Benutzerfreundliche Bildanalyse mit *HORUS*: gegenwärtiger Stand und Weiterentwicklungen

W. Eckstein, G. Lohmann, U. Meyer-Gruhl, R. Riemer,
L. Altamirano Robles, J. Wunderwald

Technische Universität München

Institut für Informatik – Lehrstuhl Prof. Radig

{eckstein,lohmann,meyergru,riemer,wunderwa,robles}@informatik.tu-muenchen.de

Vorgestellt werden die Architektur, die Datenstrukturen und die Hauptmodule des Bildanalysesystems HORUS. HORUS ist ein Werkzeug mit dem alle Stufen des Analyseprozesses von der Vorverarbeitung bis zur Interpretation problemorientiert durchgeführt werden können. Basierend auf hierarchischen Datentypen (von der Bildmatrix zu relationalen Strukturen) und funktionalen Operatoren ermöglichen verschiedene Module (Benutzerschnittstelle, Operatordatenbasis, automatische Konfiguration etc.) eine schnelle und sichere Programmentwicklung.

1 Einleitung

Das *HORUS*-System geht auf Arbeiten bei Prof. Pöppl [Eck86] zurück, die bei Prof. Radig in den letzten sechs Jahren weiterentwickelt wurden [Eck88]. Nach einer mehrfachen Reimplementierung stellt sich *HORUS* heute als ein ausgereiftes Werkzeug mit einem breit gestreuten Anwenderkreis dar, das eine große Palette von Operationen (ca. 460) aus den Bereichen Vorverarbeitung, Segmentation, Morphologie, Merkmalsgewinnung, Klassifikation, Visualisierung etc. umfaßt. Über 60 Diplomarbeiten, eine Reihe von Dissertationen ([Mes92] [Pau93] [Eck93] [Klo93]) und Industrieprojekten wurden für bzw. mit *HORUS* realisiert. Drei der für *HORUS* entwickelten Verfahren sind bei der DAGM prämiert worden [Hae86] [Lan91] [Kri92].

In der vorliegenden Arbeit soll der Aufbau von *HORUS*, wie er sich gegenwärtig darstellt, einschließlich der Erweiterungen für das nächste Jahr, vorgestellt werden.

2 Datenstrukturen

Aufbauend auf primitiven Datentypen (Basistypen) werden in *HORUS* hierarchische Strukturen erzeugt. Die Basistypen sind in numerische und ikonische Daten unterteilt. Als numerische Daten werden (variante) Tupel von ganzen Zahlen, Gleitpunktzahlen und Zeichenreihen unterstützt. Diese dienen als Steuerparameter für Operatoren und zur Konstruktion komplexer Strukturen wie Merkmalsräume oder Neuronale Netze. Auf die ikonischen Daten soll genauer eingegangen werden, da sie für das System von zentraler Bedeutung sind. Als Basistypen stehen *Bilder* und *Regionen* zur Verfügung.
Bilder sind als eine (Teil-)Menge von Pixeln eines rechteckigen Indexbereiches definiert. Als Pixeltypen gibt es Byte, Ganz- und Gleitpunktzahlen. Man kann ein Bild als eine Matrix mit einer Binärmaske interpretieren, wobei ein gesetztes Pixel in der Maske angibt, daß der Grauwert an dieser Position definiert ist. Die Maskierung der Bilder wird zur Laufzeitverbesserung von Filter- und Segmentationsoperationen verwendet.
Regionen sind beliebige Teilmengen von $\mathbb{Z} \times \mathbb{Z}$. Sie sind also nicht auf den Indexbereich der Bilder beschränkt. Dies hat entscheidenden Einfluß auf die Wirkung von Operationen (z.B. Morphologie), da per Definition keine Artefakte an Bildrändern auftreten (siehe [Eck93]). Weiterhin können sich Regionen überlappen, was einen flexiblen Umgang z.B. mit Segmentationsergebnissen oder ikonischen Modellen ermöglicht. Für die Darstellung unendlicher Regionen (die durch Komplementbildung entstehen) wurde folgende Lösung gefunden: Da sich die Komplementoperation durch Vereinigung, Durchschnitt und Differenz darstellen läßt (z.B. $\overline{A} \cup \overline{B} = \overline{A \cap B}$ oder $A \cap \overline{B} = A \backslash B$) reicht es, die Komplementbildung virtuell auszuführen (durch Setzen eines Bits) und immer mit der endlichen Version zu rechnen. Hierdurch erhält man eine formal saubere Implementierung von Regionen[1] und somit eine *korrekte* Implementierung der Morphologie.

Basierend auf den Typen Bild und Region und den darauf anwendbaren Operationen werden Objekthierarchien aufgebaut. So werden z.B. Graubilder zu Farbbildern, Einzelbilder zu Bildfolgen und zweidimensionale Bilder zu Voxelstrukturen aggregiert. Die Konstruktion erfolgt dabei in einer Objekthierarchie (Smalltalk), bei der Operationen (für Verarbeitung und Darstellung) und Eigenschaften vererbt werden. Für die ikonischen Objekte hat sich der Name *Hodget* (*HORUS*-widget) eingebürgert.

Zur Interpretation von Bildern werden semantische Netze verwendet. Die Knoten der Netze sind Hodgets (z.B. Kanten, Bögen, Flächen etc.), die mit Attributen versehen sind. Die Hodgets können mit beliebigen numerischen Relationen verknüpft werden.

3 Modulstruktur

Grundlage des *HORUS*-Systems ist die *Operatordatenbasis*. Auf sie greifen alle übrigen Module zu, um dynamisch Informationen über Konfiguration und Status des Systems

[1] Abgesehen von den Regionen R, für die gilt: $|R| = |\overline{R}| = \infty$.

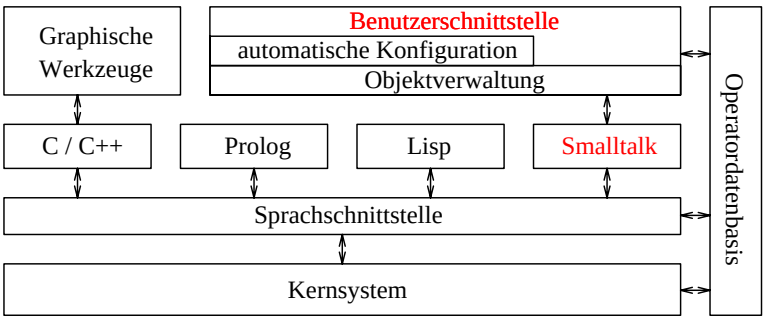

Abbildung 1: Schichtenarchitektur von *HORUS*

zu erhalten. Aussagen über Operationen (z.B. bei automatischer Konfiguration von Operatorfolgen, dynamische Struktur der Benutzerschnittstelle oder Hypertext-Manual) werden aus ihr abgeleitet. Die Abwicklung des Aufrufs von Operationen (Typisierung, Wertebereiche, Fehlermeldungen etc.) wird ebenfalls mit Hilfe der Operatordatenbasis realisiert. Das nächste Modul ist das *Kernsystem*. Es beinhaltet alle Operationen, die Ein-/Ausgabe und die Verwaltung der komplexen Datenstrukturen. Angesprochen werden die Operationen durch die *Sprachschnittstelle*, die mit Hilfe eines Compilers aus der Operatordatenbasis generiert wird. Sie ermöglicht den Zugriff auf unterschiedliche, für die Bildanalyse gängige Wirtssprachen. Basierend auf *HORUS*/C werden „einfache“ graphische Werkzeuge zur Entwicklung (Programmierung, Visualisierung und Online-Manual) zur Verfügung gestellt. *HORUS*/Prolog und *HORUS*/Lisp eignen sich zur Entwicklung komplexer Bildinterpretationssysteme ([Mes92] [Pau93]). Die Schnittstelle zu Smalltalk ist die Basis zur Entwicklung objektorientierter Tools.

4 Operatordatenbasis

In der Operatordatenbasis liegen bisher der Name und die Stelligkeit der Operatoren, sowie die Typisierung, Zusicherungen und Vorschlagswerte für die Parameter formalisiert vor. Zu den einzelnen Operatoren existiert eine umfangreiche natürlichsprachliche Beschreibung, die auf die Wirkung des Operators eingeht, seine Komplexität, die Operatorklasse sowie Querverweise und Alternativen angibt.

Die geplante Erweiterung des *HORUS*-Systems zu einem Werkzeug für Computer Aided Vision Engineering (CAVE) [Rad92] macht es erforderlich, auch die natürlichsprachliche Information zu formalisieren. Besondere Aufmerksamkeit verdient in diesem Zusammenhang die Wirkung der Parameteränderungen. Ein Beispiel hierfür ist etwa der Einfluß der Filtergröße eines Glättungsfilters auf die minimale Größe von Regionen. Bei einer Segmentation interessieren z.B. die Anzahl und Größe der Objekte sowie der Anteil der Rauschobjekte. Die Ergebnisse von statistischen und analytischen Untersuchungen über den Einfluß der Eingabeparameter auf das Operatorverhalten, sowie Abhängigkeiten

und Wechselwirkungen zwischen Operatoren werden in der Datenbasis bereitgestellt.

Besondere Anforderungen werden an die Struktur der Datenbasis gestellt, da unterschiedliche Schichten des Systems Informationen von ihr benötigen. So erwartet beispielsweise eine Sprachschnittstelle den Namen, die Stelligkeit und die Typisierung einer Operation, während die Benutzerschnittstelle Anfragen über die Wirkung einer Parameteränderung stellt.

Eine Strukturierung der Daten ergibt sich mit der Einführung von *Molekülen*, die aus einfachen Operationen zusammengesetzte Operatoren sind (siehe Abschnitt Graphische Oberfläche). Informationen über benutzerdefinierte Moleküle z.B. über die Semantik der Parameter (Wirkungsweise, Restriktionen, Voreinstellungen) müssen in die Datenbasis eingefügt werden können. Das Wissen über die erzeugten Moleküle wird dabei, soweit möglich, aus dem Wissen über die verwendeten $HORUS$-Operationen (*Atome*) propagiert, und wo dies nicht möglich ist, vom Programmierer spezifiziert. Informationen sind von unterschiedlicher Dauer:

- fest für $HORUS$-Atome und für Moleküle, die zum Standardsatz gehören
- variabel für von einer Arbeitsgruppe bzw. einem Benutzer definierte Moleküle
- temporär für Moleküle, die nur in einer Sitzung Verwendung finden.

Es werden auch verschiedene Sichtweisen auf die Datenbasis angeboten, einerseits um bei der Anpassung an eine bestimmte Zielhardware die Menge der zur Verfügung stehenden Operatoren einzuschränken und andererseits um Spezialwissen von unterschiedlichen Anwendungsdomänen aufzunehmen.

Um den genannten Anforderungen gerecht zu werden und zur Anpassung an die Benutzeroberfläche ist die Realisierung der erweiterten Datenbasis in objektorientierter Technologie geplant. Die zuletzt genannten unterschiedlichen Sichten werden mit Hilfe von Vererbung realisiert.

Die Sammlung von Bildinterpretations- und Bildanalysewissen wird ergänzt (bisher 1.5 MByte) und auch unabhängig von $HORUS$ einsetzbar sein. Die ausführliche Dokumentation der einzelnen Operatoren unterstützt die Wiederverwendbarkeit der Software. Auch bietet die Operatordatenbasis die Grundlage für ein verbessertes Hilfesystem (Hypertext-Manual) und ein Tutorsystem [Klo93].

5 Automatische Konfigurierung

Das künftige $HORUS$-System wird die Produktivität bei der Erstellung von Bildanalyseanwendungen weiter steigern. Standardmäßige, einfach spezifizierbare Bildanalyseaufgaben muß der Entwickler nicht mehr en detail ausführen, sondern er kann dies einem System zur automatischen Konfiguration überlassen.

[Mes92] stellt einen prototypischen Konfigurierer für $HORUS$-Operatoren (FIGURE) vor, der beispielsweise bei der Analyse von Straßenverkehrsszenen mit morphologischen Operatoren gute Ergebnisse erzielt.

FIGURE folgt dem hierarchischen Planungsparadigma ([McD82]): Der Entwickler spezifiziert seine Bildanalyseaufgabe in symbolischer Weise oder indem er das zu erkennende Objekt im Eingabebild interaktiv markiert. FIGURE plant nun zunächst eine Folge abstrakter Bildverarbeitungsoperatoren, verfeinert diese zu $HORUS$-Operatoren und wählt für alle in Betracht kommenden Operatorsequenzen geeigenete Parameter. Schließlich entscheidet die Anwendung der synthetisierten Programme auf das Eingabebild über das Ergebnis des Konfigurierungsprozesses, wobei die Übereinstimmung der eingezeichneten Kontur mit der entdeckten Region den Ausschlag gibt.

Die obige Vorgehensweise funktioniert nur für eine eingeschränkte Menge von Operatoren und Bildern. Deshalb verzahnt der neue Konfigurierer die Planung mit der Ausführung des Bildanalyseprogramms. Für die schrittweise Planung entwickeln wir Beurteilungskriterien für Zwischenergebnisse der Bildanalyse. Die Fähigkeit, einen nächsten Operator im Hinblick auf ein Analyseziel zu wählen, erweist sich als nützlich beim graphisch arbeitenden CAVE zur kontextsensitiven Vorschlagsgenerierung.

Im Gegensatz zum bisherigen Prototypen arbeitet der künftige Konfigurierer mit dem vollen Umfang der $HORUS$-Operatoren, wie er in der weiterentwickelten Operatordatenbasis repräsentiert ist. Besonderes Augenmerk legen wir auf die Gewinnung von Wissen über günstige Operatorenwahl und –reihenfolgen, sowie Parametrisierung. Dieses Wissen hängt stark von der Aufnahmetechnik und den Bildinhalten ab. Wir untersuchen, inwieweit sich hier Klassen bilden lassen um zu einer sinnvollen Taxonomie zu gelangen.

FIGURE erlaubt nur die Erkennung einzelner, unstrukturierter Objekte. Meistens interessieren jedoch strukturierte Objekte, wobei erst die Lagebeziehungen zwischen Teilobjekten eine robuste Objekterkennung ermöglichen. Darum verwendet der neue Konfigurierer die neuerdings in $HORUS$ vorhandene Möglichkeit, relationale Strukturen darstellen zu können.

Zu den Fähigkeiten des geplanten Synthesesystems soll auch die Erfüllung gewisser Restriktionen bei der Planung gehören. Beispielsweise erfordern viele Anwendungsbereiche die Einhaltung von Zeitschranken. Ebenso wollen wir für spezielle Bildverarbeitungsrechner mit einer eingeschränkten Zahl von Operatoren konfigurieren können. Für diese Zwecke soll der eigentliche Planungsprozeß unverändert bleiben, aber eine modifizierte Sicht auf die Wissensbasis haben.

6 Automatische Generierung von Objektmodellen

Ein weiteres Modul in $HORUS$ dient zur automatische Generierung von Modellen für zweidimensionale, strukturierte Objekte. Darüberhinaus werden die Datenstrukturen für die Repräsentation der relationen Strukturen aufgebaut und verwaltet. Im einzelnen wer-

den folgende Funktionen realisiert:

Automatische Verbesserung: Ausgangspunkt ist ein Einzelbild, welches das zu modellierende Objekt ohne Fremdobjekte beinhaltet. Es wird vorausgesetzt, daß das Objekt zweidimensional ist. Durch eine Analyse des Hindergrundes wird das Rauschverhalten geschätzt, um gegebenenfalls geeignete Operatoren und deren Parametrisierung auszuwählen und die Bildqualität zu verbessern ([Kun90]).

Automatische Kantenberechnung: In dem verbesserten Bild werden Kanten für die Objektsegmentation extrahiert. Zur Bewertung des Kantenbildes werden Gütemaße wie Kantenlänge, Kantenanzahl, usw. abgeleitet um eine automatische Adaption der Filterparameter (Filtergröße, Trennfrequenz etc.) zu erhalten.

Perceptual Grouping: Zur Reduktion irrelevanter Kanten wird das Verfahren des *Perceptual Grouping* eingesetzt. Als Ergebnis erhält man eine Menge von zusammenhängenden Kanten, d.h. Kanten, die die Gruppierungsbedingungen von Kolinearität, Parallelität und Adjazenz ([Den92]) nicht erfüllen, werden entfernt. Wichtig dabei ist, daß die Gruppierung ohne Vorwissen über Gestalt der Objekte durchgeführt wird.

Merkmalsgewinnung: Der erste Schritt zum Aufbau des Modells ist die Generierung der Knoten in der zugrundeliegenden relationalen Struktur. Dies sind Merkmale, die aus den Kanten und dazwischenliegenden Regionen abgeleitet werden. Beispiele hierfür sind Geradenstücke, Bögen, Kreise, Polygone, sowie Fläche, Form, Intensität, Farbe etc.

Berechnung der Schlüsselmerkmale: Um einen effizienten Zugriff auf die Knoten (Merkmale) für das Matching zu ermöglichen, wird deren Signifikanz nach dem Ansatz von [Kno86] bewertet und entsprechend sortiert. Verwendet wird hierfür beispielweise die Häufigkeit oder die Komplexität eines Merkmals.

Relationen zwischen Merkmalen: Im nächsten Schritt werden Relationen zwischen Merkmalen, d.h. Kanten zwischen den Knoten in der relationalen Struktur, aufgebaut. Ein Beispiel für eine solche Relation ist die Lagebeziehung „`Merkmal A ist oberhalb von Merkmal B`“ oder „`Merkmal A berührt Merkmal B`“. Hierdurch wird die räumliche Struktur des Modells aufgebaut. Anfragen an das System, wie etwa: „`Was liegt in der Nähe des Merkmales A`“ können hiermit beantwortet werden ([Cha89]). Der gleiche Ansatz kann verwendet werden, um Relationen zwischen Modellen zu erzeugen.

Modellbewertung: Abschließend wird das Modell bewertet. Hierzu wird ein Matching des Modells mit Objekten (d.h. mit deren Modell) in anderen Bildern durchgeführt. In diesen Bildern können auch Objekte auftreten, die nicht trainiert wurden. Das Matching wird mit Hilfe einer heuristischen Graphsuche durchgeführt ([Gri90]).

Durch eine Bewertung des Vergleichs erhält man einen Faktor, der angibt, wieviel Prozent der gesamten Objekte erkannt wurden. Dieser Faktor wird verwendet, um die Qualität eines Modells zu bewerten. Aus der Graphsuche werden weiterhin Informationen abgeleitet, die eine Aussage über den Grund eines teilweisen oder vollständigen Mismatch erlauben. Hiermit kann bei Bedarf die Modellgenerierung iterativ adaptiert werden.

Automatischer Aufbau der Modellbasis: Das Modell wird nach der Generierung in der Datenbank abgelegt, die eine effiziente Verwaltung vieler Modelle erlaubt. Günstig hierfür erweist sich eine Binärkodierung der Schlüsselmerkmale und Relationen, wodurch eine speicherplatzsparende Ablage und gleichzeitig ein schneller Zugriff ermöglicht wird ([Bre90]).

7 Graphische Oberfläche

Von besonderer Wichtigkeit für die Akzeptanz eines Bildanalysesystems ist die Möglichkeit zur einfachen, intuitiven Bedienung. Ohne eine geeignete graphische Oberfläche ist es heute nicht mehr praktikabel, ein komplexes System ($HORUS$ umfaßt über 6 MByte Quellcode für die Bildanalyse) zu beherrschen.

Dies gilt für Anwender, die sich z.B. für Schulungszwecke mit dem Thema Bildanalyse befassen und die deshalb ein Werkzeug zum einfachen Einarbeiten mit größtmöglicher Unterstützung benötigen, aber genauso für professionelle Benutzer, die in sehr kurzer Zeit prototypische Lösungen erstellen wollen, ohne dabei auf die Mächtigkeit von einer Programmiersprache verzichten zu wollen.

Für $HORUS$ gibt es seit einiger Zeit eine solche Oberfläche namens $HORUS$dialog (siehe Abb. 2), die Rapid Prototyping im Bereich der Bildanalyse erlaubt. Der Benutzer kann auf die nach taxonomischen Gesichtspunkten gliederten $HORUS$-Operatoren zugreifen und bekommt Parameter vorgegeben, die er abändern kann. Die bei den einzelnen Schritten entstehenden Hodgets werden als Icons dargestellt und das jeweils aktuelle wird im aktiven Fenster ausgegeben. Diverse Parameter, wie z.B. Darstellungsfarben oder Darstellungsform (konvexe Hülle, Regionenrand usw.) sind bequem einstellbar. Zur Zeit wird $HORUS$dialog u.a. im Bildanalyse-Praktikum der TU-München eingesetzt.

In Zukunft wird in Smalltalk eine Programmierumgebung erstellt werden, die noch konsequenter graphisch orientiert ist als $HORUS$dialog. Mit diesem Werkzeug wird es möglich sein, auch komplexe Programme zu kreieren. Soweit sinnvoll, bleibt dabei die textuelle Ebene der Programmierung aus dem Spiel.

Die $HORUS$-Operatoren werden für Smalltalk *funktional*, d.h. möglichst ohne Seiteneffekte, aufbereitet. Die $HORUS$-Bibliothek kann zur Zeit 2D-Bilder mit beliebig vielen Grauwertkanälen und Regionendaten verarbeiten. Ein Nebeneffekt der Vererbungsmechanismen besteht in der Möglichkeit zur Überdefinition der meisten Methoden, so daß

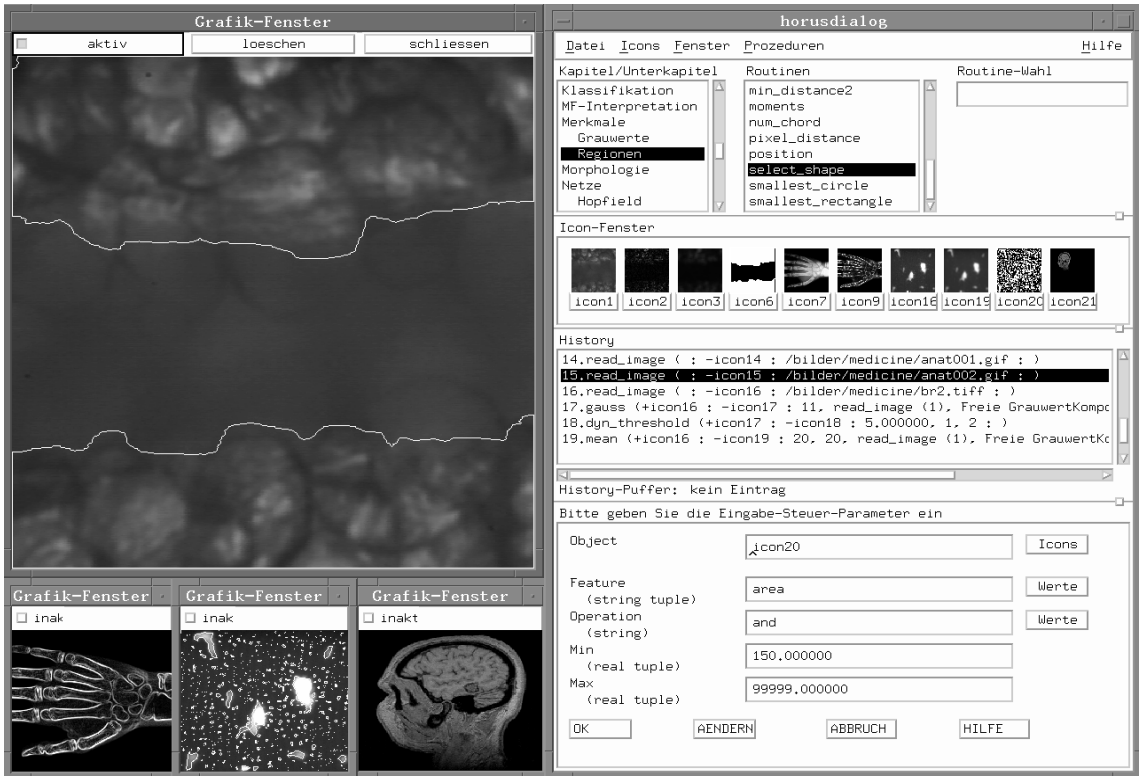

Abbildung 2: *HORUS*dialog

z.B. Bildfolgen oder Farbbilder mit demselben Operator gefiltert werden können wie Einzelbilder, indem Objekte (Hodgets) aus 2D-Bildern zusammengesetzt werden.

Die *HORUS*-Operatoren (Methoden) werden in dem neuen Werkzeug als Icons dargestellt, während die Kanten zwischen den Icons den Datenfluß repräsentieren. Zudem werden Datenquellen für die verschiedenen Eingabeobjekte (z.B. Ganzzahlen: Slider, Listen: Popups, Bilder: Fileselector) und Datensenken zur Visualisierung (z.B. Graphikfenster) verwendet (siehe Abb. 3).

Die Visualisierer für die verschiedenen Datentypen können an jede gewünschte Kante „angehängt“ werden. Während der Programmerstellung werden Veränderungen an Parametern sofort sichtbar gemacht, um eine interaktive Rückkopplung zu gewährleisten.

Der Benutzer wird auf verschiedenen Ebenen bei der Auswahl und Parametrisierung unterstützt. Einige Kriterien lassen sich bereits lokal, andere aber nur global entscheiden. Wenn z.B. in einem längeren Segmentierungsprozeß Grauwerte auf Regionendaten abgebildet werden sollen, ist a priori nicht klar, an welcher Stelle konkret dieser Übergang stattfinden soll. Solche Probleme können nur mittels einer Planungskomponente gelöst werden, die sich modular in die Oberfläche einfügt (siehe: Automatische Konfigurierung). Lokale Restriktionen können statisch, wie z.B. Anzahl und Typ der Parameter oder

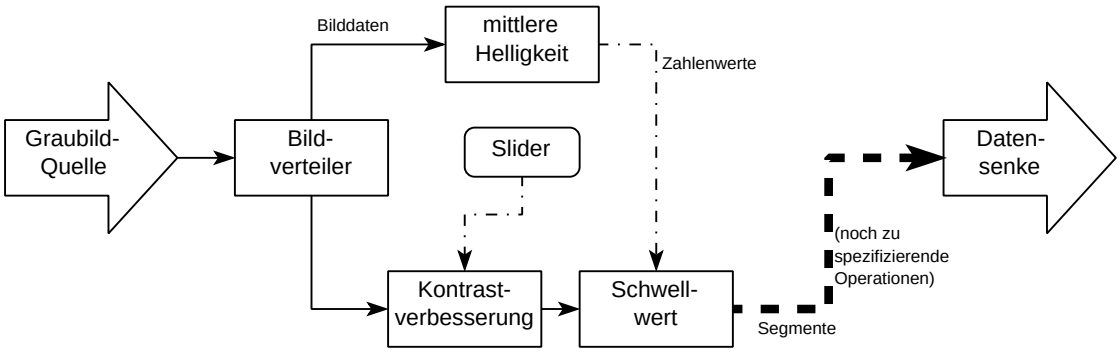

Abbildung 3: Graphisches Editieren einer Kantensegmentation

dynamisch, wie z.B. Maximum und Minimum von Schwellwerten in Abhängigkeit vom Pixeltyp (long oder byte) eines Bildobjektes sein. Aus der Operatordatenbasis werden auch Informationen über die Wirkungsweise von Parametern abgeleitet, z.B. ob der Wirkungszusammenhang eher linear oder logarithmisch ist, um zu „zielorientierten“ Parametern zu kommen. Für die höheren Ebenen der Benutzerunterstützung, z.B. Hilfe bei der Auswahl des nächsten Schrittes im Bildanalyseprozeß, ist die Planungskomponente zuständig.

Diese Programmierumgebung schafft die Möglichkeit, ein erklärtes Ziel unserer Arbeit zu realisieren, nämlich einen Satz von Operatoren zu kreieren, die der menschlichen Perzeption näherkommen, als dies die „Low-Level-Operatoren“ heute tun. Dazu sollen mit dem graphischen Editor erstellte Programm-Module nicht nur als ablauffähige Programme, sondern als Moleküle wieder in die Operatordatenbasis abgelegt werden, so daß Operationen auf höherer Abstraktionsebene erzeugt und später auch wiederverwendet werden können.

Literatur

[Bre90] T. M. Breuel. *Indexing for visual recognition from a large model base*. AI Lab. Memo 1108, 1990.

[Cha89] S. K. Chang. *Principles of Pictorial Information System Design*. Prentice Hall, Englewood Cliffs, New Jersey, 1989.

[Den92] S. Denasi, G. Quaglia, D. Rinaudi. *The use of perceptual organization in the prediction of geometric structures*. Pattern Recognition Letters, 13:529-539, 1992.

[Eck86] W. Eckstein, S.J. Pöppl. *PSIWAG - A Language for Logic Programming in Image Analysis*. Proc. 8th ICPR, Paris, 1986

[Eck88] W. Eckstein. *Das ganzheitliche Bildverarbeitungssystem $HORUS$*. Proc. 10. DAGM Symposium, Zürich, 1988

[Eck93] W. Eckstein. *Die Bildanalysesprache TRIAS*. Dissertation. Infix-Verlag, St. Augustin, 1993.

[Gri90] W. E. L. Grimson. *Object Recognition by Computer: The Role of Geometric Constraints*. MIT Press, Cambridge, Massachusetts, London, England, 1990.

[Hae86] S. Haenel, W. Eckstein. *Ein Arbeitplatz zur halbautomatischen Luftbildauswertung*. Proc. 8. DAGM Symposium, Paderborn, 1986

[Klo93] K. Klotz. *Eine mehrschichtige Architektur zur Fehlerdiagnose und Fehlerbehebung bei der Entwicklung von logischen Programmen*. Dissertation, Infix-Verlag, St. Augustin, 1993.

[Kno86] T. F. Knoll, R. C. Jain. *Recognizing partial visible objects using features indexed hypotheses*. Trans. Robotics Automation RA-2(1), 3-14, 1986.

[Kri92] H. Kristen, O. Munkelt. *Markov-Feld-basierte Bildinterpretation mit automatisch generierter Datenbasis* Proc. 14. DAGM Symposium, Dresden, 1992.

[Kun90] M. K. Kundu, S. K. Pal. *Automatic selection of object enhancement operator with quantitative justification based on fuzzy set theoretic measures*. Pattern Recognition Letters, 11:811-829, Dezember 1990.

[Lan91] S. Lanser, W. Eckstein. *Eine Modifikation des Deriche-Verfahrens zur Kantendetektion*. Proc. 13. DAGM Symposium, München, 1991

[McD82] J. McDermott *R1: A Rule-Based Configurer of Computer Systems*. Artif.Intell. 19, S.39–88, 1982

[Mes92] T. Messer. *Wissensbasierte Synthese von Bildanalyseprogrammen*. Dissertation, Infix-Verlag, St. Augustin, 1992.

[Pau93] J. Pauli. *Erklärungsbasiertes Computer-Sehen von Bildfolgen*. Dissertation, Infix-Verlag, St. Augustin, 1993.

[Rad92] B. Radig, W. Eckstein, K. Klotz, T. Messer, J. Pauli. *Automatization in the Design of Image Understanding Systems*. Proc. 5th International Conference, IEA/AIE-92, Paderborn, Springer-Verlag 1992, LNAI 604, S. 35-45