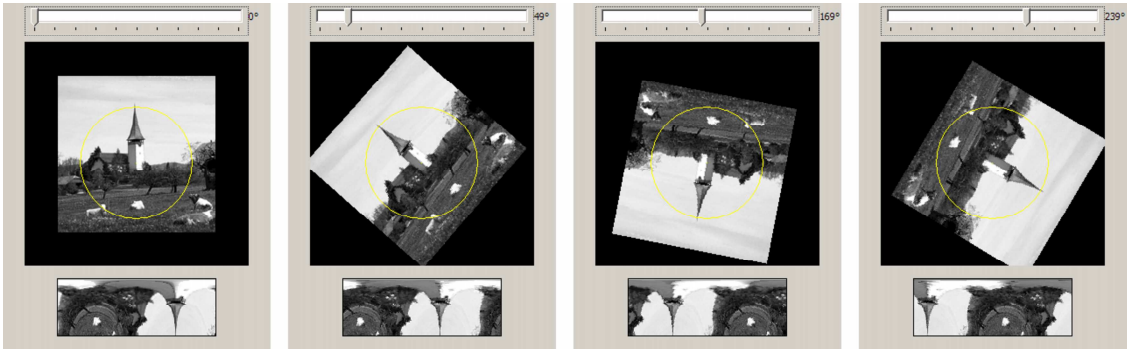


Abbildung 15: *Polarkoordinatendarstellung eines lokalen Bereiches für verschiedenen Drehungen des Bereiches.*

so erhält man eine eindimensionale Funktion $f(w)$, die sich proportional zur Drehung des lokalen Bereiches verschiebt. Eine Drehung des Bereiches um einen Winkel θ entspricht dann der Funktion $f(w+\theta)$. Formal läßt sich die Funktion $f(w)$ so darstellen:

$$f(w) = \sum_{r=0}^{R} I\left(x + r\begin{pmatrix}\cos w \\ \sin w\end{pmatrix}\right)$$

mit x dem Mittelpunkt des betrachteten lokalen Bereiches, R dem Radius des Bereiches und $I(p)$ dem Intensitätswert des Bildes an Position p.
Um nun die Verschiebung von f für einen konkreten lokalen Bereich herauszubekommen, bietet es sich an, die Fourier-Transformierte von f zu bilden:

$$F(u) = \int_{w=0}^{2\pi} f(w)\, e^{-iwu}.$$

Sei

$$F(u) = F_r(u) + iF_i(u)$$

dann berechnet sich das Phasenspektrum von $F(u)$ durch

$$\Phi(u) = \arctan2\left(F_i(u), F_r(u)\right).$$

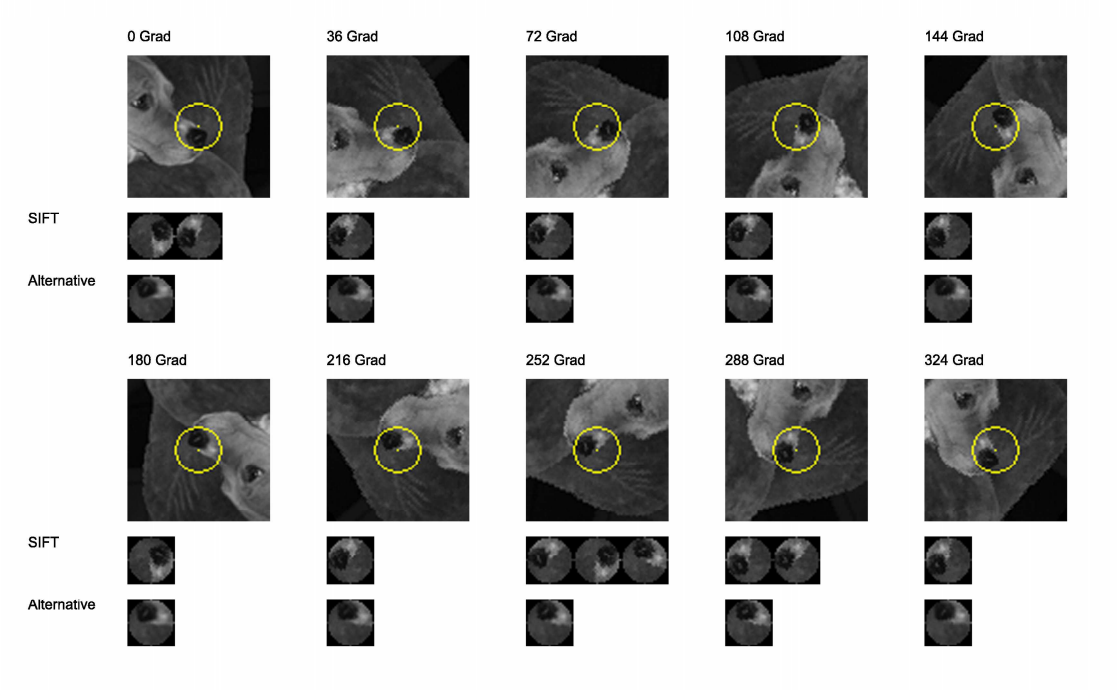


Abbildung 16: *Bestimmung der Hauptrichtung(en). Vergleich zwischen dem Ansatz von Lowe und der hier vorgeschlagenen Alternative.*

Die gesuchte Verschiebung von f, bzw. die gesuchte Drehung des lokalen Bereiches kann nun einfach durch $\Phi(1)$ berechnet werden, da $\Phi(1)$ die Phase der Frequenz enthält, deren Periodenlänge der Länge des untersuchten Bildbereiches entspricht. Versuche haben gezeigt, daß diese alternative Bestimmung der Drehung eines Bereiches deutlich stabilere Ergebnisse liefert als die von Lowe vorgeschlagene Methode. Desweiteren liefert sie immer nur *ein* Ergebnis. Einen Vergleich beider Verfahren zeigt Abbildung 16.

5.2.3 Erzeugen der Merkmalbeschreibung

Aufgrund der zuvor beschriebenen Vorverarbeitung muß die eigentliche Beschreibung der lokalen Umgebung nur noch invariant gegenüber Störungen, Helligkeitsschwankungen und Translationen sein. Wie schon beim Vorverarbeitungsschritt zur Rotationsinvarianz wird zunächst die im SIFT-Verfahren eingesetzte Merkmalbeschreibung erläutert und anschließend eine alternative Merkmalbeschreibung vorgestellt.

SIFT Methode Für die Merkmalbeschreibung wird zunächst der lokale Bereich in mehrere Unterbereiche aufgeteilt (z.B. 2x2 oder 4x4). Für jeden dieser Unterbereiche

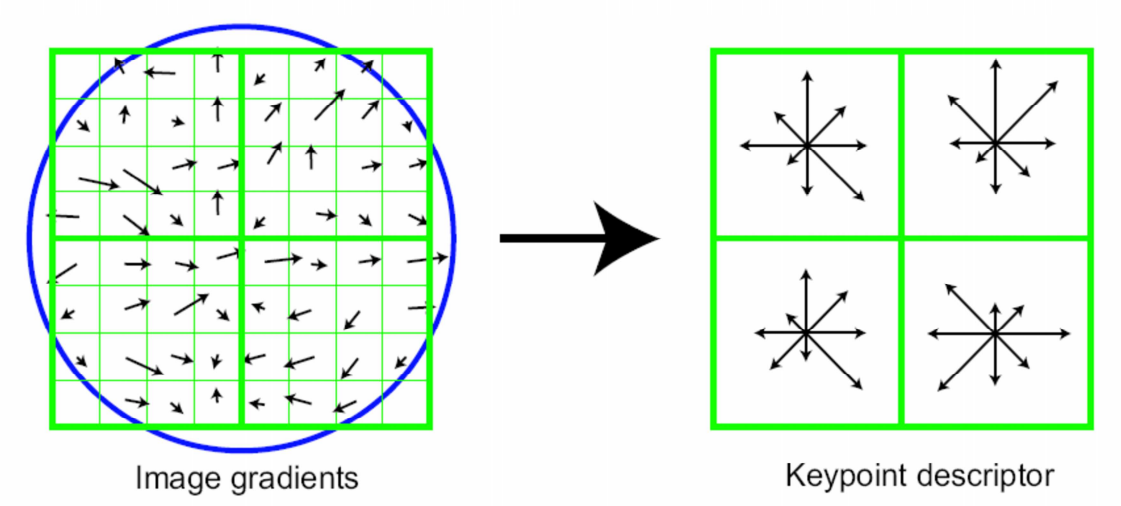


Abbildung 17: *SIFT-Deskriptor - Links: Aufteilung des lokalen Bildbereichs in 2 x 2 Unterbereiche. Die Pfeile repräsentieren Stärke und Richtung der Gradienten. Der blaue Kreis deutet die verwendete gaußsche Gewichtungsfunktion an. Rechts: Die für jeden Unterbereich erzeugten Histogramme. Jedes Bin wird über einen Pfeil für die entsprechende Größe und Richtung dargestellt. Quelle: [9]*

wird ein Richtungshistogramm erstellt. Dies geschieht äquivalent zur Erstellung des Richtungshistogramms für die Bestimmung der Hauptrichtung, mit $k = 8$ Bins. Die verwendete gaußsche Gewichtungsfunktion hat dabei ihr Zentrum nicht in der Mitte des jeweiligen Unterbereichs, sondern in der Mitte des gesamten lokalen Bereiches. Dadurch tragen die Gradienten an den Rändern der lokalen Umgebung nur noch gering zu den einzelnen Richtungshistogrammen bei. Auf diese Weise kann verhindert werden, daß eine kleine Änderung in der Position des lokalen Bereiches zu einer großen Veränderung in der Merkmalbeschreibung führt. Um abrupte Sprünge in den Histogrammen selbst zu verhindern, wird jeder Wert, der einem Histogramm hinzugefügt werden soll, mittels trilinearer Interpolation auf benachbarte Histogrammbins verteilt. Abbildung 17 zeigt die Histrogrammerzeugung für 2x2 Unterbereiche und 8 Bins pro Histogramm. Die konkrete Merkmalbeschreibung ist ein normalisierter Vektor, der die Werte aller durch eine Konstante nach oben beschränkten Histogrammbins enthält.

Die Begrenzung der Histogrammbins durch die Vorgabe eines Maximalwertes verringert den Einfluß von großen Gradienten und legt dadurch mehr Bedeutung auf die Richtungsverteilung der Gradienten. Die Normalisierung des Vektors und die

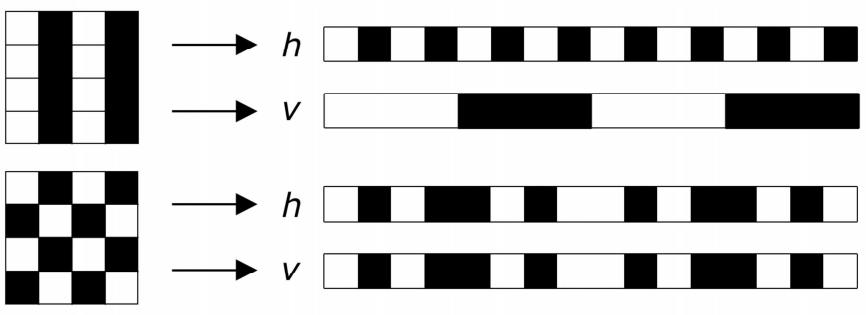


Abbildung 18: *Schematische Darstellung der Gradienten zweier lokaler Bereiche und die entsprechenden Funktionen h und v. In diesem Beispiel würde eine Beschränkung ausschließlich auf das Frequenzspektrum von h zu einer nicht ausreichenden Unterscheidung der Deskriptoren der beiden lokalen Bereiche führen.*

Verwendung von Gradienten anstelle der direkten Intensitätswerte sorgen für die Invarianz gegenüber Helligkeitsschwankungen. Eine Invarianz gegenüber Störungen wie etwa Bildrauschen besitzt der SIFT-Deskriptor nicht. Für Lowe stellt dies in [9] kein Problem dar, da der zu beschreibende lokale Bereich in einem früheren Schritt des SIFT-Verfahrens mit einem gaußschen Weichzeichner geglättet wurde und somit in diesem Fall nur geringes Bildrauschen zu erwarten ist. Die Invarianz gegenüber Translationen ist zum Teil gegeben. Solange die Bewegung des lokalen Bereiches im Vergleich zu der Seitenlänge eines Unterbereiches gering bleibt (i.d.R. 1-2 Pixel), verändern sich die resultierenden Merkmalvektoren nur kaum. Größere Bewegungen führen hingegen zu sehr unterschiedlichen Merkmalvektoren. Wie der Vergleich der verschiedene Keypoint-Detektoren gezeigt hat, tauchen bei einem Großteil der Keypoints, und damit auch bei einem Großteil der zugehörigen lokalen Bereiche, durchaus Verschiebungen von 5 und mehr Pixeln in Folge von Transformationen auf. Dies kann dazu führen, daß für ein und denselben markanten Punkt eines Objektes unter verschiedenen Transformationen sehr unterschiedliche Merkmalbeschreibungen erzeugt werden.

Alternative Die im Folgenden vorgestellte alternative Merkmalbeschreibung soll insbesondere bezüglich der Translationsinvarianz bessere Ergebnisse als der SIFT-Deskriptor liefern. Desweiteren kann die Toleranz gegenüber Bildrauschen und anderen feinen Strukturen über einen Parameter eingestellt werden. In einem ersten

Schritt werden die Gradienten G innerhalb des lokalen Bereiches I bestimmt:

$$G(x,y) = \max\left(\left|I(x,y) - I(x+1,y)\right|, \left|I(x,y) - I(x,y+1)\right|\right)$$

Mit den Gradienten G lassen sich zwei eindimensionale Funktionen h und v definieren, die einem zeilen- bzw. spaltenweisen Auslesen der Gradienten des lokalen Bereiches entsprechen:

$$h(n) = G(n \bmod b, n \operatorname{div} b)$$

$$v(m) = G(m \operatorname{div} l, m \bmod l)$$

mit l und b der Länge und Breite des lokalen Bereichs. Seien $H = H_r + iH_i$ und $V = V_r + iV_i$ die zugehörigen Fouriertransformationen der Funktionen h und v. Dann läßt sich für beide Funktionen das Frequenzspektrum berechnen:

$$|H(u)| = \sqrt{H_r^2(u) + H_i^2(u)}$$

$$|V(u)| = \sqrt{V_r^2(u) + V_i^2(u)}$$

Im Vergleich zum Phasenspektrum, das zuvor bei der Rotationsinvarianz Verwendung fand, beschreibt das Frequenzspektrum nur die Struktur des Signals, unabhängig von seiner Position. Das Frequenzspektrum gibt an, aus welchen überlagerten Frequenzen das untersuchte Signal besteht. Das Phasenspektrum gibt an, wie diese Frequenzen zueinander verschoben sind. Ignoriert man das Phasenspektrum, so liefert das Frequenzspektrum eine translationsinvariante Beschreibung des Signals. Um einen zweidimensionalen lokalen Bereich ausreichend unterscheidbar zu beschreiben, ist es notwendig, das Frequenzspektrum sowohl auf den Zeilen als auch auf den Spalten zu berechnen, wie Abbildung 18 verdeutlicht. Sei $N = l * b$ die Anzahl der Bildpunkte des lokalen Bereiches. Dann hat sowohl das horizontale wie das vertikale Frequenzspektrum N Werte. Die Anzahl der Werte, die aus den beiden Frequenzspektren in den Merkmalvektor übernommen werden, bestimmt dabei die Toleranz der Merkmalbeschreibung gegenüber Störungen bzw. feinen Strukturen. Sollen erst Frequenzen ab einer Periodenlänge von s Pixeln in die Merkmalbeschreibung einfließen, so reicht es, nur die ersten N/s Werte der Frequenzspektren in den Merkmalvektor zu übernehmen. Der Parameter s kann in der Regel immer ≥ 2 sein, da für Frequen-

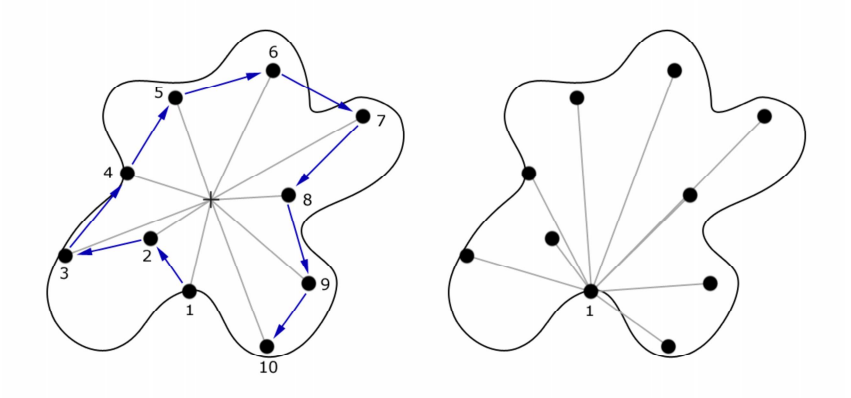


Abbildung 19: *Schematische Darstellung der Segmentbeschreibung. Die schwarzen Punkte repräsentieren Keypoints und die zugehörigen Merkmalvektoren. Links: Die Folge von Merkmalen ist durch blaue Pfeile dargestellt. Die Reihenfolge der Merkmale ist bestimmt durch die Ordnung auf den Winkeln der Geraden zum Mittelpunkt der Menge aller Merkmale. Rechts: Die relativen Winkel aller Merkmale zum ersten Merkmal der Folge. Dies entspricht der ersten Spalte der Matrix, die alle relativen Winkel der Merkmale untereinander enthält.*

zen mit einer Periode < 2 weniger als 2 Pixel pro Schwingung als Abtastwerte zur Verfügung stehen und dies entsprechend dem Abtasttheorem von Shannon für eine korrekte Rekonstruktion des Signals nicht ausreichend ist. Die in der zweiten Hälfte eines Spektrums vorgefundenen Werte sind demnach nicht die Werte besonders hoher Frequenzen, sondern die Werte von niedrigen Spiegelfrequenzen und tragen nicht zu einer besseren Beschreibung des lokalen Bereiches bei.

6 Segmentbeschreibung

Nachdem nun sowohl die Segmentierung des Bildes als auch die Merkmalvektoren interessanter Bildpunkte vorliegen, können die einzelnen Segmente des Bildes über Mengen von Merkmalvektoren beschrieben werden. Ein Segment wird dabei über zwei Dinge beschrieben. Zum einen über eine Folge von Merkmalvektoren, deren zugehörige Keypoints innerhalb des Segmentes liegen und zum anderen über eine Matrix, die die relativen Winkel dieser Keypoints zueinander enthält. Die Matrix repräsentiert auf eine redundante und skalierungs- und rotationsinvariante Weise die Form des Segmentes. Sie enthält an Stelle (i, j) den Winkel zwischen Keypoint i und Keypoint j bezüglich der Hauptrichtung des zu Keypoint i gehörenden Merkmals.

Die Hauptdiagonale der Matrix enthält Nullen. Die Folge von Merkmalvektoren beschreibt die Struktur des Segmentes. Die Reihenfolge der Merkmalvektoren wird anhand der Winkel zwischen der Horizontalen und den Geraden, die durch die zugehörigen Keypoints und den Mittelpunkt aller Keypoints des Segmentes gehen, bestimmt. Abbildung 19 zeigt eine schematische Darstellung der Segmentbeschreibung.

Für die statistische Auswertung der auftretenden Segmente bzw. Segmentbeschreibungen reicht es nicht aus, nur die exakte Gleichheit zweier Segmente überprüfen zu können. Die Segmente, die ein Objekt repräsentieren, weisen in der Regel von Bild zu Bild Veränderungen auf. So können zum Beispiel durch unterschiedliche Perspektiven oder die teilweise Verdeckung durch andere Objekte einige der Merkmale fehlen oder andere Merkmale hinzukommen. Auf der Ebene der Merkmale selbst führen schon kleine Veränderungen der lokalen Bereiche zu Veränderungen in den meist hochdimensionalen Merkmalvektoren. Ein Test auf Gleichheit der Merkmalvektoren hat dementsprechend nur einen geringen Nutzen, da er wohl in den seltensten Fällen positiv ausfallen wird. Ein geeignetes Ähnlichkeitsmaß muß derartige Veränderungen sowohl auf der Ebene der Segmentbeschreibung als auch auf der Ebene der Merkmale berücksichtigen.

Eine übliche Vorgehensweise, die Veränderung des Merkmalvektors aufgrund kleiner Veränderungen des lokalen Bereiches zu verhindern, ist der Einsatz von Dimensionsreduktionsverfahren. Die Principal Component Analysis, kurz PCA, ist hierbei sicherlich der bekannteste Vertreter. Die PCA identifiziert die Dimensionen der Merkmalvektoren, die den größten Informationsanteil tragen und reduziert die Merkmalvektoren auf genau diese Werte. Die Hoffnung ist nun, daß sich die leichten Veränderungen des lokalen Bereiches in genau den Dimensionen des ursprünglichen Merkmalvektors abspielen, die durch die PCA ausgeblendet wurden. In dem hier vorgestellten Verfahren soll jedoch nicht die PCA zum Einsatz kommen, auch wenn dies durchaus möglich wäre, sondern ein neuronaler Ansatz, mit dessen Hilfe sich die hochdimensionalen Merkmalvektoren quantisieren lassen. Desweiteren liefert dieses neuronale Verfahren eine Topologie der Merkmalvektoren, die in der Berechnung des Ähnlichkeitsmaßes der Segmente Verwendung findet.

Auf der Ebene der Segmentbeschreibung geht es zunächst darum, die Folgen von Merkmalen miteinander zu vergleichen. Betrachtet man die Segmente eines Objektes in unterschiedlichen Bildern, so können verschiedene Veränderungen in

den Merkmalfolgen des Segmentes auftreten. Es können neue Merkmale hinzukommen und bestehende Merkmale können wegfallen oder sich verändern. Desweiteren kann sich der Start der Merkmalfolge in Abhängigkeit zur Rotation des Objektes verändern, da sich die Reihenfolge der Merkmale über die absoluten Winkel zwischen der Horizontalen und den Geraden durch die Merkmale und dem Mittelpunkt der Merkmalmenge ergibt. Eine Analogie zu diesem Problem stellt der Vergleich von DNA-Sequenzen in der Bioinformatik dar (siehe [17]). Die DNA-Sequenzen können ebenfalls Insertionen, Deletionen und Mutationen erfahren. Ein Distanzmaß, das genau solche Veränderungen berücksichtigt, ist die Levensthein Distanz. Sie gibt an, wie viele Einfüge-, Lösch- und Editieroperationen mindestens nötig sind, um eine Sequenz in eine andere Sequenz zu überführen. Basierend auf der Levensthein Distanz berechnet der Smith-Waterman Algorithmus einen Wert für die Ähnlichkeit zweier Sequenzen. Der Algorithmus liefert zudem ein sogenanntes *Alignment*, eine Ausrichtung der beiden Sequenzen zueinander. Mit Hilfe dieser Ausrichtung kann schließlich auch ein Wert für die Ähnlichkeit der relativen Winkel der Merkmale zweier Segmente gefunden werden.

6.1 Quantisierung der Merkmale

Das für die Quantisierung der Merkmalvektoren eingesetze Verfahren ist ein wachsendes neuronales Gas (WNG), so wie es Bernd Fritzke in [16] vorstellt. Das WNG ähnelt in seiner Funktionsweise den Self Organizing Maps (SOM) von Teuvo Kohonen. Im Gegensatz zu den SOMs hat das WNG keine feste Anzahl an Neuronen. Es unterliegt einem datengesteuerten Wachstumsprozess, der beendet wird, wenn ein Haltekriterium erfüllt ist. Die Menge A der Neuronen des WNG wird mit zwei Neuronen initialisiert. Jedem der beiden Neuronen wird ein unterschiedlicher zufälliger Referenzvektor w zugeordnet. Desweiteren besitzt jedes Neuron $c \in A$ einen akkumulierten Fehler E_c, der mit 0 initialisiert wird. Die Kanten des WNG, die die Neuronen miteinander verbinden, besitzen ein Attribut „Alter“, daß es im Laufe des Wachstumsprozesses ermöglicht, nicht mehr benötigte Kanten zu identifizieren und zu löschen. Die Menge der Kanten zwischen den Neuronen $C, C \subseteq A \times A$ ist zu Beginn leer. Die Quantisierung einer Eingabe ξ verläuft nun folgendermaßen:
Es werden die Neuronen s_1 und s_2 gesucht, deren Referenzvektoren w_{s_1} und w_{s_2} der Eingabe ξ am nächsten bzw. zweitnächsten sind. Falls noch keine Verbindung

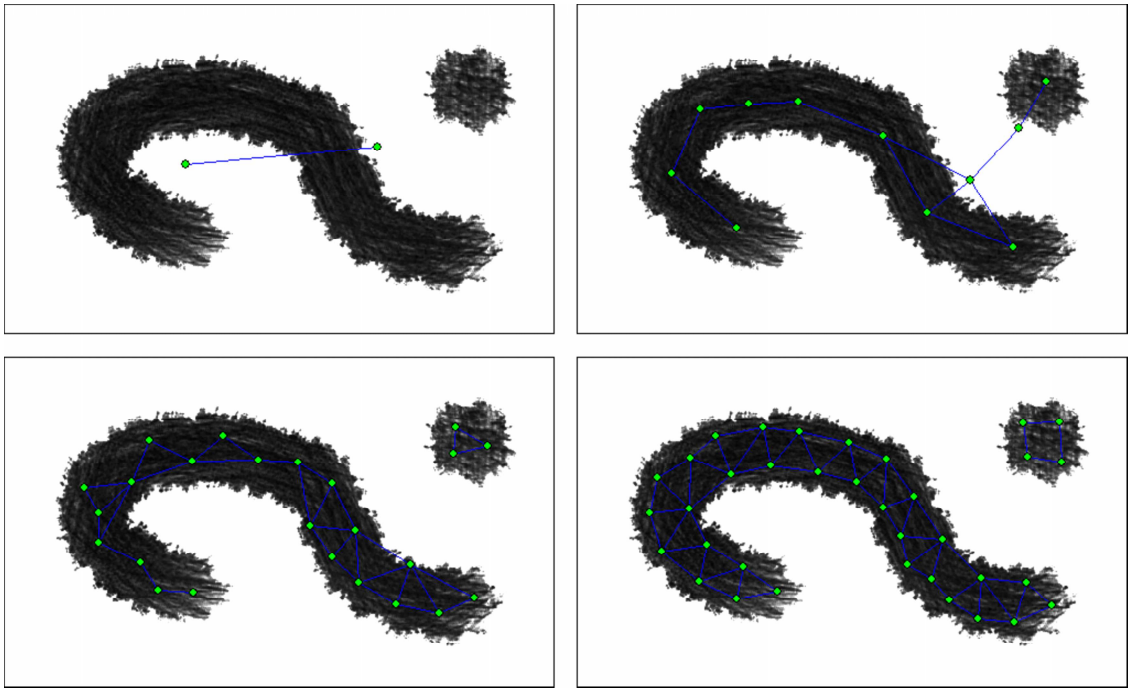


Abbildung 20: *Ein wachsendes neuronales Gas zu verschiedenen Zeitpunkten. Die dunklen Bildbereiche stellen die Wahrscheinlichkeitsverteilung der Eingabe dar. Man erkennt deutlich, wie die Neuronen (grün) des neuronalen Gases die Wahrscheinlichkeitsverteilung der Eingabe nachbilden und zugleich eine entsprechende Topologie (blau) aufbauen.*

zwischen s_1 und s_2 in C exisitert, wird eine entsprechende Kante zu C hinzugefügt:

$$C := C \cup \{(s_1, s_2)\} \ .$$

Das Alter der Kante zwischen s_1 und s_2 wird auf 0 gesetzt. Dadurch wird die Kante „aufgefrischt“. Zum akkumulierten Fehler E_{s_1} des Gewinnerneurons wird der quadratische Abstand zwischen Eingabe und Referenzvektor von s_1 hinzugefügt:

$$\Delta E_{s_1} := \|\xi - w_{s_1}\|^2.$$

In einem Adaptionsschritt wird der Referenzvektor von s_1 und werden alle Referenzvektoren der direkten Nachbarn von s_1 angepaßt:

$$\Delta w_{s_1} := \epsilon_b (\xi - w_{s_1})$$

$$\Delta w_c := \epsilon_n (\xi - w_c), \quad \forall c \in N_{s_1}$$

mit

$$N_{s_1} := \{c | (s_1, c) \in C, \quad \forall c \in A\} \ .$$

Für die Parameter ϵ_b und ϵ_n schlägt Fritzke 0.05 bzw. 0.0006 vor. Eigene Tests mit dem in Abbildung 47 gezeigten Testprogramm konnten diese Werte als geeignete Werte bestätigen. Die Parameter ϵ bestimmen in gewisser Weise die „Trägheit“ des neuronalen Gases.
Bei allen Kanten, deren Endpunkte sich im vorherigen Adaptionsschritt „bewegt“ haben, wird das Alter erhöht:

$$age(s_1, c) := age(s_1, c) + 1, \quad \forall c \in N_{s_1} \ .$$

Überschreitet das Alter einer Kante den Grenzwert a_{max}, so wird diese Kante gelöscht. Führt dies dazu, daß ein Neuron zu keiner Kante mehr inzident ist, wird dieses Neuron ebenfalls gelöscht. Für den Parameter a_{max} empfehlen sich Werte von etwa 100. Ein zu geringer Wert führt zu einer instabilen Topologie. Ein zu hoher Wert führt dazu, daß sich „eigenständige“ Gebiete des Eingaberaums nur langsam nach einer sehr hohen Anzahl Eingaben abtrennen. Ein derartiges Gebiet zeigt Abbildung 20 oben rechts.
Erreicht die Anzahl der bisherigen Eingaben ein ganzzahliges Vielfaches eines Para-

meters λ und ist das Haltekriterium für den Wachstumsprozess noch nicht erfüllt, so wird ein neues Neuron r dem WNG hinzugefügt. Der Parameter λ sollte nicht zu klein gewählt werden, da es sonst relativ lange dauert, bis ein Eingaberaum durch das WNG vollständig abgedeckt wird und es zudem deutlich mehr Neuronen für diese Abdeckung bedarf. Ein Wert von $\lambda = 300$ hat sich in Tests (s. Abbildung 21) als vernünftige Wahl erwiesen. Um eine neue Einheit r hinzuzufügen, wird das Neuron $q \in A$ bestimmt, das momentan den maximalen akkumulierten Fehler E_q besitzt. Desweiteren wird das Neuron $f \in N_q$ mit maximalem akkumuliertem Fehler bestimmt:

$$q := \operatorname{argmax}\{E_c |\ c \in A\}$$

$$f := \operatorname{argmax}\{E_c |\ c \in N_q\}.$$

Das neue Neuron r wird der Menge A hinzugefügt und erhält einen aus w_q und w_f interpolierten Referenzvektor:

$$A := A \cup \{r\}$$

$$w_r := (w_q + w_f)/2\ .$$

Der akkumulierte Fehler des neuen Neurons r wird aus den zuvor verringerten Fehlern von q und f gebildet:

$$\Delta E_q := -\alpha E_q$$

$$\Delta E_f := -\alpha E_f$$

$$E_r := (E_q + E_f)/2\ .$$

Der Parameter α charakterisiert, wie gut ein neues Neuron in der Lage ist, den akkumulierten Fehler zu reduzieren. Geringe Werte für α führen dazu, daß in Gebieten mit hohem akkumuliertem Fehler sehr viele neue Neuronen in Folge hinzugefügt werden und sich der akkumulierte Fehler nur langsam abbaut. Im Gegensatz dazu führen hohe Werte für α zu einem schnellen Abbau des akkumulierten Fehlers in einem Gebiet. Dies kann zur Folge haben, daß auch in Gebieten mit geringem Quantisierungsfehler zusätzliche Neuronen hinzugefügt werden. Der von Fritzke für α vorgeschlagene Wert ist 0.5.
Die Kantenmenge C wird um eine Kante zwischen r und q und eine Kante zwischen

r und f erweitert. Die Kante zwischen q und f wird aus der Menge C entfernt:

$$C := (C \cup \{(r,q),(r,f)\}) \setminus \{(q,f)\} \ .$$

In einem letzten Schritt werden die akkumulierten Fehler aller Neuronen um einen Bruchteil β reduziert:

$$\Delta E_c := -\beta E_c, \quad \forall c \in A \ .$$

Dieser letzte Schritt simuliert eine Art Altern der Quantisierungsfehler. Es verleiht jüngeren Quantisierungsfehlern mehr Gewicht und verhindert, daß über die Zeit angehäufte kleine Quantisierungsfehler eine zu große Bedeutung erlangen. Die Größe des Wertes β bestimmt, wie gut feine Strukturen des Eingaberaumes durch das WNG angenähert werden. Ein Wert für β, der sich in Tests (s. Abbildung 21) als gutes Mittelmaß zwischen Nachbildung feiner Strukturen und geringer Anzahl Neuronen erwiesen hat, ist 0.0005. Dies deckt sich mit dem Wert, den Fritzke in [16] angibt.

Als Ergebnis für die Quantisierung der Eingabe ξ kann der Referenzvektor oder auch nur der Index des Gewinnerneurons s_1 zurückgegeben werden. In dem hier vorgestellten Verfahren wird der Index des Gewinnerneurons verwendet. Die Folge der Merkmale bei der Segmentbeschreibung wird also durch die Quantisierung zu einer Folge der Indizes der entsprechenden Neuronen.

6.1.1 Bisherige Haltekriterien

Als Haltekriterien für den Wachstumsprozess stellt Fritzke in [16] zwei Kriterien vor. Zum einen eine maximale Anzahl an Neuronen, zum anderen einen zu erreichenden minimalen Quantisierungsfehler. Beide Kriterien eignen sich nur bedingt für das hier vorgestellte Verfahren. Die Verwendung einer maximalen Anzahl an Neuronen als Haltekriterium würde es erfordern, im voraus die Zahl der benötigten Neuronen zu bestimmen. Ist die Anzahl zu gering, sind die Quantisierungsfehler zu groß und es kann passieren, daß auf diese Weise unterschiedliche Segmente fälschlicherweise als gleich oder ähnlich angenommen werden. Ist die Anzahl zu groß, werden statistische Häufigkeiten in gewisser Weise „in die Breite gezogen“, da für im Grunde gleiche Segmente viele verschiedene Segmentbeschreibungen erzeugt werden.
Die Verwendung eines zu erreichenden minimalen Quantisierungsfehlers ist in dieser Hinsicht schon flexibler und besser geeignet, jedoch erfordert der Einsatz dieses Hal-

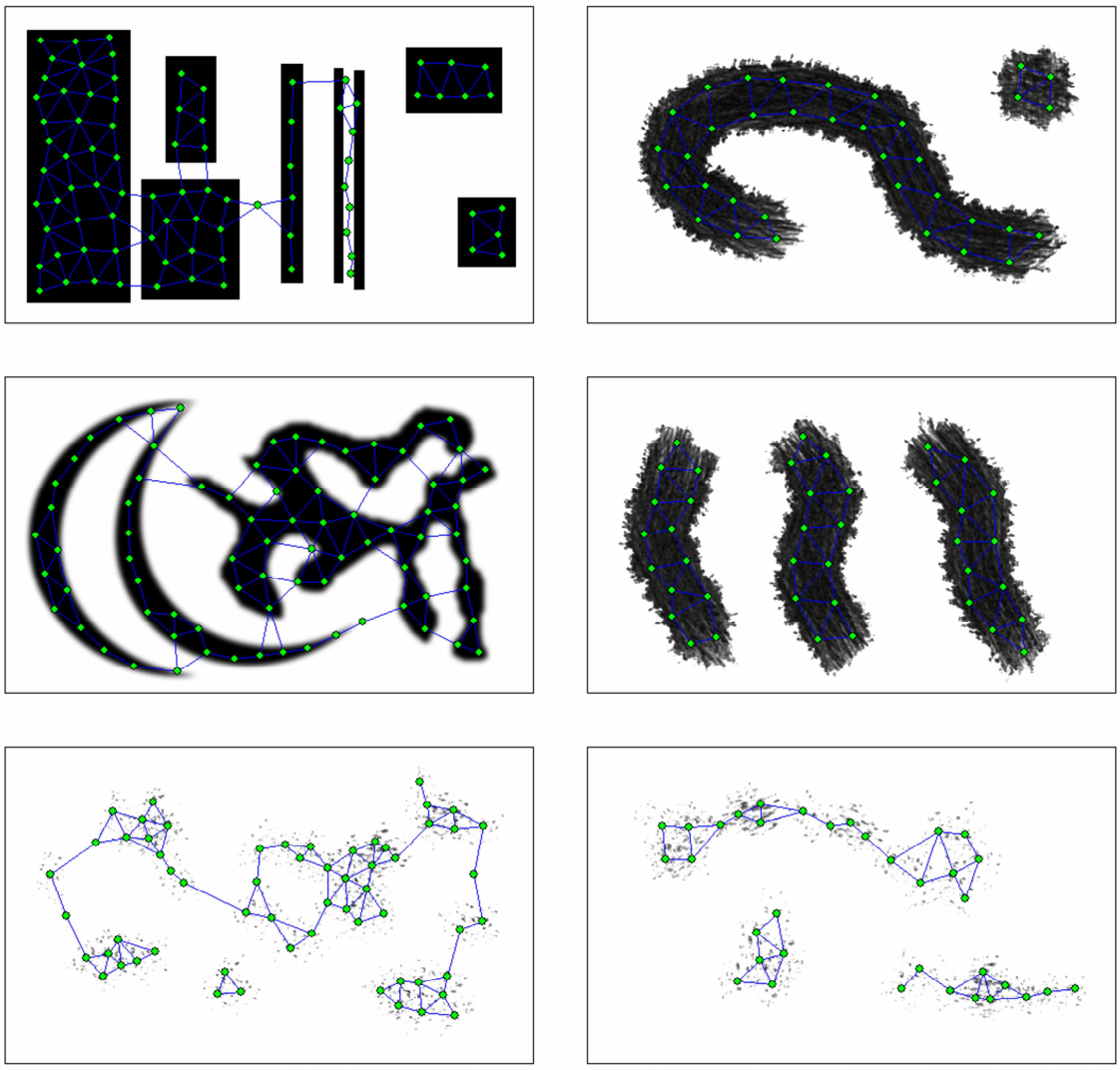


Abbildung 21: *Verschiedene Testdaten und ihre Nachbildung durch das WNG.*

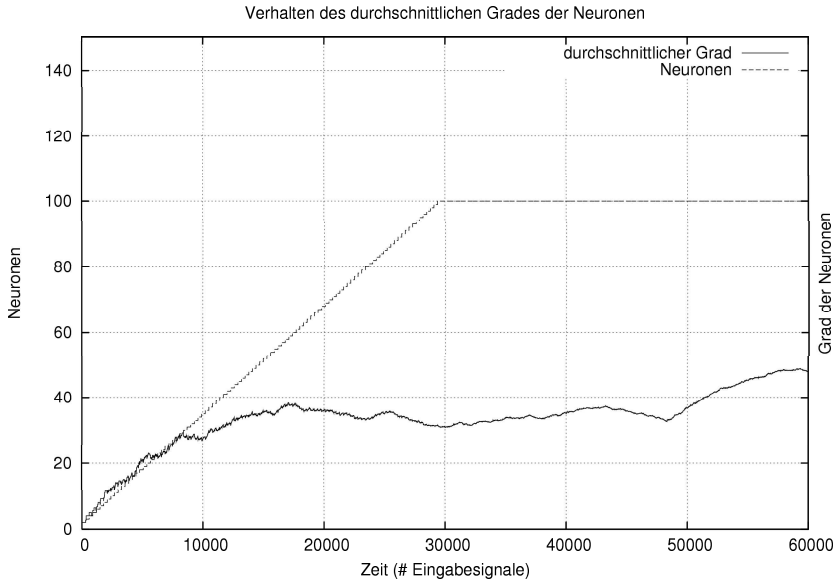


Abbildung 22: *Entwicklung des durchschnittlichen Grades der Neuronen im WNG. Nach etwa 48.000 Eingabesignalen wurden deutlich verschiedene Merkmalvektoren eingegeben. Der entstehenden „Knick" ist klar zu erkennen.*

tekriteriums ein Vorwissen über die Struktur des Eingaberaumes. Die euklidischen Abstände der Merkmalvektoren lassen aber leider keinen Schluß zu, ab welcher Distanz zwei Merkmalvektoren als gleich angesehen werden können. Wie schon Lowe in [9] über den SIFT-Deskriptor berichtet, kann ein und derselbe Abstand zwischen zwei sehr ähnlichen Merkmalen, wie auch bei zwei vollkommen unterschiedlichen Merkmalen auftreten. Der hier vorgestellte alternative Deskriptor zeigt zum Teil ähnliche Eigenschaften, wenn auch nicht so ausgeprägt wie der SIFT-Deskriptor.

6.1.2 Alternatives Haltekriterium

Eine Erweiterung des ersten Kriteriums von Fritzke, der maximalen Anzahl an Neuronen, ermöglicht es, ein Haltekriterium zu definieren, daß ohne Vorwissen über die Struktur des Eingaberaumes auskommt. Die Idee besteht darin, zunächst nur eine geringe Anzahl an Neuronen zu erlauben. Wird diese Anzahl vom WNG erreicht, wird fortan geprüft, ob diese Anzahl ausreichend ist. Fällt dieser Test irgendwann negativ aus, wird die Anzahl der erlaubten Neuronen um einen konstanten Wert erhöht und die weitere Prüfung bis zum erneuten Erreichen der maximalen Anzahl an Neuronen eingestellt. Diese Vorgehensweise erlaubt es dem WNG „je nach Be-

darf" das Wachstum einzustellen oder wieder zu aktivieren. Werden beispielsweise über einen längeren Zeitraum immer die gleichen oder sehr ähnliche Merkmalvektoren in das WNG eingegeben, so besteht kein Bedarf an einem zusätzlichen Wachstum des WNG. Ändert sich die Eingabe dann plötzlich aufgrund neuer und andersartiger Merkmalvektoren, die bislang nicht gut durch das WNG quantisiert werden können, so sollte der Wachstumsprozess des WNG erneut einsetzen, um die Quantisierung dieser Merkmalvektoren zu verbessern. Das Kernproblem dieser Idee besteht darin, auf welche Weise man prüfen kann, ob ein gegebenes WNG eine ausreichende Anzahl an Neuronen besitzt.
Hat ein WNG einmal die maximale Anzahl an Neuronen erreicht, so kann es für neue Eingaben nur noch die bestehenden Neuronen verwenden. Weichen die neuen Eingaben sehr stark von den bisherigen Eingaben ab, stammen sie also aus einem Teil des Eingaberaumes, der bislang nicht durch das WNG abgedeckt wurde, so muß das WNG bestehende Neuronen für die Abdeckung dieses neuen Teils des Eingaberaumes verlagern. Die gleiche Anzahl Neuronen muß nun einen größeren Bereich im Eingaberaum abdecken. Dies führt dazu, daß sich die potentielle Nachbarschaft jedes Neurons vergrößert und somit auch der durchschnittliche Grad der Neuronen. Der plötzliche Anstieg des durchschnittlichen Grades der Neuronen ist ein Indikator dafür, daß neue und andersartige Eingaben vom WNG verarbeitet werden müssen, die es mit der bisherigen Anzahl an Neuronen nur ungenügend verarbeiten kann. Abbildung 22 zeigt einen solchen plötzlichen Anstieg. Die Veränderung des durschnittliches Grades kann somit als Maß für die Stabilität des WNG betrachtet werden. Das auf diese Weise konstruierte Haltekriterium kommt ohne ein Vorwissen über die Struktur des Eingaberaumes aus. Es orientiert sich ausschließlich am Verhalten des WNG gegenüber neuen Eingaben.

6.2 Levensthein Distanz

In den häufigsten Fällen, in denen die Distanz zweier Folgen bzw. Vektoren $x := x_1, x_2, \ldots, x_n$ und $y := y_1, y_2, \ldots, y_n$ bestimmt werden soll, reicht eine einfache komponentenweise berechnete Metrik aus. Ein Beispiel für eine derartige Metrik ist die Minkowski-Distanz:

$$D_\lambda(x, y) := \left(\sum_{i=1}^{n} |x_i - y_i|^\lambda \right)^{1/\lambda}.$$

Für $\lambda = 1$ ergibt sich die Manhattan-Distanz, für $\lambda = 2$ der euklidische Abstand. Metriken dieser Art bewerten nur die lokale Veränderung der einzelnen Elemente einer Folge. Zudem können nur gleich lange Folgen miteinander veglichen werden, was u.U. einen Vorverarbeitungsschritt für die Angleichung der Längen erforderlich macht. Aufgrund dieser Einschränkungen sind einfache Metriken nicht für die Merkmalfolgen der hier vorgestellten Segmentbeschreibung geeignet. Die Merkmalfolgen zweier ähnlicher Segmente unterscheiden sich nicht nur durch die Veränderung einzelner Merkmale, sondern auch durch das Fehlen einiger Merkmale oder die Existenz neuer Merkmale. In diesem Sinne gleichen die Merkmalfolgen DNA-Sequenzen, bei denen durch Mutation und Vererbung ähnliche Veränderung auftreten. Ein auf DNA-Sequenzen anwendbares Distanzmaß ist die Levensthein Distanz (siehe [17]). Sie beschreibt den Unterschied zwischen zwei Sequenzen über die minimalen Kosten der Editieroperationen, die nötig sind, um die eine Sequenz in die andere zu überführen. Die hierbei betrachteten Editieroperationen sind das Einfügen (E), Löschen (L) und Ändern (Ä) von Zeichen der Sequenzen. Um beispielsweise aus dem Wort `HAUS` das Wort `HOSE` zu machen, sind u.a. folgende Umwandlungen möglich:

```
H A U S -        H A U S -        H A U S
H O - S E        H - O S E        H O S E
---------        ---------        -------
  Ä L   E          L Ä   E          Ä Ä Ä
```

Für die hier verwendete Darstellung der Umwandlungen müssen die beiden Sequenzen zunächst auf eine gleiche Länge gebracht werden, indem u.U. an geeigneten Stellen Leerzeichen `"-"` in die Sequenzen eingefügt werden. Die Kosten jeder Umwandlung ergeben sich dann als Summe der Kosten für die einzelnen Editieroperationen. Im ersten Beispiel wären dies die Kosten für eine Änderungs-, eine Lösch- und eine Einfügeoperation. Formal läßt sich die Levensthein Distanz folgendermaßen beschreiben: Seien $x := x_1, x_2, \ldots, x_n$ und $y := y_1, y_2, \ldots, y_m$ die beiden betrachteten Sequenzen mit Zeichen aus der Menge Σ und seien $\overline{x} := \overline{x}_1, \overline{x}_2, \ldots, \overline{x}_k$ und $\overline{y} := \overline{y}_1, \overline{y}_2, \ldots, \overline{y}_k$ die beiden Sequenzen mit Zeichen aus der Menge $\Sigma' := \Sigma \cup \{\epsilon\}$, die aus x und y hervorgeganen sind, nachdem diese auf eine gleiche Länge k durch das Einfügen von jeweils $k - n$ bzw. $k - m$ Leerstellen ϵ an geeigneten Stellen

gebracht wurden. Dann ist die Levensthein Distanz als

$$D := \sum_{i=1}^{k} d\left(\overline{x}_i, \overline{y}_i\right)$$

mit dem Distanzmaß d

$$\begin{array}{ll} d\left(\epsilon, a\right), & \text{Kosten für das Einfügen eines Zeichens} \\ d\left(a, \epsilon\right), & \text{Kosten für das Löschen eines Zeichens} \\ d\left(a, b\right), & \text{Kosten für das Ändern eines Zeichens} \\ a, b, \epsilon \in \Sigma' & \end{array}$$

definiert. Offen bleibt bei dieser Definition, wie die „geeigneten Stellen“ bestimmt werden können, an denen Leerzeichen eingefügt werden müssen, so daß die Kosten minimal werden. Für die Berechnung einer solchen kostenminimalen Ausrichtung der beiden Sequenzen x und y zueinander existiert ein auf dynamischer Programmierung basierender Algorithmus: Seien die beiden Teilsequenzen $x_1, x_2, \ldots, x_i$ und $y_1, y_2, \ldots, y_j$ bereits kostenminimal zueinander ausgerichtet. Diese Ausrichtung kann auf drei mögliche Zeichenpaare enden: (x_i, ϵ), (x_i, y_j) oder (ϵ, y_j).

- Endet die Ausrichtung auf das Paar (x_i, ϵ), ergeben sich die Kosten der Ausrichtung als Summe der Kosten der Ausrichtung von $x_1, x_2, \ldots, x_{i-1}$ und $y_1, y_2, \ldots, y_j$ und $d\left(x_i, \epsilon\right)$.
- Endet die Ausrichtung auf das Paar (x_i, y_j), ergeben sich die Kosten der Ausrichtung als Summe der Kosten der Ausrichtung von $x_1, x_2, \ldots, x_{i-1}$ und $y_1, y_2, \ldots, y_{j-1}$ und $d\left(x_i, y_j\right)$.
- Endet die Ausrichtung auf das Paar (ϵ, y_j), ergeben sich die Kosten der Ausrichtung als Summe der Kosten der Ausrichtung von $x_1, x_2, \ldots, x_i$ und $y_1, y_2, \ldots, y_{j-1}$ und $d\left(\epsilon, y_j\right)$.

Auf diese Weise können die minimalen Kosten für die Ausrichtungen aller möglichen Paare von Teilsequenzen

$$\left\{\left(\left(x_1, x_2, \ldots, x_i\right), \left(y_1, y_2, \ldots, y_j\right)\right) \mid \; 1 \leq i \leq n, \quad 1 \leq j \leq m\right\}$$

sukzessive berechnet werden. Die Zwischenergebnisse werden dabei in einem Array der Größe $(n+1)(m+1)$ gespeichert. Die erste Zeile und die erste Spalte des Ar-

rays werden mit den Kosten für das Einfügen von Leerstellen vor der Sequenz x bzw. vor der Sequenz y initialisiert. Das Ergebnis, die minimalen Kosten der Ausrichtung von x und y zueinander, steht am Ende des Algorithmus in Zelle (n, m) des Arrays. Die Ausrichtung der Sequenzen kann durch ein Backtracking rekonstruiert werden, wenn während der Berechnung gespeichert wird, welches Teilsequenzenpaar Vorgänger der jeweiligen Zelle ist. In Pseudocode kann der Algorithmus wie folgt beschrieben werden:

```
// Eingabe: Die Sequenzen x und y als Arrays [1..n] bzw. [1..m].
            Die Kostenfunktion d(a,b), die die Kosten für das
            Ausrichten der Zeichen a und b zueinander liefert.

// Ausgabe: Die Levensthein Distanz D(x,y).

e := ' '; // epsilon, das Zeichen für eine Leerstelle

// Initialisierung des Arrays D [0..n,0..m] :
D[0,0] := 0;
for i := 1 to n do
  D[i,0] := D[i-1,0] + d(x[i],e);
for j := 1 to m do
  D[0,j] := D[0,j-1] + d(e,y[i]);

// Berechnung einer kostenminimalen Ausrichtung:
for i := 1 to n do
  for j := 1 to m do begin
    k1     := D[i-1,j  ] + d(x[i],e);
    k2     := D[i-1,j-1] + d(x[i],y[j]);
    k3     := D[i  ,j-1] + d(e   ,y[j]);
    D[i,j] := min(k1,k2,k3);
  end;

result := D[n,m];
```

6.3 Smith-Waterman Algorithmus

Anstelle der Distanz zweier Sequenzen bzw. der Kosten für die Umwandlung der einen Sequenz in die andere Sequenz kann mit nur wenigen Änderungen am zuvor beschriebenen Algorithmus auch ein Wert für die Ähnlichkeit zweier Sequenzen be-

stimmt werden. Im Gegensatz zur Distanz nimmt der Wert für die Ähnlichkeit zweier Sequenzen zu, desto mehr sich die Sequenzen gleichen. Um mit dem beschriebenen Algorithmus einen Ähnlichkeitswert anstelle einer Distanz zu berechnen, tritt lediglich an die Stelle der Kostenfunktion d eine Scoringfunktion s. Je ähnlicher sich zwei Elemente $a, b \in \Sigma'$ sind, desto höher ist auch der Score $s(a, b)$. Die Scoringfunktion s liefert positive Werte, solange sich zwei Elemente hinreichend ähnlich sind und liefert negative Werte, wenn sich die Elemente unähnlich sind. Dadurch wird das Ausrichten ähnlicher Elemente zueinander belohnt und das Ausrichten unähnlicher Elemente zueinander bestraft. Anstelle der Minimumberechnung in der Hauptschleife des Algorithmus wird eine Maximumberechnung durchgeführt:

```
...
// Berechnung einer Ausrichtung mit maximalem Score:
for i := 1 to n do
  for j := 1 to m do begin
    k1     := S[i-1,j  ] + s(x[i],e);
    k2     := S[i-1,j-1] + s(x[i],y[j]);
    k3     := S[i  ,j-1] + s(e   ,y[j]);
    S[i,j] := max(k1,k2,k3);
  end;
...
```

Der bis jetzt vorgestellte Algorithmus berechnet eine globale Ausrichtung zweier Sequenzen. Bei Sequenzen, die sich stark in ihrer Länge unterscheiden, führt ein globales Ausrichten jedoch dazu, daß die kleinere der beiden Sequenzen über die Länge der großen Sequenz „verschmiert“ wird. Man ist daher eher an einem lokalen Sequenzalignment interessiert, daß die Struktur der kleineren Sequenz in einem gewissen Maße erhält. Der beschriebene Algorithmus kann auf einfache Weise geändert werden, so daß er eine lokale Ausrichtung anstelle einer globalen Ausrichtung berechnet:

```
// Eingabe: Die Sequenzen x und y als Arrays [1..n] bzw. [1..m].
            Die Scoringfunktion s(a,b), die die Ähnlichkeit der
            Zeichen a und b bewertet.
```

```
// Ausgabe: Ein Ähnlichkeitswert für die Sequenzen x und y.

e := ' '; // epsilon, das Zeichen für eine Leerstelle

// Initialisierung des Arrays S [0..n,0..m] :
S[0,0] := 0;
for i := 1 to n do
  S[i,0] := 0;
for j := 1 to m do
  S[0,j] := 0;

// Berechnung einer lokalen Ausrichtung mit maximalem Score:
for i := 1 to n do
  for j := 1 to m do begin
    k1     := S[i-1,j  ] + s(x[i],e);
    k2     := S[i-1,j-1] + s(x[i],y[j]);
    k3     := S[i  ,j-1] + s(e   ,y[j]);
    S[i,j] := max(k1,k2,k3);
  end;

result := Max(S);
```

Aufgrund der Initialisierung der ersten Zeile und ersten Spalte mit „0“, hat nun jede Startposition einer Ausrichtung die gleiche Chance der Beginn einer Ausrichtung mit maximalem Score zu werden. Das Einfügen von Leerzeichen vor einer der beiden Sequenzen wird somit nicht mehr bestraft. Durch diese Änderung steht der maximale Score nun nicht mehr zwingend in Zelle `S[n,m]`. Ausgehend von der Zelle mit maximalem Score, die das Ende der lokalen Ausrichtung markiert, kann mittels eines Backtrackings bis zur ersten Zelle mit Wert 0 die lokale Ausrichtung rekonstruiert werden.

6.4 Berechnung der Ähnlichkeit zweier Segmente

Der im letzten Abschnitt beschriebene Algorithmus kann nun dazu verwendet werden, die Ähnlichkeit zweier Merkmalfolgen zu berechnen. Zu diesem Zweck muß zunächst eine geeignete Scoringfunktion für die Merkmale entworfen werden. Eine Funktion, die auf der euklidischen Distanz zwischen den Merkmalvektoren aufbaut, ist problematisch. Wie schon in 6.1.1 erläutert, verlangt die Verwendung eines ein-

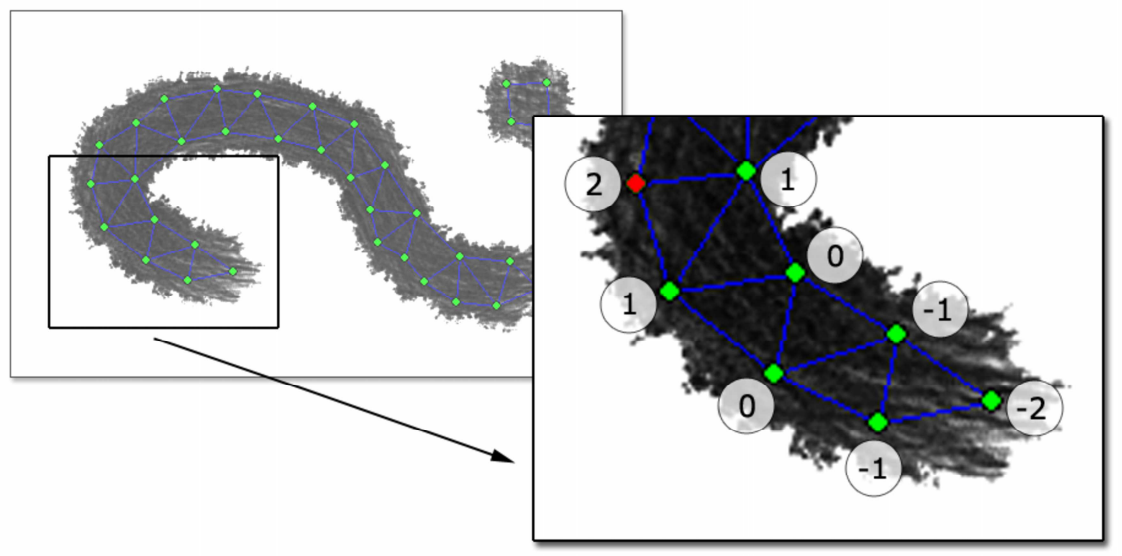
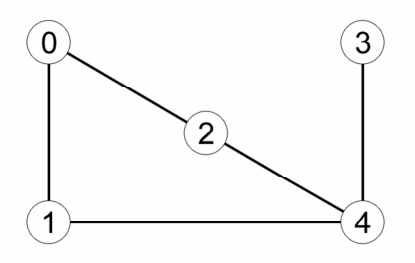


Abbildung 23: *Ähnlichkeitswerte des rot gefärbten Neurons zu den Neuronen der Umgebung (maximaler Score mxs := 2).*

fachen Distanzmaßes gewisse Vorkenntnisse über die Struktur des Merkmalraumes. Falsche Annahmen, z.B. über die Gleichförmigkeit des Merkmalraumes, können dazu führen, daß unterschiedliche Merkmale fälschlicherweise einen hohen Score bekommen, während ähnliche Merkmale nur einen geringen Score erhalten.
Anstelle eines einfachen Distanzmaßes kann die vom WNG erzeugte Topologie als Basis für eine Scoringfunktion dienen. Die erzeugte Topologie ist ein Graph, dessen Knoten den Neuronen des WNG entsprechen und dessen Kanten benachbarte und somit ähnliche Neuronen miteinander verbinden. Ziel ist es, mit Hilfe dieser Topologie des Merkmalraumes eine Scoringmatrix aufzubauen, aus der die Werte für $s(a, b)$ entnommen werden können. Die Ähnlichkeit zweier Merkmale wird dabei über die Differenz eines maximalen Scores `mxs` und der Länge des kürzesten Weges zwischen den Knoten der entsprechenden Neuronen bestimmt. Abbildung 23 zeigt exemplarisch die Ähnlichkeitswerte eines Neurons bzgl. der Neuronen seiner Umgebung. Um eine vollständige Scoringmatrix zu erhalten, müssen für jeden Knoten die kürzesten Wege zu allen anderen Knoten bestimmt werden. Dabei ist es möglich, den Rechenaufwand zu reduzieren, indem nur Wege bis zu einer Länge von $2mxs$ betrachtet werden und alle längeren Wege den konstanten Score $-(2mxs + 1)$ erhalten. Der folgende Algorithmus zur Berechnung der Scoringmatrix verwendet diese Einschränkung. Der Algorithmus nutzt den Umstand, daß über die n-te Potenz

der Adjazenzmatrix eines Graphen für jeden Knoten genau die Nachfolgeknoten bestimmt werden können, die n Kanten von dem entsprechenden Knoten entfernt sind. Ein Beispiel:



Die Adjazenzmatrix A dieses einfachen Graphen beschreibt in jeder Spalte $\vec{a}_i$ die direkten Nachbarknoten des Knoten i:

$$\begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{pmatrix}$$

Die Multiplikation eines Spaltenvektors $\vec{b} := (b_1, b_2, \ldots, b_n)^T$ mit einer Matrix A kann als eine Linearkombination der Spalten $\vec{a}_i$ von A verstanden werden:

$$A\vec{b} = b_1\vec{a}_1 + b_2\vec{a}_2 + \ldots + b_n\vec{a}_n.$$

Die Multiplikation zweier Matrizen C und D kann spaltenweise als Multiplikation der Spalten $\vec{d}_i$ von D mit der Matrix C erfolgen:

$$CD = \begin{pmatrix} C\vec{d}_1 & C\vec{d}_2 & \ldots & C\vec{d}_n \end{pmatrix}.$$

Die Quadrierung $A' = AA$ der Adjazenzmatrix A enthält also in jeder Spalte $\vec{a'}_i$ eine Linearkombination der Spalten $\vec{a}_j$ von A:

$$A' = \begin{pmatrix} \vec{a'}_1 = A\vec{a}_1 & \vec{a'}_2 = A\vec{a}_2 & \dots & \vec{a'}_n = A\vec{a}_n \end{pmatrix}.$$

Da jede Spalte $\vec{a}_i$ der Adjazenzmatrix A die Nachbarknoten des Knoten i beschreibt, beschreibt jede Spalte $\vec{a'}_i$ von A' alle Nachbarknoten der Nachbarknoten des Knoten i oder anders ausgedrückt die Knoten, die von Knoten i aus auf Wegen der Länge 2 erreichbar sind. Für das Beispiel berechnet sich die erste Spalte $\vec{a'}_1$ von A' durch:

$$\vec{a'}_1 = 0 * \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \end{pmatrix} + 1 * \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} + 1 * \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} + 0 * \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} + 0 * \begin{pmatrix} 0 \\ 1 \\ 1 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 2 \\ 0 \\ 0 \\ 0 \\ 2 \end{pmatrix}.$$

Die Werte in jeder Zelle a'_{ij} der Matrix A' geben an, wie viele Wege der Länge 2 es von Knoten i zu Knoten j gibt.
Eine weitere Multiplikation von A' und A ergibt entsprechend eine Matrix $A'' = A'A = AAA$, die in ihren einzelnen Spalten $\vec{a''}_i$ beschreibt, welche Knoten auf Wegen der Länge 3 vom jeweiligen Knoten i aus erreichbar sind. Allgemein beschreibt die n-te Potenz einer Adjazenzmatrix welche Knoten über Wege der Länge n miteinander verbunden sind.
Für den hier verwendeten Zweck, die Erstellung einer Scoringmatrix, sind die konkreten Werte innerhalb der potenzierten Adjazenzmatrix nicht von Interesse. Es genügt zu wissen, ob ein Weg der Länge n zwischen zwei Knoten existiert. Die genaue Anzahl der Wege zwischen diesen Knoten interessiert nicht. Aus diesem Grund reicht es aus, mit boolschen Werten zu arbeiten und die Addition durch die Oder-Verknüpfung und die Multiplikation durch die Und-Verknüpfung zu ersetzen. In Pseudocode sieht die Berechnung der Scoringmatrix folgendermaßen aus:

```
// Eingabe: Die Adjazenzmatrix Adj der Topologie,
//          der maximale Score mxs
```

```
// Ausgabe: Die Scoringmatrix

// verwendete Variablen:
//
// k          : Anzahl der Knoten
// Adj        :     k x k Array, das die Adjazenzmatrix enthält
// ScoreTable :     k x k Array, das die Scoringmatrix enthält
// Add        :     k x k Array als Buffer
// Tmp        : 2 x k x k Array als Buffer
// src, dst,
// i, j, u, v : allgemeine Integervariablen

// Initialisierung
src := 0;
dst := 1;
for i := 0 to k-1 do
  for j := 0 to k-1 do Begin
    Tmp[dst][i,j]   := 0;
    if i = j
      then Begin
        ScoreTable[i,j] := -mxs;
        Add[i,j]        := 1;
        Tmp[src][i,j]   := 1;
      end
      else Begin
        ScoreTable[i,j] := -mxs - 1;
        Add[i,j]        := 0;
        Tmp[src][i,j]   := 0;
      end;
  end;

// (mxs*2)-fache Multiplikation von Adj und Aufbau der Scoringmatrix
for i := 1 to mxs*2 do Begin
  for u := 0 to k-1 do
    for v := 0 to k-1 do Begin
      Tmp[dst][u,v] := 0;
      for j := 0 to k-1 do
        Tmp[dst][u,v] := Tmp[dst][u,v] or (Adj[j,v] and Tmp[src][u,j]);
      Add[u,v] := Add[u,v] or Tmp[dst][u,v];
      ScoreTable[u,v] := ScoreTable[u,v] + Add[u,v];
    end;
  j   := dst;
```

```
    dst := src;
    src := j;
  end;
```

Neben dem geeigneten Scoringschema muß noch ein weiterer Punkt bei der Ausrichtung zweier Merkmalfolgen zueinander beachtet werden. Die Folgen der Merkmale zweier gleicher Segmente enthalten zwar die Merkmale in gleicher Abfolge, jedoch bestimmt die Rotation der Segmente im Bild, mit welchem Merkmal die jeweilige Folge beginnt, da die Folgen über die *absoluten* Winkel der Merkmale zur Horizontalen aufgebaut werden. So kann z.B. die Merkmalfolge eines Segmentes in einem Bild `5,2,1,7,9` lauten, in einem anderen Bild aufgrund einer Drehung des Segmentes hingegen `1,7,9,5,2`. Um diesem Punkt Rechnung zu tragen, verdoppelt man die längere der beiden aneinander auszurichtenden Merkmalfolgen durch einfache Konkatenation. Dadurch ist es möglich, eine überlappende Ausrichtung der beiden Folgen zu erhalten:

```
5,2,1,7,9,5,2,1,7,9
-,-,1,7,9,5,2,-,-,-
```

Die Länge der Ausrichtung im Array `S` des Smith-Waterman Algorithmus bzgl. der verdoppelten Folge, darf die urpsrüngliche Länge dieser Folge nicht überschreiten. Ist die Ausrichtung zu lang und verwendet auf diese Weise Zeichen der ursprünglichen Folge mehrfach, so muß eine andere Ausrichtung mit geringerem Score gewählt werden, die kurz genug ist.

Mit einer gültigen Ausrichtung kann schließlich auch ein Score für die Ähnlichkeit der relativen Winkel zweier Segmentbeschreibungen berechnet werden. Die Ausrichtung gibt dabei vor, welche Winkel miteinander verglichen werden müssen:

```
1 2 3 4 5 1 2 3 4 5
-------------------
5,2,1,7,9,5,2,1,7,9
-,-,1,7,-,5,2,-,-,-
-------------------
    1 2   3 4
```

In diesem Beispiel müssen für das erste Matching, Merkmal 3 der ersten Folge mit Merkmal 1 der zweiten Folge, die relativen Winkel zwischen Merkmal 3 und Merkmal 4 $\angle(3,4)$, Merkmal 3 und 1 $\angle(3,1)$ und Merkmal 3 und 2 $\angle(3,2)$ der ersten Folge mit den relativen Winkeln $\angle(1,2)$, $\angle(1,3)$ und $\angle(1,4)$ der zweiten Folge verglichen werden. Mit der Vorgabe einer maximalen Winkelabweichung `mxa`, für die zwei Winkel noch als minimal ähnlich gelten, kann aus der Differenz von `mxa` und der absoluten Differenz zweier Winkel ein Score für die Ähnlichkeit der Winkel bestimmt werden. Der Gesamtscore für die Ähnlichkeit der relativen Winkel zweier Segmentbeschreibungen ist die Summe der einzelnen Scores aller zu vergleichenden Winkel.
Nachdem sowohl der Score der Ausrichtung der Merkmalfolgen als auch der Score der relativen Winkel normalisiert wurden, werden beide Scores über einen Parameter, der das Verhältnis zwischen beiden Scores bestimmt, zu einem einzigen Ähnlichkeitswert im Bereich $[-1..1]$ zusammengefaßt.

7 Statistische Daten

Mit dem im vorherigen Abschnitt entwickelten Ähnlichkeitsmaß S ist es nun möglich, statistische Daten über die Segmente zu erfassen. Zu diesem Zweck wird nicht nur die Häufigkeit H eines einzelnen Segmentes betrachtet, sondern auch die Häufigkeit der n dem jeweiligen Segment ähnlichsten Segmente. Daraus ergibt sich eine zusammengesetzte Bewertung Z eines Segmentes a:

$$Z_a := H_a + \sum_{i=1}^{n} H_i * S(a,i)$$

Um auch die Nachbarschaften der Segmente statistisch erfassen zu können, kann der für die Berechnung der Ähnlichkeit von Merkmalfolgen eingesetzte Smith-Waterman-Algorithmus angepaßt werden. Analog zu der Folge von Merkmalen in der Segmentbeschreibung kann die Nachbarschaft eines Segmentes als Folge der benachbarten Segmente aufgefaßt werden. Genau wie bei den Merkmalen wird dabei die Abfolge der Nachbarsegmente über ihre relativen Winkel zur Horizontalen durch den Mittelpunkt des zentralen Segmentes bestimmt (s. Abbildung 24). Als Scoringschema,

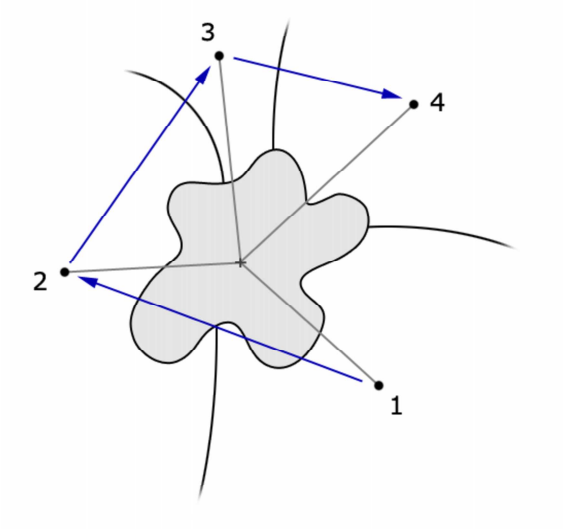


Abbildung 24: *Eine Folge von Nachbarsegmenten. Ein Ähnlichkeitsmaß kann auf die gleiche Weise wie bei den Merkmalfolgen mittels des Smith-Waterman-Algorithmus erzeugt werden.*

das die Ähnlichkeit zweier Nachbarsegmente bewertet, kann direkt das im vorherigen Abschnitt entwickelte Ähnlichkeitsmaß für Segmente verwendet werden. In diesem Sinne entspricht der Vergleich von Segmentnachbarschaften dem Vergleich von Merkmalfolgen auf einer höheren Ebene. Analog der oben beschriebenen zusammengesetzten Bewertung Z reicht es nicht aus, nur die Nachbarschaften eines einzelnen Segmentes zu betrachten. Für die Bewertung einer konkreten Nachbarschaft eines Segmentes bestimmt man die m ähnlichsten Nachbarschaften aus der Menge aller Nachbarschaften der n dem jeweiligen Segment ähnlichsten Segmente und summiert analog zur Berechnung von Z die Häufigkeiten dieser Nachbarschaften gewichtet mit ihrem Ähnlichkeitswert auf.

Die Art und Weise, wie man diese Bewertungen von Segmenten und Segmentnachbarschaften für die Objekterkennung einsetzt, hängt vom konkreten Anwendungsfall ab. Beispielsweise könnte ein Benutzer in einer interaktiven Anwendung einen einzelnen Bildpunkt eines Objektes anklicken, um das Objekt auszuwählen. Das entsprechende Programm würde dann, ausgehend von dem Bildpunkt das zugehörige Segment identifizieren und dessen Z-Wert bestimmen. Liegt der Z-Wert über einer gewissen Schwelle, z.B. dem Durchschnitt der Z-Werte aller Segmente des Bildes, so bildet das Segment den ersten Teil der Auswahl des Objektes. Im nächsten Schritt werden die Z-Werte der benachbarten Segmente berechnet. Nach-

barsegmente mit hohen Z-Werten können zweierlei Bedeutung haben. Zum einen kann es sich um ein weiteres Segment des auszuwählenden Objektes handeln, zum anderen könnte es sich aber auch um ein anderes Objekt handeln, das sich in direkter Nachbarschaft befindet. Im ersten Fall tendiert die Bewertung der Segmentnachbarschaft zu einem hohen Wert, im zweiten Fall eher zu einem niedrigen Wert. Mit Hilfe des neben der Bewertung vom Smith-Waterman-Algorithmus erzeugten Alignment kann schließlich bestimmt werden, um welche Nachbarsegmente die Auswahl erweitert wird. Von diesen Nachbarsegmenten aus werden dann auf gleiche Weise die nächsten Nachbarsegmente untersucht und ggf. zur Auswahl hinzugefügt. Ein derartiges Programm könnte z.B. als Schnittstelle für eine Bildersuchmaschine dienen, bei der der Benutzer in einem Beispielbild ein zu suchendes Objekt anklickt.
Ein anderer Anwendungsfall für eine derartige Objekterkennung liegt im Bereich der Robotik. Ein mobiler und autonomer Roboter wäre in einer unbekannten Umgebung in der Lage, im Laufe der Zeit die Objekte dieser Umgebung zu erlernen. In Kombination mit anderen Sensoren, wie z.B. einem Greifarm mit Tastsinn oder einem Laserscanner, kann der Roboter eigenständig eine präzise „Vorstellung" von seiner Umgebung aufbauen. Die Daten der anderen Sensoren werden dabei mit den entsprechenden Segmentbeschreibungen assoziiert. So können z.B. Segmente und Segmentnachbarschaften mit hohen Bewertungen Rückschlüsse auf die zu erwartenden Sensordaten der übrigen Sensoren erlauben. Eine unter Umständen aufwendige Untersuchung mit anderen Sensoren kann so vermieden werden.

Die Qualität und Aussagekraft der gewonnenen Daten hängt von einer Vielzahl an verschiedenen Faktoren ab. Zu den wichtigsten Faktoren zählt die Stabilität der Segmentierung und die Wiederholbarkeit des Keypoint-Detektors. Aber auch der Zustand des wachsenden neuronalen Gases spielt eine Rolle. Befindet sich das WNG gerade erst im Aufbau, bewegen sich die einzelnen Neuronen noch sehr stark. Dies führt dazu, daß es sehr wahrscheinlich ist, daß ein Merkmalvektor zu unterschiedlichen Zeitpunkten unterschiedlich quantisiert wird. Eine ausgeprägte statistische Häufigkeit seitens einzelner Segmente ist entsprechend erst dann zu erwarten, wenn das WNG sich nur noch geringfügig verändert. Abbildung 25 zeigt das Ergebnis einer Untersuchung des Wachstumsverhaltens des WNG bei SIFT-Merkmalbeschreibungen. Das WNG stabilisiert sich in Abhängigkeit von dem jeweiligen Schwellwert des Haltekriteriums nach der Eingabe von etwa 100.000 bis 200.000 Merkmalvektoren. Für diesen Test wurden aus der Testbildmenge (s. Abbildung 30) pro Bild etwa 200 bis

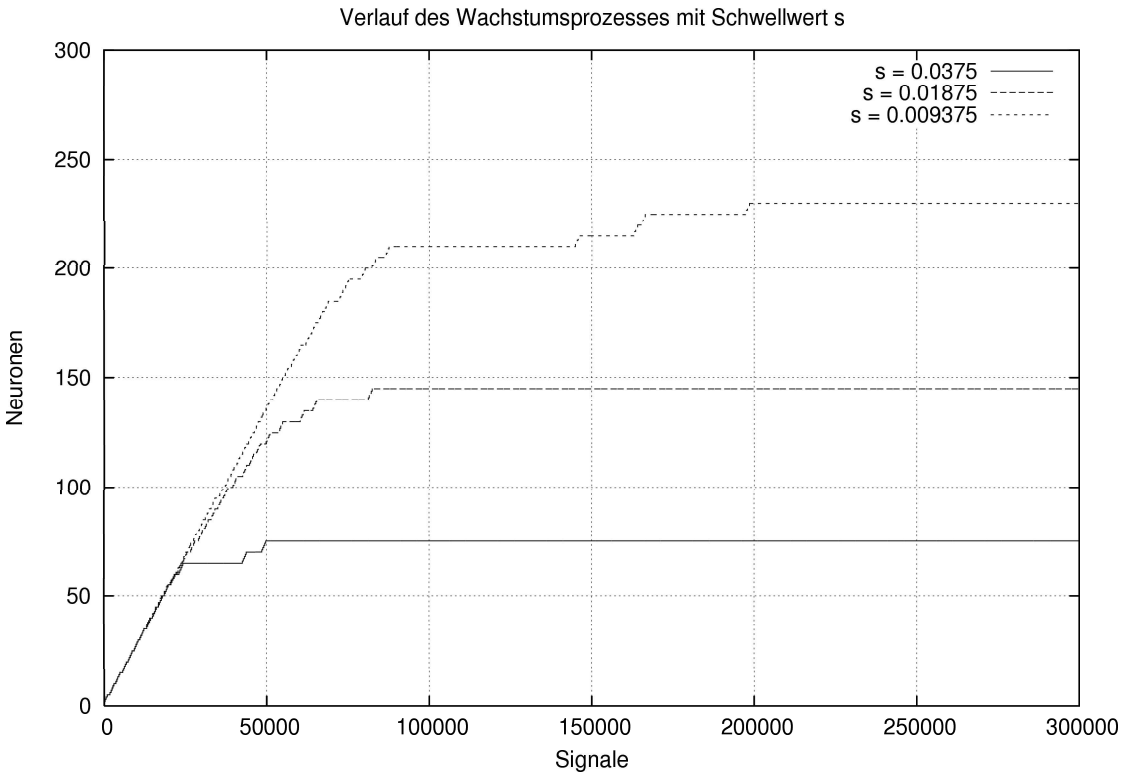


Abbildung 25: *Wachstumsverlauf des WNG für unterschiedliche Schwellwerte des hier vorgestellten Haltekriteriums.*

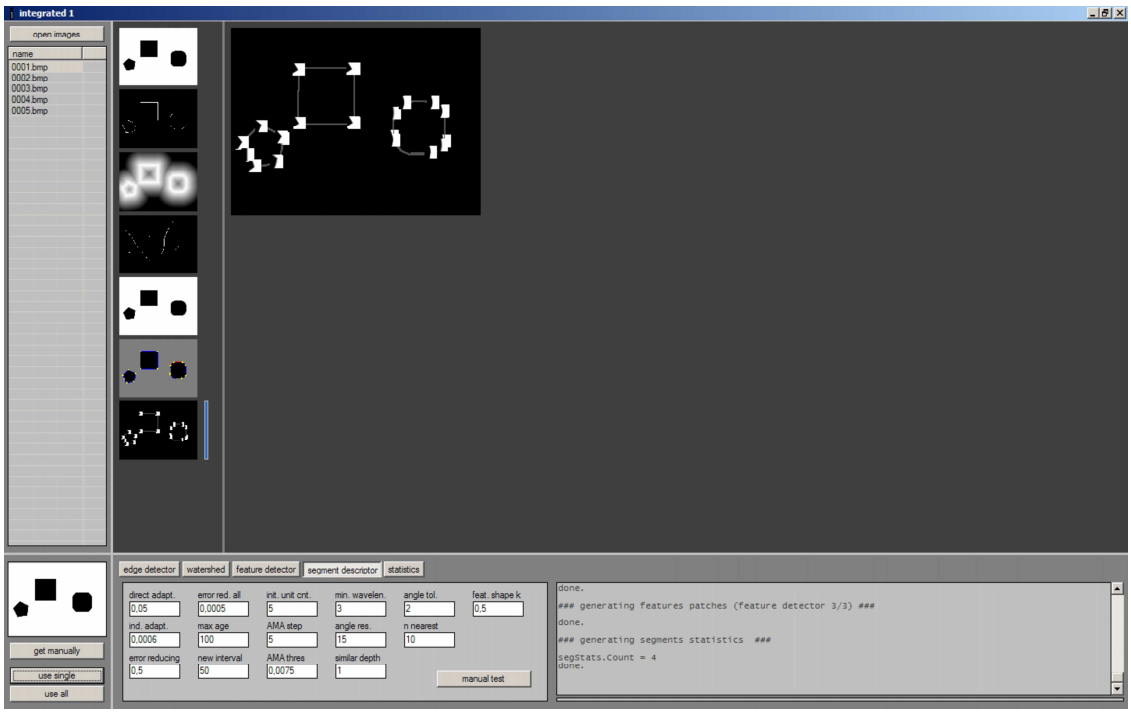


Abbildung 26: *Die entwickelte Testapplikation, die alle Schritte des Verfahrens in einer Anwendung integriert.*

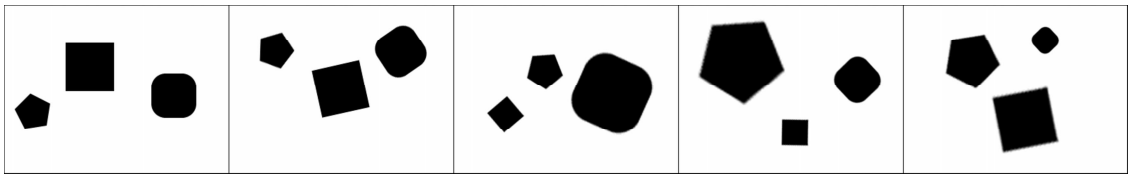


Abbildung 27: *Eine Folge einfacher Testbilder.*

250 lokale Bereiche extrahiert.

8 Ergebnisse und Ausblick

Der in dieser Arbeit vorgestellte Lösungsansatz für die Objekterkennung in Bildern zeigt einen Weg auf, wie durch die Kombination zweier Techniken der digitalen Bildverarbeitung, der Segmentierung und der Merkmalextraktion, eine Objekterkennung ohne a priori Wissen über die zu erkennenden Objekte aufgebaut werden kann. Ein zentrales Element dieses Lösungsansatzes ist das vorgestellte Ähnlichkeitsmaß, das es erlaubt, Mengen aus im Raum angeordneten hochdimensionalen Vektoren miteinander zu vergleichen und auf diese Weise eine „weiche“ Statistik über Grup-

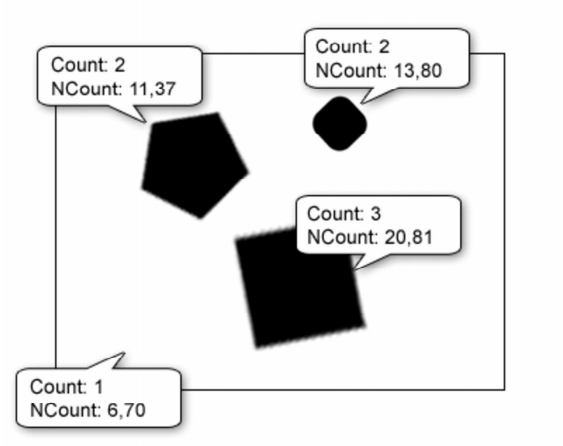


Abbildung 28: *Häufigkeitswerte nach einem Testlauf mit nur 5 Bildern.*

pen ähnlicher Mengen zu erstellen. Ohne dieses Ähnlichkeitsmaß wäre nur ein Test der Identität zweier Mengen möglich und somit nur eine „harte" Statistik über die Häufigkeit einzelner Mengen. Aufgrund der relativ starken Veränderungen, denen die Mengen unterliegen, könnte aus einer solchen harten Statistik keine ausreichende Information über mögliche zugehörige Objekte gewonnen werden. Eine weiche Statistik ermöglicht es, auch Gruppen ähnlicher Segmente zu betrachten und diese möglichen zugehörigen Objekten zuzuordnen. Der modulare Aufbau des vorgestellten Verfahrens ermöglicht es, beliebige Methoden für die Segmentierung, die Keypoint-Detektion und die Merkmalbeschreibung einzusetzen. Die in dieser Arbeit verwendeten Methoden orientieren sich daher nicht nur nach ihren Fähigkeiten, die jeweiligen Aufgaben zu lösen, sondern auch nach der Komplexität ihrer Implementierung, um in dem gesetzten Zeitrahmen bleiben zu können. Abbildung 26 zeigt die im Rahmen dieser Diplomarbeit entwickelte Testanwendung, die alle Schritte des Verfahrens in einem Programm integriert. Die verschiedenen Möglichkeiten für die einzelnen Schritte wurden zuvor in separaten Anwendungen entwickelt und getestet (siehe Anhang). Testläufe mit einfachen Bildern, wie sie Abbildung 27 zeigt, entwickeln die beschriebenen statistischen Häufigkeiten sehr schnell, da die Segmentierung dieser einfachen Bilder relativ stabil ist und das WNG nur zwischen wenigen Merkmalvektoren unterscheiden muß. Abbildung 28 zeigt die Häufigkeiten der 3 Vordergrundsegmente und des Hintergrundsegmentes nach der Eingabe der 5 in Abbildung 27 dargestellten Testbilder. Der Wert „Count" bezieht sich auf die Häufigkeit des entsprechenden Segmentes, während sich der Wert „NCount" aus

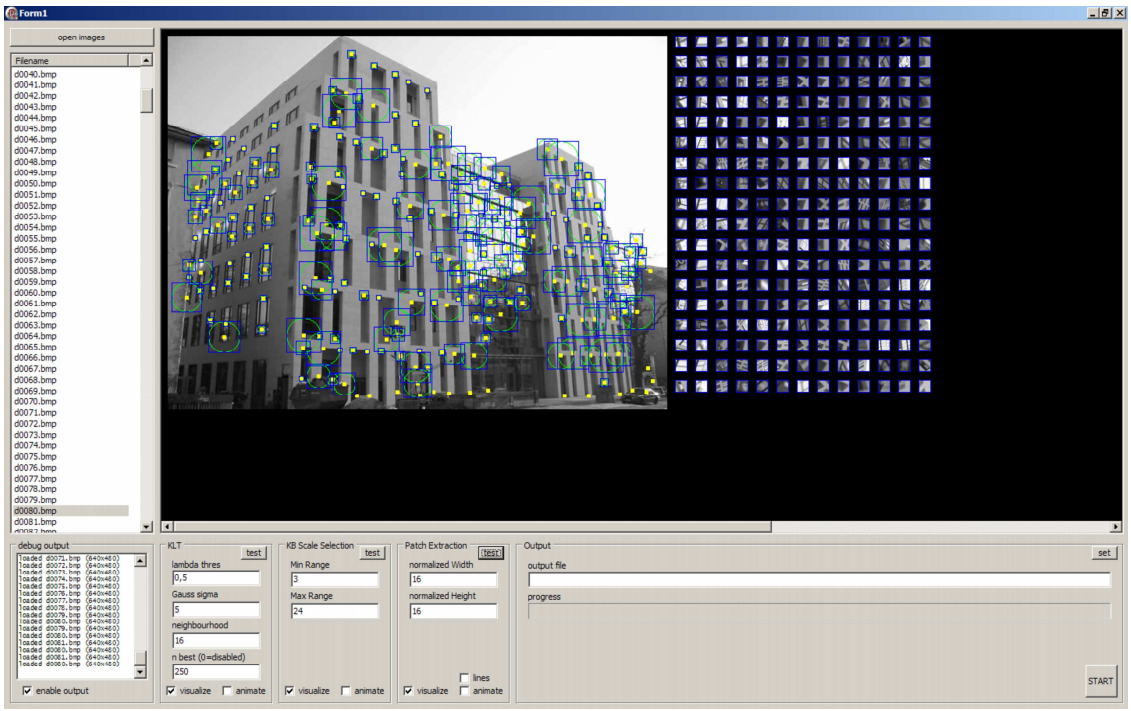


Abbildung 29: *Anwendung für die Erstellung von Testdaten für das WNG.*

der Summe der Häufigkeiten der 10 ähnlichsten Segmente gewichtet mit dem jeweiligen Ähnlichkeitswert ergibt. Schon nach der Eingabe von nur 5 Bildern ist zu erkennen, daß das Hintergrundsegment eine geringere Häufigkeit als die Vordergrundsegmente besitzt, insbesondere bzgl. des NCount-Wertes. Der KLT-Detektor liefert schon bei diesen einfachen Bildern eine recht unterschiedliche Anzahl an Keypoints für die einzelnen Flächen in verschiedenen Bildern. Im Gegensatz zur Anzahl ist hingegen die relative Position der Keypoints bzgl. der zugehörigen Flächen über die verschiedenen Bilder hinweg stabil. Auch wenn ein gleichmäßigeres Verhalten des Keypoint-Detektors wünschenswert wäre, so zeigt es doch, daß das verwendete Ähnlichkeitsmaß derartige Schwankungen in der Zahl der Merkmale pro Fläche in einer geeigneten Weise berücksichtigt.

Für die einzelnen Schritte des Verfahrens wurden zusätzlich separate Tests mit eigenen Testanwendungen durchgeführt. Abbildung 29 zeigt eine Anwendung, die für eine große Menge von Bildern die KLT-Keypoints jedes Bildes ermittelt, die Größe des lokalen Bereiches der Keypoints mit dem Entropie basierten Verfahren von Kadir und Brady bestimmt, die Hauptrichtung mit dem hier vorgestellten auf dem Phasen-

spektrum der Fouriertransformation basierenden Verfahren berechnet und schließlich jeden lokalen Bereich auf eine vorgegebene Größe normiert und extrahiert.

Der Test einer großen Anzahl an Bildern (s. Abbildung 30) hat gezeigt, daß die verwendeten Methoden auch auf sehr verschiedenen Bildern relativ gute Ergebnisse liefern. Es wurden im Schnitt ca. 207 Keypoints pro Bild generiert, mit einer Streuung von etwa 27 Keypoints. Insgesamt wurden auf diese Weise aus den Testbildern 165884 Keypoints extrahiert und für spätere Untersuchungen des WNG gespeichert. Die Wahl geeigneter Parameter, insbesondere für den KLT-Detektor, hat sich bei diesem Test als schwierig erwiesen. Wie schon zuvor beschrieben, neigt der KLT-Detektor dazu, eine recht unterschiedliche Anzahl an Keypoints auch bei sehr ähnlichen Bildbereichen zu liefern. Im Vergleich zu den sehr einfachen Bildern tritt dieses Verhalten bei natürlichen Bildern erstaunlicherweise weniger stark auf. Vermutlich liegt dies daran, daß bei sehr einfachen Bildern (vgl. Abbildung 27) die vom KLT-Detektor berechneten Eigenwerte sehr nahe beieinander liegen und somit schwerer von einem einfachen Schwellwert unterschieden werden können. Die Bestimmung der Größe des lokalen Bereiches mit dem Verfahren von Kadir und Brady liefert gute und reproduzierbare Ergebnisse, d.h. für lokale Bereiche mit ähnlichem Inhalt werden die gleichen Größen für diese Bildbereiche gewählt. Einzig bei den Bereichen, die für unterschiedliche Größen selbstähnlich sind, variiert die Wahl der Größe des Bereiches. Abbildung 31 zeigt ein Beispiel für einen derartigen lokalen Bereich. Die Auswahl der kleineren Größe (rot) führt lediglich zu leicht unscharfen Kanten in dem auf eine einheitliche Größe skalierten Bildbereich, entspricht aber ansonsten dem auf die gleiche Größe skalierten grün markierten Bildbereich. Die Bestimmung der Hauptrichtung mit dem in dieser Arbeit vorgestellten Verfahren lieferte ebenfalls gute Ergebnisse. Ein Test mit dem in Abbildung 42 gezeigten Programm ergab einen durchschnittlichen Fehler von etwa 5 Grad bei der Bestimmung der Hauptrichtung verschiedener lokaler Bereiche aus verschiedenen Bildern.

Die für alle Bilder extrahierten lokalen Bereiche wurden gespeichert, um sie in einer weiteren Anwendung genauer zu untersuchen und gleichzeitig die Quantisierung von Merkmalvektoren mittels eines wachsenden neuronalen Gases zu testen. Abbildung 32 zeigt diese Anwendung. Sie bietet verschiedene Möglichkeiten, die Aktivität und den Zustand des wachsenden neuronalen Gases zu visualisieren. Neben Kennzahlen wie der aktuellen Anzahl an verwendeten Neuronen oder dem durchschnittlichen Grad der Knoten der Topologie, wird über eine farbige Matrix der

Abbildung 30: *170 von 798 Testbildern, die für den Test der Quantisierung von hochdimensionalen Merkmalvektoren mittels eines WNG genutzt wurden.*

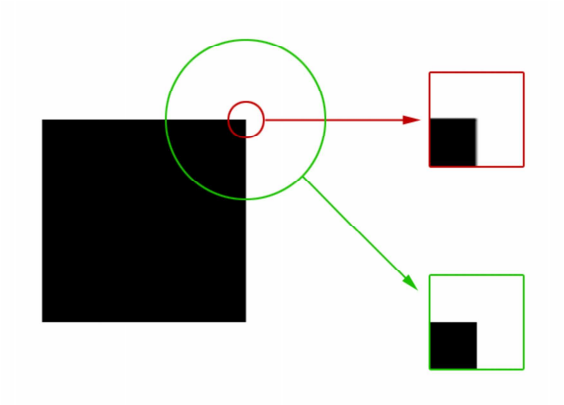


Abbildung 31: *Beispiel eines für verschiedene Größen selbstähnlichen lokalen Bereiches.*

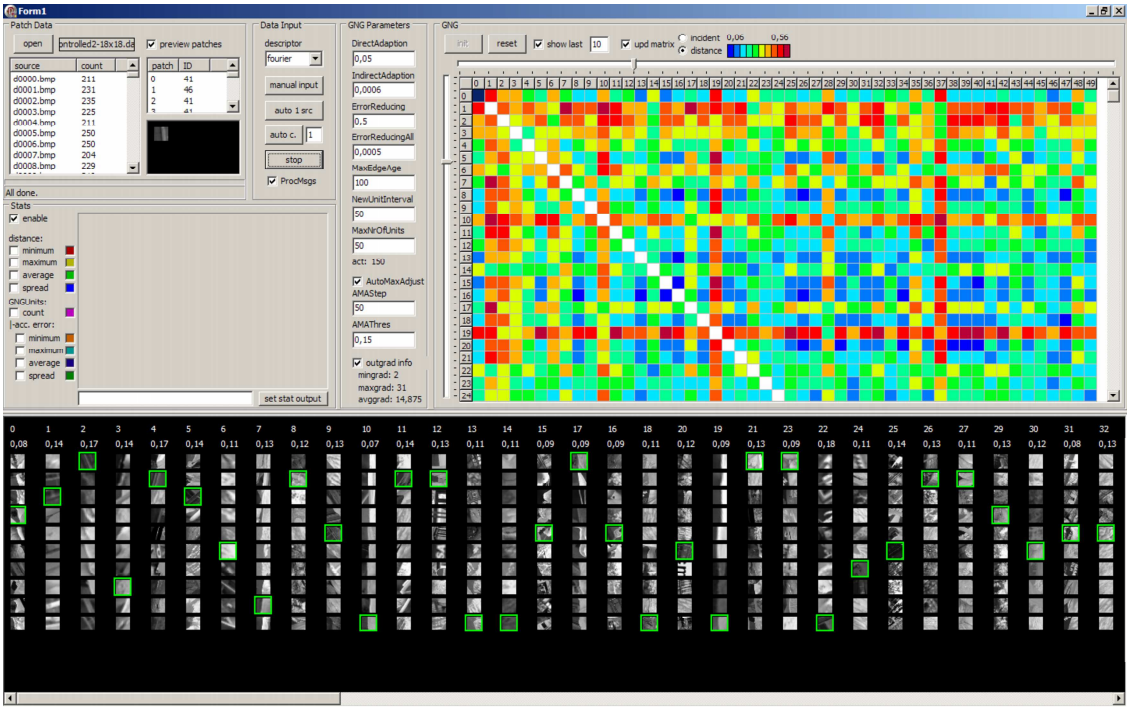


Abbildung 32: *Anwendung für den Test der Quantisierung von Merkmalvektoren mittels eines wachsenden neuronalen Gases.*

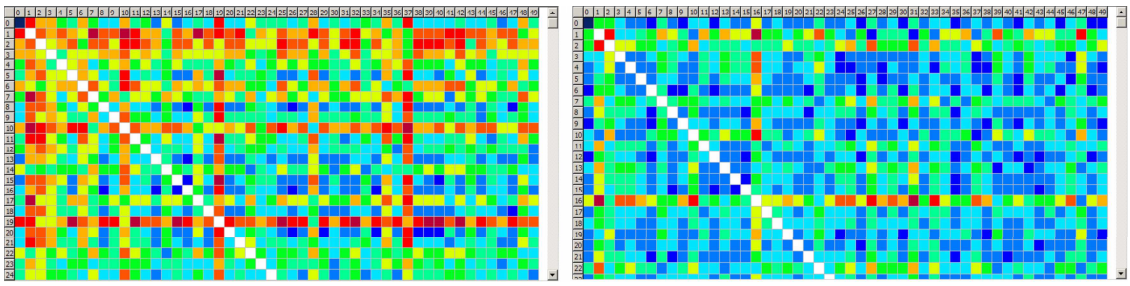


Abbildung 33: *Vergleich der euklidischen Abstände der Neuronen zweier WNGs. Die linke Seite zeigt die Eingabe des hier vorgestellten alternativen Deskriptors, die rechte Seite die direkte Eingabe der Intensitätswerte der lokalen Bereiche. Geringe Abstände sind durch Blautöne dargestellt, hohe Abstände haben rötliche Farben.*

Abbildung 34: *Darstellung der Adjazenzmatrix der Neuronen des WNG.*

euklidische Abstand der einzelnen Neuronen zueinander visualisiert. Es zeigt sich, daß für Merkmalbeschreibungen wie SIFT oder die hier vorgestellte Alternative die Merkmale im Merkmalraum nicht gleichverteilt sind. Dies wurde schon von Lowe in [9] über den SIFT-Deskriptor bemerkt und bestätigt sich hier in der vom WNG erzeugten Topologie. Verwendet man direkt die Intensitätswerte der lokalen Bereiche als Deskriptor und läßt diese von einem WNG verarbeiten, so erkennt man, das diese deutlich gleichmäßiger über den Eingaberaum verteilt sind (s. Abbildung 33). Da diese Deskriptoren aber in keinster Weise Helligkeits- und Transformationsinvariant sind, eignen sie sich nicht, um markante Punkte von Objekten zu beschreiben. Eine weitere Visualisierungsmöglichkeit ist die Darstellung der Adjazenzmatrix der Neuronen des WNG (s. Abbildung 34). Im unteren Teil der Anwendung werden für jedes Neuron die n letzten lokalen Bildbereiche dargestellt, deren Merkmalbeschreibungen auf das entsprechende Neuron abgebildet wurden. Dies ermöglicht eine visuelle Bewertung des verwendeten Merkmaldeskriptors. Sind sich z.B. die lokalen Bildbereiche, die auf ein Neuron abgebildet werden, unähnlich, so beschreibt der

verwendete Deskriptor die Bildbereiche zu allgemein. In Kombination mit der Darstellung der Adjazenzmatrix kann beobachtet werden, welche lokalen Bildbereiche bzw. welche entsprechenden Neuronen zueinander benachbart sind. Dies ist hilfreich, wenn es darum geht, einen geeigneten Wert für den Parameter `mxs` des zuvor beschriebenen Ähnlichkeitsmaßes zu bestimmen. Die Darstellung der auf die Neuronen abgebildeten lokalen Bereiche illustriert zudem, mit welcher Granularität das WNG die Merkmaldeskriptoren quantisiert. Anhand dieser kann bei Bedarf der Schwellwert des oben vorgestellten alternativen Haltekriteriums des WNG angepaßt werden. Repräsentieren zu viele Neuronen die gleichen oder ähnliche Merkmalvektoren, so kann der Schwellwert des Haltekriteriums erhöht werden. Werden sehr unterschiedliche Merkmalvektoren durch ein einzelnes Neuron repräsentiert, so muß der Schwellwert des Haltekriteriums verringert werden.

Für den Test der Quantisierung hochdimensionaler Merkmalvektoren mittels des wachsenden neuronalen Gases wurden zunächst für eine große Menge und hohe Bandbreite an unterschiedlichen Bildern (s. Abbildung 30) die durch einen KLT-Detektor bestimmten lokalen Bereiche extrahiert. Für die Umsetzung dieser Testdaten in Merkmalvektoren wurden verschiedene Merkmalbeschreibungen getestet. Neben der direkten Eingabe der Intensitätswerte als Merkmalvektor, wurden sowohl der SIFT-Deskriptor als auch die hier vorgestellte alternative Merkmalbeschreibung (Fourier) eingesetzt. Auch wenn die direkte Eingabe der Intensitätswerte als Merkmalvektor, wie bereits oben erwähnt, aus Sicht der Objekterkennung keinen Sinn macht, so bietet sie doch für einen menschlichen Beobachter die Möglichkeit, die Quantisierung von hochdimensionalen Vektoren durch das WNG auf einfache augenscheinliche Weise bewerten zu können. Wie Abbildung 35 zeigt, werden bei der Eingabe der direkten Intensitätswerte in das neuronale Gas, ähnliche lokale Bereiche auf gleiche Neuronen abgebildet. Bei der Eingabe von SIFT-Deskriptoren oder Deskriptoren der hier vorgestellten Alternative ist die korrekte Quantisierung weniger offensichtlich, da der Betrachter sich zunächst für jeden betrachteten lokalen Bereich vorstellen muß, wie dieser vom verwendeten Deskriptor beschrieben wird. Neben der Quantisierung hochdimensionaler Merkmalvektoren wurde ebenfalls getestet, wie gut das in dieser Arbeit vorgestellte Haltekriterium funktioniert. Bei Testbildmengen der Art von Abbildung 30 mit einem breiten Spektrum an verschiedenen Merkmalen kann ein sich stetig verlangsamender Wachstumsprozess bis zum vollständigen Stop des Wachstums beobachtet werden (s. Abbildung 25). Der Schwellwert des

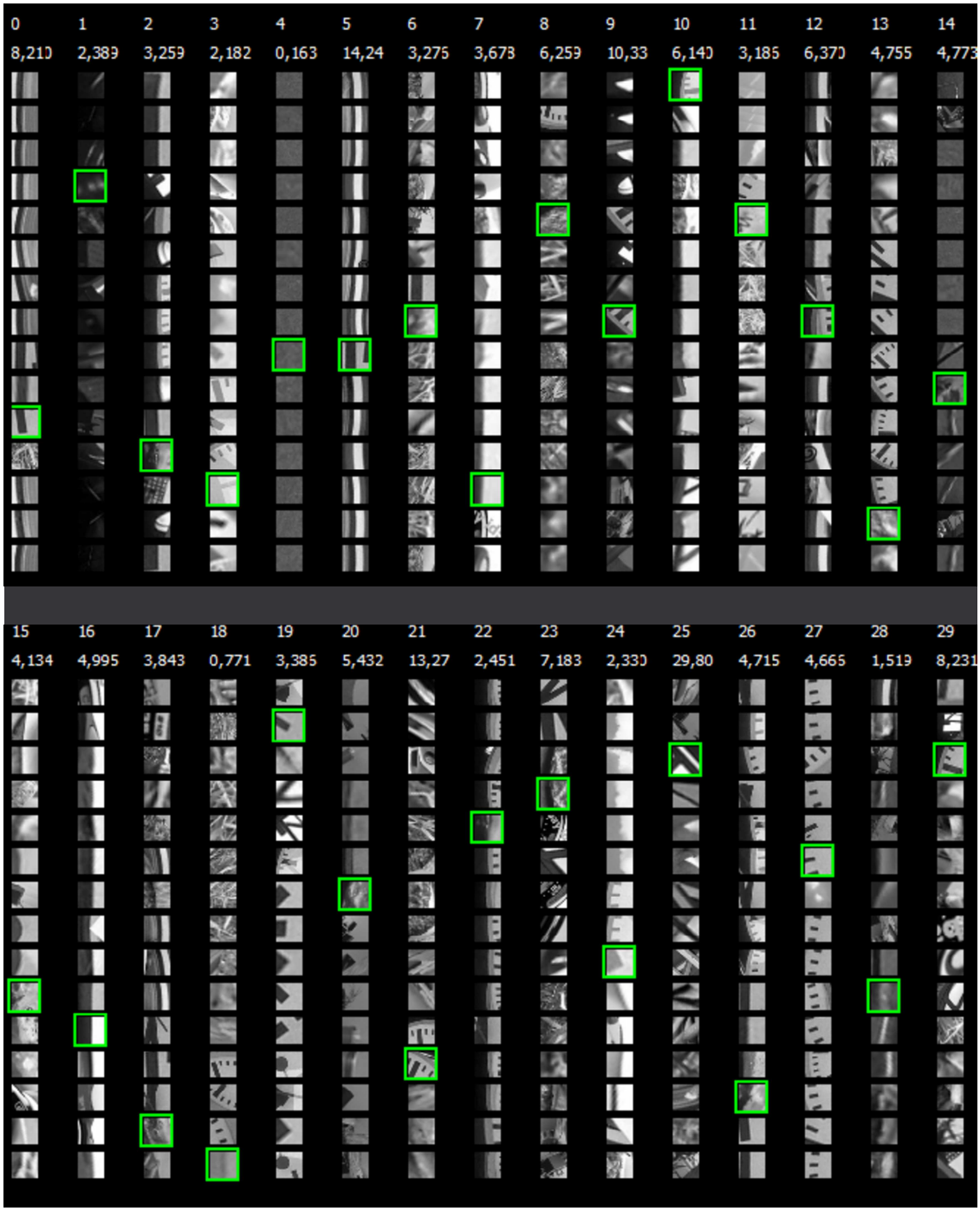


Abbildung 35: *Für 30 Neuronen die jeweils 15 jüngsten lokalen Bereiche, die auf die Neuronen abgebildet wurden.*

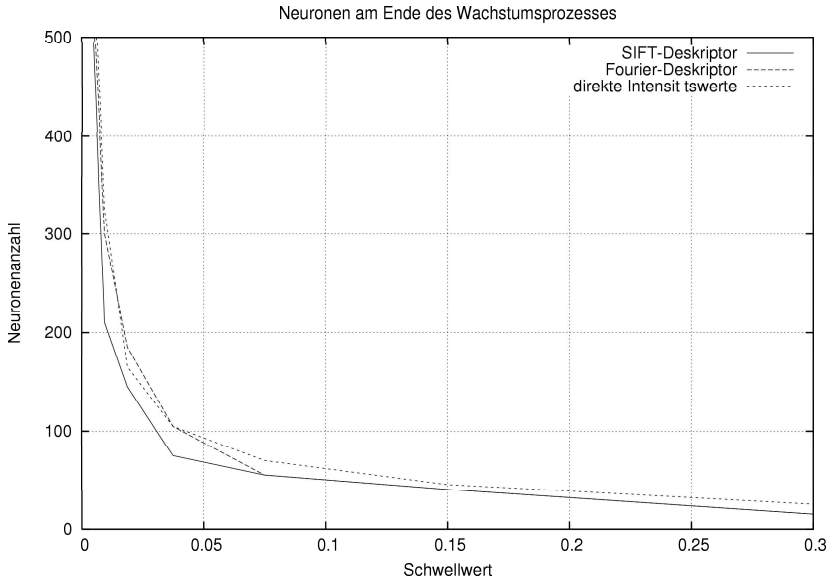


Abbildung 36: *Test des Haltekriteriums. Der Graph beschreibt das Verhältnis zwischen dem Schwellwert des Haltekriteriums und der Neuronenanzahl am Ende des Wachstumsprozesses.*

Haltekriteriums ist dabei umgekehrt proportional zur absoluten Anzahl der Neuronen am Ende des Wachstumsprozesses und bestimmt damit die Auflösung der erzielten Quantisierung (s. Abbildung 36). Obwohl die hier beschriebenen Merkmalvektoren i.d.R. deutlich mehr als 100 Dimensionen besitzen, werden sie selbst bei kleinen Schwellwerten für das Haltekriterium dennoch nur auf wenige hundert Neuronen abgebildet. Dies ist deutlich weniger als derart hochdimensionale Vektoren zunächst vermuten lassen und läßt den Schluß zu, daß der durch die Merkmalvektoren beschriebene Eingaberaum nicht sehr dicht besetzt ist. Bei Testbildmengen der Art von Abbildung 37, die der Eingabe bzw. Verarbeitung von Videosequenzen entsprechen, kann ein anderes Wachstumsverhalten beobachtet werden. Aufgrund der relativ ähnlichen Merkmale der aufeinanderfolgenden Bilder der Videosequenzen endet das Wachstum des neuronalen Gases relativ schnell. Erst wenn sich die Merkmale z.B. durch einen Szenenwechsel ändern, setzt das Wachstum erneut ein, bis das neuronale Gas auch die neuen Merkmale ausreichend gut quantisieren kann. Dem Schwellwert des Haltkriteriums kommt hierbei eine entscheidene Rolle zu. Ist der Schwellwert zu hoch, dann setzt das Wachstum bei einem Szenenwechsel nicht wieder ein und die neuen Merkmale „ziehen“ die Neuronen von den Positionen der alten Merkmale ab. Ein zu hoch gewählter Schwellwert kann also bei der Verarbei-

Abbildung 37: *Auszug aus Testbildsequenzen, die für den Test der Quantisierung von Merkmalvektoren mittels WNG verwendet wurden.*

tung von Videosequenzen zu einer für das in dieser Arbeit beschriebenen Verfahren unerwünschten ständigen Bewegung des neuronalen Gases führen.

Das in dieser Arbeit vorgestellte Verfahren ist lediglich ein Lösungsansatz, der in erster Linie auf relativ einfachen Bildern funktioniert. Eine Verbesserung der Segmentierung, z.B. durch den Einsatz der in 4.4 und 4.5 beschriebenen Segmentierungsverfahren, wäre für den Einsatz auf komplexeren Bildern vorteilhaft. Die hier eingesetzte Segmentierungsmethode kann vor allem durch eine Verbesserung des Kantenfilters, z.B. durch eine dynamische Parametrisierung, und durch eine Verbesserung der Segmentverschmelzung im Nachbearbeitungsschritt optimiert werden. Ein weiterer Verbesserungspunkt ist der Keypoint-Detektor. Der eingesetzte KLT-Detektor liefert zwar brauchbare Ergebnisse in dem Sinne, daß die gefundenen Keypoints eine recht hohe Wiederholrate aufweisen, die Anzahl der Keypoints schwankt jedoch zwischen verschiedenen Bildern erheblich. So kann es passieren, das bei gleichen Parametern in einem Bild an den augenscheinlich richtigen Stellen Keypoints gefunden werden, bei einem anderen Bild jedoch gar keine oder viel zu viele Keypoints gefunden werden.

Neben diesen Punkten, die der Verbesserung bedürfen, können wiederum viele Teile des Verfahrens ihrerseits zu einer Verbesserung von anderen Bildverarbeitungsverfahren beitragen. Der vorgestellte Kantendetektor liefert ähnliche Ergebnisse wie der Canny-Edge-Detektor und stellt durch seine andere Parametrisierung (exponentiell statt linear) eine Alternative für kantenbasierte Verfahren dar. Die vorgestellte Erweiterung der Hough-Transformation kann dazu verwendet werden, separate Linienabschnitte in den von der Transformation detektierten Geraden ausfindig zu machen. Das Verfahren zur Bestimmung der Hauptrichtung eines lokalen Bereiches erzielt deutlich stabilere Ergebnisse als die SIFT-Variante und der vorgestellte Merkmaldeskriptor kann ebenfalls einige Schwächen des SIFT-Deskriptors ausgleichen. Damit stellen diese beiden Methoden interessante Alternativen für verschiedene merkmalbasierte Bildverarbeitungsverfahren dar, insbesondere für jene, die bislang mit SIFT arbeiten. Schließlich liefert die Kombination von wachsendem neuronalen Gas und dem hier beschriebenen Ähnlichkeitsmaß auf Basis des Smith-Waterman-Algorithmus eine neue Methode, wie Mengen von im Raum angeordneten hochdimensionalen Vektoren miteinander verglichen werden können.

Literatur

[1] Viola, P. ; Jones, M. *Rapid object detection using a boosted cascade of simple features. In CVPR.* 2001

[2] Swain, Michael J. ; Ballard, Dana H.: Color indexing. In: *Int. J. Comput. Vision* 7 (1991), Nr. 1, S. 11–32. – ISSN 0920–5691

[3] Huang, J. ; Kumar, S. R. ; Mitra, M. ; Zhu, W.-J.: Spatial Color Indexing and Applications. In: *ICCV*, 1998, S. 602–607

[4] Nistico, W. ; Röfer, T.: Improving percept reliability in the Sony Four-Legged League. In: *RoboCup 2005: Robot Soccer World Cup IX / Lecture Notes in Artificial Intelligence*, Springer-Verlag, 2005

[5] Veltkamp, R. C. ; Hagedoorn, M.: State of the art in shape matching. (2001), S. 87–119. ISBN 1–85233–381–2

[6] Borenstein, E. ; Ullman, S.: Class-Specific, Top-Down Segmentation. In: *ECCV '02: Proceedings of the 7th European Conference on Computer Vision-Part II*, Springer-Verlag, 2002. – ISBN 3–540–43744–4, S. 109–124

[7] Schmid, C. ; Mohr, R.: Local Grayvalue Invariants for Image Retrieval. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 19 (1997), Nr. 5, S. 530–535

[8] Kadir, T. ; Brady, M.: Saliency, Scale and Image Description. In: *Int. J. Comput. Vision* 45 (2001), Nr. 2, S. 83–105. – ISSN 0920–5691

[9] Lowe, D. *Distinctive image features from scale-invariant keypoints.* 2003

[10] Moravec, H.: Rover Visual Obstacle Avoidance. In: *proceedings of the seventh International Joint Conference on Artificial Intelligence*, 1981, S. 785–790

[11] Harris, C. ; Stephens, M.: A combined corner and edge detector. In: *proceedings of The Fourth Alvey Vision Conference*, 1988, S. 147–151

[12] Freeman, W. T. ; Adelson, E. H.: The design and use of steerable filters. In: *IEEE Trans. Pattern Analysis and Machine Intelligence* 13 (1991), Nr. 9, S. 891–906

[13] Roweis, S.T. ; Saul, L.K.: Nonlinear Dimensionality Reduction by Locally Linear Embedding. In: *Science* 290 (2000), Nr. 5500, S. 2323–2326

[14] Ke, Y. ; Sukthankar, R.: PCA-SIFT: A more distinctive representation for local image descriptors. In: *In Proceedings of IEEE Computer Vision and Pattern Recognition*, 2004

[15] Mikolajczyk, K. ; Schmid, C.: A Performance Evaluation of Local Descriptors. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 27 (2005), Nr. 10, S. 1615–1630. – ISSN 0162–8828

[16] Fritzke, B.: *Vektorbasierte Neuronale Netze*. Shaker Verlag, 1998

[17] Merkl, R. ; Waack, S.: *Bioinformatik Interaktiv - Algorithmen und Praxis*. WILEY-VCH, 2003

[18] I. Dahm, M. H. ; Osterhues, A.: Robust color classification for robot soccer. In: *7th International Workshop on RoboCup 2003, Lecture Notes in Artificial Intelligence* (2003)

[19] Ritter, Helge ; Ontrup, Jörg: Perceptual Grouping in a Neural Model: Reproducing Human Texture Perception / SFB 360, Bielefeld University. 1998 (TR 98/6). – Forschungsbericht

[20] Ritter, Helge: A Spatial Approach to Feature Linking. In: *INNC-90* 2 (1990), S. 898–901

[21] Shi, Jianbo ; Malik, Jitendra: Normalized Cuts and Image Segmentation. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22 (2000), Nr. 8, S. 888–905

[22] Gonzalez, R.C. ; Woods, R.E.: *Digital Image Processing - Second Edition*. Prentice Hall, 2002

[23] Canny, J: A computational approach to edge detection. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 8 (1986), Nr. 6, S. 679–698. – ISSN 0162–8828

[24] Noble, J. A.: *Description of Image Surfaces*, Department of Engineering Science, University of Oxford, UK, Diss., 1989

[25] Tomasi, Carlo ; Kanade, Takeo: Detection and Tracking of Point Features / Carnegie Mellon University. 1991 (CMU-CS-91-132). – Forschungsbericht

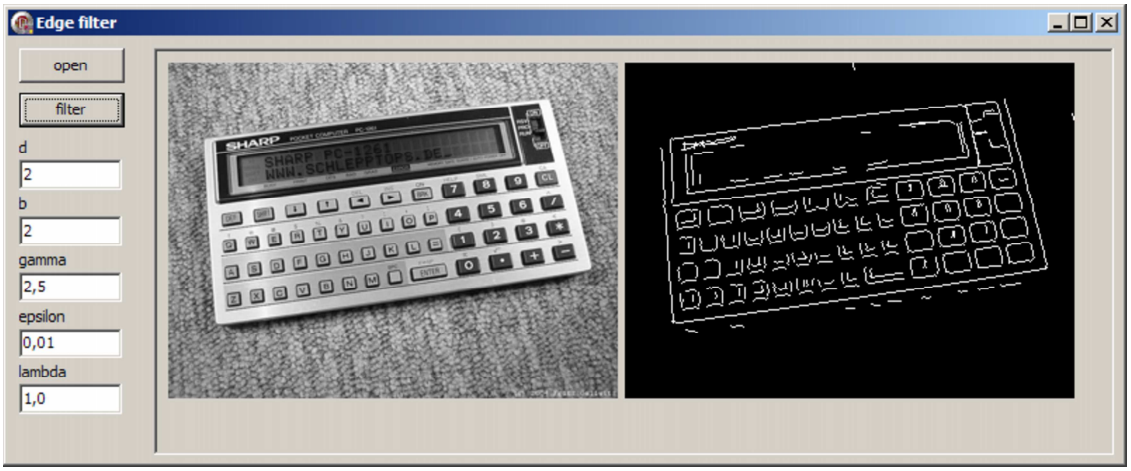


Abbildung 38: *Testprogramm für den in dieser Arbeit vorgestellten Kantendetektor.*

A Softwaredokumentation

Im Folgenden werden einige der Programme dokumentiert, die im Rahmen dieser Diplomarbeit erstellt wurden. Alle Programme wurden mit Borland Delphi 2005 geschrieben. Das verwendete Bildformat für das Einladen von Bildern ist bei allen Programmen BMP (24Bit). Intern werden die Bilder durch Arrays vom Typ `double` und mit Intensitätswerten aus dem Bereich [0..1] repräsentiert. Die Programme liegen sowohl compiliert als auch im Quelltext auf einer DVD dieser Arbeit bei.

A.1 Kleinere Testprogramme

Das in dieser Arbeit vorgestellte Verfahren ist relativ komplex und beherbergt eine Reihe von Teilproblemen, für die es verschiedene Lösungsansätze gibt. Dementsprechend wurden viele kleine Programme geschrieben, die den Test von Lösungen für die einzelnen Teilprobleme erlauben.

A.1.1 Kantendetektor

Abbildung 38 zeigt das Testprogramm, welches für den hier vorgestellten Kantendetektor (s. 4.6.1) erstellt wurde. Über die „open"-Schaltfläche kann ein Graustufenbild geöffnet werden. Auf der linken Seite des Programmfensters befinden sich Eingabefelder, über die die einzelnen Parameter des Kantendetektors $(d, b, \gamma, \epsilon, \lambda)$ eingestellt werden können. Die Schaltfläche „filter" wendet den Detektor auf das

Abbildung 39: *Testprogramm für schnelles Weichzeichnen mittels Summenbild.*

geöffnete Bild an und gibt das berechnete Kantenbild rechts neben dem Originalbild aus.

A.1.2 Schneller Weichzeichner

In gleicher Weise wie das zuvor beschriebene Testprogramm für den Kantenfilter ist auch das in Abbildung 39 gezeigte Programm aufgebaut, welches für den Test des von Viola und Jones in [1] vorgestellten schnellen Weichzeichners erstellt wurde. Der Parameter, der über das Eingabefeld auf der linken Seite des Programmfensters eingestellt werden kann, ist die Reichweite r des Weichzeichners. Das lokale Fenster, über das der Algorithmus den Durchschnitt ermittelt ist entsprechend $(2r)^2$ Pixel groß.

A.1.3 Scanline Verfahren

Das in 4.6.2 vorgestellte Scanline-Verfahren zur Segmentbildung auf Kantenbildern wurde in dem in Abbildung 40 gezeigten Programm implementiert. Als Eingabe erwartet das Programm ein Kantenbild, das über die Schaltfläche „open“ geladen werden kann. Die Schaltfläche „reload“ lädt das zuletzt geöffnete Bild erneut. Dies kann erwünscht sein, wenn mögliche Debug-Zeichnungen aus einem vorherigen Lauf des Algorithmus wieder aus dem dargestellten Bild gelöscht werden sollen. Die Schaltfläche „get Segments“ führt den Scanline-Algorithmus auf dem geladenen Kantenbild aus. Unterhalb des Bildes befindet sich ein mehrzeiliges Textfeld, das den Fortschritt des Algorithmus dokumentiert. Desweiteren befinden sich dort zwei Listen, welche die erzeugten Linien- bzw. Pixelcluster nach Ablauf des Algorithmus

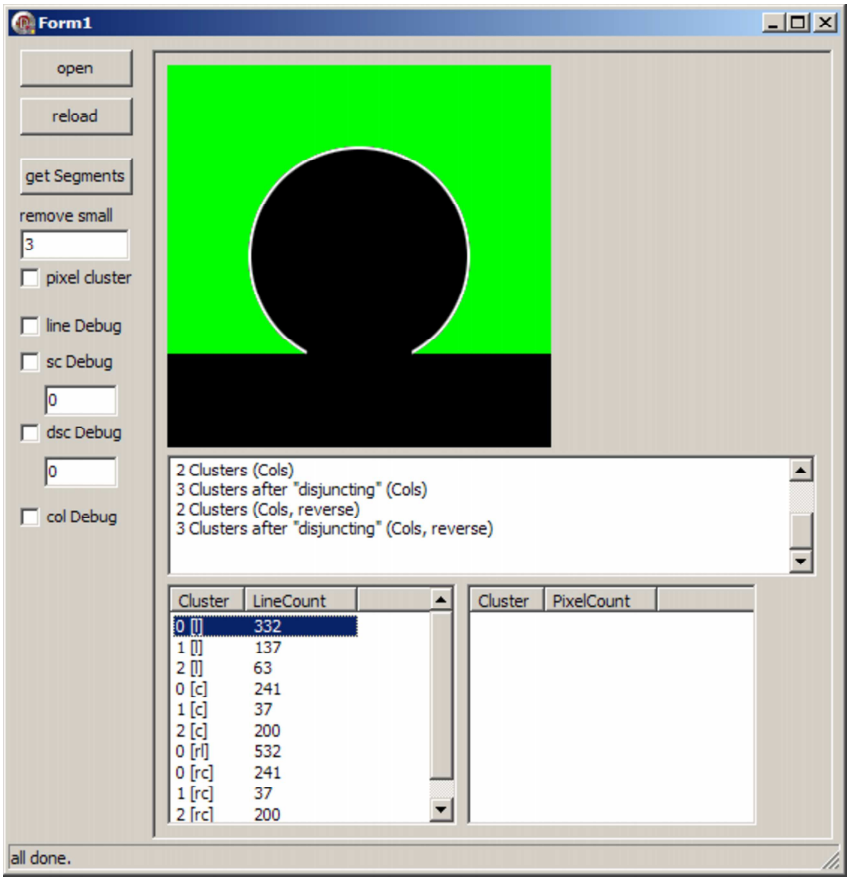


Abbildung 40: *Testprogramm für die Segmentierung mittels Scanline-Verfahren.*

enthalten. Das implementierte Verfahren erstellt Liniencluster für die Suchrichtungen „zeilenweise von oben nach unten“ (l), „spaltenweise von links nach rechts“ (c), „zeilenweise von unten nach oben“ (rl) und „spaltenweise von rechts nach links“ (rc). Die für die einzelnen Suchrichtungen gefundenen Cluster werden in der linken Liste mit dem entsprechenden Kürzel (l,c,rl,rc) für die jeweilige Suchrichtung aufgeführt. Die rechte Liste wird nur dann gefüllt, wenn die Option „pixel cluster“ (auf der linken Seite des Programmfensters) ausgewählt wurde. Die Liste enthält disjunkte Mengen von Bildpunkten (Pixelcluster), die aus dem Schnitt der Punkte der Liniencluster (linke Liste) erzeugt wurden. Über das Eingabefeld „remove small“ kann angegeben werden, ab welcher Anzahl von Bildpunkten ein Pixelcluster nicht mehr in die rechte Liste aufgenommen werden soll.
Wird in einer der beiden Listen ein Eintrag mit der Maus markiert, so werden die entsprechenden Bildpunkte im angezeigten Bild hellgrün eingefärbt. Weitere Optionen auf der linken Seite des Programmfensters erlauben es, sich einen Teil der Zeilen- und Spaltenabschnitte anzeigen zu lassen („line debug“„col Debug“), die der Algorithmus auf dem Kantenbild bestimmt. Die Option „sc Debug“ ermöglicht es, einen durch das zugehörige Eingabefeld bestimmten, noch nicht disjunkten Liniencluster anzeigen zu lassen, bevor der Algorithmus die Liniencluster durch entsprechende Mengenoperationen in disjunkte Cluster aufteilt. Die Option „dsc Debug“ ist veraltet und hat die gleiche Funktion wie das Auswählen eines Clusters in der linken Liste.

A.1.4 Schnelle Fourier Transformation

Abbildung 41 zeigt das Programm, das für den Test der Implementation der schnellen Fourier Transformation geschrieben wurde. Die Transformation selbst wurde als Klasse `TFFT` in der Unit `gfxstuff.pas` umgesetzt. Die Klasse bietet Methoden für die Transformation ein- und zweidimensionaler Felder an, sowie Methoden für die Konvolution zweidimensionaler Felder. Die Testanwendung ist in zwei funktionsgleiche Bereiche aufgeteilt. Jeder Bereich bietet die Möglichkeit ein Bild einzuladen, es zu transformieren und es mit dem Bild des anderen Bereiches zu falten. Die Option `-1^(x+y)` verschiebt die Transformation um eine halbe Periode für jede Dimension in die Mitte des Bildes, um eine zentrierte Ansicht der Transformation zu erhalten. Das Auswahlfeld „display“ bestimmt, welcher Teil des transformierten Bildes angezeigt werden soll. Dabei kann zwischen dem Realteil, dem Imaginärteil, dem Spektrum

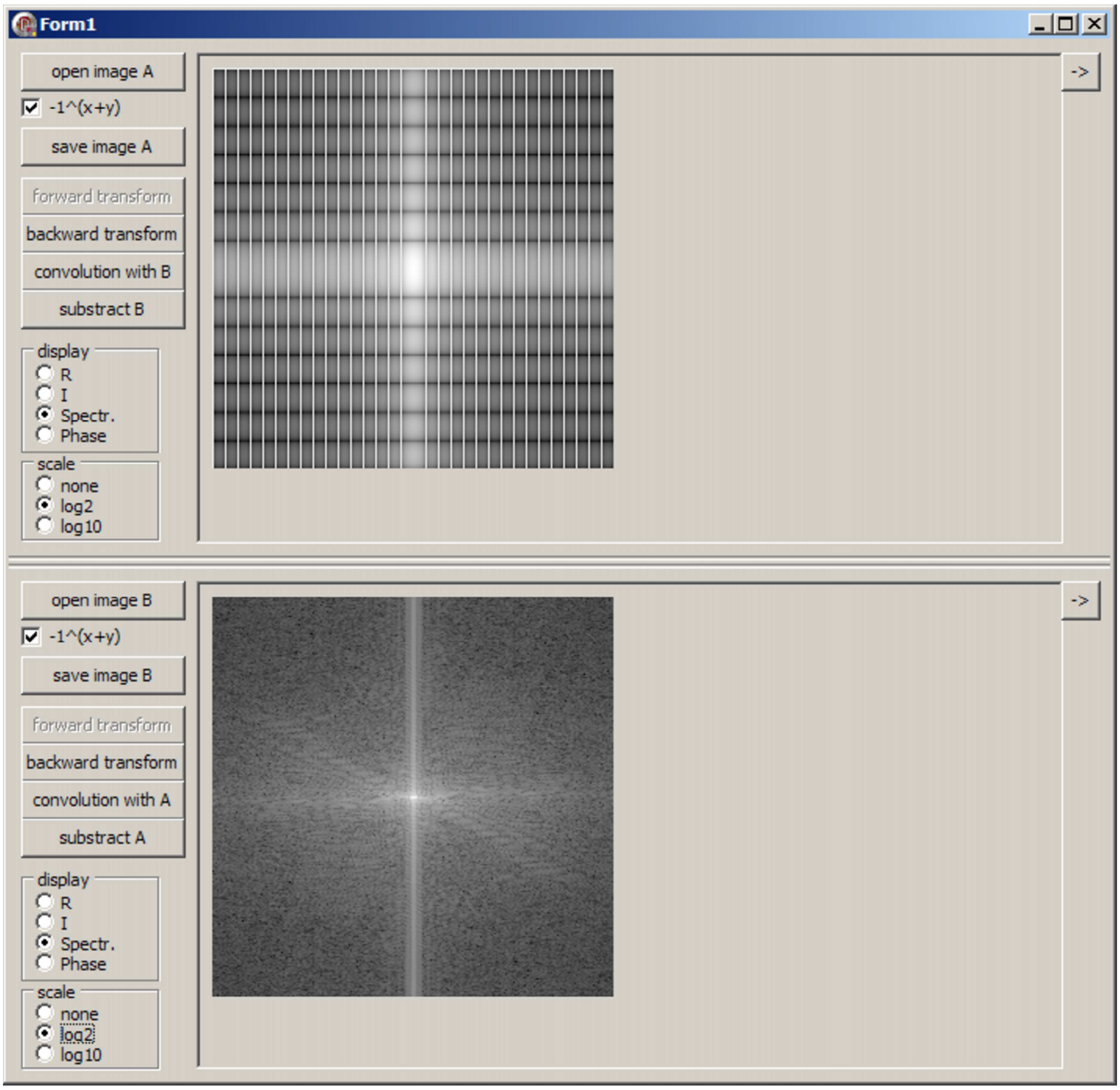


Abbildung 41: *Testprogramm für die FFT-Implementation.*

und der Phase gewählt werden. Das Auswahlfeld „scale“ bietet die Möglichkeit, anstelle der linearen Darstellung der Daten (*none*) eine logarithmische Darstellung zu verwenden (*log2* bzw. *log10*).

A.1.5 Rotationsinvarianz

Für den Test und den Vergleich der in Abschnitt 5.2.2 vorgestellten Methoden zur Bestimmung der Hauptrichtung eines lokalen Bildbereiches wurde die in Abbildung 42 gezeigte Anwendung erstellt. Die obere Ansicht zeigt die Programmoberfläche mit aktivem Reiter für das SIFT-Verfahren, die untere Ansicht zeigt die Oberfläche mit aktivem Reiter für die in dieser Arbeit vorgeschlagene Alternative.
Über die Schaltfläche „open img“ kann ein Bild eingeladen werden. Mit der Maus kann innerhalb des Bildes der Mittelpunkt des zu verwendenden lokalen Bereiches ausgewählt werden. Die Größe des Bereiches wird über das Eingabefeld „Radius“ bestimmt. Für das SIFT-Verfahren wird der lokale Bereich über einen roten Kreis markiert. Ein kleinerer blauer Kreis ($\mathrm{Radius}/3$) deutet den Radius der im SIFT-Verfahren verwendeten gaußschen Gewichtung an. Für das alternative Verfahren wird der lokale Bereich mit einem gelben Kreis markiert. Nachdem ein Bereich auf diese Weise ausgewählt wurde, kann mit dem Schieberegler oberhalb des Bildes das Bild gedreht werden. Der aktuelle Winkel wird dabei rechts vom Schieberegler in Grad angezeigt. Jede Bewegung des Schiebereglers bewirkt eine Neuberechnung der Hauptrichtung des aktuellen lokalen Bereichs über das zu diesem Zeitpunkt sichtbare Verfahren.
Ist das SIFT-Verfahren ausgewählt, befindet sich auf der rechten Seite des Programmfensters ein Balkendiagramm, das das aktuelle Richtungshistogramm darstellt. Neben dem Diagramm befindet sich eine Liste, die die Winkel der gefundenen Histogrammmaxima enthält. Unterhalb des Histogramms wird der lokale Bereich in jeder der gefundenen Hauptrichtungen gedreht angezeigt. Über das Eingabefeld „Peak Thres“ kann angegeben werden, wie stark lokale Maxima bzgl. des globalen Maximums sein müssen, um ebenfalls als eine Hauptrichtung zu gelten. Der voreingestellte Wert von `0,8` entspricht dem von Lowe in [9] angegebenen Wert. Das Eingabefeld „ScanDist“ ist eine Erweiterung des originalen Verfahrens. Bei einem Wert von 1 für ScanDist verhält sich der implementierte Algorithmus exakt gleich dem Original. Für Werte größer 1 werden die betrachteten Gradienten folgender-

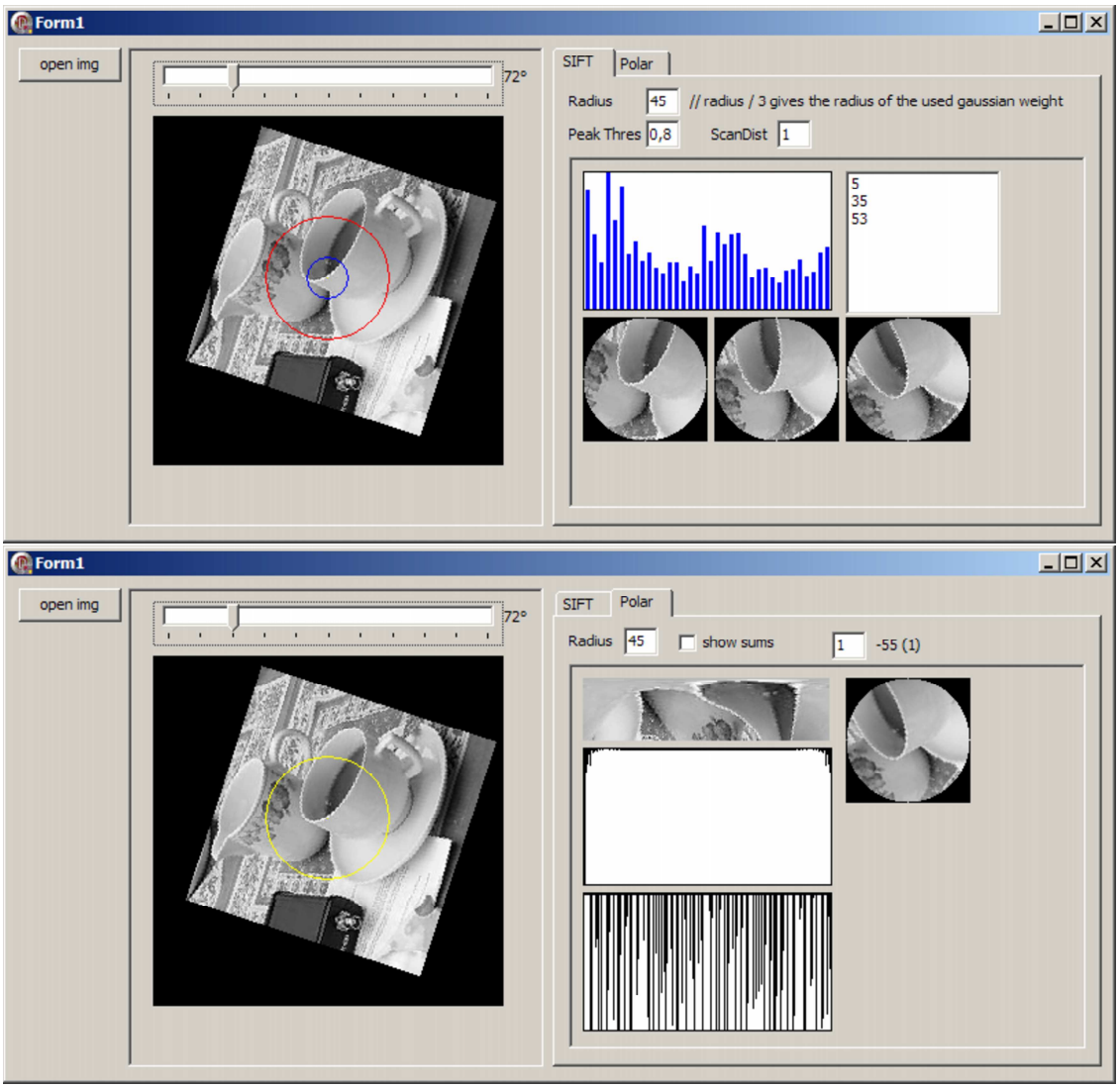


Abbildung 42: *Testprogramm zur Rotationsinvarianz mit den Parametern für SIFT (oben) und den Parametern für die hier vorgestellte Alternative (unten).*

maßen berechnet:

$$\frac{\partial I}{\partial x} := \sum_{u=1}^{ScanDist} I(x+u, y) - \sum_{u=1}^{ScanDist} I(x-u, y)$$

$$\frac{\partial I}{\partial y} := \sum_{v=1}^{ScanDist} I(x, y+v) - \sum_{v=1}^{ScanDist} I(x, y-v).$$

Ein Wert größer 1 für ScanDist kann die Ergebnisse des Verfahrens bei stark verrauschten Bildern verbessern. Bei Bildern mit geringen Störungen konnte keine signifikante Verbesserung festgestellt werden.
Für das alternative Verfahren wird auf der rechten Seite des Programmfensters der lokale Bereich in Polarkoordinaten dargestellt. Über die Option „show sums“ kann das Signal, das durch Projektion der Polarkoordinatendarstellung auf eine Dimension entsteht, über der Darstellung des lokalen Bereiches eingeblendet werden. Darunter befinden sich zwei Diagramme, die das Fourier- und das Phasenspektrum des eindimensionalen Signals anzeigen. Daneben befindet sich eine Darstellung des lokalen Bereiches, der in die gefundene Hauptrichtung gedreht wurde. Das zweite Eingabefeld gibt an, welcher Wert des Phasenspektrums für die Bestimmung der Hauptrichtung genutzt werden soll. Neben dem Eingabefeld wird die aktuelle Hauptrichtung in Grad angezeigt.

A.2 SIFT-Komponente

Für die Untersuchung des SIFT-Verfahrens wurde dieses anhand der Beschreibung in [9] als *Delphi-Komponente* implementiert. Eine Delphi-Komponente ist eine spezielle Klasse, die sich nahtlos in die IDE von Delphi integriert und dadurch sehr einfach per „drag and drop“ in anderen Anwendungen verwendet werden kann. Da es keine frei verfügbaren Quellen der Originalimplementation von Lowe für das inzwischen patentierte SIFT-Verfahren gibt, kann nicht garantiert werden, daß die für diese Arbeit implementierte Version 1:1 der Originalversion von Lowe entspricht. Gerade im Bereich des Keypoint-Detektors, der mit Differenzbildern der unterschiedlich stark mit einem Gaußfilter geglätteten Eingabe arbeitet, gibt es zahlreiche offene Fragen, die durch die Beschreibung in [9] nur unzureichend beantwortet werden. Ein Blick in den Quellcode der freien libsift[3]-Implementierung offenbart, daß der Author von

[3] http://user.cs.tu-berlin.de/~ nowozin/libsift/

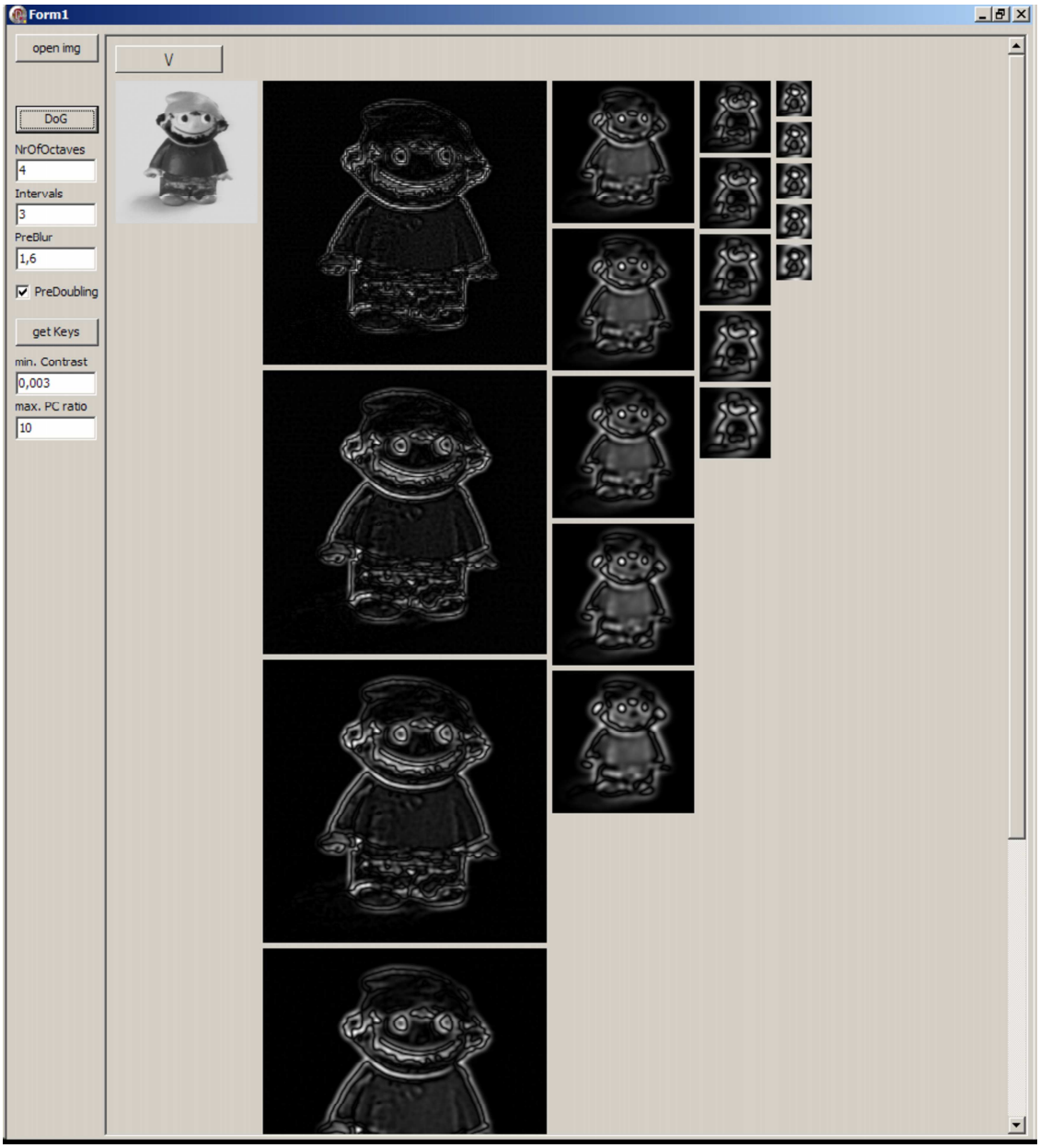


Abbildung 43: *Testprogramm für die SIFT-Komponente.*

libsift die gleichen offenen Fragen hat und es sich demnach ebenfalls nur um eine SIFT-ähnliche Implementierung handelt.
Das in Abbildung 43 dargestellte Programm dient dem Test der SIFT-Komponente und bietet insbesondere die Möglichkeit, die einzelnen Differenzbilder, die für den Keypoint-Detektor des SIFT-Verfahrens erstellt werden, zu betrachten. Über die Schaltfläche „open img“ kann ein Graustufenbild eingeladen werden. Die Schaltfläche „DoG“ startet die Berechnung der Differenzbilder mit den Parametern „NrOfOctaves“, „Intervals“, „PreBlur“, sowie „PreDoubling“. Der Parameter „NrOfOctaves“ gibt an, für wieviele Skalierungen (Oktaven) des Eingabebildes die Differenzbilder erstellt werden sollen. Der Parameter „Intervals“ bestimmt, wieviele Zwischenschritte pro Oktave berechnet werden. Dabei ist die Zahl der Zwischenschritte um 2 größer als der mit „Intervals“ angegebene Wert, da der Keypoint-Detektor für jedes zu untersuchende Differenzbild ein Vorgänger- und Nachfolgebild innerhalb der Oktave benötigt und somit die äußeren Differenzbilder einer Oktave für den Keypoint-Detektor nicht nutzbar sind. Der Parameter „PreBlur“ gibt an, wie stark das Eingabebild vor der Erstellung der Differenzbilder geglättet werden soll, um den Einfluß von möglichem Bildrauschen zu reduzieren. Die Option „PreDoubling“ gibt an, ob das Eingabebild in seiner Größe verdoppelt werden soll, bevor die Differenzbilder berechnet werden.
Mit der Schaltfläche „get Keys“ und den zugehörigen Parametern „min. Contrast“ und „max. PC ratio“ lassen sich schließlich die Keypoints auf den zuvor berechneten Differenzbildern bestimmen. Der Parameter „min. Contrast“ gibt einen Schwellwert für die minimale Stärke eines Keypoints bzgl. seiner direkten Nachbarpunkte an. Der Parameter „max. PC ratio“ (PC = principal curvature) ist ein Schwellwert für die „Eckigkeit“ eines Keypoints und soll verhindern, daß einfache Kanten als Keypoints ausgewählt werden. Die Idee entspricht dabei den Überlegungen, wie sie beim Harris-Detektor bzgl. der Eigenwerte λ_1 und λ_2 getroffen wurden. Nur wenn beide Eigenwerte eine ähnliche und ausreichende Größe haben, handelt es sich bei dem betrachteten Punkt um eine Ecke, ansonsten um eine Kante oder eine Fläche. Wenn $\lambda_1 = r\lambda_2$, dann darf r nicht größer als „max. PC ratio“ sein, damit der untersuchte Punkt als Eckpunkt betrachtet wird.

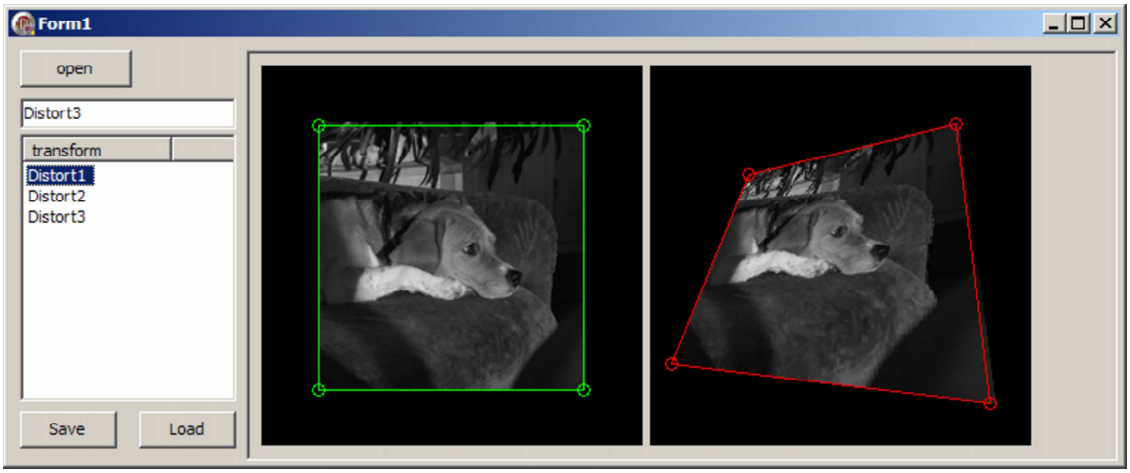


Abbildung 44: *Programm für die Definition einer Transformationsmenge.*

A.3 Programme für den Wiederholbarkeitstest der Keypoint Detektoren

Für den Test der Wiederholbarkeit der verschiedenen in dieser Arbeit vorgestellten Keypoint-Detektoren (s. Abschnitt 5.1.5) wurden drei separate Programme geschrieben. Das erste Programm (s. Abbildung 44) dient der Definition einer Menge von Transformationen. Im späteren Test wird jedes Testbild diesen Transformationen unterzogen und für jede Transformation werden die Keypoints in jedem Bild bestimmt. Durch die entsprechende Rücktransformation können schließlich die gefundenen Keypoints mit denen der untransformierten Bilder verglichen und die Wiederholrate der einzelnen Keypoint-Detektoren bestimmt werden. Um eine Menge von Transformationen mit der in Abbildung 44 dargestellten Anwendung zu definieren wird zunächst über die Schaltfläche „open“ ein Bild eingeladen. Das Bild wird in zweifacher Ausführung nebeneinander dargestellt. Es ist einmal von einem grünen, einmal von einem roten Viereck umrandet. Mit der Maus können die Ecken der beiden Vierecke interaktiv bewegt werden. Das grüne Viereck definiert einen Ursprungsbereich, das rote Viereck einen Zielbereich. Durch Eingabe einer Bezeichnung in das Eingabefeld unterhalb der „open“-Schaltfläche und Bestätigung durch die Return-Taste, wird die über die beiden Vierecke definierte Transformation in die Liste auf der linken Seite des Programmfensters eingetragen. Über die Schaltflächen „Save“ und „Load“ kann die Liste der Transformationen gespeichert oder geladen werden. Die Auswahl eines Elementes der Transformationsliste lädt die entsprechen-

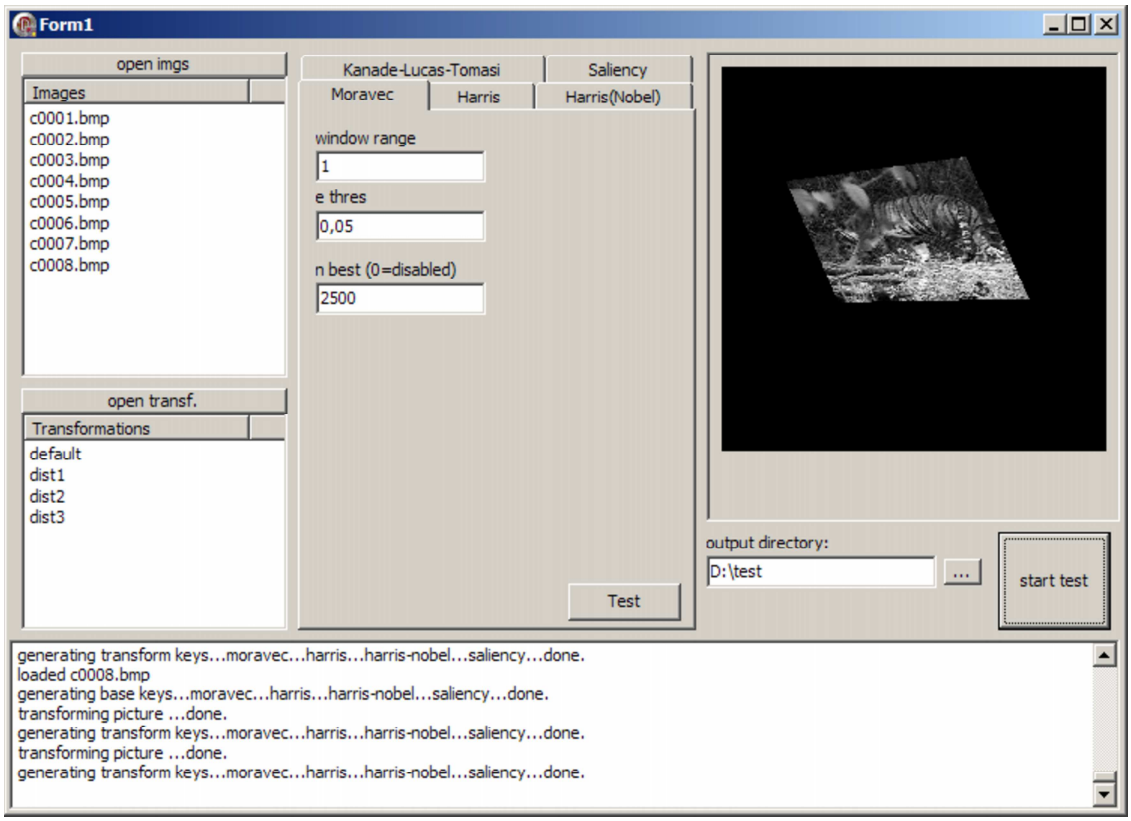


Abbildung 45: *Programm für die Durchführung des Wiederholbarkeitstests.*

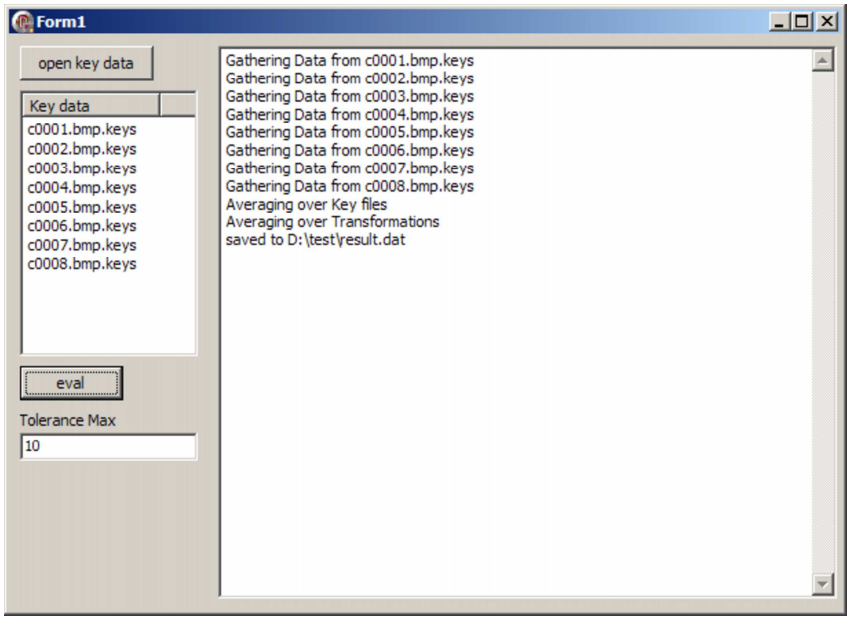


Abbildung 46: *Programm für die Auswertung des Wiederholbarkeitstests.*

de Transformation und stellt diese über die beiden Vierecke dar.
Die auf diese Weise erstellte und gespeicherte Menge von Transformationen kann mit dem zweiten Programm (s. Abbildung 45) eingeladen werden. Das zweite Programm dient der eigentlichen Durchführung des Wiederholbarkeitstests. Über die Schaltfläche „open imgs“ kann eine Menge von Testbildern eingeladen werden. Die Dateinamen der geladenen Bilder werden in einer Liste unterhalb der Schaltfläche angezeigt. Durch die Auswahl eines Elementes dieser Liste wird das entsprechende Bild auf der rechten Seite des Programmfensters in einem Vorschaubereich angezeigt. Über die Schaltfläche „open transf.“ kann die mit dem ersten Programm erstellte Liste der Transformationen eingeladen werden. Im mittleren Bereich des Programmfensters können über mehrere Reiter die Parameter für die einzelnen Keypoint-Detektoren eingestellt werden. Als Keypoint-Detektoren stehen der Moravec-, Harris- und KLT-Detektor, sowie der Detektor von Kadir und Brady zur Verfügung. Wird ein Bild im Vorschaubereich auf der rechten Seite angezeigt, so können die Parameter des aktuell ausgewählten Keypoint-Detektors über die Schaltfläche „Test“ an diesem Bild ausprobiert werden. Die erzeugten Keypoints werden dabei farbig im Bild markiert. Die Schaltfläche „start test“ startet den Test der Wiederholbarkeit auf allen Bildern, die in der Bildliste auf der linken Seite des Programmfensters aufgeführt sind und verwendet die eingeladenen und mit dem ersten Programm definierten Transformationen. Die Daten des Tests werden für jedes Testbild einzelnd in einer `.keys` Datei in dem durch das Eingabefeld „output directory“ bestimmten Verzeichnis abgespeichert. Der Aufbau dieser Dateien sieht wie folgt aus:

```
[c0001.bmp] // Name des Bildes

  [Transformation 0]    // Keypoints ohne Transformation

    [Moravec Keys]      // Keypoints des Moravec-Detektors
      90,63
      42,55
      ...
    [Harris Keys]       // Keypoints des Harris-Detektors
      42,53
      78,56
      ...
```

```
  [Harris/Nobel Keys] // Keypoints des Harris-Detektors
    42,53             // mit der Modifikation von Nobel
    78,56
    ...
  [KLT Keys]          // Keypoints des KLT-Detektors
    42,52
    78,56
    ...
  [Saliency Keys]     // Keypoints des Detektors von
    79,134            // Kadir und Brady
    142,123
    ...

[Transformation 1]    // Keypoints der ersten Transformation
  ...
```

Das dritte Programm (s. Abbildung 46) dient der Auswertung dieser Daten. Über die Schaltfläche „open key data“ können die `.keys` Dateien eingeladen werden. Die Schaltfläche „eval“ startet die Auswertung. Diese berechnet die durchschnittliche Wiederholrate der einzelnen Keypoint-Detektoren bzgl. aller verwendeten Transformation und aller verwendeten Testbilder. Über das Eingabefeld „Tolerance Max“ kann angegeben werden, bis zu welchem Abstand ein rücktransformierter Keypoint noch als wiederholt gefundener Keypoint gilt. Ist die Auswertung beendet, fordert das Programm auf, eine Datei anzugeben, in der die Ergebnisse gespeichert werden können. Eine grafische Auswertung dieser Datei kann z.B. durch gnuplot[4] erfolgen (s. Abbildung 13).

A.4 WNG-Komponente

Für das in dieser Arbeit eingesetzte wachsende neuronale Gas (s. Abschnitt 6.1) wurde eine Delphi-Komponente erstellt. Für den Test der Komponente und die Untersuchung des Verfahrens wurde die in Abbildung 47 gezeigte Testanwendung programmiert. Mit Hilfe dieser Anwendung ist es möglich, daß Verhalten des neuronalen Gases auf zweidimensionalen Daten zu beobachten. Über die Schaltfläche „open map“ kann ein Graustufenbild eingeladen werden, welches die Wahrscheinlichkeitsverteilung eines zweidimensionalen Eingaberaumes repräsentiert. Die dunklen Stellen

[4] http://www.gnuplot.info

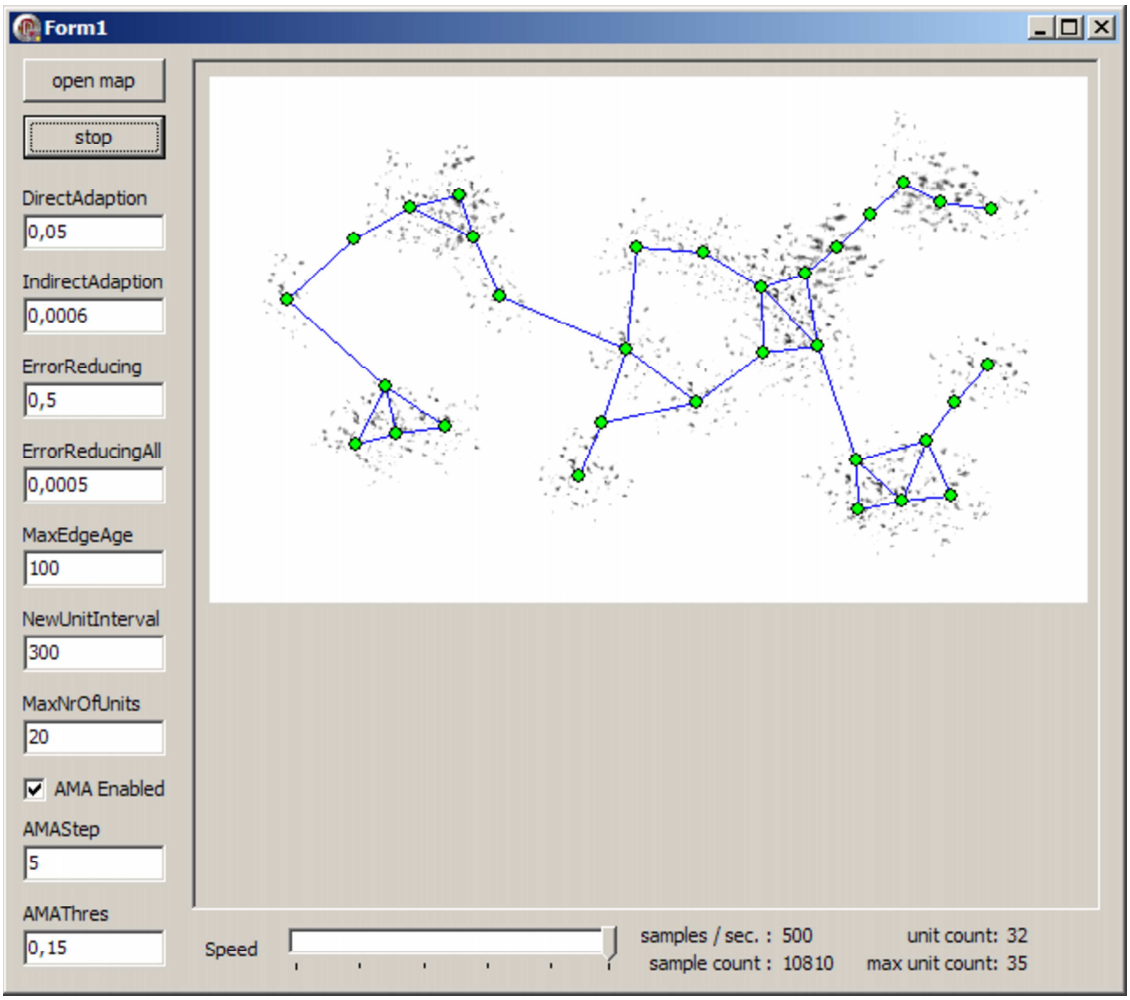


Abbildung 47: *Testprogramm für die WNG-Komponente.*

des Bildes markieren die Bereiche hoher Wahrscheinlichkeit für das Auftreten eines Eingabesignales für das wachsende neuronale Gas. Auf der linken Seite des Programmfensters befinden sich die Eingabefelder für die verschiedenen Parameter des WNG. Die Bezeichnungen im Programm weichen von den in diesem Text verwendeten Bezeichnungen ab:

Programm	Text
DirectAdaption	ϵ_b
IndirectAdaption	ϵ_n
ErrorReducing	α
ErrorReducingAll	β
MaxEdgeAge	a_{max}
NewUnitInterval	λ

Der Parameter „MaxNrOfUnits“ bestimmt die maximale Anzahl an Neuronen, die das neuronale Gas enthalten kann. Ist die automatische Größenanpassung („AMA Enabled“, AMA = Auto Max Adjust) aktiviert, so kann sich dieser Wert im Laufe des Algorithmus nach oben verändern. Der Parameter „AMAStep“ gibt an, um wieviele Neuronen der Wert „MaxNrOfUnits“ pro Anpassungsschritt vergrößert wird. Der Parameter „AMAThres“ gibt an, wie stark der durchschnittliche Grad der Neuronen bzgl. des letzten lokalen Minimums ansteigen muß, damit eine Größenanpassung erfolgt. Dieser Parameter steuert in gewisser Weise, mit welcher Auflösung das WNG die eingegebenen Werte quantisiert.
Über die Schaltfläche „start/stop“ kann die Eingabe von zufälligen der durch das Graustufenbild gegebenen Wahrscheinlichkeitsverteilung entsprechenden Signalen gestartet und gestoppt werden. Die Geschwindigkeit, mit der die Signale erzeugt werden, kann über den Schieberegler im unteren Teil des Programmfensters gesteuert werden. Die Neuronen des WNG werden als grüne Kreise an den durch ihre zugehörigen Referenzvektoren gegebenen Positionen angezeigt und sind entsprechend der durch das WNG erzeugten Topologie mit blauen Kanten verbunden.

A.5 Test der Quantisierung von Merkmalvektoren

Das zuvor beschriebene Testprogramm bietet zwar die Möglichkeit das Verhalten des wachsenden neuronalen Gases zu beobachten und die Auswirkungen der einzelnen Parameter zu studieren, jedoch werden dabei stets zweidimensionale Signale

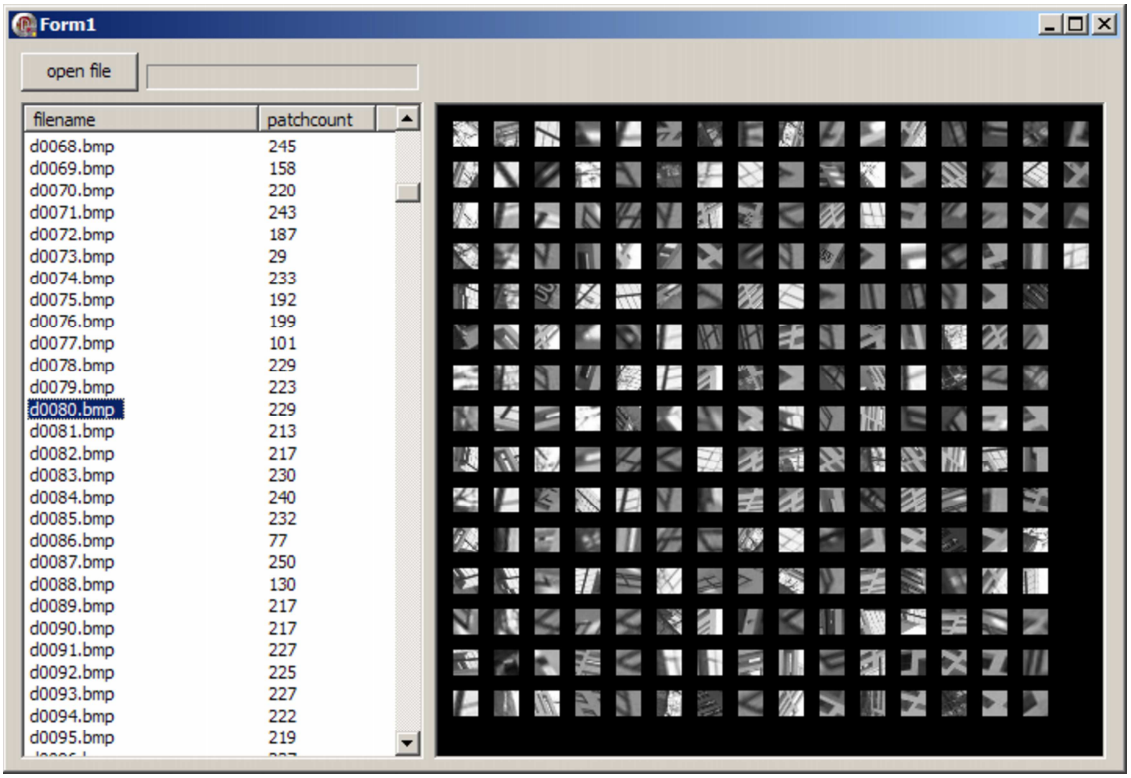


Abbildung 48: *Betrachtungsprogramm für die Testdaten der Quantisierung von Merkmalsvektoren.*

verwendet. Um das Verhalten des WNG auch für hochdimensionale Signale, in diesem Fall hochdimensionale Merkmalvektoren, beobachten zu können, wurden zwei weitere Testanwendungen erstellt. Die erste Anwendung, dargestellt in Abbildung 29, dient der Erzeugung einer großen Zahl von Merkmalvektoren, die als Testdaten für das WNG benötigt werden. Über die Schaltfläche „open images“ kann eine Menge von Bildern eingeladen werden, die anschließend auf der linken Seite des Programmfensters in einer Liste angezeigt werden. Durch die Auswahl eines Elementes der Liste wird das entsprechende Bild angezeigt. Im unteren Bereich des Programmfensters befinden sich Eingabefelder, mit denen die Parameter für den KLT-Detektor, die Parameter für die Größenbestimmung der lokalen Bereiche mittels des Verfahrens von Kadir und Brady und die Parameter für die Extraktion der lokalen Bereiche bestimmt werden können. Für jedes dieser Verfahren exisitert eine Schaltfläche „test“, mit der die jeweiligen Parameter auf dem aktuell geladenen Bild getestet werden können. Damit die Ergebnisse der einzelnen Verfahren auch angezeigt werden, muß jeweils die Option „visualize“ angewählt sein. Hat man auf diese Weise geeignete Parameter bestimmt, kann über die Schaltfläche „START“ die Detektion der Keypoints und die Extraktion der in ihre Hauptrichtung gedrehten und auf eine einheitliche Größe skalierten lokalen Bereiche auf allen Bildern gestartet werden. Das Eingabefeld „output file“ gibt dabei an, in welcher Datei die erzeugten lokalen Bereiche abgespeichert werden sollen. Über die Schaltfläche “set“ kann ein entsprechender „Datei speichern“-Dialog aufgerufen werden. Abbildung 48 zeigt ein Betrachtungsprogramm, mit dem die erzeugten Testdaten kontrolliert werden können.

Die zweite Anwendung, dargestellt in Abbildung 32, dient dem eigentlichen Test der Quantisierung von hochdimensionalen Merkmalvektoren mittels eines wachsenden neuronalen Gases. Die zuvor erstellten Testdaten können im Bereich „Patch Data“ über die „open“-Schaltfläche eingeladen werden. Zwei Listen in diesem Bereich geben Auskunft über die eingeladenen Testdaten. Eine Liste enthält alle Bilder, aus denen die Daten gewonnen wurden und zeigt für jedes Bild die Anzahl der extrahierten lokalen Bereiche an. Wird ein Element dieser Liste selektiert, so zeigt die zweite Liste alle lokalen Bildbereiche (Patches) des entsprechenden Bildes an. Durch die Auswahl eines Patches in der zweiten Liste, wird dieses in einem Vorschaubereich unterhalb der Liste angezeigt (falls die Option „preview patches“ angewählt ist). Im angrenzenden Bereich „Data Input“ kann der zu verwendende Merkmal-

deskriptor über eine Drop-Down-Box ausgewählt werden. Es stehen die Optionen „direct“, „SIFT“, „alternative“ und „fourier“ zur Verfügung. Die Option „direct“ faßt die lokalen Bereiche direkt als Merkmalvektoren auf. Auch wenn diese Art der Merkmalbeschreibung aufgrund ihrer Sensitivität bzgl. Intensitätsschwankungen und Transformationen jeglicher Art nicht wirklich als Merkmalbeschreibung geeignet ist, so ist sie jedoch für einen Menschen am einfachsten nachvollziehbar. Die Option „SIFT“ legt den SIFT-Deskriptor als zu verwendende Merkmalbeschreibung fest. Bei diesem Deskriptor ist darauf zu achten, daß die geladenen lokalen Bereiche eine Größe von 18x18 Pixeln haben. Die Option „alternative“ ist *nicht* die in dieser Arbeit vorgestellte alternative Merkmalbeschreibung. Der über „alternative“ ausgewählte Merkmaldeskriptor verwendet das Spektrum der zweidimensionalen Fouriertransformation des zu beschreibenden lokalen Bereiches und ist als Beispiel für einen schlechten Merkmaldeskriptor im Programmcode belassen worden. Die Option „fourier“ legt schließlich den in dieser Arbeit beschriebenen alternativen Merkmaldeskriptor als zu verwendende Merkmalbeschreibung fest. Die weiteren Elemente des Bereichs „Data Input“ werden erst später erläutert, da das neuronale Gas zunächst initialisiert werden muß, bevor diese Elemente verwendet werden können. Die Parameter des wachsenden neuronalen Gases können im Bereich „GNG Parameters“ bestimmt werden. Die Bezeichnungen der Parameter entsprechen denen, die im oben beschriebenen WNG-Testprogramm verwendet werden. Über die Option „outgrad info“ können zusätzliche Informationen über den minimalen, maximalen und durchschnittlichen Grad der Neuronen des neuronalen Gases angezeigt werden. Im Bereich „GNG“ kann das wachsende neuronale Gas über die Schaltfläche „init“ initialisiert und über die Schaltfläche „reset“ zurückgesetzt werden. Bevor das neuronale Gas initialisiert werden kann, müssen, wie zuvor beschrieben, die Testdaten eingeladen worden sein. Desweiteren enthält der Bereich „GNG“ eine Tabelle, die wahlweise die Distanz oder die Inzidenzbeziehung zwischen den Neuronen des neuronalen Gases beschreibt. Die Distanz zwischen den Neuronen wird dabei über eine Farbskala dargestellt, deren Minimal- und Maximalwerte ebenfalls im Bereich „GNG“ angezeigt werden. Über die Schieberegler an den Achsen der Tabelle kann die Breite bzw. die Höhe der Zellen der Tabelle eingestellt werden. Die Option „upd matrix“ gibt an, ob die Matrix während der Eingabe der Daten aktualisiert werden soll. Die Option „show last“ und das zugehörige Eingabefeld bestimmen, ob und wieviele zuletzt auf ein Neuron abgebildete lokale Bereiche im unteren Teil des Programmfensters an-

gezeigt werden sollen.
Nachdem das neuronale Gas initialisiert wurde, kann die Dateneingabe beginnen. Diese wird über die Schaltflächen des Bereiches „Data Input“ gesteuert. Die Schaltfläche „manual input“ gibt nur den aktuell selektierten lokalen Bereich (2. Liste im „Patch Data“-Bereich), beschrieben durch den gewählten Merkmaldeskriptor, in das neuronale Gas ein. Die Schaltfläche „auto 1 src“ gibt alle lokalen Bereiche des aktuell gewählten Bildes (1. Liste im „Patch Data“-Bereich) in das neuronale Gas ein. Über die Schaltfläche „auto c.“ werden alle Testdaten eingegeben. Das zugehörige Eingabefeld bestimmt dabei, wie oft dies geschehen soll. Über die Schaltfläche „stop“ kann die Eingabe der Daten angehalten werden. Dabei wird jedoch immer erst die Eingabe der Daten des aktuellen Bildes beendet, so daß es u.U. zu einer leichten Verzögerung kommen kann. Die Option „procMsgs“ legt fest, ob das Programm während der Eingabe der Daten auf Maus- und Tastatursignale des Anwenders achten soll oder nicht. Ein Verzicht führt zu einer schnelleren Eingabe der Daten. Diese läßt sich dann aber nicht mehr durch die Schaltfläche „stop“ frühzeitig abbrechen. Im Bereich „stats“ kann eine Datei angegeben werden, in die statistische Daten über den Zustand des neuronalen Gases abgespeichert werden. Die Programmierung des Bereichs „stats“ ist zu diesem Zeitpunkt noch nicht vollständig abgeschlossen, so daß die angezeigten Auswahlboxen derzeit keine Funktion haben.

A.6 Testanwendung des vollständigen Algorithmus

Für den integrierten Test aller Einzelschritte des in dieser Arbeit vorgestellten Verfahrens wurde die in Abbildung 26 gezeigte Anwendung erstellt. Für jeden Einzelschritt befindet sich im unteren Bereich des Programmfensters eine Schaltfläche, über die die entsprechenden Eingabefelder für die Parameter des jeweiligen Einzelschrittes eingeblendet werden können.
Die Parameter für den Kantendetektor werden über die Schaltfläche „edge detector“ angezeigt. Die Eingabefelder „distance“, „blur radius“, „gamma“, „gamma thres.“, sowie „length thres.“ entsprechen dabei den in diesem Text mit d, b, γ, ϵ, λ bezeichneten Parametern.
Die Parameter für den Watershed-Algorithmus werden über die Schaltfläche „watershed“ angezeigt. Das Eingabefeld „max. range“ und die Auswahlbox „measure“ sind Parameter für die Erzeugung des Höhenbildes aus dem Kantenbild und ge-

ben an, wie groß der noch berücksichtigte maximale Abstand zu einer Kante ist und welches Abstandsmaß (Euklid oder Manhattan) verwendet wird. Die Eingabefelder „skel. width“ und „skel. thres“ sind Parameter für die Skelettbildung auf dem erzeugten Höhenbild und dienen dem Auffinden geeigneter Startpunkte für das Watershed-Verfahren. Das Eingabefeld „max. grow“ ist ein Parameter für das Watershed-Verfahren und gibt an, um wieviele Bildpunkte jedes Segment in einem Schritt des simultanen Bereichswachstums vergrößert werden darf. Die Eingabefelder „average thres“, „spread thres“, „entropy thres“ und „min votes“ sind Parameter eines Nachbearbeitungsschrittes, bei dem ähnliche benachbarte Segmente miteinander verschmolzen werden. Hierfür werden die drei Eigenschaften Durchschnittsintensität, Streuung und Entropie der Segmente miteinander verglichen. Liegen für zwei Segmente mindestens „min votes“ Differenzen dieser Eigenschaften innerhalb der durch die „thres“-Parameter vorgegebenen Grenzen, so werden die Segmente miteinander verschmolzen. Die für die Berechnung der Entropie eines Segmentes benötigte Wahrscheinlichkeitsverteilung der Intensitätswerte wird über ein Intensitätshistogramm angenähert. Das Eingabefeld „hist. bins“ legt dabei fest, wieviele Bins das Intensitätshistogramm hat.

Die Parameter für den Keypoint-Detektor werden über die Schaltfläche „feature detector“ angezeigt. Die Eingabefelder „lambda thres“, „gauss sigma“, „neighbourh.“ und „n best“ sind Parameter des KLT-Detektors. Die Eingabefelder „min. radius“ und „max. radius“ sind Parameter der Größenbestimmung mittels des Verfahrens von Kadir und Brady. Die Eingabefelder „norm. width“ und „norm. height“ legen fest, auf welche Größe die gefundenen lokalen Bereiche normiert werden.

Die Parameter für das wachsende neuronale Gas und die Segmentbeschreibung werden über die Schaltfläche „segment descriptor“ angezeigt. Die Eingabefelder „direct adapt.“, „ind. adapt.“, „error reducing“, „error red. all“, „max age“, „new interval“, „init. unit cnt.“, „AMA step“ und „AMA thres“ sind Parameter des wachsenden neuronalen Gases. Das Eingabefeld „min. wavelen.“ ist Parameter der in dieser Arbeit vorgestellten alternativen Merkmalbeschreibung und gibt die Größe der kleinsten Strukturen an, die noch von der Merkmalbeschreibung berücksichtigt werden sollen. Die Eingabefelder „angle res.“, „similar depth“, „angle tol.“ und „feat. shape k“ sind Parameter der Segmentbeschreibung. Der Parameter „angle res.“ gibt an, mit welcher Winkelauflösung die relativen Winkel zwischen den Merkmalen verglichen werden. Der Parameter „angle tol.“ gibt an, wieviele Einheiten der entsprechen-

den Winkelauflösung zwei relative Winkel voneinander abweichen dürfen, so daß sie noch als minimal ähnlich betrachtet werden. Der Parameter „similar depth“ bestimmt, wie lang der Weg zwischen zwei Neuronen auf der durch das WNG erzeugten Topologie sein darf, so daß die beiden Neuronen und die damit assoziierten Merkmalbeschreibungen noch als minimal ähnlich gelten. Der Parameter „feat. shape k“ bestimmt das Verhältnis zwischen Ähnlichkeit der Merkmalfolgen und Ähnlichkeit der relativen Winkel zwischen den Merkmalen. Ein Wert von 1 führt zu einer ausschließlichen Verwendung der Ähnlichkeit der Merkmalfolgen, ein Wert von 0 führt zu einer ausschließlichen Verwendung der Ähnlichkeit der relativen Winkel zwischen den Merkmalen. Das Eingabefeld „n nearest“ bestimmt für ein Segment die Anzahl der Segmente, für die die berechneten Ähnlichkeitswerte gespeichert und für die statistische Auswertung verwendet werden.

Um das Verfahren auf eine Menge von Bildern anzuwenden, können diese über die Schaltfläche „open images“ eingeladen werden. Die Bilder werden anschließend in einer Liste unterhalb dieser Schaltfläche angezeigt. Wird ein Element dieser Liste ausgewählt, so wird das entsprechende Bild in einem kleinen Vorschaufenster unterhalb der Liste angezeigt. Über die Schaltfläche „get manually“ kann das ausgewählte Bild in den Hauptbereich geladen werden. Der Hauptbereich besteht aus einer Übersichtsspalte auf der linken Seite und einem großen Anzeigebereich auf der rechten Seite. Für die einzelnen Schritte des Verfahrens werden Miniaturansichten der Ergebnisse der Einzelschritte in der Übersichtsspalte angezeigt, insofern der jeweilige Einzelschritt schon berechnet wurde. Ein blauer Balken neben den Miniaturansichten zeigt an, welches Zwischenergebnis momentan im großen Anzeigebereich dargestellt wird. Durch Auswahl einer anderen Miniaturansicht mit der Maus, kann das zugehörige Ergebnis eines anderen Einzelschrittes in den großen Anzeigebereich geladen werden. Die Schaltflächen „manual test“ in den Parameterbereichen der einzelnen Schritte des Verfahrens erlauben es, nur den entsprechenden Schritt auf dem aktuell im Hauptbereich geladenen Bild auszuführen. Es ist jedoch notwendig, das alle Schritte, die Vorbedingung eines bestimmten Schrittes sind, zuvor auf dem gleichen Bild ausgeführt wurden. So kann z.B. der Watershed-Algorithmus erst dann über die Schaltfläche „manual test“ getestet werden, wenn zuvor der Kantendetektor ausgeführt wurde und somit ein Kantenbild als Eingabe für den Watershed-Algorithmus zur Verfügung steht. Über die Schaltfläche „use single“ auf der linken Seite des Programmfensters werden alle Einzelschritte des Verfahrens auf das ak-

tuell in der Bilderliste selektierte Bild mit den momentan eingestellten Parametern nacheinander ausgeführt. Die Schaltfläche „use all“ führt das Verfahren nacheinander auf alle in der Liste enthaltenen Bilder aus.
Wird das Segmentierungsergebnis (5. Miniaturansicht) eines Bildes im großen Anzeigebereich dargestellt, werden die einzelnen Segmente des Bildes sichtbar, wenn der Mauszeiger über die entsprechenden Bereiche des Bildes geführt wird. Verharrt man einen Moment mit dem Mauszeiger über einem Segment, werden diverse Daten für dieses Segment eingeblendet:

Bezeichnung	Erklärung
Segment:	Nr. des Segmentes
AvgVal:	durchschnittliche Intensität
Spread:	Streuung
Entropy:	Entropie
B2BRatio:	border to body ratio - Verhältnis zwischen Rand und Fläche
Neighbours:	Anzahl der Nachbarsegmente
Patches:	Merkmalfolge des Segmentes
Count:	Anzahl des Segmentes
NCount:	gewichtete Anzahl der n-ähnlichsten Segmente

Intern werden die Daten der Segmente in zwei Listen verwaltet. Die Liste `segStats` vom Typ `TList` verwaltet die Segmentdaten, die über mehrere Bilder hinweg gespeichert werden. `TList` ist eine Klasse von Delphi, die eine Liste von Pointern verwaltet. Jedes Listenelement von `segStats` zeigt auf einen record vom Typ `TSegDescr`, der ein Segment wie folgt beschreibt:

```
TSegDescr = packed record
  patches        : array of integer;
                 // enthält die Indize der Neuronen,
                 // über die die Merkmale des Segmentes
                 // quantisiert wurden

  relAngles      : array of array of double;
                 // enthält die relativen Winkel der
                 // Merkmale zueinander
```

```
    count          : double;
                   // Häufigkeit des Segmentes

    nNearest       : TList;
                   // enthält Pointer auf die
                   // n ähnlichsten Segmente (TSegDescr)

    neighbourhoods : TList;
                   // enthält Pointer auf die Nachbarschaften
                   // (TSegNeighbourhood), in denen das
                   // Segment bislang auftrat.
  end;

  TSegNeighbourhood = packed record
    neighbours : array of PSegDescr;
               // enthält die Nachbarsegmente des jeweiligen
               // Segmentes

    nNearest   : TList;
               // enthält Pointer auf die n ähnlichsten
               // Nachbarschaften (TSegNeighbourhood)
               // innerhalb der Nachbarschaften des Segmentes

    count      : double;
               // Häufigkeit der Nachbarschaft
  end;
```

Die Liste `segments` hingegen speichert die Segmentinformation, die nur während der Bearbeitung eines Bildes benötigt wird. Die Elemente der Liste zeigen auf records vom Typ `TSegment`, die die Segmentdaten wie folgt beschreiben:

```
  TSegment = record
    points     : TList;
               // enthält Pointer auf die Bildpunkte
               // (TPoint), die zum Segment gehören.

    border     : TList;
               // enthält Pointer auf die Bildpunkte
               // (TBorderPoint), die den Rand des
```

```
            // Segmentes bilden und die am jeweiligen
            // Punkt angrenzenden Nachbarsegmente.

  neighbours : TList;
            // enthält Pointer auf die Nachbarsegmente
            // (TNeighbour) des Segmentes

  keypoints  : TList;
            // enthält Pointer auf die Keypoints des
            // Segmentes (TKeypoint)

  pd         : array of double;
            // Wahrscheinlichkeitsverteilung der
            // Intensitätswerte des Segments

  avgVal     : double;
            // durchschnittlicher Intensitätswert

  spread     : double;
            // Streuung der Intensitätswerte

  entropy    : double;
            // Entropie der Intensitätswerte

  b2bRatio   : double;
            // border to body ratio
            // Verhältnis zwischen Rand und Fläche des
            // Segments
end;

TNeighbour = record
  segment    : PSegment;
            // Pointer auf die TSegment-Struktur des
            // Nachbarsegmentes

  border     : TList;
            // enthält Pointer auf die Bildpunkte
            // (TBorderpoint), die die Grenze zwischen
            // den Segmenten bilden

  localAngle : double;
            // relativer Winkel zum Nachbarsegmentes bzgl.
```

```
            // der Mitte des zugehörigen Segmentes.
end;

TKeyPoint = record
  borderPoint : PBorderPoint;
            // Bildpunkt des Keypoints

  kltVal      : double;
            // Wert des KLT-Detektors

  radius      : integer;
            // Größe des lokalen Bereiches um den
            // Keypoint

  angle       : double;
            // Hauptrichtung des lokalen Bereiches

  patch       : TGrayPic;
            // Bildpunkte des lokalen Bereiches

  localAngle  : double;
            // relativer Winkel zum Keypoint bzgl.
            // der Mitte des zugehörigen Segmentes.
end;
```

Der Speicher, der durch die Daten der `segments`-Liste belegt wird, wird nach der Bearbeitung des jeweiligen Bildes wieder freigegeben, nachdem die dauerhaft zu speichernden Segmentdaten des Bildes in der Liste `segStats` abgelegt wurden.
Die Keypoints werden bei dieser Implementation nur auf den Randpunkten der Segmente gesucht, da i.d.R. innerhalb der Segmente nur selten Keypoints gefunden werden und so durch die Beschränkung auf die Randpunkte die Berechnung der Keypoints etwas beschleunigt wird.