Verfolgung von farblich markierten Objekten
Seminarvortrag Computergrafik II

Sascha Lange

07.02.2003

Zusammenfassung

Die Bildanalyse ist neben der Bilderzeugung ein großer Bereich der Computergrafik. Es wird ein kurzer Überlick über die in diesem Gebiet verwendeten Begriffe und Methoden gegeben, um dem Neuling tiefere Einblicke in das Computer Vision Tool Kit geben zu können.

1 Bildanalyse

Im folgenden wird ein weiterer, kurzer Versuch unternommen, die Bildanalyse zu definieren und einen kurzen Überblick über die “klassiche” Bildverarbeitung zu geben. Abweichungen von diesem klassischen Paradigma in der neueren Zeit werden aufgezeigt.

Anschließend liegt die Konzentration in dieser Seminararbeit auf der Erzeugung des Regionenbildes, aber auch andere Techniken werden kurz angerissen.

1.1 Definition

Das Feld der Bildanalyse wird sehr unterschiedlich definiert und gegenüber den anderen Bereichen der Computergrafik abgesteckt. Konsenz besteht aber darüber, dass die Bildanalyse sich mit Techniken zur Generierung von Beschreibungen und Interpretation von Bildern beschäftigt. Allgemein ist die Bildanalyse der Weg von der ikonischen Darstellung des Bildes zu einer symbolischen - oder zumindest andersartigen - Beschreibung. Mit dem Wort “symbolisch” muß man an dieser Stelle hinsichtlich der neueren Entwicklungen vorsichtig sein, denn z.B. wird im Bereich des Image/Exemplar Based Content Retrieval unter Umständen auf einer Analysestufe abgebrochen, die zwar effiziente Vergleiche zwischen Bildern ermöglicht aber von einer bedeutungsvollen Beschreibung des Bildinhaltes weit entfernt ist

und unter dieser ursprünglichen Verwendung des Begriffs nicht als “symbolische Beschreibung” bezeichnet werden würde.

1.2 “Klassische” Bildverarbeitung und neuere Entwicklungen

Bei Jähne [5] findet man für die Aufgabe des Bildverstehens die in Abbildung 1 dargestellte hierarchische Strukturierung der notwendigen Bildoperationen zur Analyse eines Einzelbildes im Sinne der “klassischen” Bildverarbeitung. Die Bildanalyse wird hierbei als mehrstufiger Prozess von der Bildaufnahme bis zum Bildverstehen aufgefaßt.

Die Strukturierung enthält keinerlei Rückkopplungen zwischen den einzelnen Verarbeitungsschritten, sondern versucht, die Analyse in klare, sauber getrennte Einzelschichten aufzuteilen. Die einzelnen Operationen lassen sich auch grob in die Kategorien Bilderfassung, Bildverbesserung, Segmentierung, Bildsymbolverarbeitung und reine Symbolverarbeitung aufteilen. Die Bilderfassung bezeichnet die physikalische Erzeugung und Erfassung des Bildes mit einem Sensor. Bevor die eigentliche Analyse begonnen werden kann, gilt es oftmals das Bild zu verbessern. Die Operationen der Bildverbesserung “arbeiten” auf den erfassten Bildern - haben also sowohl als Ein- wie auch als Ausgabe ein Bild. Unter dem Oberbegriff Segmentierung werden Operationen zusammengefaßt, die Bildregionen identifizieren und markieren. Anschließend muß nun mittels bildsymbolverarbeitender Operationen eine rein symbolische Beschreibung aus dem Merkmalsbild, das noch ikonische Darstellungen der Regionen und Merkmale enthält, generiert werden. Hier schließen sich nun beliebige, rein symbolische Verarbeitungsstufen an. Nach Jähne läßt sich jede denkbare Operation in diese Hierarchie einordnen.

Ziel dieser klassischen Verarbeitungskette ist kurz gefaßt die Erzeugung einer symbolische Beschreibung des ikonischen Bildes (“Objektbeschreibung” und “Objektklassen”). Ein häufiges Zwischenziel ist das sogenannte Regionenbild.

In der neueren Zeit gibt es aber immer mehr Anwendungen, die den Umweg über eine symbolische Beschreibung im herkömmlichen Sinne nicht mehr machen. So werden zum Beispiel Reaktionen auf bestimmte visuell erfaßte Situationen oder Reize “direkt auf den Bildern” gelernt, ohne daß in einer Verarbeitungstufe eine Beschreibung des Bildinhaltes, geschweige denn einer Klassifizierung der erfaßten Objekte vorhanden wäre (Beispiel: EROSAL). Auch in der Biometrie ist eine symbolische Beschreibung bzw. Klassifizierung oftmals nicht notwendig. Vielmehr gilt es hier sogenannte Feature-Points zu matchen (s. Abbildung 2). Ein weiteres großes Feld, in dem Inhalte manchmal nicht mehr explizit repräsentiert werden, ist der Bereich des “Content Based Image Retrieval”. Dort sind unter anderem Systeme zu finden, die zu einem vorgelegt Bild oder einer einfachen Skizze Bilder mit gleichem Inhalt aufgrund der Ähnlichkeit zur Vorlage finden (Query by example). In

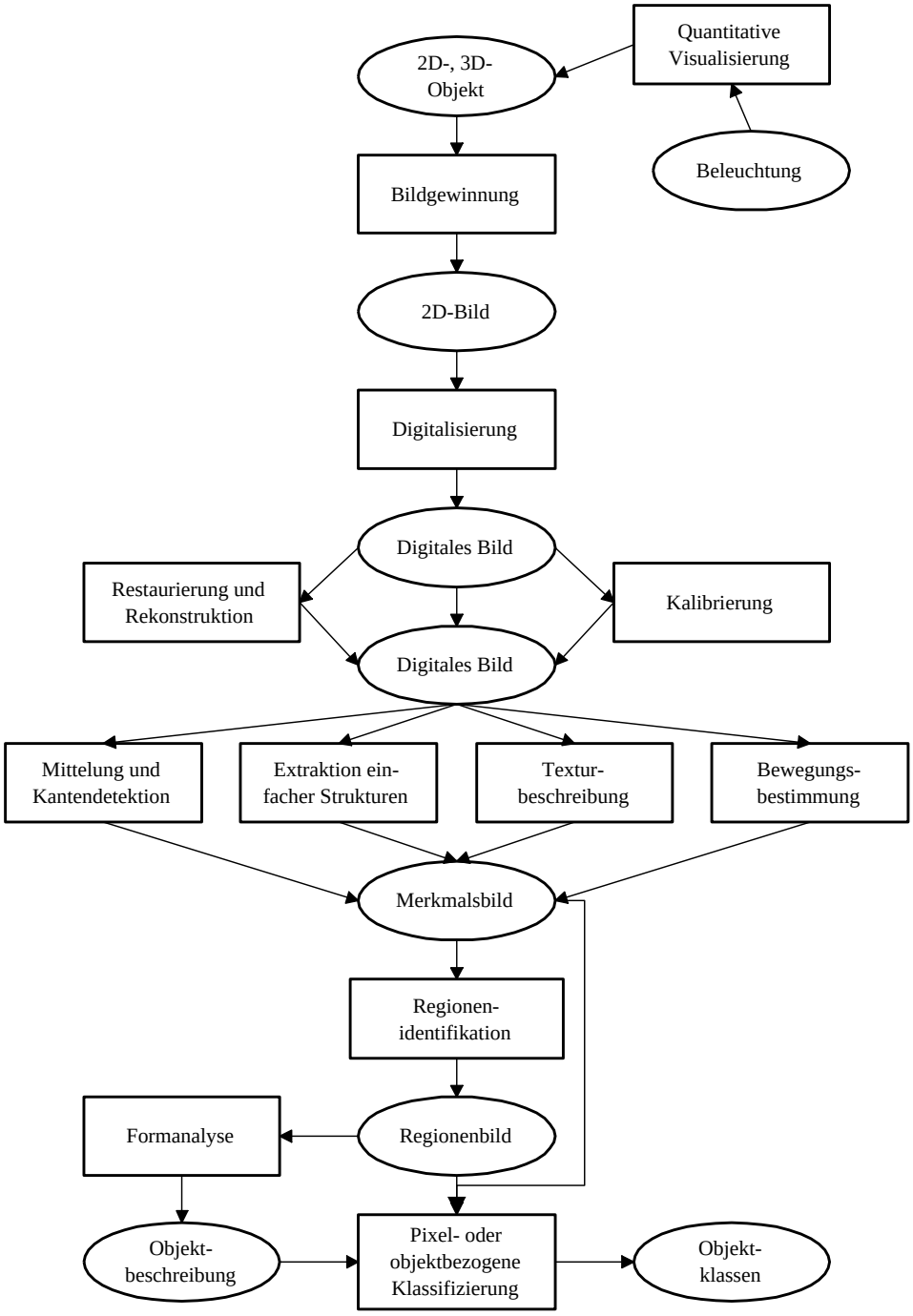

Abbildung 1: Hierarchie der Bildverarbeitungsoperationen nach Jähne. Rechtecke entsprechen den Operationen, Ovale den Datenstrukturen.

Abbildung 2: Zwischenschritte bei der Analyse eines Fingerabdruckes. Ziel ist es, die Position der rechts dargestellten markanten “Features” zu finden und mit einer Datenbank zu vergleichen. (Herkunft unbekannt)

Abbildung 3: Anfrage und Ergebnis im Query by example Verfahren. Links sieht man die Analyse des Anfragebildes, links die drei besten, bzw. ähnlichsten Ergebnisse, die das System als Antwort liefert. (Fotos von http://elib.cs.berkeley.edu/photos/blobworld/)

vielen Systemen wird nicht die Ähnlichkeit einer symbolischen Beschreibung des Bildinhaltes, sondern eine “abstrakte” Bildbeschreibung verglichen. Die verwendeten Ähnlichkeitsmaße arbeiten oftmals bloß auf der Farbe oder der Form einzelner “wichtiger” Bildregionen. Eine Garantie, daß solche ähnlichen Bilder auch gleiche Inhalte zeigen, gibt es natürlich nicht. “Bloß” eine hohe Wahrscheinlichkeit kann postuliert werden (s. Abbildung 3).

In der Tabelle 1.2 sind einige praktische Anwendungen der Bildverarbeitung aufgelistet.

1.3 Segmentierung

Ein wichtiges Zwischenziel der Verabeitungskette ist das Regionenbild, dessen Gewinnung im folgenden näher untersucht werden soll. Die zur Erzeugung des Regionenbildes verwendeten Techniken werden oftmals unter dem Oberbegriff “Segmentierung” zusammengefaßt. Ursprünglich bezeichnet “Segmentierung” eigentlich nur den Vorgang der Trennung, bzw. Hervorhebung von Bildvordergrund gegenüber dem Hintergrund. Im weiteren Sinne zählt aber auch die Identifikation und Abgrenzung einzelner Bildregionen hinzu. Diese Bildregionen dürfen aber nicht mit einer Identifikation der Objektflächen verwechselt werden. Während Objektflächen eine “erzeugende” Ursache für eine abgrenzbare Bildregion sein können,

Medizin	Tomographie, Radiologie, Gewebeschnitte
Industrielle Automatisierung	Qualitätskontrolle, Sortierung, Transport
Überwachung	Gebäudeüberwachung, Suchen in Menschenmengen
Vermessungswesen / Fernerkundung	Landvermessung, Satellitenbildauswertung, Radarbilder
Human Computer Interface	Gesture Recognition
Biometrie	Fingerabdrücke, Authentifizierung
Archivierung	Content Based Image Retrieval

Tabelle 1: Einige Anwendungen der Bildanalyse.

entstehen solche Regionen aber auch durch Schatten auf den Oberflächen, Farb- oder Texturänderungen oder einfache Überdeckungen. Bildregionen sind also nicht mit Objektflächen identisch. Um solch eine Beziehung herstellen zu können, sind vielmehr weitere, zumeist symbolische Verarbeitungsschritte notwendig.

1.3.1 Kantendetektion

Eine Möglichkeit, die Bildregionen zu finden, führt über die Regionengrenzen selbst. Mit Hilfe sogenannter Kantendetektoren können diese Grenzen gefunden werden. Eine Kante ist dabei zumeist eine abrupte Helligkeitsänderung.

Die meisten Verfahren verwenden zur Berechnung eines “Kantenbildes” eine Faltung (siehe Anhang) des zu untersuchenden Bildes mit einem bestimmten Kernel - dem Kantendetektor. Ergebnis ist eine Pixelmatrix, die pro Pixel einen Wert enthält, deren Betrag die “Kantenstärke” angibt. Je nach Kernel kann das Vorzeichen ebenfalls eine Bedeutung, z.B. über den Farbverlauf (hell nach Dunkel, Dunkel nach Hell) der Kante besitzen. Diese Bilder werden in der Regel zur Darstellung, bzw. zur Weiterverarbeitung normiert. Mit Hilfe eines Schwellwertes können nur Regionenkanten einer gewünschten Intensität durchgelassen werden.

Die einfachste Art von Detektoren, die hier vorgestellt werden soll, verwendet die 1. Ableitung des Bildes, betrachtet also den Helligkeitsgradienten. Die Ableitung wird dabei durch Differenzfilter, die teilweise das Bild vorher senkrecht zur horizontalen oder vertikalen Ableitungsrichtung mitteln, berechnet (s. Abbildung 4).

Solche mittelnden Operatoren sind die Sobel-Operatoren für die vertikale und

horizontale Kantenrichtung:

$$G_x = D_x B_y^2 = \frac{1}{8}\begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix}, \quad G_y = D_y B_x^2 = \frac{1}{8}\begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}$$

Wobei D_x den Differenzoperator

$$D_x = \frac{1}{2}\begin{pmatrix} 1 & 0 & -1 \end{pmatrix}$$

und B_y den Mittelungsspaltenoperator

$$B_y = \frac{1}{4}\begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix}$$

bezeichnen. D_y und B_x bezeichnen die entsprechenden Spalten- bzw. Zeilenvektoren.

Den Absolutbetrag des Gradienten erhält man nach getrennter Anwendung der beiden Kernel z.B. durch eine einfache euklidsche Kombination:

$$|G| = \sqrt{G_x^2 + G_y^2}$$

Natürlich geht aber durch diese einfache Kombination die in den unterschiedlichen Werten von G_x und G_y kodierte Information über die Richtung der Kante verloren.

Das Ansprechen des Sobeloperators wächst also mit der betragsmäßigen Größe des Helligkeitsgradienten. Zwar nimmt dieser Operator schon eine einfache Glättung senkrecht zur Ableitungsrichtung vor, aufgrund der kleinen Umgebung ist er aber relativ anfällig für Rauschen. Außerdem erhält man von diesem Operator nur breite, zumeist diffuse Linien.

Demgegenüber kann der Laplace-Operator, der auf der zweiten Ableitung arbeitet, Kanten jeglicher Orientierung detektieren:

$$L = \begin{pmatrix} 1 & -2 & 1 \end{pmatrix}\begin{pmatrix} 1 \\ -2 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 1 & 1 & 0 \end{pmatrix}$$

Die Verwendung der zweiten Ableitung hat gegenüber der ersten Ableitung den Vorteil, daß bei diffusen Kanten durch die Detektion des Schnittpunktes mit der x-Achse der zweiten Ableitung (Änderung des Helligkeitsgradienten) deren Mitte gefunden werden kann. Damit erhält man in jedem Falle einen klaren Kantenverlauf.

Abbildung 4: Anwendung des Sobel Operators. (oben links) Originalbild (oben rechts) Ergebnis des “vertikalen” Sobel Operators (unten links) Ergebnis des “horizontalen” Sobel Operators (unten rechts) Kombination der beiden Ergebnisse nach Anwenung eines Thresholds.

Allerdings ist dieser Operator genau wie der Sobel Operator anfällig gegenüber Rauschen.

Hildreth und Marr schlugen dagegen einen verbesserten Operator vor, der zudem den auf der menschlichen Retina vorgenommenen Verarbeitungen entsprechen soll [9]. Der Operator ist eine Kombination aus dem Laplace Operator mit einem Gauss'schen Glättungsoperator gleicher "Größe" und wird dementsprechend "Laplace of Gaussian" (kurz: LoG) oder nach seinen Begründern Hildreth-Marr Operator genannt. Der Faltungskernel berechnet sich folgendermaßen:

$$H(x,y) = c(\frac{x^2+y^2-\sigma^2}{\sigma^4})e^{-\frac{x^2+y^2}{2\sigma^2}}$$

wobei c einen normalisierenden Koeffizienten und σ die Standardabweichung des der Gaussglocke bezeichnet (nach [15]). Die diskrete Approximierung eines 5x5 Kernels sieht zum Beispiel so aus:

$$H^5 = \begin{pmatrix} 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & -2 & -1 & 0 \\ -1 & -2 & 16 & -2 & -1 \\ 0 & -1 & -2 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 \end{pmatrix}$$

Der Operator liefert einen dünnen Kantenverlauf, dessen Feinheit durch die Standardabweichung der Gaußglocke kontrolliert werden kann. Es treten keinerlei Kanten auf, die nicht durch die zweiten Ableitung unterstützt werden. Das verhindert zum Beispiel, dass in einer Objektfläche, die auf eine Lichtquelle zuläuft und deshalb einen gleichmäßigen, hohen Helligkeitsgradienten aufweist, Kanten gefunden werden.

1.3.2 Constant Thresholding

Andere Methoden versuchen nicht die Kanten, sondern die Bildregionen direkt anhand der Farbwerte zu identifizieren.

Beim "constant thresholding" wird ein konstanter Schwellwert t verwendet, um für jedes Pixel anhand des Helligkeitswertes einzeln zu entscheiden, in welche von zwei Klassen es fällt. Diese Klassen sind "Hintergrund" und "Vordergrund", hier bezeichnet mit 0 und 1. Liegt ein Mehrkanalbild, z.B. $f(x,y) = (r,g,b)$ vor, kann man als einfachste Lösung eine Funktion $l(r,g,b)$ verwenden, die eine wie auch immer geartete Luminosität berechnet und damit das Bild in ein Grauwertbild transformiert. Natürlich gibt es auch spezielle Farbsegmentierungsverfahren, die direkt mit den gesamten Kanälen arbeiten.

Das segmentierte Bild $s(x, y)$ berechnet sich dann durch die einfache Entscheidung, ob die Luminosität unter oder oberhalb des Schwellwertes liegt:

$$s(x, y) = s\left(l\left(f(x, y)\right)\right)$$

$$s(l) = \begin{cases} 0, & l \leq t \\ 1, & l > t \end{cases}$$

Auf diesem Binärbild wird eine Region nun über die Nachbarschaft der Pixel definiert: Alle Pixel, die den gleichen Wert zugewiesen bekommen haben und im Sinne der 4er oder 8er Nachbarschaft benachbart sind, gehören zur selben Region. Ausgehend von einem Pixel einer Region, läßt sich damit rekursiv die gesamte Region identifizieren.

Probleme bereiten diesem einfachen Verfahren starke Beleuchtungsgradienten oder helle zu erkennende Objekte vor einem dunklen Hintergrund, auf die starke Schatten geworfen werden. In solchen Situation ist häufig kein Schwellwert mit befriedigender Qualität (im Sinne richtig klassifizierter Pixel) zu finden.

1.3.3 Adaptive Thesholding

Um dem Problem mit ungleichmäßig ausgeleuchteten Bildern entegenzuwirken, kann man anstelle des konstanten, für das gesamte Bild gültigen Schwellwertes einen von der Bildposition abhängigen Schwellwert verwenden. Solche “adaptive” Thresholds können Helligkeitsunterschiede zwischen verschiedenen Bildregionen recht gut ausgleichen, sofern sich möglichst große, zusammenhängende Bereiche finden lassen, die relativ uniform ausgeleuchtet wurden.

An dieser Stelle seien zwei Verfahren zur Bestimmung dieses adaptiven Thresholds genannt: “Local thresholding” und Adaptive Thresholding nach Chow und Kaneko.

Beim Local Thresholding wird für jedes Pixel anhand seiner Umgebung mit Hilfe einer statischen, geeigneten Funktion (z.B. Median, Durchschnitt) ein eigener Schwellwert berechnet. Dabei kann die Umgebungsgröße variiert werden. Probleme bereiten hier aber unter Umständen uniforme Bildregionen, die größer als die betrachtete Umgebung sind. Außerdem ist dieses Verfahren relativ langsam, da sich die Vergrößerung der pro Pixel betrachteten Umgebung direkt auf die Rechenzeit auswirkt. Sei die Größe durch n (nxn Pixel) gegeben, so beträgt der Rechenaufwand unabhängig von der verwendeten Funktion zumindest $O(n^2)$.

Beim Verfahren nach Chow und Kaneko wird daher nicht für jedes Pixel die lokale Umgebung betrachtet. Das Bild wird vielmehr in n rechteckige Regionen aufgeteilt, für die jeweils wie beim einfachen Constant Thresholding ein Schwellwert berechnet wird. Nach Chow und Kaneko ist die Wahrscheinlichkeit der uniformen Ausleuchtung bei diesen Teilbildern größer. Der lokale Schwellwert für

ein Pixel wird nun durch Interpolation der Schwellwerte der Regionen berechnet. Je näher die Region am Pixel liegt, desto mehr geht deren Schwellwert ein. Bei ausreichend großen Teilbildern ist dieses Verfahren bei ähnlichen Ergebnissen wesentlich schneller als das Local Thresholding.

1.3.4 Regiongrowing

Eine weitere Methode zum direkten Identifizieren von Regionen ist das sogenannte "Regiongrowing". Für das Regiongrowing definiert man wiederum eine Nachbarschaftsrelation - üblicherweise die 4er oder 8er Nachbarschaft - und eine zusätzliche Homogenitätsbedingung für Regionen

$$H(R_i) = \begin{cases} TRUE \\ FALSE \end{cases}$$

wobei im finalen Regionenbild dann gelten muss $H(R_i) = TRUE$ für alle R_i und $H(R_i \cup R_j) = FALSE$ für alle benachbarten Regionen R_i, R_j. Zwei Regionen sind hierbei benachbart, wenn zwei beliebige Pixel $x_i \in R_i$ und $x_j \in R_j$ in der Nachbarschaftsrelation stehen. Als Homogenitätskriterien kommen zum Beispiel uniforme Helligkeit, Farbe oder Textur sowie beliebige Kombinationen in Frage.

Mittels dieser Definition eines "gültigen" Regionenbildes lässt sich iterativ ein Regionenbild direkt aus dem Pixel- oder vorsegmentierten Bild berechnen. Dabei sind im wesentlichen zwei Vorgehensweisen zu unterscheiden:

1. Beim "Region Merging" wird mit einem übersegmentierten Bild begonnen (z.B. dem Pixelbild). Die Regionen werden iterativ mit Hilfe eines Verschmelzungskriteriums verschmolzen, bis die oben definierten Bedingungen zutreffen. Als Verschmelzungskriterium kann zum Beispiel direkt $H(R_i \cup R_j)$ dienen.

2. Beim "Region Splitting" wird hingegen mit einem untersegmentierten Bild (z.B. gesamtes Bild als eine Region) begonnen, und die Regionen werden iterativ aufgeteilt (z.B. geviertelt), bis $H(R_i)$ für alle Regionen erfüllt ist.

Die beiden Verfahren können durchaus unterschiedliche Ergebnisse für das gleiche Bild liefern. Aus Gründen der Effizienz sollte beachtet werden, dass für eine Implementierung in der Regel eine Datenstruktur zur Repräsentierung der Adjazenzen benötigt wird. Das neuerliche Nachprüfen der Nachbarschaft zwischen der betrachteten mit allen anderen Regionen ist aufgrund der Definition über die Nachbarschaft zweier "Elementpixel" zu aufwändig.

Das Verfahren, dass nun tatsächlich häufig verwendet wird, ist eine Kombination dieser beiden Verfahren. Beim "Split and Merge" Verfahren wechseln sich

Schritte des Teilens und Verschmelzens ab. Die Adjazenzen werden hierbei in einem Quadtree repräsentiert, der bei den einzelnen Iterationen aufgebaut und aktualisiert wird. Die Regionen werden beim Splitten deshalb immer geviertelt.

1.4 Regionen beschreiben

Nachdem nun die Regionen identifiziert und in einem Binärbild kodiert wurden, gilt es eine platzsparendere und effizienter zu bearbeitende Beschreibung zu finden. Man macht nun den Schritt von der ikonischen Darstellung hin zur symbolischen Verarbeitung. Dabei sind in der Regel möglichst topologisch invariante Beschreibungen gewünscht.

Die Vielzahl der möglichen Darstellungen lassen sich wiederum grob in zwei Klassen einteilen:

1.4.1 Konturbasierte Beschreibungen

Konturbasierte Darstellungen beschreiben den Kantenverlauf der Region. Übliche Kodierungen sind z.B. die Kettenkodes oder die Fourierdescriptoren. Auf diesen Beschreibungen lassen sich wiederum konturbasierte Maße definieren, die für einen Vergleich zweier Regionen herangezogen werden können. Typisch sind hier zum Beispiel die “Kurvigkeit” (engl: “curvature”), die Kantenlänge / der Umfang, oder auch die Biegungsenergie (engl: “bending energy”).

1.4.2 Regionenbasierte Beschreibungen

Regionenbasierte Beschreibungen versuchen dagegen die Form der Region an sich zu analysieren, wobei der exakte Kantenverlauf “verloren” gehen kann. Beispiele sind z.B. die Skelettierung (“nicht-zerteilende” Erosion der Region bis auf ein 1-Pixel dickes Skelett) oder alle dekompositionellen Beschreibungen (z.B. mittels eines Baums). Auch auf diesen Beschreibungen lassen sich dann regionenbasierte Vergleichsmasse, je nach gewählter Beschreibung mehr oder weniger einfach berechnen. Typische Maße sind hier: Fläche, Zentrum, Höhe und Breite, die Exzentrizität (das Verhältnis zwischen Haupt- und Nebenachse), die “Länglichkeit” (engl: “Elongatedness”, größter Durchmesser durch Fläche) oder die Kompaktheit (Umfang durch Fläche).

1.5 Objekterkennung

Mit Hilfe der nun vorhandenen effizient zu bearbeitenden Beschreibungen der Bildregionen lassen sich aufwändigere symbolische Verfahren zur Objekterkennung bzw. -klassifizierung durchführen. Eine solche Klassifizierung kann nicht

mehr alleine mit den Bilddaten durchgeführt werden, es wird nun zusätzliches (Welt-)Wissen benötigt.

Zur Klassifizierung der Objekte stehen verschiedene Techniken zur Auswahl. Statistische Verfahren aus der Mustererkennung, Neuronale Netze, syntaktische Mustererkennung und das Graphenmatching seien hier exemplarisch als Vertreter genannt. Auf eine Beschreibung dieser Techniken und weiterer Optimierungsverfahren wird an dieser Stelle verzichtet.

1.6 Bildfolgen

Wenn die Verarbeitung von ganzen Bildfolgen gefordert ist, gilt es nunmehr nicht nur die Objekte in den einzelnen Bildern herauszufinden und zu bezeichnen. Zusätzlich kann auch eine Korrespondenz der in den einzelnen Bildern aufgezeichneten Objekte durch die Bildfolge hindurch hergestellt werden. Die Verarbeitung ist dann oftmals durch die Fragestellung "Wohin hat sich Objekt x aus Bild A in Bild B bewegt, bzw. ist es in Bild B überhaupt noch vorhanden?" gekennzeichnet.

1.6.1 Feature Point Tracking

Nach der Analyse der Einzelbilder sind die Positionen von sogenannten "Feature Points" im Bild bekannt. Feature Points können hier interessante Punkte (z.B. Ecken) oder auch ganze Objekte sein. Die Problemstellung für die Bildserie bleibt die gleiche: Liegt zum Zeitpunkt t eine Menge P_t von Feature Punkten p_i^t und zum Zeitpunkt $t+1$ eine Menge von P_{t+1} Feature Punkten vor, welche zwei Punkte $p_i^t \in P_t$ und $p_j^{t+1} \in P_{t+1}$ "bezeichnen" denselben Punkt, bzw. dasselbe Objekt?

Man könnte meinen, dass ein einfacher Vergleich der Position ausreichen würde und jeweils die zwei am nächsten zusammenliegenden Punkte aus P_t und P_{t+1} korrespondien. Wie man aber leicht an Abbildung 5 erkennen kann, führt eine solche globale Minimierung des Abstandes alleine nicht unbedingt zum Erfolg. Daher ist eine nähere Betrachtung dieses Problems unumgänglich, wenn Objekte durch eine Bildserie verfolgt werden sollen.

Ziel dieses "Feature Point Trackings" ist also das Finden von Trajektorien $T_i = (p_i^1, p_i^2, ..., p_i^N)$ für die durch die Einzelbilder der Serie definierten diskreten "Zeitscheiben" $t \leq N$ [10].

Zur Lösung dieser Aufgabe wird in der Regel eine "Pfadkohärenzfunktion" definiert, die eine gegebene Trajektorie bewertet. Die Funktion verwendet oftmals vereinfachte physikalische Modelle, um die Plausibilität oder Wahrscheinlichkeit einer in der Trajektorie kodierten Bewegung abzuschätzen (zum Beispiel durch die Berücksichtigung des Trägheitsgesetzes, das plötzliche, starke Beschleunigungsänderungen ohne Zusammenstöße zumindest unwahrscheinlich macht). Mit

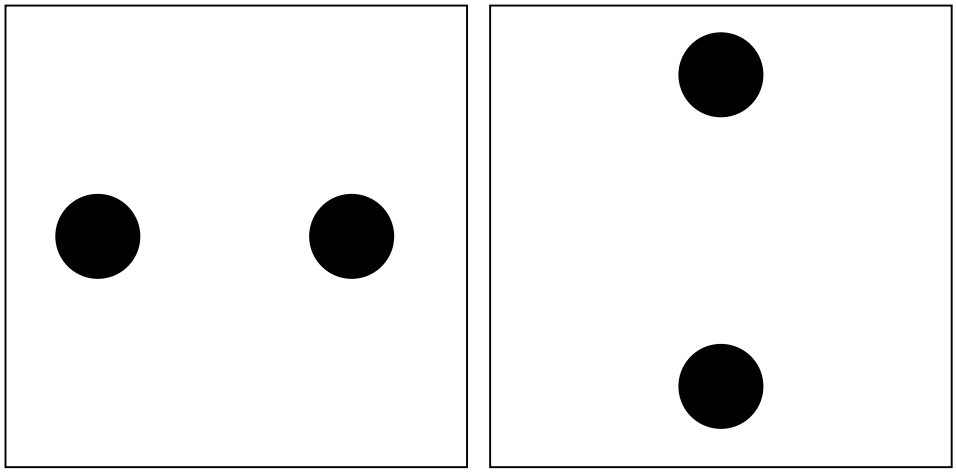

Abbildung 5: Segmentierung zweier aufeinanderfolgender Einzelbilder. Es ist nicht feststellbar, welches der beiden Objekte sich nach oben und welches sich nach unten bewegt hat. Diese Zuordnung kann nur anhand der gesamten Bewegung vollzogen werden.

einer solchen Kohärenzfunktion $h(T_i)$ erhält man dann ein Optimierungsproblem, bei dem die Trajektorien in Hinblick auf die "globale" Kohärenz (z.B. gewichete Summe von $h(T_i)$ für alle T_i) maximiert werden muß.

Die optimale Pfadkohärenzfunktion existiert hierbei alleine schon aus Gründen der Effizienz nicht, es gilt vielmehr mittels eines physikalischen Modells der zu beobachtenden Bewegungen geignete Constraints für diese Funktion zu finden. In einem System, in dem sich Objekte passiv und ohne Zusammenstösse bewegen (z.B. bei der Beobachtung von mehreren, in verschiedenen Ebenen frei schwingenden Pendel), ist in der Regel eine andere Funktion sinnvoll als zum Beispiel bei der Beobachtung eines Autoscooters, bei dem die Objekte einen eigenen Antrieb besitzen und des öfteren zusammenstossen.

Bei der häufig verwendeten Pfadkohärenzfunktion nach Sethi und Jain [14] werden zum Beispiel Richtungs- und Geschwindigkeitsänderungen berücksichtigt. Die angegebene Funktion wird dabei immer für eine Zeitscheibe $t-1$ berechnet und berücksichtigt die Positionen p_i^{t-2} und p_i^t um eine "lokale" Bewertung zu berechnen:

$$\delta\left(p_i^{t-2}, p_j^{t-1}, p_k^t\right) \quad = \quad w_1 \left(1 - \frac{\left(p_i^{t-2} - p_j^{t-1}\right)\left(p_j^{t-1} - p_k^t\right)}{\|p_i^{t-2} - p_j^{t-1}\| \|p_j^{t-1} - p_k^t\|}\right)$$

$$+ \quad w_2 \left(1 - \frac{2 \left[\| p_i^{t-2} - p_j^{t-1} \| \| p_j^{t-1} - p_k^t \| \right]^{\frac{1}{2}}}{\| p_i^{t-2} - p_j^{t-1} \| \| p_j^{t-1} - p_k^t \|} \right)$$

Der erste Term berechnet hierbei den Kosinus der Richtungsvektoren $(p_i^{t-2} - p_i^{t-1})$ und $(p_i^{t-1} - p_i^t)$ (0, wenn senkrecht zueinander), der zweite Term bestraft Längenunterschiede zwischen diesen beiden Richtungsvektoren.

Der Wert dieser Kohärenzfunktion wird nun für alle möglichen Trippel $p_i^{t-2}, p_j^{t-1}, p_k^{t-1}$ berechnet, um das globale Maximum herauszufinden und dementsprechend die Paarungen in den Trjaektorien an der Stelle $t-1$ lokal zu optimieren.

Ein Problem dieser Funktion ist natürlich das Finden der richtigen Gewichte w_1 und w_2. Außerdem wird sie oftmals in Systemen angewendet, die nicht ganz dem Modell entsprechen (z.B. im Robocup, wo der Ball mit Robotern zusammenstossen und doch plötzlich die Richtung ändern kann).

Neben dem Finden geeigneter Koheränzfunktionen stellen Ein- und Austritte von Objekten in der Bildserie ein Problem dar. Auch Überdeckungen oder zwischenzeitlich nicht detektierte Punkte breiten Probleme [13].

1.6.2 Active Vision

In den Anfängen der Computer Vision wurde die Verarbeitung von Bildfolgen eher als zusätzliche Erschwerung des Problems gesehen. Man hat sich in der Regel auf die Verarbeitung und Analyse von Einzelbildern, unabhängig von den vorhergehenden oder nachfolgenden Bildern beschränkt und allenfalls eine nachträgliche Integration der so gewonnenen unabhängigen Daten bemüht. Neuerdings wird die Verarbeitung von Bildfolgen auch als Chance bzw. als Vereinfachung des Problems wahrgenommen.

Wenn die Daten eines Einzelbildes alleine nicht für eine eindeutige Klassifizierung ausreichen, kann beim “Active Vision” das System aktiv die Kontrolle über die Kamera bzw. die Datenaquirierung im Allgemeinen übernehmen, um mit Hilfe einer vorläufigen Analyse gezielt weitere Daten zu aquirieren, die die Analyse verbessern. Hier steht aber letztendlich immer noch die Analyse eines statischen Bildes im Vordergrund.

Andere Ansätze versuchen aber aus der in einer Bildfolge erfassten Bewegung selbst Daten zu gewinnen, die sonst üblicherweise aus einem Einzelbild gewonnen wurden: Bei den Verfahren “Shape from Motion” wird z.B. die Form von Regionen, bzw. Objekten aus einer Bildserie “herausgerechnet” (Vereinfacht: Featurepunkte, die sich in der gleichen Richtung und Geschwindigkeit bewegt haben, gehören vermutlich zum gleichen Objekt).

Solche Ansätze werden außerdem durch physiologische und psychologische Untersuchungen an den Sehapparaten von Menschen und Tieren unterstützt. Als Beispiel seien hier der Mechanismus, mit dem Frösche Fliegen fangen und die "kaskadischen" Augenbewegungen des Menschen genannt, der niemals ein "komplettes Einzelbild" in einem Rutsch analysiert, sondern vielmehr geschickt lokale "Einzelanalysen" integriert.

Obwohl an dieser Stelle nicht näher auf die einzelnen Verfahren eingegangen werden kann, sei dennoch erwähnt, dass die "klassische" Aufgabenstellung "Einzelbild vollständig verstehen" "illformed" und damit nicht vollständig und korrekt lösbar ist. Man bedenke, dass die Projektion des dreidimensionalen Raumes auf eine zweidimensionale Fläche mehrdeutig ist, solange nur eine Ansicht zur Verfügung steht. Unserem Gehirn gelingt eine solche Verarbeitung, wenn wir uns ein Auge zuhalten, dennoch recht gut, weil es Erfahrungen und bestimmte Gesetzmäßigkeiten berücksichtigt. Es gibt aber eine Vielzahl von untersuchten Sinnestäuschungen, mit denen sich jeder dieser Mechanismen gezielt überlisten lässt. Erst durch das Hinzuziehen von Bildserien (oder verschiedenen Ansichten) erhält man eine wohlgeformte Problemstellung.

Durch die aktuellen Entwicklungen in der Robotik kommt solchen dynamischen, echtzeitfähigen Ansätzen ("Vision as Process") zunehmende Bedeutung bei.

1.7 Exemplar Based Image Recognition

Nicht unerwähnt bleiben soll hier ein an unserer Universtität verfolgter Ansatz, der einen komplett anderen Weg geht. Im Studentenprojekt "EROSAL" [**?**] werden Einzelbilder nicht mehr analysiert und symbolisch repräsentiert um eine Klassifizierung bzw. Entscheidung, herbeizuführen. Stattdessen werden alle gesehenen Bilder zusammen mit ihrer getroffenen Klassifizierung und einem von der Umgebung mittelbar auf diese Klassifizierung erhaltenen Feedback (Reinforcement Signal) exemplarisch abgespeichert. Wird ein neues Bild aufgenommen, so wird es zum Zwecke der Klassifizierung mit den vorhanden Exemplaren mit Hilfe eines Ähnlichkeitsmaßes verglichen. Die Klassifizierung wird dann anhand der ähnlichsten Exemplare und deren Reinforcement bestimmt.

Die Analyse geschieht hier also nicht mehr durch eine symbolische Verarbeitungskette und vorgegebenes Weltwissen, sondern wird über die Ähnlichkeit von Bildern moduliert. Interessant ist hierbei auch der Ansatz, die Bilder nicht in vorgegebene (Objekt-)Klassen einzuteilen, sondern direkt mit einer Handlung zu verknüpfen. Klassen entstehen hierbei dynamisch über die gleichen "richtigen", also die positiv verstärkten Handlungen. Sind auf zwei Bilder unterschiedliche Reaktionen nötig, um positives Reinforcement zu erhalten, gehören diese Bilder folglich auch zwei verschiedenen Klassen an.

Die Herausforderung stellt hierbei das Finden eines für die Aufgabe geeigneten Ähnlichkeitsmaßes dar. In ein solches Ähnlichkeitsmaß können dann auch zum Teil die traditionellen analytischen Schritte einfließen. Biologisch motivierte Ähnlichkeitsmaße werden zum Beispiel von den Psychophysikern des MPI in Tübingen untersucht (Stichworte: “Early Vision”, “Visual Recognition”, vgl. [1]).

2 Computer Vision Tool Kit

Im folgenden wird kurz das “Computer Vision Tool Kit” [6] vorgestellt, dass als öffentlich zugängliche Lösung für die Small Size League des Robocups [11] entstanden ist, aber auch in ähnlichen Problemstellung zur Anwendung kommen kann. So war das Aufzeichnen von Pendelschwingungen ein weiteres Benchmark.

2.1 Überblick

Das CVTK implementiert eine einfache klassische Bildverarbeitungskette von den Pixelmatrizen zu Obejktbeschreibungen (bottom up), wobei zu den einzelnen Objekten Korrespondenzen über die Einzelbilder der von der Videokamera gelieferten Bildserie hinweg hergestellt werden. Dabei wurden die einzelnen, die notwendigen Schichten realisierenden Techniken speziell für die Domäne des Robocups - globale Kamera, 2 dimensionale Spielebene und farblich markierte Objekte - und hinsichtlich der folgenden allgemeinen Designkriterien ausgewählt.

- (Echtzeitverarbeitung) Die Bildverarbeitung muß in Echtzeit, soll heißen, mit mindestens 25 fps und geringer Verzögerung arbeiten.
- (Robustheit) Das System muß möglichst zuverlässig arbeiten und auch über längere Zeit keine Fehler machen oder diese selbstständig bemerken und korrigieren können.
- (Kalibrierung) Das System muß so angelegt sein, daß der Benutzer bei der Kalibrierung unterstützt wird und die hierzu benötigten Werkzeuge leicht bereitgestellt werden können.
- (Flexibilität) Die Bilderkennung muß sich leicht auf wechselnde Anforderungen und Umgebungen anpassen lassen. Es sollte nicht ein spezielles Problem, sondern ein größerer Bereich, eine Problemklasse, abgedeckt werden.
- (Modularität) Einmal erarbeitete Datenstrukturen, Algorithmen und Oberflächen sollen möglichst wiederverwendbar und austauschbar sein.

In Abbildung 6 sieht man eine schematische Darstellung der verwendeten, zu Jähnes Klassifizierung der Bildverarbeitungstechniken analogen Schichtenarchitektur. Bemerkenswert ist hier, dass zwischen Bildgewinnung und Segmentierung keine Bildverbesserungsschichten zu finden sind. Aufgrund der Bedingung “Echtzeitverarbeitung” scheiden sinnvolle Filteroperationen wie Glättungen und Helligkeitsabgleich leider aus. Solche Operationen lassen sich bei der geforderten Auflösung (Die meisten Kameras liefern 640x480 Pixel mit 24, bzw. 32 bpp) für

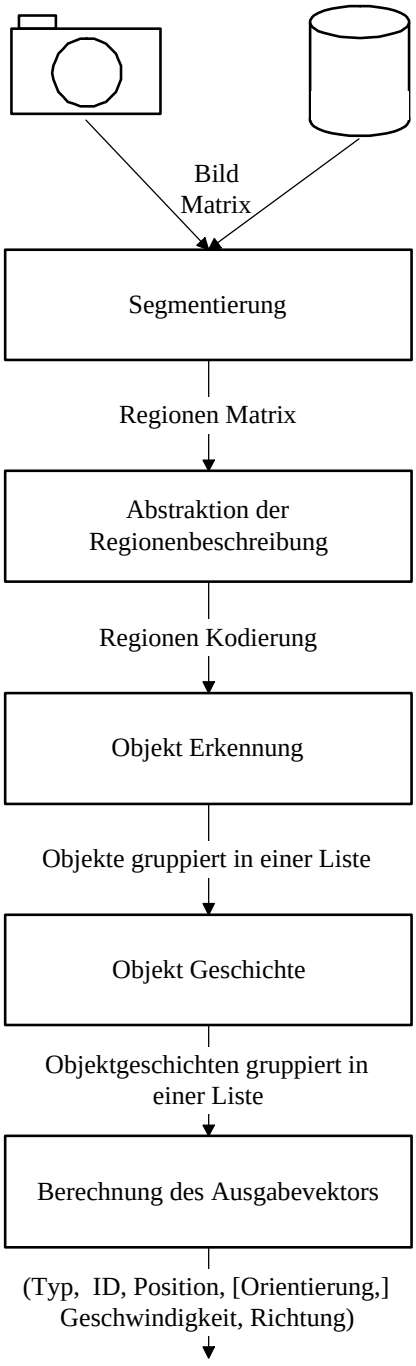

Abbildung 6: Schichtenarchitektur des Bildanalysesystems auf einer konzeptionellen Ebene.

größere Pixelumgebungen entweder nur in Hardware (Verwendung von spezialisierten DSPs) oder unter Zuhilfenahme spezieller optimierter Befehlssätze (MMX und SSI) möglich. Da solche Bildverbesserungsoperatoren aber direkt auf den Pixelmatrizen arbeiten, lassen sie sich auch noch ohne weiteres nachträglich einfügen.

2.2 Farbsegmentierung

Für CVTK wurde eine einfache Farbsegmentierung gewählt. Dabei wird der RGB-Wert jedes Pixels durch eine konstante Funktion $(s : (r, g, b) \mapsto c$ aus dem RGB Raum auf eine Farbklasse c abgebildet. 0 kodiert dabei den Hintergrund, Werte > 0 bezeichnen beliebig viele definierte Farbklassen.

Um dem Benutzer beliebige Abbildungen des RGB-Raumes auf diese Farbklassen zu ermöglichen (nicht bloß eine konstante Schwelle oder die üblichen Voronoi-Polytope) wird die eigentliche Arbeit der problembezogenen Aufteilung des RGB-Raumes in Farbklassen auf ein externes Kalibrierungstool verlagert. Diese Tools müssen eine Nachschlagetabelle vorbereiten, in der die komplette diskrete Funktion s für die übliche Kodierung des RGB-Raumes (rgb-Tripplet mit $0 \leq r, g, b \leq 255$) abgespeichert ist. Diese Nachschlagetabelle wird dann in die Segmentierungsschicht geladen, die die Farbklasse dann für jedes einzelne Pixel durch eine einfache Nachschlageoperation “berechnen” kann.

2.3 Kettenkodierung

Die in der ikonischen Darstellung vorliegenden Regionen werden nun wahlweise mit der Vierer- oder Achternachbarschaft in der Freeman - (Ketten)kodierung [4] kodiert. Dabei wird zu jeder Region ein Anfangspunkt des Kettenkodes, der auf der Kontur der Region liegt, zusammen mit einer Sequenz von Nummern abgespeichert. Die Sequenz kodiert hierbei die Richtungen in der beim Abschreiten der Kontur zu gehenden Einzelschritte (siehe Abbildung 7). Diese Art der Kodierung ermöglicht die schnelle Berechnung von Zentrum und Fläche, die in den folgenden Schichten für die Erkennung der Objekte verwendet werden sollen.

2.4 Objekterkennung

Zum Finden von Objekten in dem Bild wird die Liste der gefunden Regionen nach vorgegebenen, expliziten Beschreibung der Objekte durchsucht. Es werden mehrere Regionen, die ein bestimmtes Objekt bilden, und ihre Lage zueinander benannt. Bei der Beschreibung der Regionen werden nur die Farbe und optional die Fläche verwendet. Die Form der Regionen bleibt unbeachtet. Ein Objekt besteht aus mindestens einer Fläche, die das Zentrum markiert. Eine weitere Region des Objektes

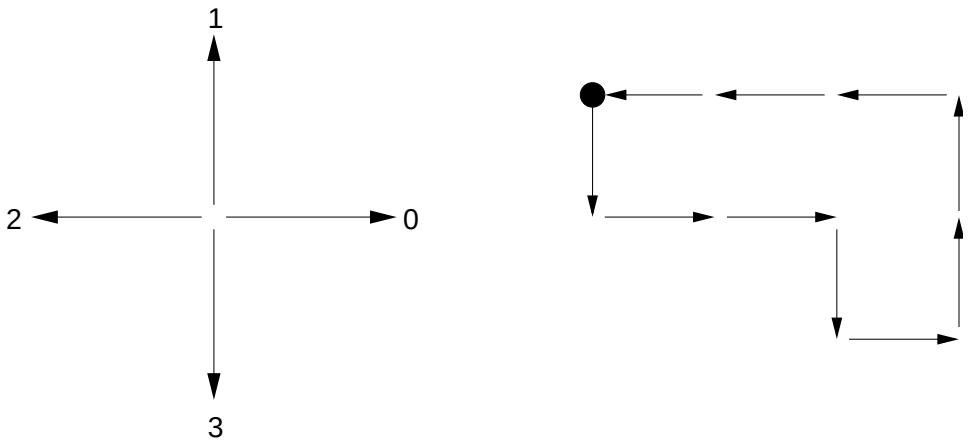

Abbildung 7: Nummerierung der Nachbarn einer 4er-Nachbarschaft und Kodierung einer Kontur. Die Kontur wird durch folgenden Kettenkode beschrieben: 3,0,0,3,0,1,1,2,2,2

etabliert dann eine Objektachse. Hier wird außerdem der (Euklidsche) Abstand $\|\vec{p} - \vec{p}'\|$ vom Zentrum angegeben:

$$\|\vec{p} - \vec{p}'\| = \sqrt{(p_1 - p_1')^2 + (p_2 - p_2')^2} \tag{1}$$

wobei $\vec{p}$ und $\vec{p}'$ die Zentren der beiden Regionen seien. Besitzt das Objekt weitere Marker, wird ebenfalls ihr Abstand zum Zentrum und ihr Winkel zu der durch die ersten zwei Marker etablierten Achse angegeben (Abbildung 8).

Diese Beschreibung ist im Gegensatz zu Schablonen rotationsinvariant und muß daher nicht in verschiedenen Rotationen getestet werden. Zur Ablage dieser Beschreibungen wird eine sehr einfache Objektbeschreibungssprache samt Parser und Generator definiert. Die Beschreibungsdateien liegen dabei in für Menschen lesbarer und dokumentierter Form vor, so daß das Editieren von Hand leicht möglich ist.

Für die Definition der Objekte wird eine einfache Grammatik verwendet:

```
BESCHREIBUNGSDATEI:
 OBJEKT
 [OBJEKT..]

OBJEKT:
 object
 ID
```

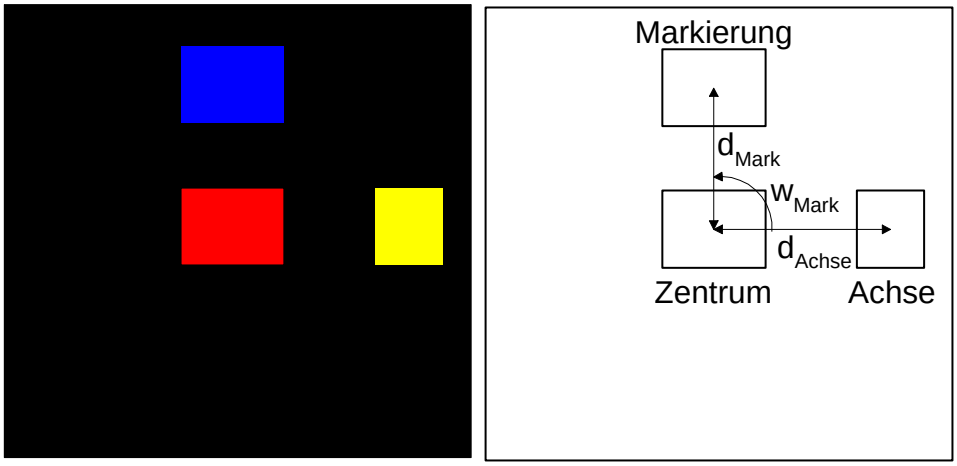

Abbildung 8: Links die Ansicht eines Objektes mit Farbmarkern und rechts die Visualisierung einer möglichen Beschreibung. d_{Mark} ist dabei der Abstand der Markierung vom Zentrum, und w_{Mark} der Winkel des Positionsvektors der Markierung zur Objektachse.

```
 ZENTRUM
 [ACHSE
  [MARKIERUNG..]]
 end

ZENTRUM:
 center FARBKLASSE [FL\"ACHE FL\"ACHE_TOLERANZ]

ACHSE:
 axis FARBKLASSE ABSTAND ABSTAND_TOLERANZ
      [FL\"ACHE FL\"ACHE_TOLERANZ]

MARKIERUNG:
 mark FARBKLASSE ABSTAND ABSTAND_TOLERANZ
      WINKEL WINKEL_TOLERANZ
      [FL\"ACHE FL\"ACHE_TOLERANZ]

ID, FARBKLASSE, ABSTAND, FL\"ACHE,
ABSTAND_TOLERANZ, FL\"ACHE_TOLERANZ,
WINKEL_TOLERANZ:
```

```
 eine positive ganze Zahl

WINKEL:
 eine positive ganze Zahl kleiner als 360
```

wobei die in eckige Klammern gesetzte Angabe der Fläche samt Toleranz eine optionale Angabe darstellt. Die in Abbildung 8 gezeigte Beschreibung könnte zum Beispiel (maßstabsfrei) folgendermaßen formuliert werden:

```
object
0
center 1 200 30
axis 0 60 10 160 25
mark 3 60 10 90 10 200 30
end
```

Eine solche Beschreibung der Objekte soll aber nicht von Hand, sondern vielmehr mit dem bereitgestellten grafischen Werkzeug erstellt werden. Hier sollen auch die Toleranzgrenzen sinvoll eingestellt werden.

Übereinstimmungen werden durch einen einfachen, intuitiven Algorithmus gesucht. Hierbei werden rekursiv die einzelnen spezifizierten Markierungen mit den gefundenen Regionen verglichen. Da die Regionen nach ihrer Farbe gruppiert vorliegen, müssen nicht für jede Markierung alle Regionen, sondern nur die mit gleicher Farbklasse verglichen werden. Nachdem eine Übereinstimmung gefunden wurde, wird nach weiteren Übereinstimmungen der gleichen Schablone gesucht, bis alle in Frage kommenden Regionen untersucht wurden. Hierbei handelt es sich im Prinzip um eine Tiefensuche in einem “Zuordnungsbaum”, bei dem in jedem Sohn eine weitere Markierung einer Region zugeordnet wird. In den Blättern des Baumes finden sich dann vollständige Zuordnungen. Die Tiefe des Baumes wird durch die Anzahl der Markierungen in der Beschreibung bestimmt und ist damit beschränkt.

Obwohl diese einfache Suche keine Heuristiken verwendet, ist sie bei der auftretenden Zahl von Objekten (11 im RoboCup) mehr als schnell genug.

Zurückgegeben werden die gefundenen Objekte nach ihrem Typ (die ID der Beschreibung, mit der sie gefunden wurden) zusammengefaßt. Ein Objekt ist dabei eine Instanz der Klasse `CObject`, wobei die Eigenschaften Klasse, Position und Orientierung gültige Werte besitzen. Die ID und die Geschwindigkeit müssen erst noch bestimmt werden.

2.5 Objektverfolgung

Für die Bestimmung der Geschwindigkeit und zur Aufzeichnung von objektbezogenen Daten über einen längeren Zeitraum gilt es, eine Korrespondenz zwischen den Objekten aufeinanderfolgender Einzelbilder zu etablieren. Können die Objekte nicht anhand eindeutiger Merkmale differenziert werden, gestaltet sich die Zuordnung als nicht triviales Problem. In einem solchen Fall hat man eine Menge $\vec{O}_{t-1}$ von m ununterscheidbaren Objekten gleichen Typs in einem Bild B_{t-1} und eine Menge $\vec{O}_t$ mit n Objekten im aktuellen Bild B_t gefunden. Hierbei bezeichne $\vec{p}_{t,i}$ die Position des i-ten Objektes aus $\vec{O}_t$. Aufgrund der schon erläuterten Möglichkeit der unvollständigen Pfade muß nicht zwingend $m = n$ gelten und jedes Objekt der einen Menge einen Nachfolger bzw. Vorgänger in der anderen Menge besitzen.

Um die Zuordnungen zu finden, soll die Pfadkohärenz nach Sethi und Jain maximiert werden. Die Minimierung wird hier durch die Formulierung als Suchproblem realisiert werden. Bezüglich unvollständiger Pfade werden einige vereinfachende Annahmen aufgestellt:

1. Ein- und Austritte treten nur einzeln auf. Wenn $m > n$ gilt, sind $m - n$ Objekte ausgetreten. Gilt $m < n$, sind $n - m$ Objekte eingetreten, gilt hingegen $n = m$, liegen weder Ein- noch Austritte vor. Diese Annahme ist nicht nur in Umgebungen, wo Ein- und Austritte nur selten auftreten, plausibel, sondern ist wohl auch mit einer hohen Wahrscheinlichkeit ganz allgemein korrekt, wenn man eine hohe Bildrate und relativ wenige Objekte voraussetzt.

2. Ein einmal ausgetretenes Objekt tritt nicht wieder ein. Das bedeutet, daß die ID eines ausgetretenen Objektes nicht wieder vergeben wird. Das Unterscheiden von Ein- und Wiedereintritten spielt für die Bestimmung der Geschwindigkeit wie auch für reaktive Agenten keine Rolle. Sollte die Entdeckung z.B. für die Generierung von Steuerbefehlen benötigt werden, muß ein Wiedereintritt in einer höheren Verarbeitungsschicht festgestellt werden.

Verdeckungen werden durch ein Fortschreiben nicht entdeckter Objekte anhand von linearen Modellen gelöst. Erst wenn das Objekt in mehreren aufeinanderfolgenden Bildern nicht wiedergefunden werden kann, wird das Objekt gelöscht und gilt als ausgetreten.

Bevor der aufwendige Matching-Algorithmus selbst angeworfen wird, werden zuerst alle Objektklassen untersucht und die Fälle aussortiert, für die eine Zuordnung ad hoc möglich ist. Eine Zuordnung kann in folgenden Fällen direkt vorgenommen werden:

n=0 Es wurden im aktuellen Bild keine Objekte gefunden. Alle Objekte der Menge O_{t-1} werden entweder anhand eines linearen Modells fortgeschrieben

oder gelöscht.

m=0 Alle Objekte sind neu eingetreten und erhalten daher eine neue Identifizierungsnummer und eine leere Geschichte.

m=n=1 Das einzige Objekt in $\vec{O}_{t-1}$ kann direkt dem einzigen Objekt in $\vec{O}_t$ zugeordnet werden.

Nur für die restlichen Fälle wird ein Algorithmus, mit dem die Schicht konfiguriert wurde, zum Finden einer wahrscheinlichen Zuordnung angestoßen.

Zum Finden der Zuordnung wird hier eine Uniform-Cost-Suche verwendet [12]. In jeder Ebene des Suchbaumes wird eine Zuordnung mehr vorgenommen. Dabei werden nacheinander den Objekten der Menge $\vec{O}_t$ alle Objekte der Menge $\vec{O}_{t-1}$ zugeordnet. Die Suche beginnt also mit einem Knoten, in dem noch keine Zuordnungen getroffen wurden. Wird dieser Knoten expandiert, werden dem ersten Objekt aus $\vec{O}_t$ in m neuen Knoten alle Objekte aus $\vec{O}_{t-1}$ zugeordnet. Wird wiederum einer dieser neuen Knoten expandiert, werden nun in $m-1$ neuen Knoten dem zweiten Objekt aus $\vec{O}_t$ alle in $\vec{O}_{t-1}$ verbliebenen Objekte zugeordnet. Ein Sonderfall ergibt sich beim Expandieren, wenn $m \neq n$ gilt. In einem solchen Fall bleiben Elemente einer Menge unzugeordnet. Dies wird beim Aufspannen berücksichtigt, indem, falls $m < n$ gilt, ein Knoten angelegt wird, bei dem das aktuell zuzuordnende Objekt aus $\vec{O}_t$ explizit als unzugeordnet gespeichert wird. Die Reihenfolge, in der der Suchbaum durchlaufen wird, wird bei der Uniform Cost Search anhand einer Kostenfunktion bestimmt. Die Söhne des "billigsten" Knotens werden zuerst untersucht. Die Uniform-Cost Search ist hierbei vollständig und optimal, d.h. es wird in jedem Fall die günstigste Zuordnung gefunden.

Als Orientierung diente die von Russell und Norvig beschriebene Suchshell wie in Abbildung 9 dargestellt, mit der sich verschiedene Suchalgorithmen von der Tiefensuche bis hin zu A* implementieren lassen. Kern dieser Implementierung sind die Expandierungsfunktion, die die Söhne eines Knoten liefert, eine Einreihungsfunktion, die die expandierten Knoten anhand der Strategie in eine Verarbeitungsliste einordnet und ein Goal-Test, der Knoten auf den Zielzustand testet.

Im Ergebnis dieser Schicht liegt dann eine Gruppierung der Objektgeschichten nach ihrem Typ vor. Jede dieser Objektgeschichten besitzt eine eindeutige ID.

2.6 Berechnung der Ausgabe

Anhand der nun vorliegenden in der "Objektgeschichte" kodierten Trajektorien kann nun die benötigte Ausgabe berechnet werden. Für jedes gefundene Objekt wird die Position, Orientierung, Bewegungsrichtung und die Geschwindigkeit zurückgegeben. Allerdings werden diese Werte z.Zt. nicht geglättet, was gerade bei lang-

```
function GENERAL-SEARCH(problem, QUEUING-FUNCTION)
    returns a soulution or failure

  nodes ⟵ MAKE-QUEUE(MAKE-NODE(
      INITIAL-STATE[problem]))
  loop do
    if nodes is empty then return failure
    node ⟵ REMOVE-FRONT(nodes)
    if GOAL-TEST[problem] applied to STATE(node)
        succeeds then return node
    nodes ⟵ QUEUING-FUNCTION(nodes, EXPAND(
        node, OPERATORS[problem]))
  end
```

Abbildung 9: Allgemeiner Suchalgorithmus nach Russell und Norvig.

samen Geschwindigkeiten größere Schwankungen hervorrufen kann. Hier ist später noch eine geeignete Glättung und die Verwendung von statistischen Methoden (Stichwort: Kalman-Filter) vorgesehen.

2.7 Ausblick

Obwohl die Bibliothek für die Small-Size-League und nur für 2-dimensionale Bilderkennungsaufgaben konzipiert wurde, werden Teile von ihr derzeit in ein Team der MidSize-League integriert. So haben die “Brainstormer Tribots” (vorläufiger Name) aus Dortmund ihre Entwicklungen ausgehend von dieser Bibliothek begonnen.

Außerdem soll diese Bibliothek die bisherige mechanische Sensorik beim in Karlsruhe und Dortmund verwendeten Pendelwagen ersetzen. Anhand der visuell erfassten Stellung eines Stabes soll ein Reinforcement Regler darauf trainiert werden, diesen freistehenden Stab auf einem Wagen zu balancieren.

A Faltung

Unter “Faltung” versteht man bezüglich zweidimensionaler, “diskreter” Pixelmatrizen folgende Operation:

$$G(x,y) = \sum_{a=1}^{A} \sum_{b=1}^{B} F(x+a-1, x+b-1) K(a,b)$$

wobei $F_{M,N}$ die Bildmatrix und $K_{A,B}$ mit $A <= M$ und $B <= N$ den sogenannten “Kernel” oder “Faltungsmaske” (engl.: “convolution mask”) bezeichnen. Das Ergebnis dieser Operation ist die kleinere Matrix $G_{M-A+1,N-B+1}$, wobei die Operation für jede Stelle $G(x,y)$ der resultierenden Matrix berechnet werden muß. Zur Berechnung der resultierenden Matrix wird der Kernel also quasi über die Bildmatrix bewegt. Dabei werden die Einträge des Kernels mit dem “darunter liegenden” Pixel multipliziert. Die Summe dieser Produkte bildet dann den Eintrag an der aktuellen Stelle $G(x,y)$.

Je nach Kernel hat die Faltung eines Bildes andere Auswirkungen. Typische Anwendungen sind Filteroperationen. So kann das Bild zum Beispiel durch Auswahl eines einfachen Gausskernels geglättet werden.

Literatur

[1] MPI for Biological Cybernetics, *Publications*, http://www.kyb.tuebingen.mpg.de/bu/publication.html, Tübingen, 2003

[2] Chetverikov, D., Verstóy, J., Feature Point Tracking for Incomplete Trajectories. *Computing*, 1999

[3] Fisher, R., Perkins, S., Walker, A., Wolfart, E., *Image Processing Learning Resources*, http://www.dai.ed.ac.uk/HIPR2/,2002

[4] Freeman, H., On the encoding of arbitrary geometric configuration. *IRE Transactions on Electronic Computers*, EC-10(2):260-268, 1961

[5] Jähne, B., *Digitale Bildverarbeitung*, 4., neubearbeitete Auflage, Springer-Verlag, Berling und Heidelberg 1997

[6] Lange, s., *Computer-Vision-Tool-Kit Projekt Dokumentation und Bachelor Arbeit*, http://www.cvtk.sf.net, Osnabrück 2001

[7] Liedtke, C.-E., *Digitale Bildverarbeitung*, Script zur Vorlesung, http://www.tnt.uni-hannover.de/edu/vorlesungen/Bildverarb/ ,Hannover 2000

[8] Lüning, A., *Kantenerkennung und Skelettierung*, Seminarvortrag Computervision, Berlin 1999

[9] Marr,D., *Vision*, W.H. Freeman and Company, New York 1982

[10] Rangarajan, K., Shah, M., Establishing motion correspondence. *CVGIP: Image Understanding*, 54:56-73, 1991

[11] RoboCup Federation, *Rules and Regulations*, http://www.robocup.org

[12] Russel, S., Norvig P, *Artificial Intelligence: A Modern Approach*, Prentice Hall, New Jersey 1995, S.46-47

[13] Salari, V., Sethi, I.K., Feature point correspondence in the presence of occlusion. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 12:87-91, 1990

[14] Sethi, I.K., Jain, R., Finding trajectories of feature points in a monocular image sequence. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 9:56-73, 1987

[15] Sonka, M., Hlavac, V., Boyle, R. *Image Processing, Analysis, and Machine Vision*, second edition Brooks/Cole Publishing Company, Pacific Grove 1999

[16] Vornberger, O., Müller, O., *Computergrafik*, Script zur Vorlesung, http://www-lehre.inf.uos.de/∼cg/2000/, Osnabrück 2000