Mitteilung Nr. 163/88

Gestaltung und Implementation einer graphischen Dialogschnittstelle auf einer Lisp-Maschine nach dem Vorbild eines datenfluß- und objektorientierten Bildfolgenanalysesystems

Ralf Möller

FBI-HH-M-163/88

November 1988

Fachbereich Informatik
Universität Hamburg
Bodenstedtstraße 16

D-2000 Hamburg 50

Zusammenfassung

Dieser Beitrag stellt die Konzepte der objektorientierten graphischen Dialogschnittstelle VIPEX (Visual Programming of Experimental Systems) vor. Das Grundkonzept basiert auf einem datenflußorientierten Verarbeitungsschema. Zu verarbeitende Daten werden über Datenpfade zu den entsprechenden Verarbeitungseinheiten transportiert. Das System wurde in eine Lisp-Umgebung (Symbolics™) eingebettet und visualisiert einem Benutzer durch graphische Darstellungen die zu verarbeitnden Daten, die Verarbeitungseinheiten wie auch die Datenpfade, d.h. die Verknüpfung von Verarbeitungsfunktionen. Neben ihrer Verknüpfung ist auch die Wirkungsweise von Verarbeitungseinheiten durch interaktive Änderung von Parameterwerten in gewissen Rahmen von einem Schnittstellenbenutzer veränderbar.

Der Bericht schildert weiterhin Konzepte zur Strukturierung und Zusammenfassung von Verarbeitungseinheiten. Durch Verbindung von Verarbeitungseinheiten über Datenpfade enstehen sog. Netze. VIPEX gestattet es einem Benutzer, Teilnetze zusammenzufassen und wie elementare Verarbeitungseinheiten zu benutzen.

Die Interaktionskonzepte werden anhand einer Beispielanwendung in der Verarbeitung von Bildfolgen aufgezeigt. Sie eignen sich aber auch für andere Bereiche, in denen die Verarbeitung durch Aneinanderreihung verschiedener Schritte gekennzeichnet ist, bei denen die Feinabstimmung der Verarbeitungseinheiten, d.h. die Wahl der Steuerparameter, interaktiv erfolgen soll oder muß.

Abstract

This report presents the interaction concepts of the graphical interface VIPEX (Visual Programming of Experimental Systems). VIPEX is based on a data flow model and uses an object-oriented interaction style. Data objects are processed by service units connected via data paths. The system is integrated in a Lisp environment (Symbolics™) and uses graphical presentations to visualize the data to be processed, the service units, and the data paths, i.e. the connection structure of functions to be applied. Service units can be modified interactively by editing a set of parameter values assigned to them. In addition to this fine tuning, users can create or modify connection structures of service units. These nets of service units can be divided into subnets which are used like simple service units.

The concepts are presented with image sequence analysis as an application domain. They are suitable in any area where processing is characterized by sequent steps to be coordinated interactively.

Ich danke meinem Betreuer Herrn Dr. Volker Haarslev für seine Unterstützung während der Anfertigung dieser Arbeit, ganz besonders für seine ständige Gesprächsbereitschaft und die sorgfältige Durchsicht des Manuskripts.

Mein Dank richtet sich auch an die Mitarbeiter der Arbeitsgruppe "KOGS" des Fachbereichs Informatik Herrn M. Mohnhaupt und Herrn Dr. C. Sielaff, die mir im Umgang mit dem Programmsystem "Image Calc" und der "Symbolics"-Programmierumgebung behilflich waren. Meinem Studienkollegen Herrn R. Joswig danke ich für seine Anmerkungen sowohl zur Gestaltung der Benutzerschnittstelle als auch zur Abfassung der vorliegenden Arbeit.

Ralf Möller, April 1988

Inhaltsverzeichnis

Kapitel 1

Einleitung

In Bildverarbeitungsexperimenten und -anwendungen werden verschiedene Operationen und Funktionen auf Bilder angewendet (Filter, Kantenfinder, Fouriertransformation, Objekterkennung, ...). Bedingt durch Beleuchtung, Oberflächenbeschaffenheit der abgebildeten Objekte und weiterer Einflüsse läßt sich sowohl die Kombination der Funktionen als auch die Einstellung ihrer Parameter (z.B. Schwellwerte, Verteilungskenngrößen, ...) meist nicht ohne vorherige Testläufe bestimmen. Um die Zeitkomponente erweitert, ist es bei der Verarbeitung von Bildfolgen erforderlich, Funktionen testen und bewerten zu können, die als Argumente mehrere Bilder einer Folge übernehmen. Als einfaches Beispiel wäre eine Differenzbildung von zwei aufeinanderfolgenden Bildern zur einfachen Bewegungsbestimmung abgebildeter Objekte zu nennen.

Um den Entwicklungsvorgang der Funktionskomposition und Parametereinstellung einfach zu halten, wird in dieser Arbeit ein Vorschlag zur Gestaltung einer interaktiven Benutzerschnittstelle vorgestellt. Ausgehend von einer objektorientierten Sicht wird ein Interaktionsmodell bereitgestellt, das zur Verarbeitung benötigte Funktionen durch Objekte mit zugehörigen graphischen Darstellungen repräsentiert.

Kapitel 2 dieses Berichts schildert das konzeptuelle Modell der Interaktion [*Foley et al. 84*] von VIPEX (Visual Programming of Experimental Systems). Um ein datenflußgesteuertes Verarbeitungsschema zu visualisieren, werden Verknüpfungen von Verarbeitungseinheiten (Objekten) durch "Leitungen" symbolisiert, über die Daten, z.B. Bilder, "transportiert" werden. Die graphische Darstellung dieses Transports läßt sich auch als Animation der Verarbeitungsfolge und des jeweiligen Verarbeitungsfortschritts auffassen. Die vorgestellte Dialogschnittstelle ist aus Erfahrungen mit ODISA [*Haarslev 86, Haarslev 87a, Haarslev 87b*] entstanden und bietet weiterführende Möglichkeiten insbesondere zur Strukturierung von Einzelobjekten. Durch Leitungen verbundene Verarbeitungseinheiten können als Netze interpretiert werden. In VIPEX ist es möglich, Teilnetze zu neuen abstrakten Verarbeitungseinheiten zusammenzufassen.

Das nächste Kapitel geht auf die Darstellung von Objekten ein. Eine objektorientierte Darstellungsweise einer Verarbeitungsfunktion mit einem Piktogramm erlaubt einem Benutzer eine direktmanipulative Interaktionsform [*Hutchins et al. 85*]. Parameterwerte von Funktionen (i.d.S. Objekten) werden direkt mit dem Piktogramm dargestellt und können auf einfache Weise den zu verarbeitenden Daten (z.B. Bildern) angepaßt werden. Zusätzlich zur

Darstellung von zusammengefaßten Teilnetzen werden verwendete Techniken zur Visualisierung der Ablaufsteurung und zur Darstellung von Zwischenergebnissen einer Verarbeitungssequenz geschildert.

Die entscheidend von der Programmierumgebung (Symbolics™, Symbolics-Common-Lisp™) beeinflußte Syntax der Benutzerschnittstelle - es konnte auf vordefinierte Objektklassen zurückgegriffen werden - sowie die lexikalische Ebene (Maus-, Tastaturbenutzung) schildert das vierte Kapitel. Das Zusammenspiel der Objekte innerhalb des Datenfluß-modells wird hier mit Hilfe der Netztheorie erläutert. Es werden außerdem einige Erfahrungen, Vorteile und Probleme festgehalten, die sich aus der Programmiersprache Lisp und der verwendeten Programmierumgebung (Maus, Fenster, Objektklassen, Prozesse, ...) ergaben.

Ein Anwendungsbeispiel zu den erläuterten Überlegungen aus dem Bereich der Bildverarbeitung bietet Kapitel 5. Abschließend werden in Kapitel 6 erweiternde Möglichkeiten zur Verschaltung von Objekten diskutiert.

Kapitel 2

Konzepte der Interaktion

Einem Benutzer sollen Komponenten einer Bildverarbeitungsumgebung (z.B. Kameras, Bildspeicher, Programmfunktionen, ...) durch eine interaktive Benutzerschnittstelle zugänglich gemacht werden. Durch Objektbezogenheit wird eine direktmanipulative, modusvermeidende Interaktionsform unterstützt, die es einem Benutzer gestatten soll, auf einfache Weise die benötigten Informationen zu erhalten sowie notwendige Anweisungen zu geben [*Hutchins et al. 85*].

In den nachfolgenden Abschnitten werden die in ODISA [*Haarslev 87a*] zur Interaktion entwickelten Objektklassen sowie deren Attribute kurz zusammengefaßt. Als Erweiterungen sind nachfolgend Klassen zur Strukturierung, d.h zur Zusammenfassung und Gruppierung von Objekten aufgezeigt.

2.1 Elementare Interaktionsobjekte

Alle zu verarbeitenden Daten werden als *Datenobjekte* bezeichnet. Es kann sich beispielsweise um Bilder, repräsentiert durch ein zweidimensionales Feld, um Wissensdarstellungen oder Verweise hierauf sowie auch um "einfachere" Objekte wie z.B. Zahlen handeln.

Zur Bearbeitung von Datenobjekten wurde die Klasse der *Verarbeitungsobjekte* geschaffen. Verarbeitungsobjekte dienen als Modell für periphere Geräte, wie z.B. Kameras, Bildspeicher, Monitore. Sie können weiterhin Spezialprozessoren, Programme sowie auch bekannte Bildverarbeitungsfunktionen bzw. -operatoren darstellen.

Verarbeitungsobjekte bilden eine bestimmte Zahl von Eingangsdatenobjekten in eine Anzahl von Ausgangsdatenobjekten ab. Im Falle fehlender Eingänge fungiert ein Verarbeitungsobjekt als Quelle, im Fall fehlender Ausgänge als Senke. Als Beispiel für ein Verarbeitungsobjekt, das als Quelle wirkt, wäre eine Kamera zu nennen. Ein Verarbeitungsobjekt zur Speicherung von Bildern (Datenobjekten) in einem Bildspeicher könnte als Senke ausgebildet werden.

Datenhaltungsobjekte dienen der (evt. kurzfristigen) Speicherung oder Pufferung von Datenobjekten. Es handelt sich hierbei um virtuelle Objekte, die keine speziellen Bildspeicher oder Datenbanken modellieren, sondern eine gewisse Anzahl von Datenobjekten aufbewahren und zeigen können. Einem Benutzer wird eine Möglichkeit geboten, wäh-

rend der Experimentierphase bestimmte Datenobjekte, zur späteren Verwendung zwischenzulagern. Datenhaltungsobjekte können als Senke ausgebildet sein, während eine Funktion als Quelle nicht in Betracht kommt. Bildspeicher (auch im Sinne von Bildquellen) etc. sind durch Verarbeitungsobjekte darstellbar, zumal sie häufig schon eine Vorverarbeitung (Verknüpfung) von Bildern unterstützen.

Im folgenden seien Objekte, die über Ein- oder Ausgänge verfügen, also bisher Verarbeitungs- und Datenhaltungsobjekte, zusammenfassend als *funktionale Objekte* bezeichnet.

Funktionalobjekte sollen miteinander verknüpft werden. Das bedeutet, daß von diesen Objekten keine Information über Vorgänger- oder Nachfolgeobjekte verwendet werden darf, da sonst keine problemlose, kontextunabhängige Kopplung möglich ist.

Zur Kopplung von funktionalen Objekten wurden *Leitungsobjekte* eingeführt, die vom Ausgang des einen zum Eingang eines anderen Funktionalobjekts "verlegt" werden. Die Aufgabe besteht im "Transport" von Datenobjekten und damit zur Regelung der Kommunikation zwischen Funktionalobjekten. Der Transport verläuft unidirektional vom Eingangsobjekt einer Leitung (Sender) zu deren Ausgangsobjekt (Empfänger). Leitungsobjekte sollen keine Konvertierungs- oder Anpassungsfunktionen übernehmen. Ein Datenobjekt darf also über ein Leitungsobjekt nur dann befördert werden, wenn der Empfänger den Typ des übermittelten Datenobjektes auch verarbeiten kann. Zur Einhaltung dieser Konvention sind den Ein- und Ausgängen *Typen* zugeordnet. Es sind nur Kopplungen von Ein- und Ausgängen des gleichen Typs möglich. In Hinblick auf anzukoppelnde Leitungsobjekte sollen Ein- und Ausgänge als *Anschlüsse* bezeichnet werden.

2.2 Objektattribute

Um Funktion und Verhalten von Bildverarbeitungskomponenten modellieren und darstellen zu können, wird jedem Interaktionsobjekt eine Menge von Attributen zugeordnet. Es wurde versucht, die Attributmengen der Interaktionsobjekte möglichst einheitlich zu bilden, um durch ähnliche graphische Darstellungsweisen eine benutzerfreundliche Interaktionsform zu unterstützen.

2.2.1 Beschreibungsattribute

Kennzeichnend für ein funktionales Objekt sind sein *Name* sowie die *Typen* der Ein- und Ausgänge und damit auch deren Anzahl. Daten- und Leitungsobjekte haben einen anonymen Charakter und sind nicht durch beschreibende Attribute charakterisiert.

2.2.2 Parameter

Jedem funktionalen Objekt ist eine Menge von *Parametern* zugeordnet. Die Parameter werden durch die modellierte Funktion bestimmt. Dient ein Verarbeitungsobjekt z.B. zur Darstellung einer Kamera, so können Neigungs-, Kipp- und Schwenkwinkel als beschreibende Parameter eingeführt werden. Für Bildverarbeitungsoperatoren können Schwellwerte maßgeblich sein, während für Mustererkennungsklassifikatoren bestimmte kalibrierende Verteilungskenngrößen und andere Parameter einstellbar bzw. änderbar sein sollen.

2.2.3 Ablaufsteuerung

Weitere Objektattribute für funktionale Objekte ergeben sich durch die Ablaufsteuerung. Um die Wirkung eines Verarbeitungsschrittes bewerten zu können, ist es notwendig, die Verarbeitung bzw. Weiterleitung eines Datenobjekts zeitweilig aufhalten zu können. Wenn das entsprechende Datenobjekt vorliegt, kann untersucht werden, wie sich der Verarbeitungsschritt ausgewirkt hat. Für Bilder mag eine graphische Darstellung sinnvoll sein, während für andere Datenobjekte eine ihnen entsprechende Darstellungsform gewählt werden muß.

Zur Ablaufsteuerung dient ein Attribut zur Beschreibung der *Verarbeitungsart* mit Werten 'step', 'stop' und 'cycle'. Während 'stop' eine Verarbeitungs- und Weiterleitungssperre bewirkt, gewährleistet 'cycle' eine ungehinderte Verarbeitung und Weiterleitung. Die Verarbeitungsart 'step' führt eine einschrittige 'cycle'-Operation aus und geht dann nach 'stop' über.

2.3. Kompositionsobjekte

Um einen gewünschten Verarbeitungsschritt zu realisieren, wird es häufig nötig sein, mehrere Interaktionsobjekte zusammenzusetzen. Zur Zusammenfassung von Objekten wurde die Klasse der *Kompositionsobjekte* definiert, um eine zum bisherigen Konzept konsistente, objektorientierte Repräsentation zu erhalten. Kompositionsobjekte können sowohl elementare Interaktionsobjekte (Kap. 2.1) als auch weitere Kompositionsobjekte enthalten. Es ergibt sich eine baumartig strukturierte Menge von Interaktionsobjekten (Abbildung 2.1).

Ein analoges Beispiel für das Konzept der Kompostionsobjekte findet sich in Logik- oder Schaltkreissimulatoren, in denen z.B. bestimmte Elementarglieder zu einem Halbaddierer zusammengefaßt werden. Zwei Halbaddierer und ein Oder-Glied wiederum bilden, je nach Leitungsstruktur, einen Volladdierer.

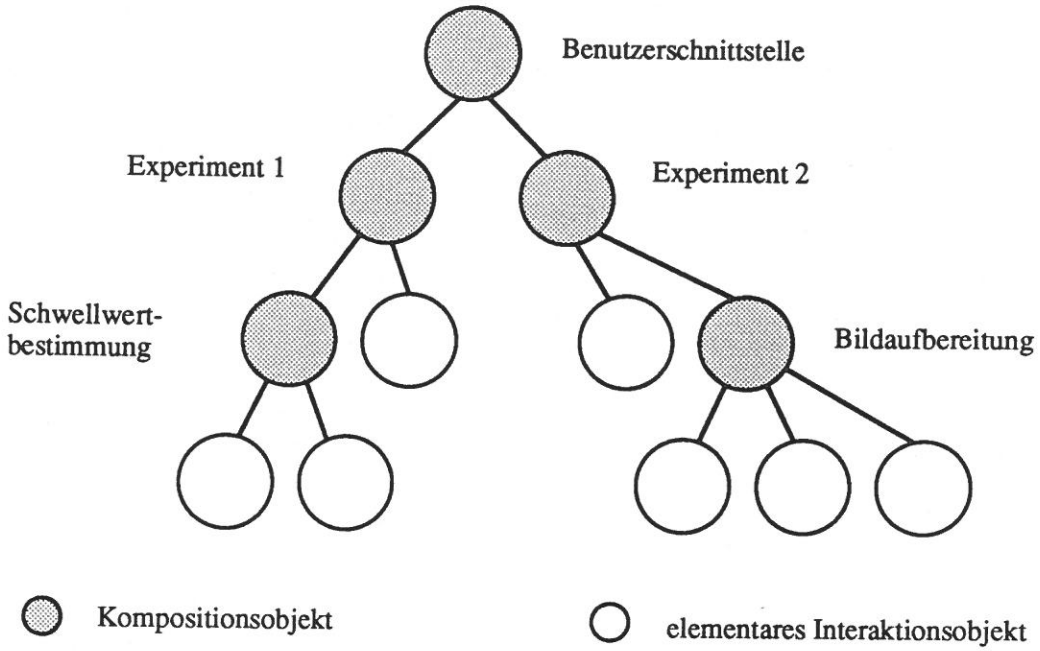


Abbildung 2.1: Baumartige Strukturierung von Interaktionsobjekten

2.3.1 Attribute von Kompositionsobjekten

Auch einem Kompositionsobjekt ist eine Menge von Attributen nach Abschnitt 2.2 zugeordnet. Der Name wird von der Gesamtwirkung der zusammengefaßten Objekte (z.B. "Experiment", "Halbaddierer", "Objekterkennung") beeinflußt sein. Die Typen der Ein- und Ausgänge eines Kompositionsobjekts ergeben sich aus den Typen der Anschlüsse derjenigen Objekte, die mit Leitungen aus der "Umgebung" gekoppelt sind. Da Kompositionsobjekte über Ein- und Ausgänge verfügen, zählen sie zu den funktionalen Objekten (s. Abschnitt 2.1).

Die Ablaufsteuerungsmöglichkeiten funktionaler Objekte gelten auch für Kompositionsobjekte. Die Ablaufattribute beziehen sich hier auf den Transport von Datenobjekten von und zur "Umgebung" (Abbildung 2.2). In der Ablaufart 'stop' wird ein Datenobjekt weder an ein Interaktionsobjekt, das in einem Kompositionsobjekt enthaltenen ist, weitergeleitet noch "nach außen abgegeben". Der Wert 'cycle' garantiert einen ungehinderten Weitertransport von Datenobjekten; 'step' wirkt – wie in Abschnitt 2.2.3 vorgestellt – als einschrittige 'cycle'-Operation mit anschließendem Übergang nach 'stop'.

Die Kopplung der Umgebung an die in einem Kompositionsobjekt zusammengefaßten Interaktionsobjekte wird genauer in Abschnitt 3.3 dargestellt. An dieser Stelle soll die schematische Darstellung aus Abbildung 2.2 das Zusammenfassungskonzept verdeutlichen. Das Kompositionsobjekt dieser Abbildung enthält zwei Funktional- und vier Leitungsobjekte

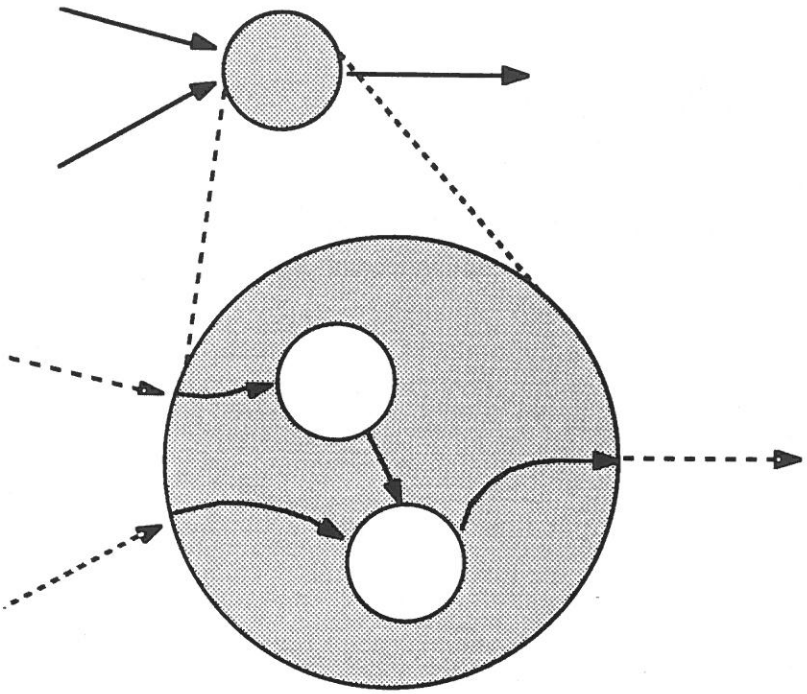


Abbildung 2.2: Schematische Darstellung eines Kompositionsobjektes

(dargestellt durch Kreise bzw. Pfeile). Drei der Leitungsobjekte sind mit der "Umgebung" verbunden. Die äußeren Leitungsobjekte werden in diesem Sinne durch die inneren "fortgesetzt".

Nun sind die zusammengefaßten Funktionalobjekte durch Parameter nach Abschnitt 2.2.2 gekennzeichnet. Parameter eines Objekts müssen von einem Benutzer interaktiv manipulierbar sein. Auch Kompositionsobjekte besitzen Parameter ähnlich wie Verarbeitungsobjekte. Die Werte dieser Parameter "vererben" sich (in noch zu spezifizierender Weise) auf die enthaltenen Objekte. Überlegungen hierzu sind im nachfolgenden Abschnitt festgehalten.

2.3.2 Vererbung von Parameterwerten

Interaktionsobjekte können (hierarchisch) strukturiert werden; Einzelobjekte werden zu Kompositionsobjekten zusammengefaßt. Es erweist sich als günstig, diese Anordnung der Objekte auszunutzen, um eine (eventuell notwendige) Änderung eines Parameterwertes bei jedem Einzelobjekt nur an einer (zentralen) Stelle vornehmen zu müssen. Als Beispiel für Parameter, die bei mehreren Verarbeitungsobjekten gleichermaßen auftreten, sind die Wir-

kungsradien (Diameter) von Bildverarbeitungsoperatoren zu nennen. Sollen alle Radien geändert werden, entfällt eine umständliche (lokale) Änderung jedes einzelnen Wertes, wenn das die Verarbeitungsobjekte enthaltende Kompositionsobjekt einen (globalen) Parameter zur Kennzeichnung des Wirkungsradius' führt. Eine Werteänderung beim Kompositionsobjekt vererbt sich auf die Einzelobjekte.

Die Probleme dieses Ansatzes ergeben sich aus einem vielleicht verständlichen Wunsch eines Benutzers, während der Test- und Anpassungsphase einer Funktion einen - durch ein Kompositionsobjekt "gebundenen" - Parameterwert eines Objektes lokal zu verändern.

Eine Bindung von Parameterwerten könnte über die Namen der Parameter realisiert sein. Der (gleiche) Wert liegt dann in Form mehrerer Kopien bei den einzelnen Objekten vor. Durch lokale Änderungen entstehen (evt. kurzfristig) Inkonsistenzen. Läßt man jedoch Inkonsistenzen innerhalb durch Vererbungsmechanismen gebundener Parameterwerte zu, so treten Schwierigkeiten bei der (globalen) Änderung des Wertes beim Kompositionsobjekt zutage: Werden die (temporär) inkonsistenten Werte in der "Vererbungshierarchie" automatisch auf den neuen Wert geändert? Sollte der Benutzer über den inkonsistenten Zustand (bei jeder globalen Änderung) informiert werden? Muß über ein Menü ausgewählt werden, welche Objekte mit inkonsistenten Parameterwerten (bezüglich des veränderten globalen Wertes) den neuen Wert übernehmen? Die hier etwas vage formulierten Fragen spiegeln die durch Inkonsistenzen hervorgerufene Unsicherheit eines Benutzers wider.

Eine strengere Möglichkeit der Parameterbindung ergibt sich, wenn die zusammengefaßten lokalen Objekte keine Kopie eines Parameterwertes, sondern nur eine Referenz auf das globale Parameterobjekt verwenden. Inkonsistenzen sind bei diesem Vorgehen nicht möglich. Nach dieser Methode wurde bei der implementierten Schnittstelle verfahren.

Eine andere Variante der Parameterbindung bestünde in der expliziten Festlegung allgemeiner funktionaler Abhängigkeiten (Constraints) zwischen Parametern (im Gegensatz zur oben geschilderten "direkten" Abhängigkeit).

Einen weiteren Gesichtspunkt, der zu bedenken ist, wenn man lokale Parameteränderungen erlaubte, schildert der nächste Abschnitt.

2.3.3 Abstraktionsbildung

Das Ein-Ausgabeverhalten zusammengesetzter Objekte läßt sich in geeigneter Weise interpretieren. Die in Abbildung 2.3. gezeigte Diodenschaltung z.B. läßt sich je nach Zuordnung von "0" und "1" zu entsprechenden Spannungsbereichen - losgelöst vom Verhalten der einzelnen Bauelemente - als OR- oder als AND-Gatter deuten. Auch in Kompositionsobjekten treten die zusammengefaßten Einzelobjekte in den Hintergrund ("Black-Box"). Eine

Auswahl repräsentativer Parameter erlaubt eine abstrakte Sicht auf die zusammengesetzten Objekte. Durch globale Parameteränderungen kann das Gesamtverhalten eines Kompositionsobjekts gesteuert werden, ohne die Wirkungsweise der Einzelobjekte betrachten zu müssen. Unter diesem Gesichtspunkt werden Kompositionsobjekte synonym als *Abstraktionsobjekte* bezeichnet.

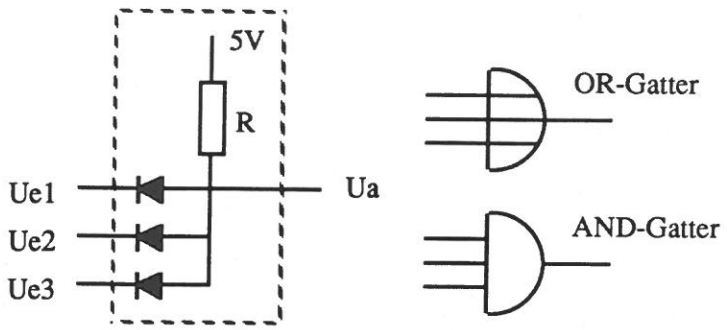


Abbildung 2.3: Diodenschaltung

Lokale Parameteränderungen, um die obige Diskussion wieder aufzugreifen, würden dem Abstraktionsgesichtspunkt widersprechen. Sie sind in der erstellten Implementation deshalb gänzlich ausgeschlossen.

2.4 Gruppenobjekte

Wie aus Abbildung 2.2 hervorgeht, werden die in Kompositionsobjekten zusammengefaßten funktionalen Objekte durch Leitungsobjekte mit der "Umgebung" verbunden. Das bedeutet, daß ein Interaktionsobjekt nur unter genau einem Kompositions- oder Abstraktionsobjekt eingeordnet werden kann, da sich sonst die angekoppelten Leitungsobjekte in irgendeiner Form "aufspalten" müßten.

Eine Steuerung zusammengefaßter Objekte (Parametervererbung) ist bisher nur von einem Kompositionsobjekt aus möglich. Gerade in der Experimentierphase erweist es sich jedoch als umständlich, jedesmal ein (neues) Kompositionsobjekt zu "erzeugen", wenn - zwecks schnellerer Beeinflussung - mehrere Parameter gebunden werden sollen. Im Sinne der Abstraktionsobjekte ist eine solche pragmatische Zusammenfassung auch nicht adäquat.

Um diesen Schwierigkeiten entgegenzutreten, wurde die Klasse der *Gruppenobjekte* geschaffen. Durch Gruppenobjekte kann eine bezüglich der "Hierarchieebenen" (Abbildung 2.1) transparente Zusammenfassung von Objekten modelliert werden. Ähnlich wie bei den Kompositionsobjekten kann eine Bindung von Parameterwerten erreicht werden. Gruppenobjekte verfügen nicht über Anschlüsse; daher kann ein Objekt in mehreren Gruppenobjekten auftreten, also unter verschiedenen Aspekten zusammengefaßt werden. Als weitere Möglichkeit läßt sich hier auch eine Bindung der Ablaufsteuerungsattribute modellieren, so daß gruppierte Objekte in einem Schritt, z.B. auf 'stop' gesetzt werden können.

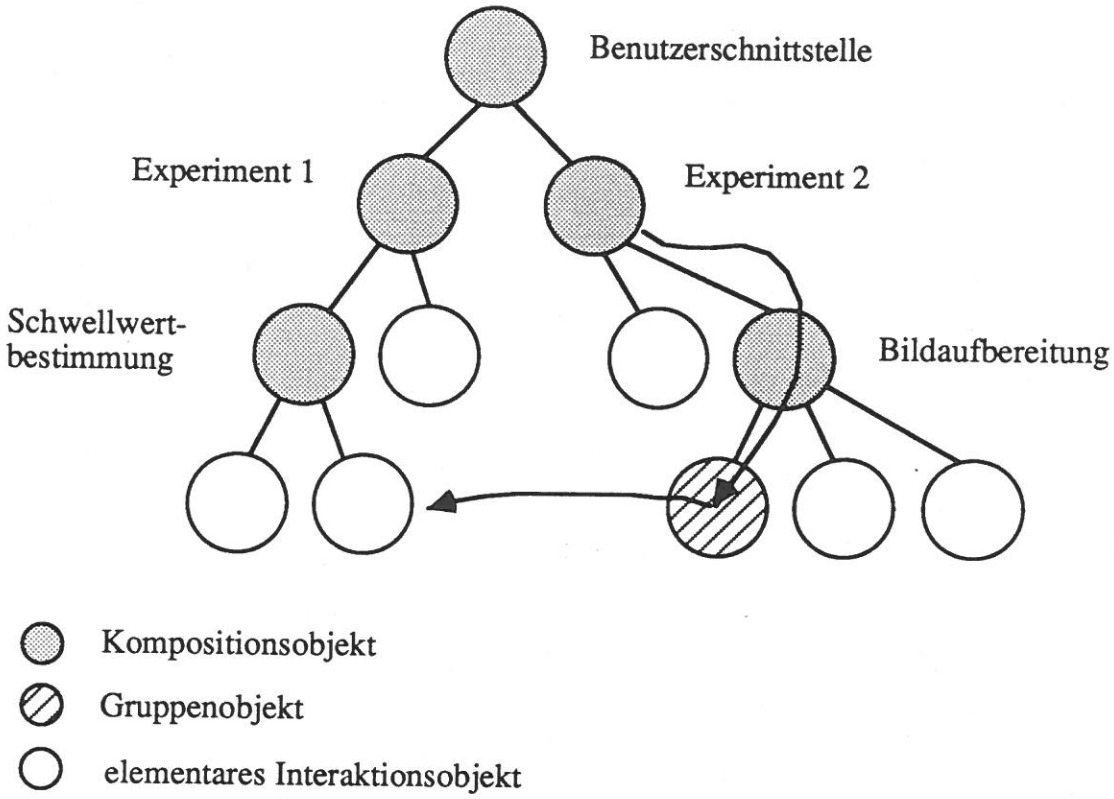


Abbildung 2.4: Parameterbindungsgeflecht

Gruppenobjekte können in Kompositionsobjekte eingeordnet werden. Wie aus Abbildung 2.4 hervorgeht, kann bei entsprechender Parameterbindung ein "Geflecht" aus Bindungen und Beziehungen entstehen. Sei ein Parameterwert eines Gruppenobjektes durch Vererbung über "Bildaufbereitung" von "Experiment 2" gebunden. Nun vererbt sich jedoch die Änderung eines Parameters des Gruppenobjekts wieder auf andere Objekte, z.B. auf ein Objekt in einer anderen Seitenlinie von "Benutzerschnittstelle" als "Experiment 2" (in Abbildung 2.4 durch Pfeile angedeutet)!

Es wird deutlich, daß Parameterbindungen von einem Benutzer eventuell nicht mehr durchschaut und verstanden werden, wenn sie (von ihm) zu kompliziert aufgebaut werden. In einem restriktiveren Ansatz könnte eine Vererbung von Werten an Parameter von Gruppenobjekten untersagt sein. Die in Abbildung 2.4 aufgezeigte Situation träte dann nicht auf, die Flexibilität der Gruppenobjekte jedoch litte in nicht unerheblichem Maße (siehe auch Kap. 3.3).

2.5 Umgang mit Interaktionsobjekten

Die benötigten Informationen zur Erzeugung von Objekten stellt ein sog. *Generierungssystem* der Benutzerschnittstelle zur Verfügung (in *Faasch 87* auch als Verarbeitungssystem bezeichnet). Für Verarbeitungsobjekte sind dies u.a. Funktionen und Prozeduren (zur Implementierung von Bildverarbeitungsoperationen). Ebenfalls können von einem solchen System Objekte über Leitungen zusammengeschaltet und (vor-)konfiguriert werden. Schon bei der Generierung von Objekten können z.B. mit Kompositionsobjekten Parameterbindungen und Strukturierungen festgelegt werden. Ein Benutzer kann dann den Ablauf der Interaktion im Einzelnen bestimmen. In der vorliegenden Implementation der Schnittstelle ist dieses schon weitgehend unterstützt; eine Beispielsitzung wird in Kapitel 5 dargestellt.

Zusätzlich soll es auch möglich sein, daß ein Benutzer interaktiv Objekte erzeugen kann. Insbesondere gilt dies für Gruppen- und Kompositionsobjekte. Es sind Parameterbindungen festzulegen sowie Anschlüsse vorzusehen. Leitungsobjekte werden interaktiv an Funktionalobjekte angekoppelt oder wieder von diesen gelöst. Objekte können in Kompositionsobjekten zusammengefaßt oder aus diesen entfernt werden. Im Sinne von Abstraktionsobjekten werden konstruierte Zusammenfassungen gesichert, d.h. für eine spätere Verwendung aufbewahrt (Bibliothek von Abstraktionsobjekten). Weiterführende Überlegungen dieser Art sind aus Zeitgründen bisher bei der Implementierung unberücksichtigt geblieben, bieten aber ein vielversprechendes Interaktionskonzept.

Kapitel 3

Repräsentation von Objekten

Die einem Interaktionsobjekt zugeordneten Attribute sollen für einen Benutzer in geeigneter Weise (graphisch) dargestellt werden. In Ergänzung hierzu müssen bei einem Kompositionsobjekt auch die in diesem Objekt enthaltenen Interaktionsobjekte zugänglich sein. Die zusammengefaßten Objekte legen die *Struktur* der Kompositionsobjekte fest. Es ergibt sich, daß nicht nur Kompositionsobjekte in diesem Sinne eine Struktur haben. Auch für andere Objektklassen können strukturelle Merkmale aufgezeigt werden. Im weiteren werden die *Darstellungsklassen* geschildert, die die benötigten Darstellungsarten repräsentieren.

Für die Darstellung bietet sich die Verwendung von *Piktogrammen* an, indem z.B. jede Objektklasse durch ein Piktogramm gekennzeichnet wird. Während sich das "Piktogramm" für die Klasse der Leitungsobjekte in natürlicher Weise als ein Linienzug ergibt, soll die Metapher der Darstellung der anderen Objektklassen zunächst erläutert werden.

3.1. Metapher der Objektdarstellung

Wie aus den vorigen Kapiteln deutlich wurde, befördern Leitungen Datenobjekte zu den funktionalen Objekten (Verarbeitungs-, Datenhaltungs- und Kompositionsobjekte). Die "Leitungen" sind also im Sinne von *Rohrpostsystemen* zu interpretieren, durch die (kleine) *Container* versandt werden. Die Container symbolisieren die Datenobjekte; das Piktogramm hierzu ist in Abbildung 3.1 skizziert. Einem Benutzer muß visualisiert werden, daß Datenobjekte bei einem funktionalen Objekt zur Übernahme bzw. zur Abholung bereitstehen. Dafür sind entsprechende *Anschlußkästchen* vorhanden, in denen vorhandene Datenobjekte durch deren Piktogramme gezeigt werden (Abbildung 3.1). Gleichzeitig dienen die Anschlußkästchen als Kopplungspunkte für Leitungsobjekte. Um einem Benutzer anzuzeigen, daß Datenobjekte in einem Objekt zur Betrachtungszeit gerade transformiert werden, sind die Container während dieser Zeit an den Eingängen geöffnet dargestellt (Abbildung 3.1).

Die allgemeine Form einer Darstellung erfordert außerdem Flächen für ein Piktogramm und einen Objektnamen sowie für eine Darstellungsfläche, auf der weitere Informationen präsentiert werden. Eine Skizze der gewählten Aufteilung findet sich in Abbildung 3.2. Es wurde keine viereckige Gesamtform gewählt, um die Auffälligkeit des Piktogramms

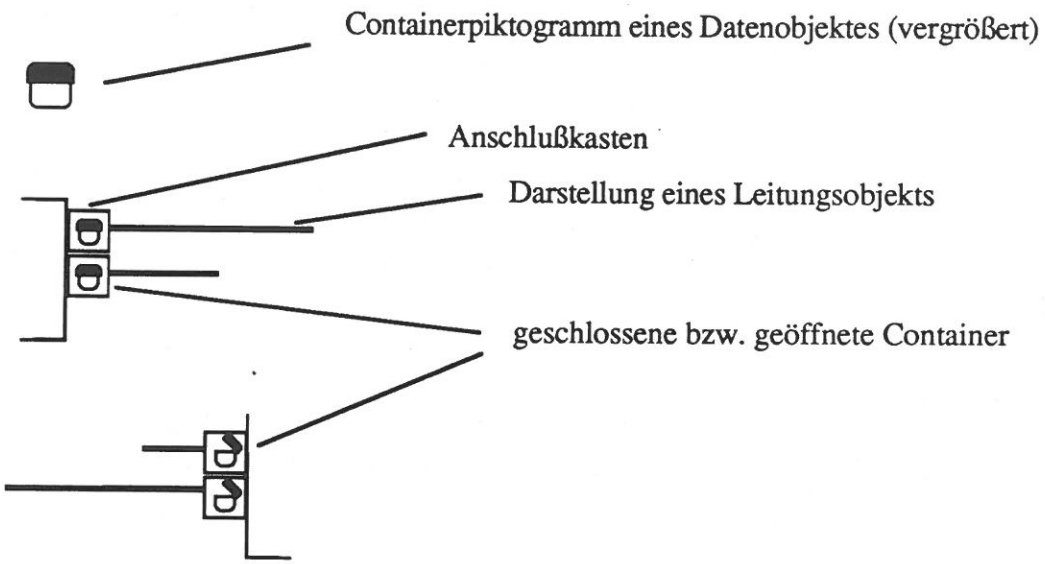


Abbildung 3.1: Datenobjektdarstellungen

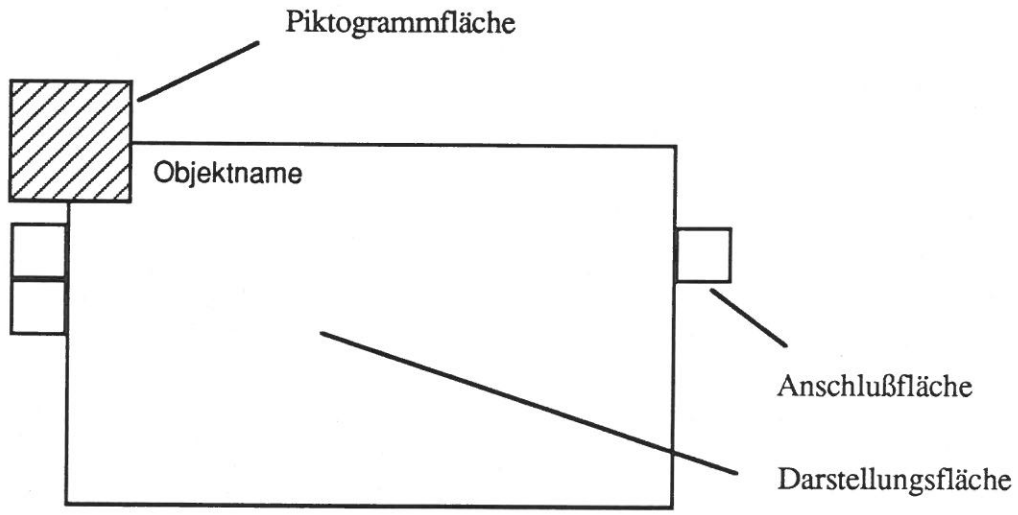


Abbildung 3.2: allgemeines Darstellungslayout

links oben steigern zu können. Weiterhin werden die Anschlußkästchen in ihrer Funktion als Verbindung zur Umgebung hervorgehoben.

Die Standard-Piktogramme für die Objektklassen aus Kapitel 2 sollen anhand von Abbildung 3.3 erläutert werden. Wie schon in der Einleitung zu diesem Kapitel erwähnt, gehören zur Präsentation eines Interaktionsobjektes außer der Attributdarstellung - Statusdarstellung genannt - noch weitere Informationen, die die Struktur der Objekte betreffen (Strukturdarstellung). Nähere Erläuterungen hierzu finden sich in den Abschnitten 3.2 und 3.3. In diesem Abschnitt werden die zur Kennzeichnung der Darstellungsklassen verwendeten Pikto-

gramme eingeführt. Um einen Benutzer nicht mit verschiedenen Interaktionsarten zu verwirren [*Smith et al. 82*], soll sowohl für die Status- als auch für die Strukturdarstellungen aller Funktionalobjekte die in Abbildung 3.2 skizzierte Grundform beibehalten werden.

Verarbeitungsobjekte sind dadurch gekennzeichnet, daß sie eine Vielzahl verschiedener Bildverarbeitungskomponenten modellieren können. Beispiele hierzu sind in Kapitel 2 erwähnt. Berücksichtigt man dieses, so wird deutlich, daß ein Standard-Piktogrammsymbol für Verarbeitungsobjekte eine sehr allgemeine Beschreibung darstellen muß [*Bewley et al. 83*]. In einem Verarbeitungsobjekt wird der Inhalt eines Containers, ein Datenobjekt, aus diesem *entnommen*, *verarbeitet* und anschließend wieder in den Container "*verpackt*". In diesem Sinne ist das Piktogramm aus Abbildung 3.3 zu deuten.

Das Piktogrammsymbol für Kompositionsobjekte ergibt sich als Umschließung einer Fläche durch einem Linienzug. Hierdurch wird eine Zusammenfassung von Objekten kenntlich gemacht. Datenhaltungsobjekte sind im Sinne einer Aneinanderreihung von Datenobjekten aufzufassen. Die Aneinandereihung ist in dem gezeigten Piktogrammsymbol durch (drei) hintereinanderliegende Datenobjektpiktogramme ausgedrückt. Eine Gruppierung wird durch Hintereinandersetzung stilisierter Darstellungsschemata nach Abbildung 3.2 symbolisiert.

Die Piktogramme zur Strukturdarstellung (rechts in Abbildung 3.3) unterscheiden sich von denen zur Statusdarstellung, indem der jeweils "gefüllte" Teil des Piktogrammsymbols der Statusdarstellung hier "durchsichtig" gestaltet ist.

Abbildung 3.3: Standard-Piktogramme

3.2. Statusdarstellung

Die Statusdarstellung funktionaler Objekte sowie auch die der Gruppenobjekte erfolgt nach einem einheitlichen Schema. In der Darstellungsfläche (siehe Abbildung 3.2) werden die Parameter mit Namen und zugehörigen Werten präsentiert. Ein einfaches Beispiel einer Statusdarstellung eines Verarbeitungsobjekts zeigt Abbildung 3.4.

Da die Anzahl der Parameter besonders bei Abstraktions- oder Gruppenobjekten recht groß werden kann, ist ein Sichtfenster-Mechanismus (Scrolling) erforderlich, falls der zur Verfügung stehende Platz nicht zur Darstellung aller Parameter ausreicht. Die genaue Syntax der Parameteränderung wird in Kapitel 4 dargestellt. Es soll hier noch darauf hingewiesen werden, daß von der in *Haarslev 86 (Kap. 6.2.3)* vorgestellten Sichtweise der Darstellungsobjekte insofern abgewichen wird, daß ein Darstellungsobjekt immer die vollständige Beschreibung eines Interaktionsobjekts erhält. Gegebenenfalls wird zwar ein

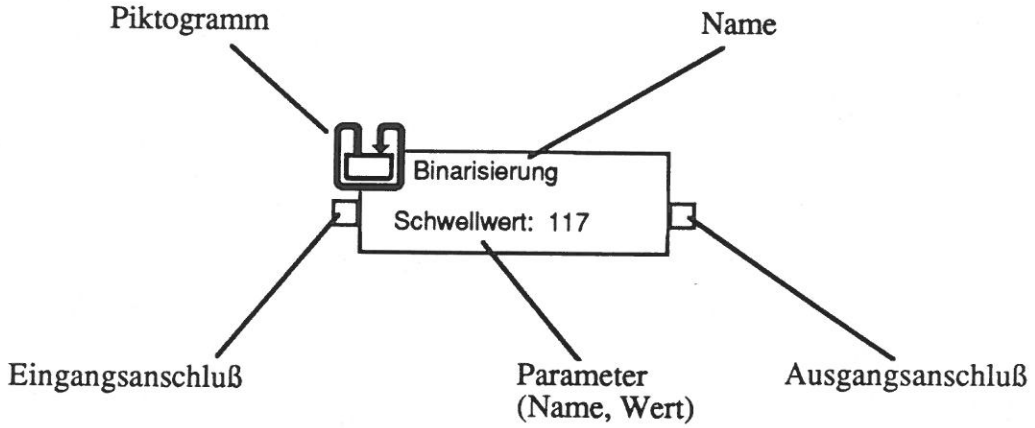


Abbildung 3.4: Beispiel einer Statusdarstellung eines Verarbeitungsobjekts

Sichtfenstermechanismus nötig sein, aber eine nur ausschnittsweise Darstellung der Parameter wird hier nicht vorgenommen. Wünscht der Benutzer eine ausschnittsorientierte Darstellung ohne aktuelles Sichtfenster, so kann er sich ein Gruppenobjekt erzeugen, welches nur das betreffende Objekt zusammenfaßt. Die zu "gruppierenden" Attribute können dann ausgewählt werden. Diese extreme Sichtweise eines Gruppenobjektes als Zusammenfassung von genau einem Objekt findet damit eine sinnvolle Interpretation als Ausschnittsdarstellung der Parameter eines Objekts.

Die Statusdarstellungen der Daten- und Leitungsobjekte weichen von dem Schema aus Abbildung 3.2 ab. Datenobjekte werden nur mit einem Piktogramm in den Anschlußkästchen der Funktionalobjekte dargestellt (Abbildung 3.1). Die Wege der Linienzüge zur Repräsentation der Leitungsobjekte werden automatisch so berechnet, daß sich ein optisch ansprechendes Erscheinungsbild ergibt. Der hierzu benötigte Linienführungs-Algorithmus [*Hightower 69*] wird in Anhang A kurz skizziert. Durch die automatische Leitungsführung ergibt sich, daß die Statusobjekte beliebig verschiebbar sind.

Der "Status" eines Datenobjekts läßt sich als Position innerhalb einer Verarbeitungskette definieren, der eines Leitungsobjekts ergibt sich als Koordinaten der "Eckpunkte" des Linienzugs. Für die anderen Objekte sind die aktuellen Parameterwerte, die Verarbeitungsart ('step', 'stop' oder 'cycle') sowie auch der Verarbeitungszustand (geöffnete Container - s.

Abbildung 3.1) bzw. der Verarbeitungsfortschritt maßgeblich. Sowohl Verarbeitungsart als auch der (relative) Verarbeitungsfortschritt sollen in der Statusanzeige sichtbar werden. Da beides auch in der nachfolgend geschilderten Strukturdarstellung präsentiert werden soll, wird hier auf Abschnitt 3.5 und 3.6 verwiesen.

3.3. Strukturdarstellung

Läßt sich der Status darzustellender Objekte noch in vergleichsweise einheitlicher Form zeigen, so treten bei der Strukturdarstellung erhebliche Unterschiede zutage. Wie schon in Abschnitt 3.1 erläutert, soll das Darstellungsschema aus Abbildung 3.2 auch für Strukturdarstellungen verwendet werden. Die Piktogramme richten sich nach den Objektklassen (Abbildung 3.3).

Kompositionsobjekte fassen bestimmte Interaktionsobjekte zusammen (Kapitel 2.3). Die Strukturdarstellung ergibt sich deshalb als Darstellung der zusammengesetzten Interaktionsobjekte. Ein Beispiel wird in Abbildung 3.5 gezeigt. Das als Beispiel vorgestellte Kompositionsobjekt "Median-Binarisierung" besteht aus fünf Interaktionsobjekten, zwei Verarbeitungs- und drei Leitungsobjekten. Die Interaktionsobjekte erscheinen in ihrer Statusdarstellung.

Der Parameter ("Schwellwert") des Verarbeitungsobjekts "Binarisierung" ist durch das Kompositionsobjekt gebunden, während der Parameter von "Median" lokal geändert werden kann.

Die Umgebung wird in dieser Darstellungsform durch die äußeren Anschlußkästchen der Strukturdarstellung repräsentiert. An eine Ankopplung von Leitungsobjekten von außen

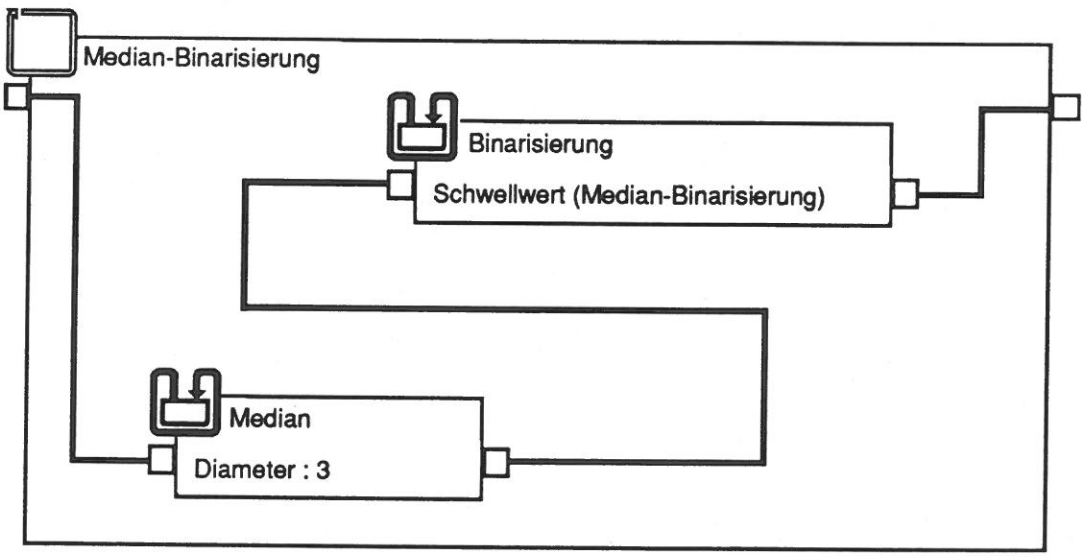


Abbildung 3.5: Strukturdarstellung eines Kompositionsobjektes

an die Strukturdarstellung ist nicht gedacht. Ankopplungen von außen sind nur an Statusobjekte möglich. Das zu "Median-Binarisierung" gehörende Statusobjekt befindet sich in der Strukturdarstellung eines anderen Kompositionsobjekts, etwa innerhalb eines "Experiments" (siehe Abbildung 3.6). Die von außen an das Statusobjekt von "Median-Binarisierung" heranführenden Leitungen werden "innen" (Abbildung 3.5) in der Strukturdarstellung fortgesetzt. Dieses Schema ist von Schaltkreissimulatoren bekannt.

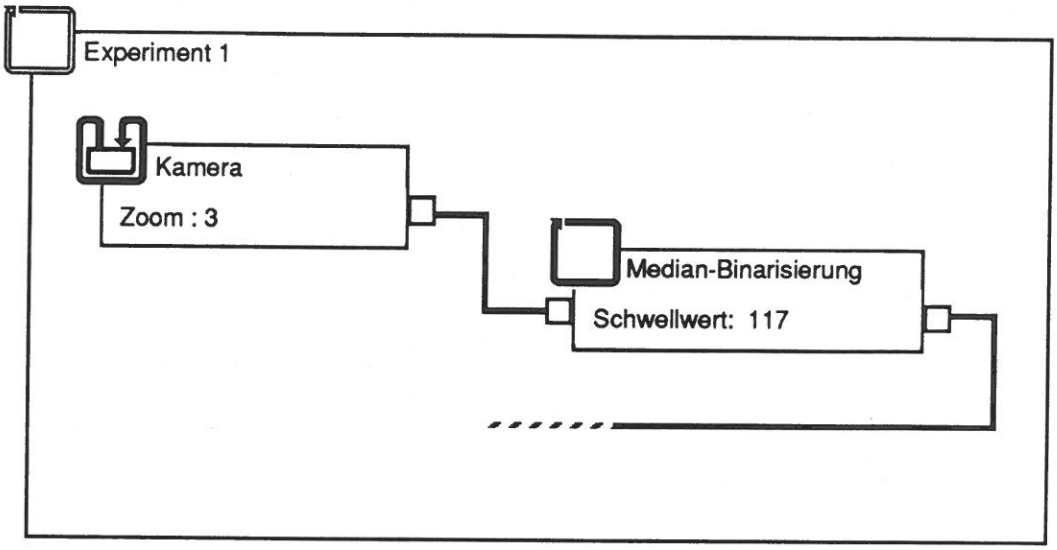


Abbildung 3.6: Strukturdarstellung eines Kompositionsobjektes namens "Experiment 1"

Einen besonderen Charakter haben Kompositionsobjekte ohne äußere Anschlüsse. Um bei bekannten Beispielen zu bleiben, wäre wieder das "Experiment" zu nennen. Hier wird ein nach außen abgeschlossenes System repräsentiert. Kompositionsobjekte können auch als Quelle oder Senke mit entsprechenden Abschlüssen an nur einer Seite definiert werden.

Die Verarbeitungsobjekte dienen, wie aus Kapitel 2 bekannt, zur Modellierung von Kameras, Bildschirmen, Spezialprozessoren, insbesondere aber von Programmen, und Funktionen (im Lisp-Sinne). Eine Strukturdarstellung von Verarbeitungsobjekten soll deshalb von einem Editor übernommen werden. Das Editorfenster wird innerhalb der Darstellungsfläche aus Abbildung 3.2 plaziert; eine Skizze hierzu findet sich in Abbildung 3.7. Es handelt sich in der vorliegenden Fassung um eine Instanz der auf der Zielmaschine "Symbolics" verwendeten ZMACS-Editor-Objektklasse [*Symbolics 85*]. Programmfunktionen können geändert und aus dem Editor heraus neu evaluiert bzw. übersetzt werden. Bildverarbeitunskomponenten wie Kameras und Spezialprozessoren seien in einer Strukturdarstellung durch ihre Treiberprogramme vertreten.

Einer andersartigen Behandlung bedürfen wiederum die Datenobjekte. Der "Inhalt" eines Containers soll nicht explizit gezeigt werden. Da die Container nicht durch einen Na-

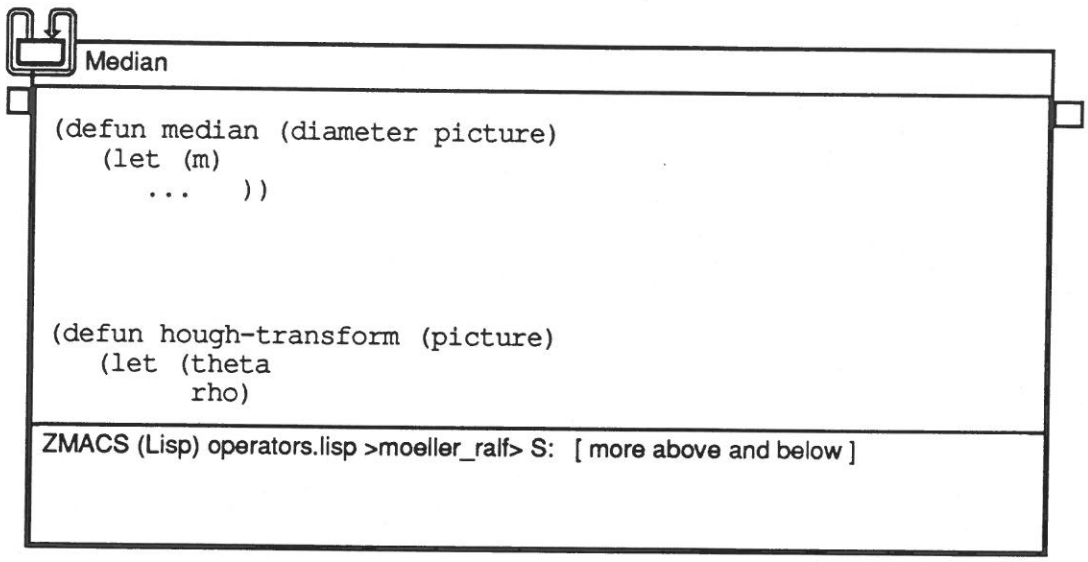


Abbildung 3.7: Strukturdarstellung eines Verarbeitungssobjektes

men gekennzeichnet sind und sich ihre Position verändert, wird nicht deutlich, zu welchem Container eine Darstellung gehört. Die Darstellung von Datenobjekten ist deshalb an die Anschlüsse (sie erscheinen hier als Piktogramm) der funktionalen Objekte gekoppelt. Die Überlegungen hierzu werden im Abschnitt 3.4 erläutert.

In den Darstellungen von Funktionalobjekten könnte es gemäß den in Kapitel 2.5 geschilderten Überlegungen zusätzlich nötig sein, interaktiv Anschlüsse hinzuzufügen bzw. zu löschen. Parameterbindungen müssen erstellt oder verändert werden. In der bisher implementierten Benutzerschnitttelle können solche Operationen noch nicht interaktiv vorgenommen werden. Auch Strukturdarstellungen für Datenhaltungs- und Gruppenobjekte sind in der vorliegenden Implementation bisher noch nicht realisiert.

Die in einem Datenhaltungsobjekt zu einem bestimmten Zeitpunkt enthaltenen Datenobjekte sollten einsehbar sein. Die Ausgabeweise richtet sich nach dem Typ der enthaltenen Objekte (siehe auch Abschnitt 3.4).

Die Struktur der Gruppenobjekte besteht aus den gruppierten Objekten. Für in der Strukturdarstellung aufgeführte Objekte sei angegeben, welche ihrer Parameter gebunden sind. Parameterbindungen könnten einem Benutzer durch eine Netzdarstellung, wie in Abbildung 2.4 angedeutet, präsentiert werden.

3.4. Anschlußdarstellung

Die Anschlüsse funktionaler Objekte enthalten zur Übernahme bzw. Abgabe bereitstehende Datenobjekte. In den Anschlußkästchen wird dies durch Darstellung von Containern angedeutet (Abbildung 3.1). Ein Benutzer kann auf Wunsch den "Inhalt" der Container ein-

sehen, um die Wirkung einer Funktion zu bewerten. Der Inhalt eines Containers kann als Struktur eines Datenobjektes interpretiert werden. Die Struktur ist durch den anzugebenden "Typ" gekennzeichnet.

Das Darstellungslayout aus Abbildung 3.2 ist zur Strukturdarstellung eines Datenobjektes ebenfalls geeignet. Als Piktogramm links oben wird das für Datenobjekte, bekannt aus Abbildung 3.1, verwendet. Die Darstellung innerhalb der "Darstellungfläche" hängt von den darzustellenden Objekttypen ab.

Ein Beispiel ist in Abbildung 3.8 aufgezeigt. Der Name des Darstellungsobjekts deutet darauf hin, daß es sich um den Inhalt des Containers im ersten Eingangsanschlußkästchen eines "Median"-Verarbeitungsobjektes (z.B. aus Abbildung 3.5) handelt.

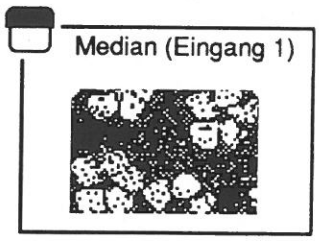


Abbildung 3.8:
Anschlußdarstellung mit Beispielbild

3.5. Darstellung von Ablaufsteuerungsattributen

Die Ablaufsteuerung ist durch Attribute der Interakionsobjekte festgelegt (s. Abschnitt 2.2.3). Diese Ablaufsteuerungsattribute sind durch Verwendung von Farbe dargestellt. So ist für 'cycle' das Piktogramm der Darstellungsobjekte in grün, für 'step' in gelb und für 'stop' in rot zu sehen. Es wird dadurch an einen Ampelmechanismus erinnert, durch den die Abarbeitung geregelt sein soll.

Sowohl das Piktogramm der Status- als auch das der Strukturdarstellung eines Objekts zeigt die Ablaufart in grüner, gelber bzw. roter Farbe, während das der Anschlußdarstellungen neutral in schwarz gehalten ist.

Leitungsobjekte, die gerade "leitend" sind, seien durch (schwach) gelb (-> 'step') blinkende Linienzüge gekennzeichnet, während nicht leitende in roter (-> 'stop') Farbe erscheinen.

3.6. Darstellung des Verarbeitungsfortschritts

Da Verarbeitungsfunktionen häufig recht lange Rechenzeiten erfordern, erscheint es wünschenswert, eine zumindest relative Anzeige des Fortschritts der Verarbeitung zu zeigen. Zu diesem Zweck wurde vorgesehen, unter dem Namen der Objektdarstellung (siehe Abbildung 3.2) eine Linie zu ziehen, deren relative Länge im Verhältnis zur Gesamtbreite (bis zum rechten Begrenzungsrand) den Verarbeitungsfortschritt anzeigt. In der Beispielabbildung 3.9 ist für ein Fourier-Transformationsobjekt eine ca. 33-prozentige Verar-

beitungsleistung angedeutet. Auch in der Strukturdarstellung eines Verarbeitungsobjekts wird der Verarbeitungsfortschritt in gleicher Weise kenntlich gemacht.

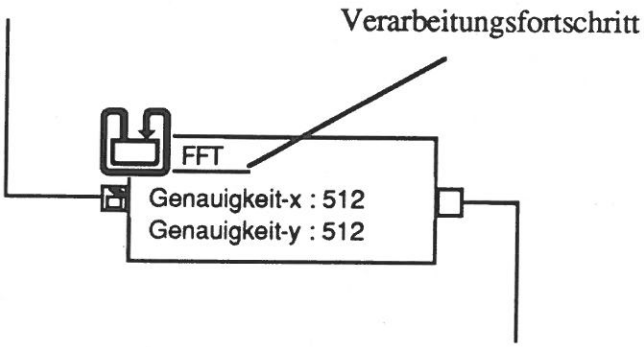


Abbildung 3.9: Darstellung des Verarbeitungsfortschritts

3.7. Darstellung der Wurzel der Objekthierarchie

Durch Kompositionsobjekte wird eine hierarchische Strukturierung von Interaktionsobjekten erreicht, wie in Abbildung 2.1 festgehalten. Das Wurzelobjekt der Hierarchie (dort als "Benutzerschnittstelle" bezeichnet) enthält alle verwendeten Objekte, wie z.B verschiedene "Experimente". Es stellt die Verbindung zur Umgebung der anderen Programmsysteme dar. Deshalb wird für dieses Objekt eine gesonderte Behandlung in bezug auf Status- und Strukturdarstellung vorgeschlagen. In der Statusdarstellung wird die objektabhängige "momentane" Bedeutung der Maustasten verdeutlicht. Hierzu bot sich die auf der Zielmaschine "Symbolics" übliche Darstellungsweise in der "schwarzen" Statuszeile unten am Bildschirmrand an. Die Strukturdarstellung übernimmt ein viereckiges Fenster, wie auch bei anderen Programmsystemen (z.B. Editor, Inspector, ZMail). Es erscheinen hier ähnlich wie in Abbildung 3.5 die Statusdarstellungen der enthaltenen Objekte. Eine Beispieldarstellung zeigt Kapitel 5. Fehlerzustände zeigt das vorhandene System zur Fehlerbehandlung (Debugger) in einem separat erscheinenden Fenster an.

Kapitel 4

VIPEX: Implementation der Schnittstelle

Begleitet von Hinweisen auf die Einbettung der Schnittstelle in die Sprache Lisp und die verwendete Programmierumgebung (speziell: Symbolics Common Lisp) geht dieses Kapitel auf syntaktische Merkmale der Interaktion ein. Dabei wird der Kontrollfluß der Implementierung (VIPEX : Visual Programming of Experimental Systems) anhand von Netzen dargestellt.

4.1 Das Schalten von Verarbeitungsobjekten als Evaluierung von Lisp-Funktionen

Die in Kapitel 2.1 vorgestellten Verarbeitungsobjekte sind innerhalb der Benutzerschnittstelle u.a. zur Modellierung von Programmfunktionen vorgesehen. Von einem Generierungssystem (s. Kap. 2.5) soll eine Lisp-Funktion bereitgestellt werden, die einen vorgesehenen Algorithmus implementiert.

4.1.1 Funktionsparameter

Als Parameter der von einem Generierungssystem bereitgestellten Funktion treten zunächst alle Parameter eines Objekts nach 2.2.2 auf. Weiterhin sind die Eingangsdatenobjekte als Parameter vorgesehen. Als weiteren Parameter kann eine Funktion, die das Verhalten eines Verarbeitungsobjektes implementiert, eine von der Benutzerschnittstelle zur Verfügung gestellte Funktion zur Darstellung des Verarbeitungsfortschritts nach Abschnitt 3.6 übernehmen. Diese Funktion soll mit (reellwertigen) Argumenten von 0.0 bis 100.0 (in aufsteigender Reihenfolge) innerhalb der Bildverarbeitungsfunktion evaluiert werden und stellt den Verarbeitungsfortschritt prozentual (wie in Abbildung 3.9) dar. Nicht jede Funktion muß dieses Vorgehen unterstützen, sie bietet dann jedoch dem Benutzer keine Auskunft über den Verarbeitungsfortschritt.

4.1.2 Zusammenwirken der Funktionen

Der Funktionsaufruf kann in Anlehnung an die Netz-Theorie [*Reisig 82*, *Jessen & Valk 87*] auch als *Schalten* des Verarbeitungsobjekts (im Sinne einer Transition) aufgefaßt werden. Wegen der nicht anonymen "Marken" (Datenobjekte) zeigt sich das

vorgestellte System als spezielle Klasse von Pr/T-Netzen (auch als gefärbte Netze bekannt).

Ein Verarbeitungsobjekt heißt *aktiviert*, wenn

a) alle Eingangsdatenobjekte vorhanden und
b) alle Ausgangsanschlüsse frei sind.

Die Kombination "Ausgangsanschluß-Leitungsobjekt-Eingangsanschluß" (im folgenden auch als "Kanal" bezeichnet) kann als zwischen zwei Verarbeitungsobjekten liegende Kombination "Stelle-Transition-Stelle" interpretiert werden (Abbildung 4.1). Da jede Stelle höchstens eine Eingangs- und Ausgangstransition besitzt, ergibt sich, daß ein zyklenfreies System aus zusammengeschalteten Verarbeitungsobjekten ein (mit nicht anonymen Marken) markiertes *Kausalnetz* darstellt und damit als *Auftragssystem* gedeutet werden kann [*Jessen & Valk 87*].

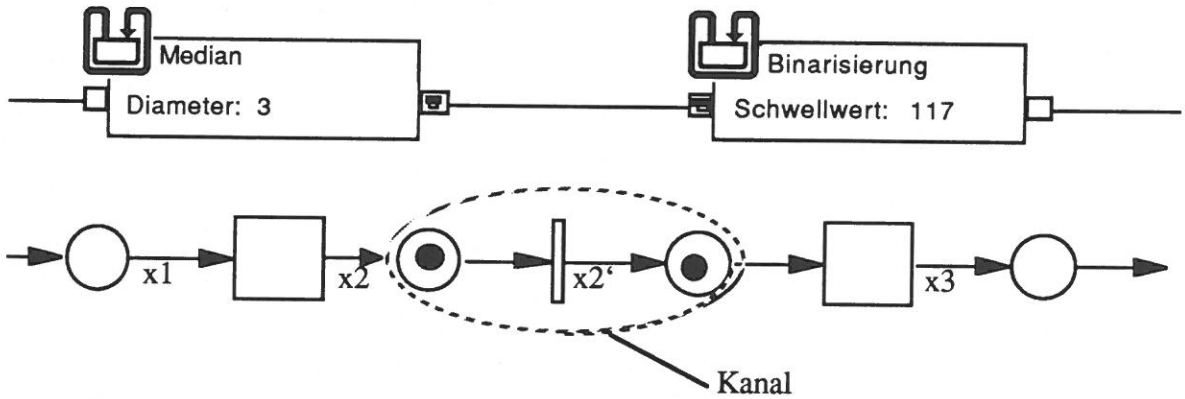


Abbildung 4.1: Vergleich mit Pr/T-Netzen

In "VIPEX" wurde ein Schalten nach dieser Vorgehensweise für alle funktionalen Objekte vorgesehen. Das bedeutet, daß auch Abstraktionsobjekte z.B. ihre Eingangsdatenobjekte erst dann nach innen weiterleiten, wenn alle Eingangsanschlüsse belegt sind. Sieht man Abstraktionsobjekte als Vergröberungen, so ist das vergröberte Netz also wieder ein Pr/T-Netz. Da sowohl auf der Eingangs- als auch auf der Ausgangsseite eines funktionalen Objekts Anschlußkästchen vorhanden sind, ergibt sich die in Abbildung 4.1 dargestellte Situation mit zwei Stellen pro Verbindungskanal.

4.1.3 Funktionswerte als Datenobjekte

Sobald ein Verarbeitungsobjekt aktiviert ist, schaltet es, d.h. die zugehörige Lisp-Funktion wird evaluiert. Die Ausgangsdatenobjekte ergeben sich als *Werte* der Funktionsaus-

wertung. Die Verwendung von einfachen Lisp-Funktionen legt die Implementierung der Datenobjekte durch "S-Expressions" nahe. Die Funktionen können damit unabhängig von der Dialogschnittstelle entworfen, implementiert und ausgetestet werden. Dies soll jedoch nicht besagen, daß der Lisp-Code der Funktionen von der Schnittstelle aus nicht zugänglich ist. Durch die in Abschnitt 3.3 vorgestellte Strukturdarstellung für Verarbeitungsobjekte (Abbildung 3.7) wird ein Editor zur Funktionsmodifikation angeboten. Der Philosophie von Lisp folgend, kann hier leicht jede Funktion geändert und für sich neu evaluiert bzw. übersetzt werden.

Zur einfachen Implementierung gerade solch komplizierter Systeme wie Editoren wurde die Verwendung des als CommonLisp-Erweiterung angebotenen "Flavor-Systems" [*Symbolics 85*] erforderlich. Die Verwendung der in einem besonderen Modul (Package) zur Programmierung des Fenstersystems vordefinierten Objektklassen (Flavors) soll im nächsten Abschnitt erläutert werden.

4.2 Darstellung von Objekten mithilfe von Fenstern

Einem Benutzer muß die Möglichkeit geboten werden, die Darstellungen nach seinen Vorstellungen auf dem Bildschirm plazieren zu können. Aufgrund des beschränkten Platzes sind Überlagerungen nicht zu vermeiden. Implementierungstechnisch lassen sich diese Anforderungen durch Fenstermechanismen am einfachsten erfüllen, da Fenster leicht verschoben werden können und z.B. durch Verschiebung freigelegte Bildschirmbereiche vom System automatisch neu gezeichnet werden usw.

Das Symbolics-Fenstersystem ist recht mächtig und anpassungsfähig. Es lassen sich aus vordefinierten Objektklassen einfache Fenster mit oder ohne Rand (Border), mit oder ohne Titelzeile (Label) etc. erzeugen. Es können aber auch ganz spezielle Fenster wie Menüs oder Editoren instanziiert werden. Alle Fenster bilden ein Rechteck. Für das in Kapitel 3 vorgestellte "Layout" der Darstellung (Abbildung 3.2 ff.) bedeutet dies, daß aufgrund der hervorstehenden Anschlußkästchen "tote" Zonen um die Ränder entstünden, falls man ein (einziges) Fenster zur Darstellung benutzte. Daher wurde das Darstellungslayout durch Verwendung mehrerer *verbundener* Fenster realisiert. Das Konstruktionsprinzip wird in Abbildung 4.2 durch Darstellung der Fenster in "explodierter" Form veranschaulicht. Die bei Darstellung vieler Objekte hohe Anzahl aktiver Fenster bereitet der Maschine jedoch in keiner Weise Probleme, obwohl durch die Verwendung von Fenstern als "Bausteine" für die Darstellungsobjekte hohe Anforderungen gestellt werden.

Durch das Weglassen von bestimmten Rändern (borders) wird der Eindruck eines zusammenhängenden "Gebietes" erweckt (Abbildungen 3.4 ff.). Der Objektname wird als Name (Label) des Namensfensters verwendet. Anschlußfenster werden in benötigter Anzahl links und rechts plaziert. Das Piktogramm wird als Zeichen eines speziellen Fonts im Pikto-

grammfenster dargestellt. Die zur Piktogrammdarstellung voreinstellungsmäßig verwendeten Zeichen sind vorgegeben (Abbildung 3.3), obwohl ein Generierungssystem auch eigene Piktogramme mithilfe eines Fonteditors bereitstellen kann.

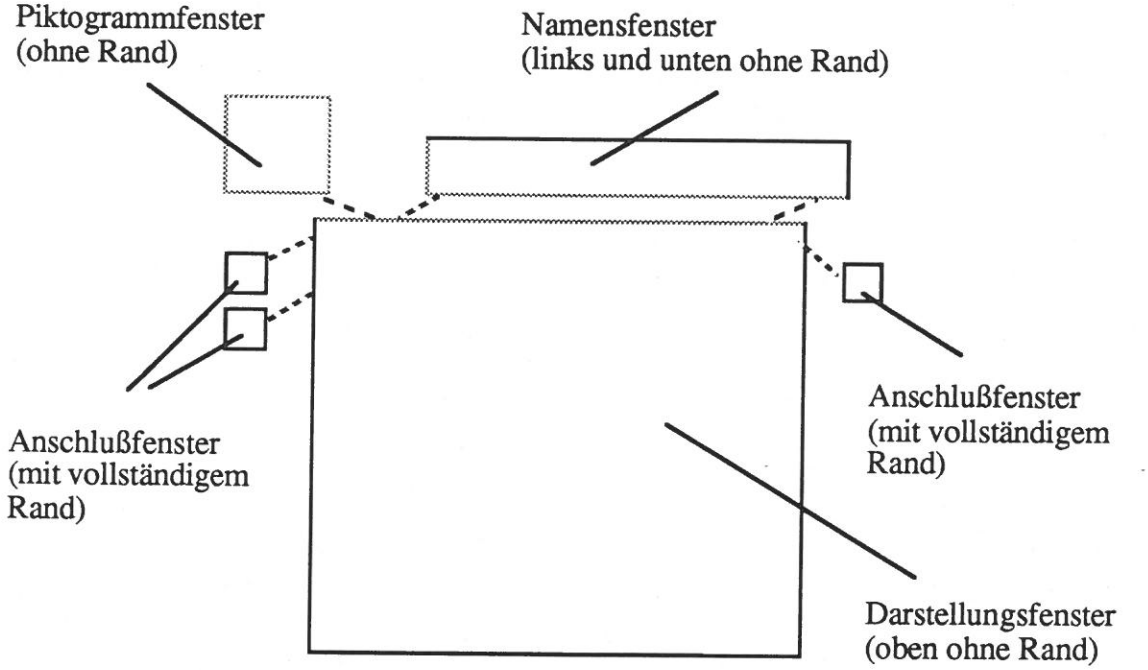


Abbildung 4.2: Windows als Bausteine in explodierter Darstellung

Die Klassen der Darstellungsfenster unterscheiden sich je nach Darstellungsklasse erheblich. Zur Parameterpräsentation (Abschnitt 3.2) konnte auf eine spezielle Menü-Objektklasse zur Eingabe und Änderung von Variablenwerten (choose-variable-values-menu) zurückgegriffen werden. Durch diese mächtige Objektklasse ist es möglich, verschiedene Eingabemöglichkeiten auf einfache Weise nur unter Angabe einer Liste, bestehend aus einem Namen (String), zugehöriger Variable (Symbol) sowie auch einer Typangabe (z.B `:number`, `:pathname`, `:string`), zu implementieren.

Um einen maussensitiven Bereich wird, wie aus der Programmierumgebung bekannt, ein Rechteck gezeichnet, sobald der Mauszeiger in einem solchen Bereich bewegt wird. In der Statuszeile (vgl. Abschnitt 3.7) kann die Bedeutung der Maustasten abgelesen werden. Je nach Maustastenklick kann nun ein Parameterwert neu eingegeben (Linksklick) oder nur editiert werden (Mittelklick).

Da die einem Eingabe-Menü zur Verfügung zu stellenden Angaben einen Typ enthalten - vordefinierte sind vorhanden, neue können definiert werden - kann nach Abschluß der Eingabe (mit der Return-Taste) von dem Menüobjekt (choose-variable-values-menu) selbst eine Fehleingabenüberprüfung vorgenommen werden. Das System unterstützt eine ggf. gewünschte Sichtfensterverschiebung, falls der zur Verfügung gestellte Platz nicht zur Dar-

stellung aller Parameter ausreicht. Der verwendete Mechanismus ist identisch mit dem Verschiebemechanismus (Scrolling) des ZMACS-Editors und anderer Systemkomponenten, bedarf also keiner Eingewöhnung, sofern diese Systeme bekannt sind.

Die Forderung nach einfacher Parameteränderung ist durch einfaches Anklicken der zu ändernden Werte erfüllt. Durch die Anzeige der maussensitiven Bereiche wird einem Benutzer direkt visualisiert, welche Bedeutung die (drei) Maustasten je nach Position des Mauszeigers haben.

Ein weiteres Beispiel für die Verwendung vordefinierter Klassendefinitionen stellt die schon häufiger erwähnte Strukturdarstellung eines Verarbeitungsobjekts (Abbildung 3.7) dar. Die Klassendefinition der Darstellungfläche entspricht einem "ZMACS-Frame", einer instanziierbaren Editorfensterklasse. Die Bedienung ist auf die Lisp-Programmierung abgestimmt.

Durch Einbettung solch geeigneter Klassendefinitionen kann die Codelänge der Implementation erheblich reduziert werden. Leider gilt dies nicht unbedingt für den Programmieraufwand, da die Einarbeitungs- und Vertiefungszeiten erheblich sind und die Tücke oft im Detail des Zusammenwirkens der verwendeten Objektklassen steckt. Für ein System wie "VIPEX", welches die Möglichkeiten des Fenstersystems weit ausschöpft, sind vertiefte Kenntnisse vonnöten, die über das, was in den angebotenen Handbüchern steht, (zu) weit hinausgehen.

Verbundene Fenster, die eine graphische Darstellung bilden, erscheinen einem Benutzer als ein Ganzes, also als ein einziges (nicht viereckiges) "Fenster", und können mit vertrauten Kommandos wie "MOVE", "RESHAPE", "OPEN", etc. manipuliert werden. Hierbei erweist es sich als vorteilhaft, daß die Leitungsobjekte sich ihren Weg selbst berechnen. Durch Leitungen verbundene Statusanzeigen (wie z.B in Abbildung 3.5) können verschoben werden, ohne die Leitungen "von Hand" neu verlegen zu müssen.

4.3 Interaktion durch Mausklick oder Menüauswahl

Die vom "Star"-System [*Smith et al. 82*] bekannte einprägsame Interaktionsform der *universellen Kommandos* sollte weitestgehend übernommen werden. Durch Verwendung vordefinierter Objektklassen ist die Interaktionsgestaltung in einigen Bereichen jedoch schon festgelegt (der ZMACS-Editor orientiert sich nicht an diesem Prinzip). Es wurde jedoch versucht, bezüglich der in Kapitel 3 geschilderten Darstellungsform eine möglichst einheitliche Interaktionsform zu finden.

Das zur Objektrepräsentation dienende Piktogramm ist maussensitiv, hat also eine bzgl. Mausklick definierte Funktion. Ebenfalls maussensitiv sind die Anschlußfenster. Durch *Einfachklick mit der linken Maustaste* auf maussensitive Darstellungsobjektbereiche er-

scheint das in Abbildung 4.3 dargestellte - aus einer vordefinierten Objektklasse instanziierte - Menüobjekt. Durch Auswahl kann eine Reihe von allgemein geltenden Kommandos gegeben werden: ob nun ein Darstellungsobjekt vergrößert, bewegt oder aber ein Anschluß eines Abstraktionsobjekts gelöscht werden soll. Nach *Preece et al. 87* könnte man Menüobjekte als Meta-Objekte bezeichnen.

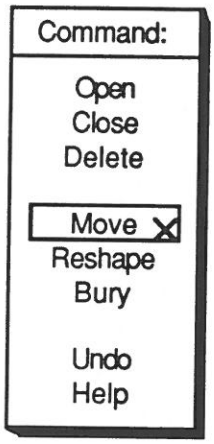


Abbildung 4.3
Menüobjekt
zur Interaktion

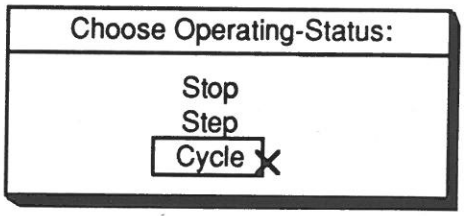


Abbildung 4.4.:
Menü zur Auswahl der Ablaufart

Einer schnelleren Ausführung wegen wurden einige der Kommandos aus dem Menü in Abbildung 4.3 auf "Doppelklick-Links" gelegt. *Doppelklick mit der linken Maustaste* auf das Piktogrammsymbol einer Statusanzeige bewirkt z.B. das "Öffnen" eines Objekts ("Open" in Abbildung 4.3) bzw. ein "Nach-oben-Holen" der entsprechenden Darstellung, falls diese schon "geöffnet" ist. Dieses gilt sowohl für die Statusdarstellung, die "geöffnet" werden kann, um die Struktur des Objekts zu zeigen, als auch für die Anschlüsse, die "geöffnet" werden können, um den Inhalt der Container für die Datenobjekte freizulegen. Auf eine ausführliche Erläuterung aller Kommandos soll hier zugunsten einer mehr an Konzepten orientierten Darstellung verzichtet werden.

Diese allgemeinen Mechanismen sind z.B. dem Apple Macintosh angelehnt und weitläufig bekannt. Die obere "Menüleiste" dieses Systems wurde jedoch nicht übernommen. Um einem Benutzer das Gefühl einer direkten Manipulation dargestellter Objekte zu geben, wurden sog. "Pop-Up"-Menüs vorgezogen. Auf dem zur Verfügung stehenden 20" Bildschirm wären (je nach Position des Mauszeigers) die zur Auswahl eines Kommandos zurückzulegenden Wege zur Menüleiste außerdem recht lang.

Die Veränderung der Attribute zur Festlegung der Ablaufart (Abschnitt 2.2.3) soll ebenfalls leicht möglich sein. Wiederum durch Klick auf das Piktogramm, diesmal jedoch mit der *mittleren Maustaste,* erscheint das Menüobjekt zur Auswahl der Ablaufart aus Abbildung

4.4. Doppelklick mit der mittleren Maustaste bewirkt hier ein einfaches Fortschalten von rot (-> 'stop') auf gelb (-> 'step'), gelb auf grün (–> 'cycle'), usw., stellt also wieder die verkürzende Interaktionsform dar.

Die verwendeten Menüfenster sind mit einer Umrandung, die Schattenwirkung darstellt, versehen und heben sich durch die räumliche Wirkung von der Masse der Darstellungsobjekte deutlich ab.

Wie in anderen Programmsystemen auf der Symbolics-Maschine üblich, bewirkt ein Klick auf die *rechte Maustaste* das Erscheinen eines Interaktionsmenüs mit Funktionen wie "Create", "Save", etc., während ein Doppelklick der rechten Maustaste das globale Systemmenü der Symbolics-Programmierumgebung erscheinen läßt. Die Funktion der rechte Maustaste ist unabhängig von den maussensitiven Bereichen.

4.4 Nebenläufigkeit implementiert durch Prozesse

Um einem Benutzer eine datenflußorientierte Visualisierung bieten zu können, erscheint eine nebenläufige Evaluierung der zu Verarbeitungsobjekten gehörenden Lisp-Funktionen eine einfache Implementierungsstrategie zu sein (im Gegensatz zu Agenda-Mechanismen). Da bei parallelen Leitungspfaden auch ein kollaterales Leiten zu modellieren ist, wurde auf vordefinierte Prozeßobjektklassen zurückgegriffen. Jedem Verarbeitungs- und Leitungsobjekt ist also eine Prozeß-Instanz zugeordnet. In diesen Prozessen werden die entsprechenden Funktionen dann nebenläufig evaluiert. Je nach Eintreten der aus Kapitel 4.1 bekannten Aktivierungsbedingungen, die über Vorhandensein oder Abwesenheit von benötigten Datenobjekten Auskunft geben, wird ein Prozeß in den Zustand 'run' oder 'wait' versetzt. Die Kontrollflußsteuerung konnte somit vollständig der Prozeßverwaltung (Scheduler) des Betriebssystems übertragen werden, wobei eine Synchronisation durch Kommunikation über Nachrichten an Objekte erfolgt, die allen Prozessen zugänglich sind. Spezielle Timing-Probleme konnten durch ein zur Verfügung stehendes Konstrukt zur Erzeugung von unteilbaren Handlungen [*Jessen & Valk 87*] gelöst werden.

Kapitel 5

Darstellung eines Ablaufbeispiels

Das Zusammenspiel der in vorangegangenen Kapiteln vorgestellten Konzepte und Darstellungsformen soll nun anhand eines Anwendungsbeispiels erläutert werden. Die vorgestellten Bildschirmkopien sind dabei als "Schnappschüsse" einer Interaktionssitzung aufzufassen. Leider können die durch Farbe repräsentierten Ablaufsteuerungsattribute (Kapitel 3.5) nicht in den Darstellungen deutlich gemacht werden. Ebenso werden die gezeigten Grauwertbilder nur andeutungsweise wiedergegeben.

Die Objekte des gezeigten Anwendungsbeispiels sind schon vorkonfiguriert (in *Faasch 87* als statische Konfiguration bezeichnet); der Benutzer kontrolliert durch seine Interaktionen den Ablauf des "Experiments". Als Beispiel zur Interaktion wird eine bekannte Folge von Realweltbildern gezeigt, auf der ein Abbiegevorgang eines Taxis in eine Seitenstraße festgehalten ist.

5.1 Strukturdarstellung von Kompositionsobjekten

In Abbildung 5.1 sind zwei Kompositionsobjekte dargestellt. Das eine beinhaltet eine Objektkonfiguration zur Bildfolgenaufzeichnung ("Image Sequence Recording"), das andere ein Experiment zur Erkennung von bewegten Objekten ("Object Detection"). Während für "Image Sequence Recording" das Standard-Piktogramm für Kompositionsobjekte nach Abbildung 3.3 beibehalten wurde, wird "Object Detection" durch ein eigenes Symbol gekennzeichnet. Die Piktogrammgrundform für Kompositionsobjekte (Abbildung 3.3) ist jedoch erhalten geblieben. Durch Einhaltung dieser Konvention kann sofort die Objektklasse erkannt werden.

Beide Kompositionsobjekte dienen mehr der Verwaltung und Zusammenfassung von Objekten denn der Abstraktionsbildung. Um ein Experiment "durchzuführen", werden die Kompositionsobjekte nach Bedarf durch Mausklick (siehe Abschnitt 4.3) "geöffnet". Dies sei in Abbildung 5.2 für "Object Detection" bereits geschehen.

Der gewählte Algorithmus zur Erkennung von bewegten Objekten spiegelt sich in der Strukturdarstellung wider: Nachdem jeweils zwei Bilder einer Folge durch das Objekt "Image Pair" geliefert sind, werden sie durch "Difference" voneinander subtrahiert. Anschließend wird das Differenzbild in "Object Mask" durch Schwellwertbildung zu einem Binärbild weiterverarbeitet. Bereiche, durch die sich Objekte bewegt haben, erscheinen als schwarze Objektmasken. Dadurch kann auf bewegte Objekte zurückgeschlossen werden.

Auf die Funktionen der Gruppenobjekte wird später im Zusammenhang mit den Parameterbindungen und der Steuerung des Ablaufs noch eingegangen.

Die in Abbildung 5.3 gezeigte Struktur von "Image Pair" besteht aus zwei Verarbeitungsobjekten zum "Laden" jeweils eines Bildes sowie nachgeschalteten Abstraktionsobjekten zur Bildaufbereitung. Die Bildaufbereitungsobjekte setzen sich jeweils aus einem Median-Filter und einem Objekt zur Kontrasterhöhung zusammen. Abbildung 5.4 zeigt die Struktur von "Difference" und "Object Mask".

5.2 Parameterbindungen

Sei die zu untersuchende Bildfolge etwa durch "Image Sequence Recording" im Dateisystem unter dem Pfad "S:>Bilder>taxi>" in jeweils einer Datei pro Bild abgelegt (zur Syntax des Pfadnamens siehe *Symbolics 85*). Der Name der Bilddateien sei "tax1.image", "tax2.image", ... "tax44.image". Das Verarbeitungsobjekt "Load Odd Images" lädt (modulo 44) die ungeraden, "Load Even Images" die geraden Bilder der Folge. Da beide Ladeobjekte denselben Pfadnamen etc. benötigen, sollten diese Parameter durch ein Gruppenobjekt oder durch das "übergeordnete" Kompositionsobjekt gebunden werden. In dem Beispiel ist letzteres der Fall. Angezeigt wird dieses durch den Zusatz "(Image Pair)" bei den Parametern der Ladeobjekte. Lokale Änderungen sind nicht möglich (siehe Kapitel 2.3.2). Zusätzlich wird durch "Image Pair" der "Enhance Factor" der Objekte zur Kontrasterhöhung in den Bildaufbereitungsobjekten gebunden. Hier liegt eine mehrstufige Vererbungsstruktur vor. Der Parameter "Diameter" des Objekts "Median 1" kann durch das Gruppenobjekt "Median Diameter" geändert werden. In Abbildung 5.4 ist zudem zu erkennen, daß in der Beispielkonfiguration der Parameter "Threshold" des Schwellwertoperators "Thresholding" an das übergeordnete Abstraktionsobjekt "Object Mask" gebunden ist. Anmerkung: Der Schwellwert kann direkt aus dem Histogramm berechnet werden. Der Histogrammanschluß des Schwellwertoperators "Thresholding" steuert in diesem Beispiel den Kontrollfluß. Das Schwellwertobjekt kann nur nach Berechnung des Histogramms aktiv werden (Abbildung 5.4 und 5.9). Das Histogramm kann so vor der Binarisierung untersucht werden, wenn der Verarbeitungsmodus von "Thresholding" auf 'stop' gesetzt ist. Durch das Gruppenobjekt "Histogram Style" ist die Verarbeitungsart von "Histogram 2" gebunden (nicht in den Abbildungen erkenntlich, da die Strukturdarstellung von Gruppenobjekten noch nicht implementiert ist). Durch Änderung der Histogrammverarbeitungsart durch "Histogram Style" kann der Kontrollfluß also auch gesteuert werden, wenn die Struktur von "Object Mask" nicht (vollständig) zu sehen ist.

5.3 Ablaufüberwachung

Durch "Öffnen" von Anschlußdarstellungen (Kap. 3.4) wird der "Weg" eines Bildpaares durch die Verarbeitungspipeline(s) verfolgt. Anschlußdarstellungen sind durch die Präfixe "Arr." (Arrival) oder "Dep." (Departure) und einer Anschlußnummer vor dem Namen des zugehörigen Interaktionsobjekts gekennzeichnet.

Die ersten beiden Bilder der zu verarbeitenden Bildfolge werden geladen nachdem die Verarbeitungsart der beiden Ladeobjekte z.B. auf 'cycle' gesetzt ist (Abbildung 5.5). Durch das Gruppenobjekt "Loader Group" (in "Object Detection") seien die Verarbeitungsarten der beiden Ladeobjekte "Load Odd Images" und "Load Even Images" gebunden. Es genügt daher, die Verarbeitungsart von "Loader Group" auf 'cycle' zu setzen, um beide Ladefunktionen zu "starten".

Abbildung 5.6 zeigt ein Stadium, in dem beide Bilder weitergeleitet sind. Die Anschlußdarstellung von "Enhance Image 1" präsentiert das erste Taxibild. Da die Ladeobjekte auf 'cycle' gesetzt sind, liefern sie sofort das nächste Bildpaar nach. In Abbildung 5.7 ist die Verarbeitung soweit fortgeschritten, daß sich die geladenen Bilder durch die Bildaufbereitungsobjekte bis in die Ausgangsanschlüsse von "Image Pair" bewegt haben. Das aufbereitete Bild kann in der Anschlußdarstellung des ersten Ausgangs von "Image Pair" betrachtet und mit dem "Originalbild" verglichen werden. Zur Berechnung der Objektmaske ist es u.U. nicht nötig, ja sogar störend, die Bilder wie gezeigt aufzubereiten; dieses wird hier zu Demonstrationszwecken durchgeführt.

Obwohl sich im Eingangsanschluß von "Enhance Image 1" kein Datenobjekt mehr befindet (kein Container im Anschlußkasten), wird das vormalige Datenobjekt in der Anschlußdarstellung noch gezeigt (Abbildung 5.7). Müßte die Anschlußdarstellung aus Konsistenzgründen eigentlich "gelöscht" werden, so möchte man doch die (auf dem Bildschirm vorhandene) Information nicht verschenken, sondern zu Vergleichszwecken erhalten. Ähnliche Konsistenzprobleme sind in *Smith et al. 82* erwähnt (Soll ein zum Druckerpiktogramm bewegtes Dateipiktogramm - und damit die zu druckende Datei - nach dem Druckvorgang gelöscht werden?). In der vorliegenden Implementation wird das Anschlußdarstellungpiktogramm in blau gezeigt, wenn sich ein Datenobjekt im Anschluß befindet, sonst ist es in schwarz gehalten.

Haben alle Objekte innerhalb von "Image Pair" die Verarbeitungsart 'cycle', kann der "Bilderfluß" durch die Verarbeitungsart von "Image Pair" selbst gesteuert werden. Wird diese auf 'step' gesetzt, so erscheint genau ein Bildpaar in den Ausgängen. Sind die Bilder geliefert, so schaltet das Objekt auf 'stop'. Diese Bilder werden solange "festgehalten", bis die Verarbeitungsart entweder auf 'step' oder 'cycle' gesetzt wird. Die Strukturdarstellung von "Image Pair" kann also geschlossen werden.

Das Gruppenobjekt "Loader Group" wird daher nicht unbedingt zur Steuerung des Kontrollflusses benötigt. Durch folgende Überlegung erhält dieses Objekt jedoch noch eine andere Bedeutung: Durch Änderung des Pfadnamens "Pathname", des Dateinamens "Filename" sowie der anderen Parameter von "Image Pair" kann eine andere Bildfolge zur Verarbeitung geladen werden. Falls nach der Änderung des Pfadnamens z.B. noch der Dateiname geändert werden muß, könnte in diesem temporär inkonsistenten Zustand von den Ladeobjekten versucht werden, eine nicht existierende Datei anzusprechen (Verarbeitungsarten auf 'cycle'). Durch das Gruppenobjekt "Loader Group" können beide Ladeobjekte während der Parameteränderung "auf Eis gelegt" und eine Fehlermeldung vermieden werden. Die Probleme mit inkonsistenten Zuständen sind nicht durch die Benutzerschnittstelle entstanden, sondern in diesem Beispiel durch die Aufteilung der Dateispezifikation in mehrere Teile bedingt.

Die nächste Abbildung (5.8) beschreibt die Verarbeitungsfolge, nachdem das Differenzbild berechnet wurde. Dieses ist in der Eingangsanschlußdarstellung von "Object Mask" zu erkennen. Die weiteren Anschlußdarstellungen dienen dazu, in der weiteren Ablauffolge das berechnete Histogramm des Differenzbildes sowie das binarisierte Differenzbild darzustellen. Abbildung 5.9 zeigt die Situation nach der Schwellwertanwendung auf das Differenzbild. An der Eingangsdarstellung von "Store Object" wie auch am Histogramm erkennt man die ungünstige Schwellwertwahl von "128" (etwa in der Mitte der Spitze des Histogramms).

5.4 Parameteränderung

Der Parameter "Threshold" von "Object Mask", der sich auf das Objekt "Thresholding" vererbt, soll geändert werden. Durch Anklicken (Mittelklick) des Wertes "128" (s. Kap. 4.2) kann dieser editiert werden. Der neue Wert betrage "100" (Abbildung 5.10).

Das Gruppenobjekt "Histogram Style" bestimmt den Ausgabestil der von "Histogram 2" berechneten Histogramme. Der Parameterwert kann durch Anklicken eines der vier Werte "direkt" ausgewählt werden. Der aktuelle Wert ist durch die Schriftart "bold" kenntlich gemacht. In Abbildung 5.11 wurde der Stil des Histogramms für das nächste Bildpaar auf "LINES" geändert. Nachdem das nächste Bildpaar die Pipelines durchlaufen hat, kann aus den Anschlußdarstellungen entnommen werden, daß "100" in dieser Situation ein besserer Schwellwert ist (Abbildung 5.11).

Wie schon in Kapitel 4.2 erwähnt, können die Sichtfenster der Parameterdarstellungen verschoben werden, falls der zur Verfügung stehende Platz nicht ausreicht. Abbildung 5.12 und 5.13 zeigen einige Variationen der Parameterdarstellungen von "Image Pair" und "Histogram Style".

5.5 Strukturdarstellung von Verarbeitungsobjekten

Für die verwendeten Verarbeitungsfunktionen konnte auf Funktionen eines vorhandenen Bildverarbeitungssystems ImageCalc™ [*Image Calc 84*] zurückgegriffen werden (auch die Funktionen zur Bild- und Histogrammausgabe in den Anschlußdarstellungen stammen aus dem System - siehe hierzu Anhang B). In Abbildung 5.14 ist das Verarbeitungsobjekt "Thresholding" in seiner Strukturdarstellung gezeigt. Wie im ZMACS-Editor können die Funktionen geändert und neu übersetzt bzw. evaluiert werden usw.

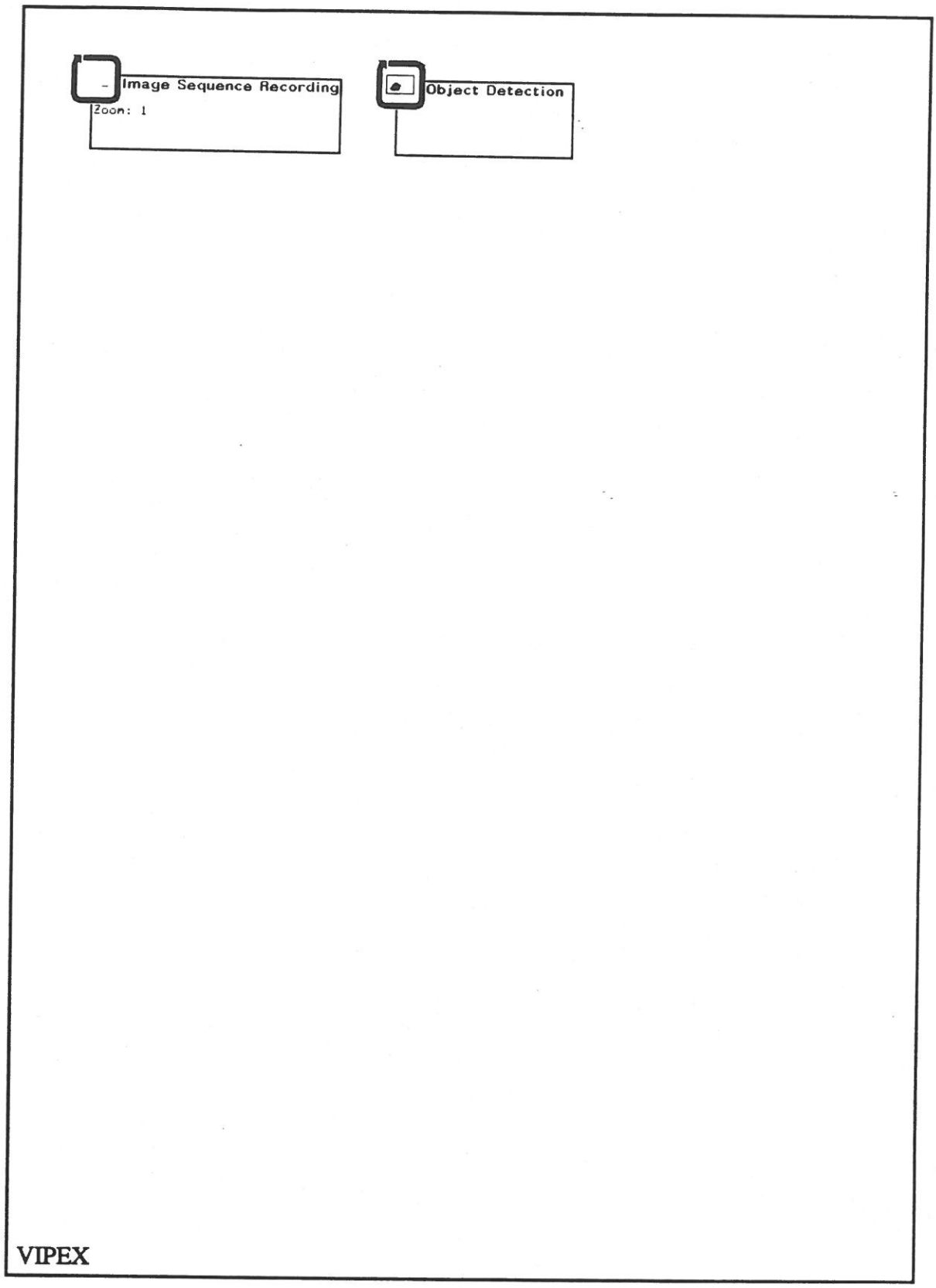


Abbildung 5.1

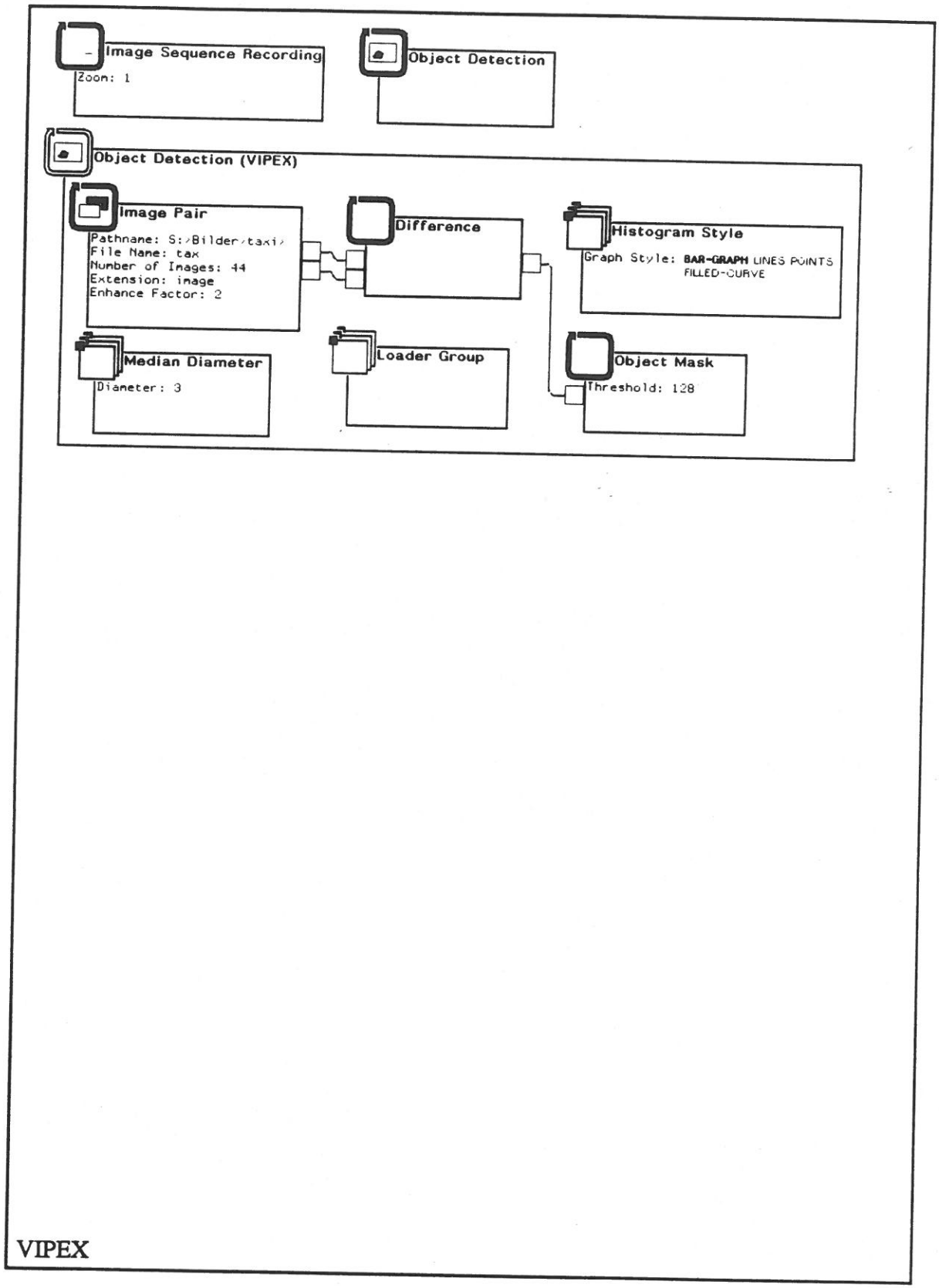


Abbildung 5.2

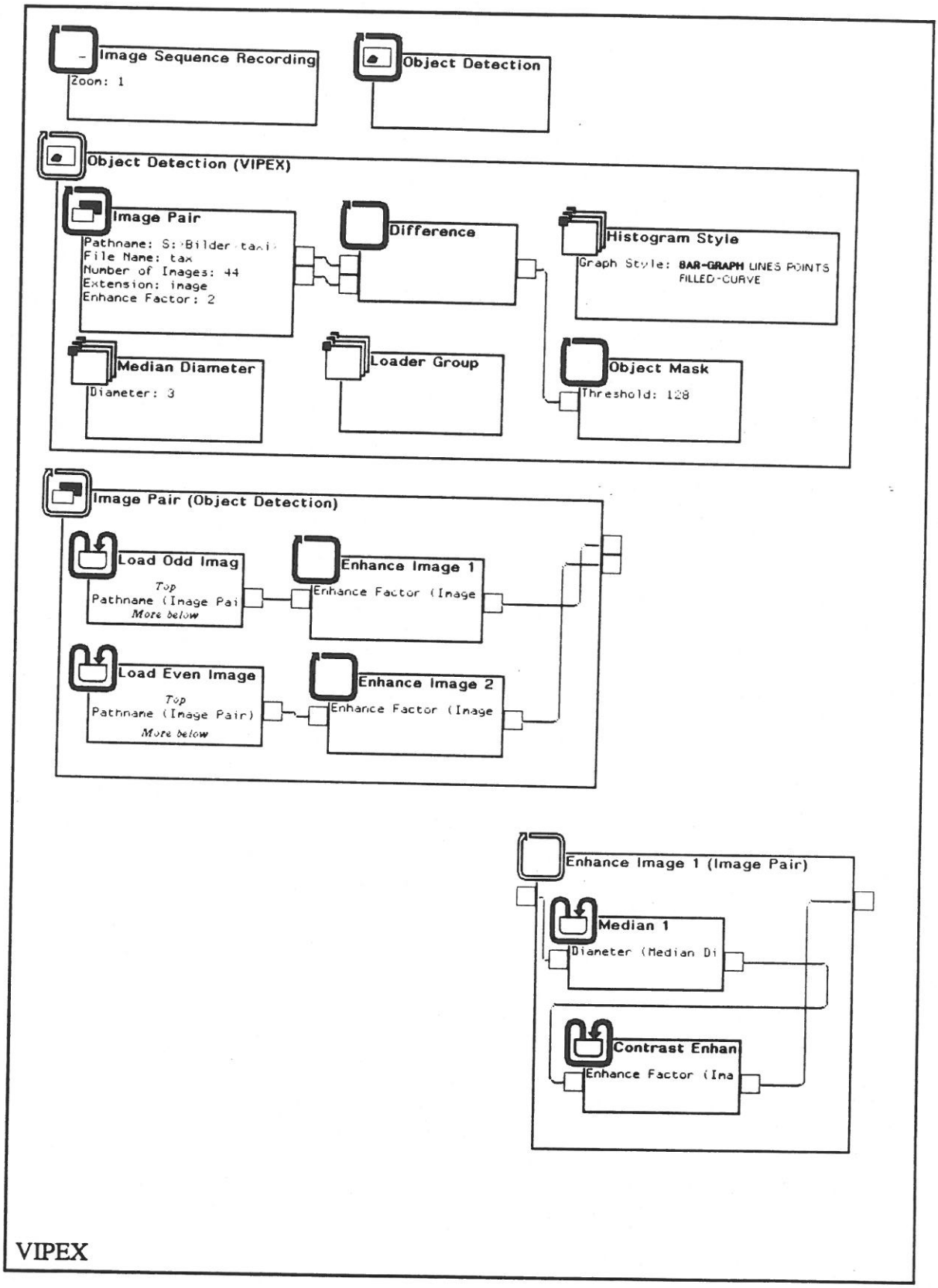


Abbildung 5.3

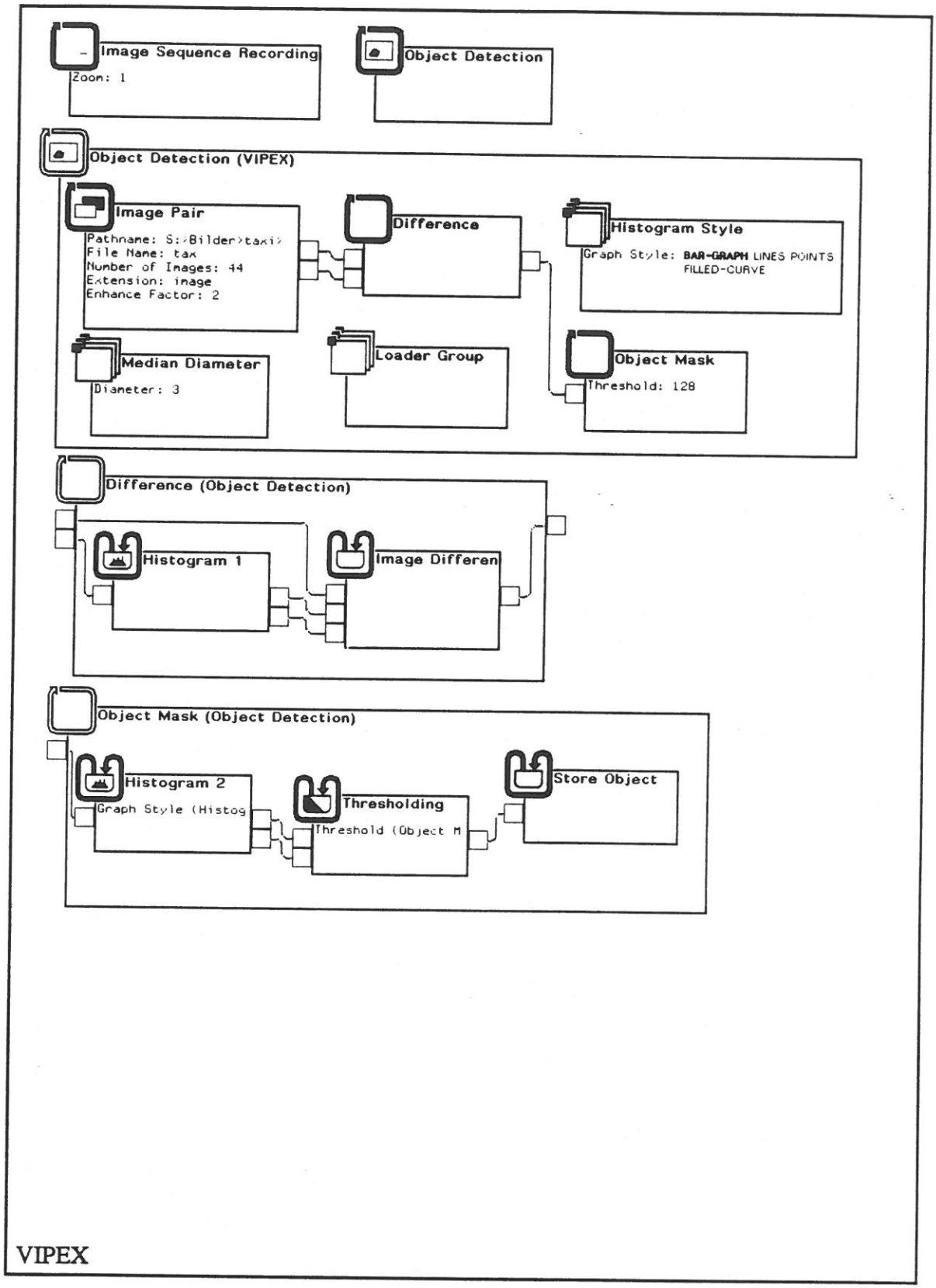


Abbildung 5.4

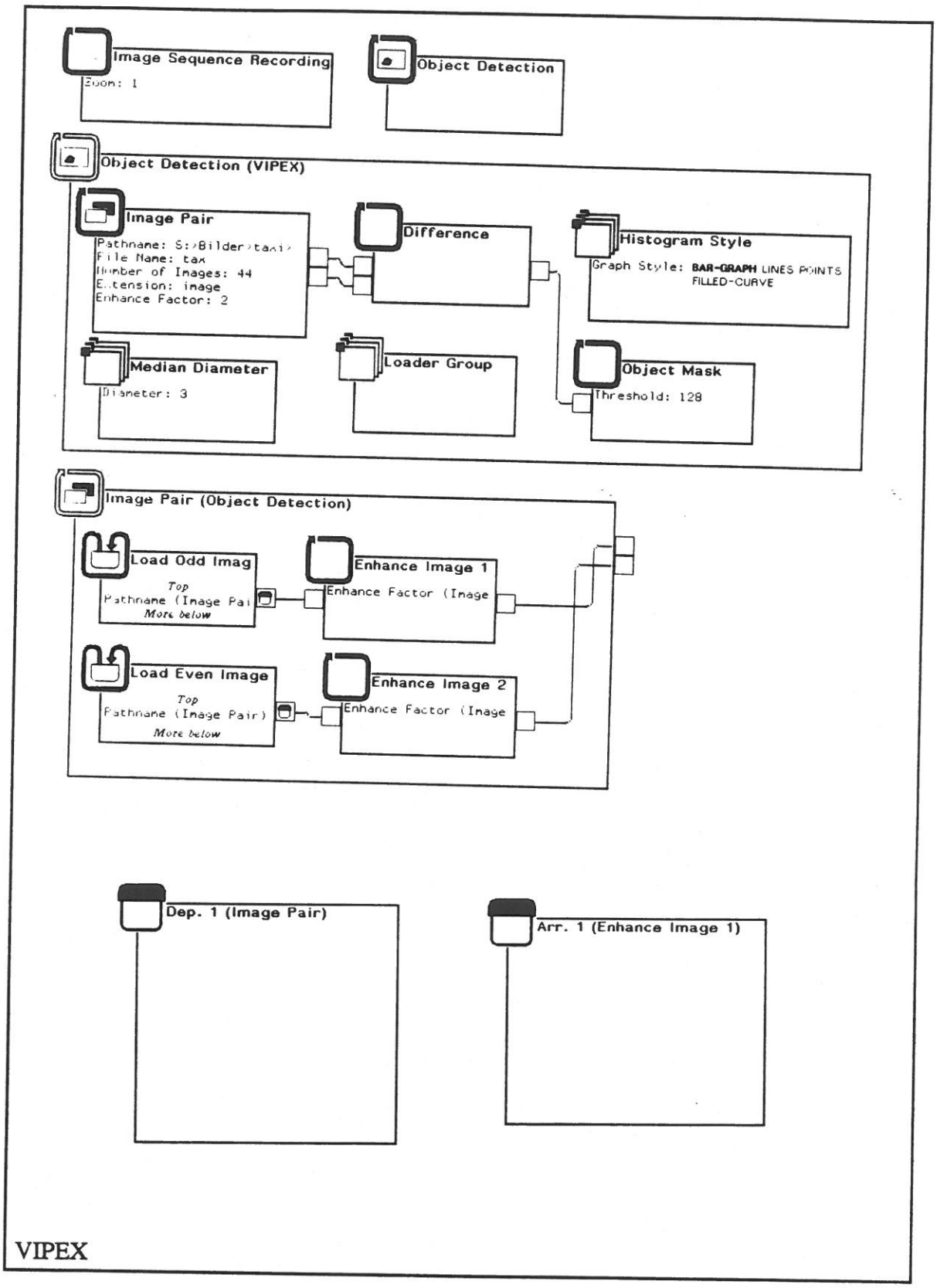


Abbildung 5.5

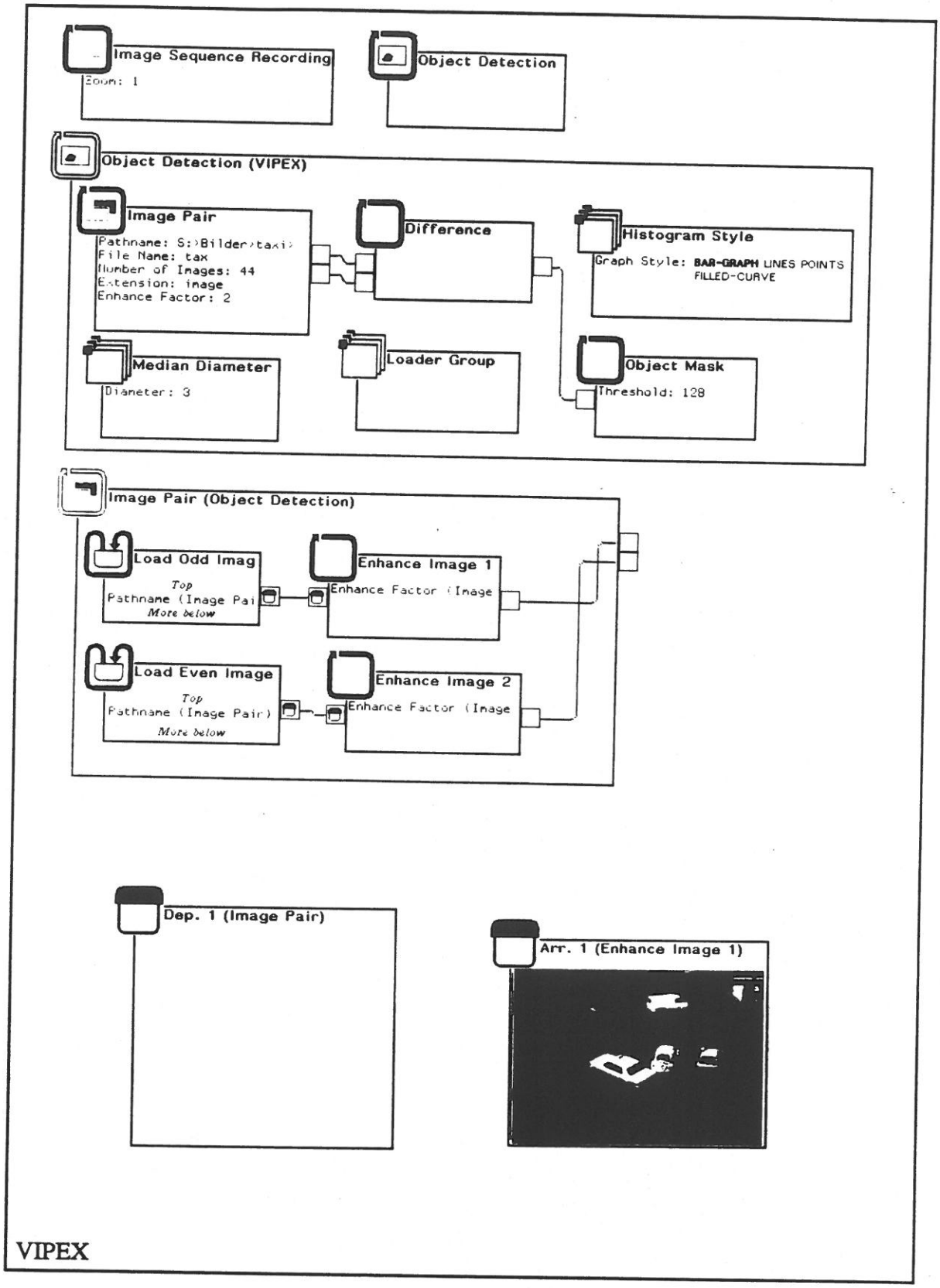


Abbildung 5.6

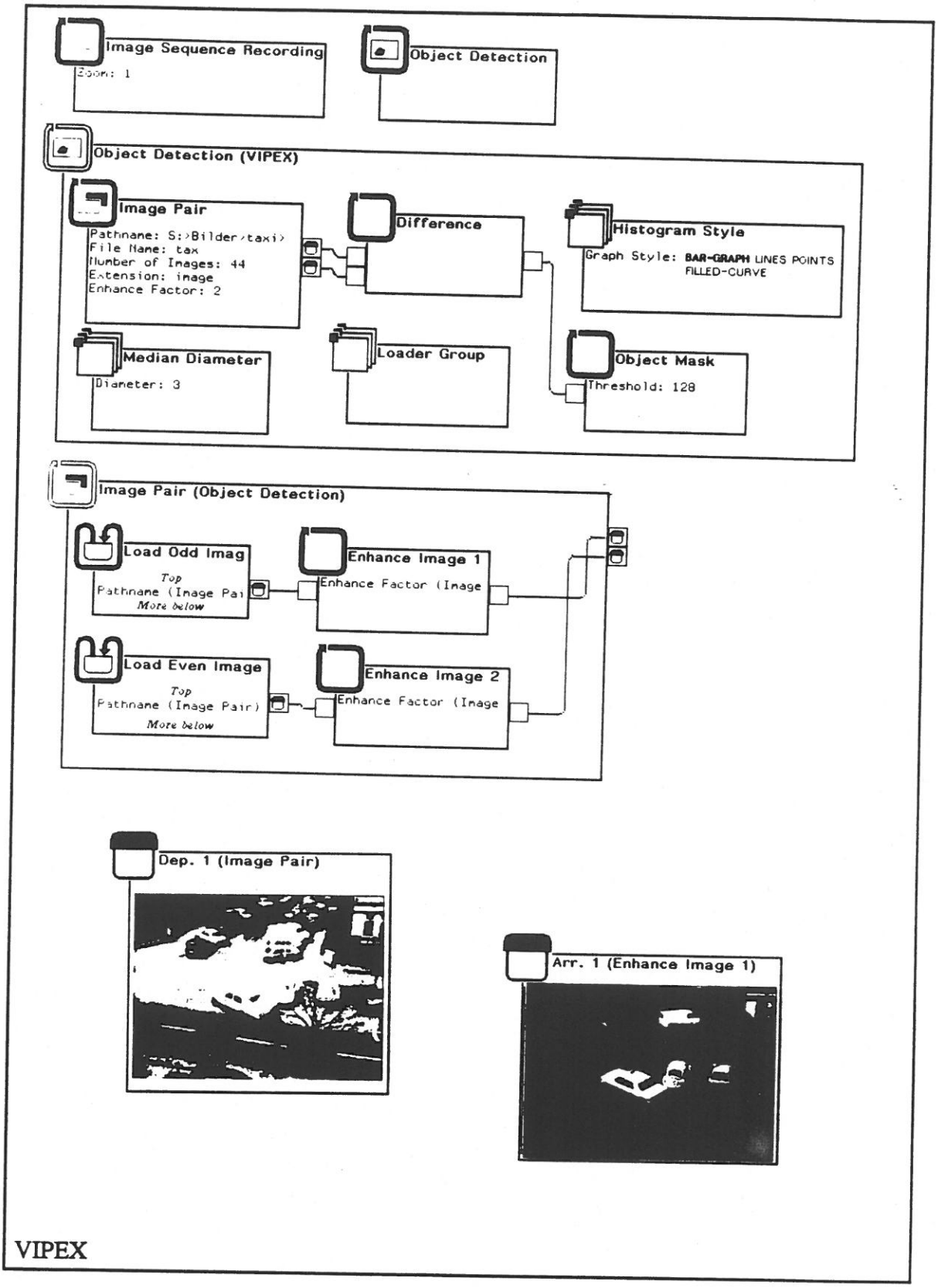


Abbildung 5.7

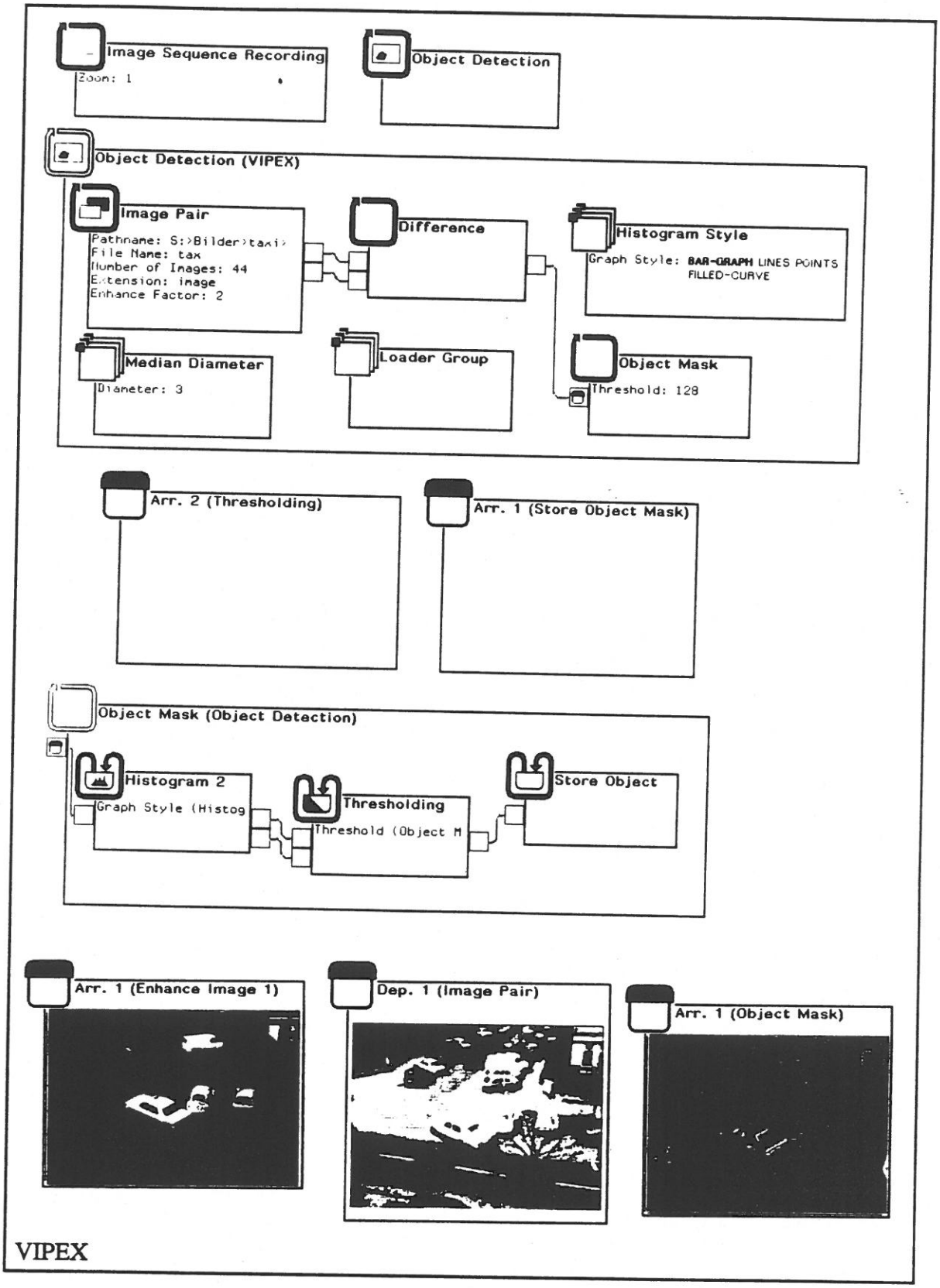


Abbildung 5.8

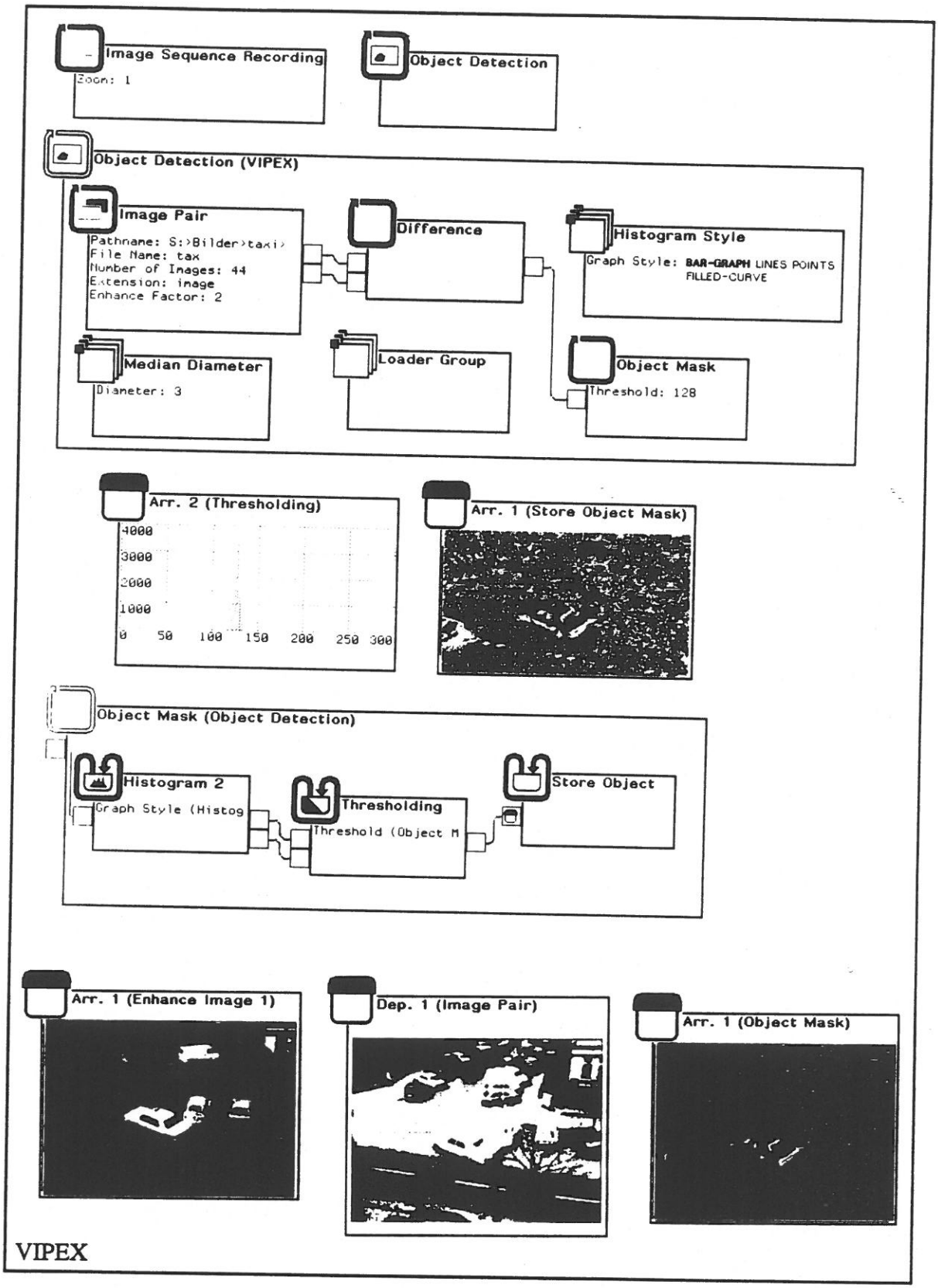


Abbildung 5.9

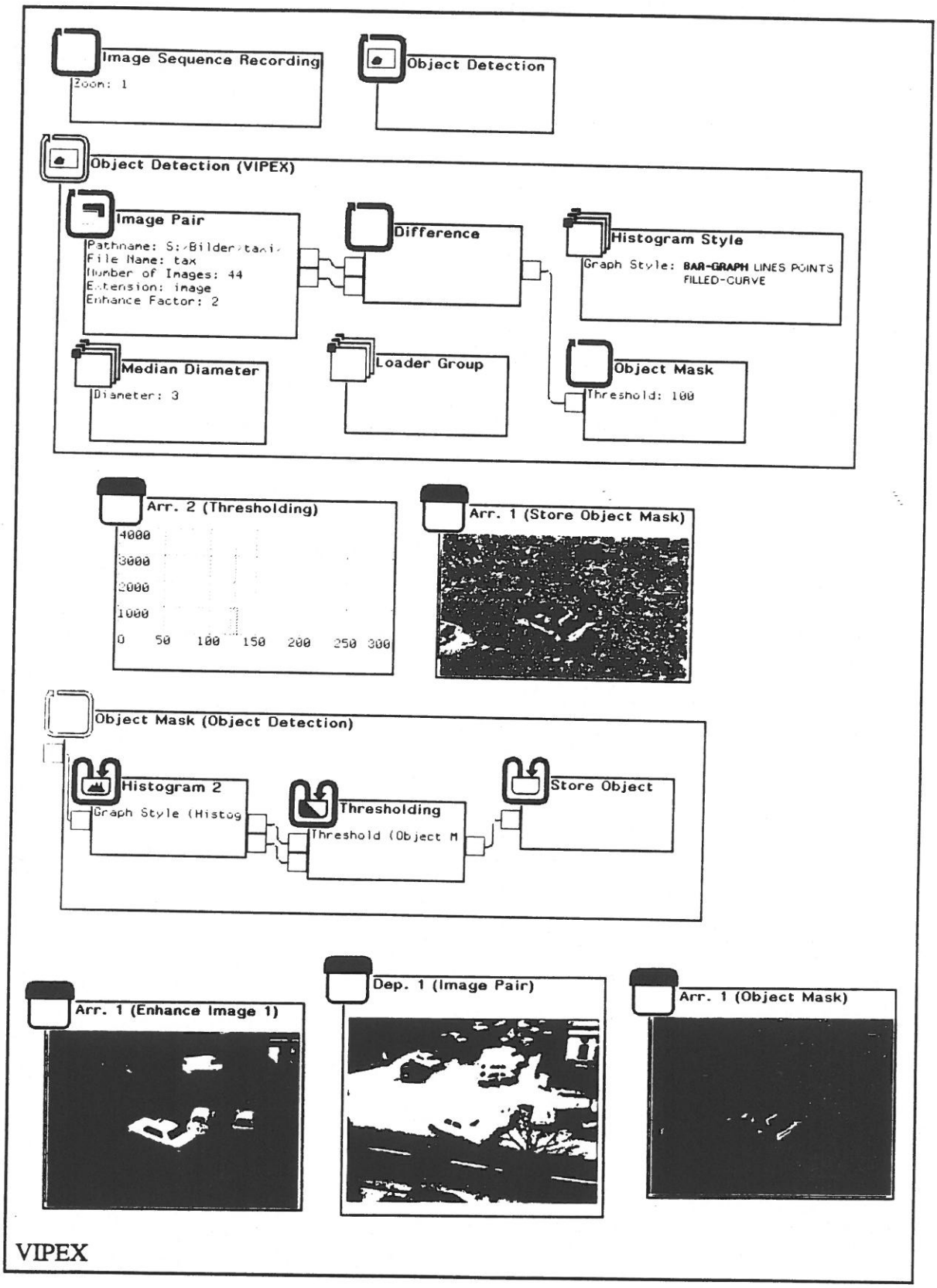


Abbildung 5.10

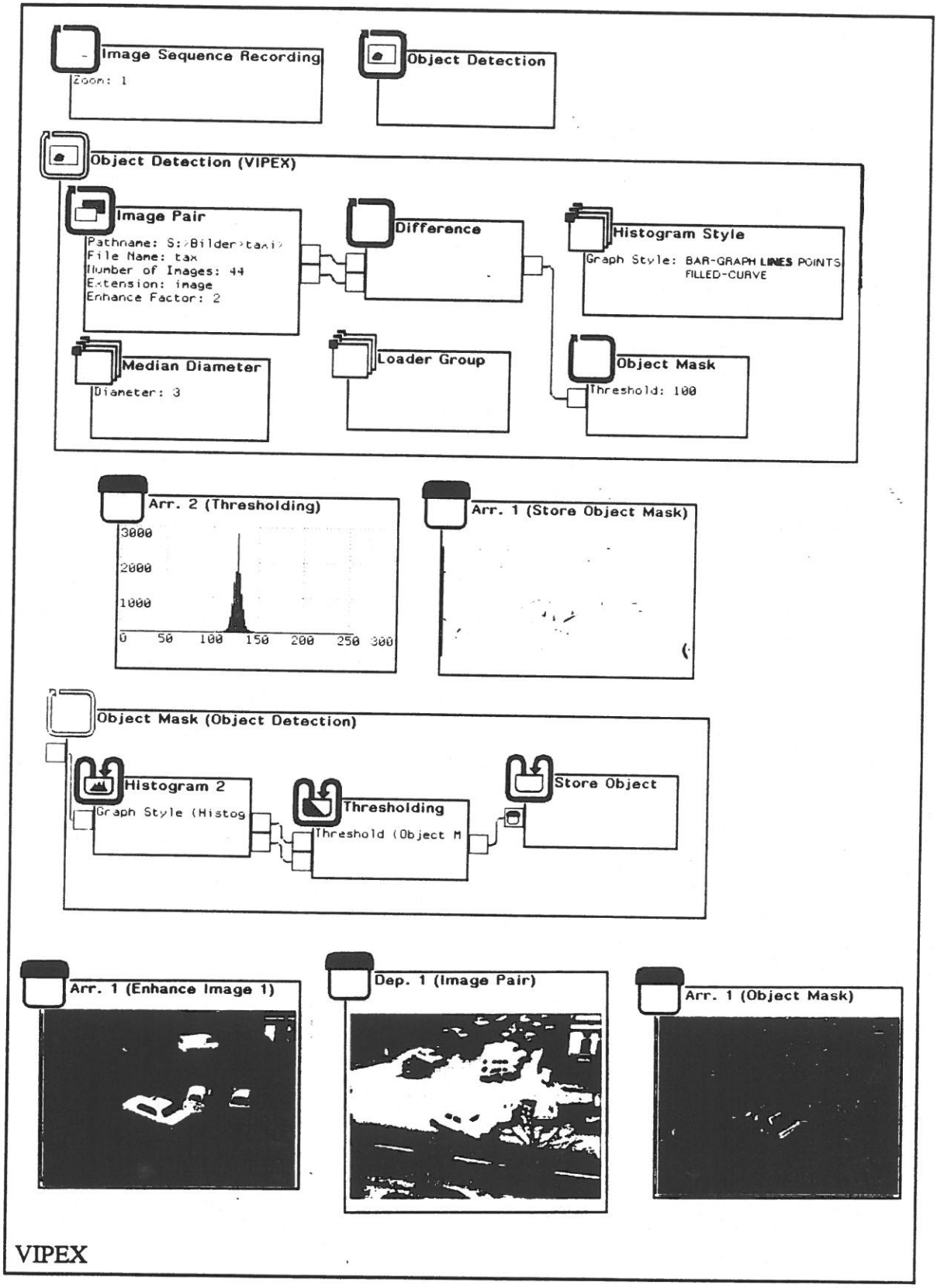


Abbildung 5.11

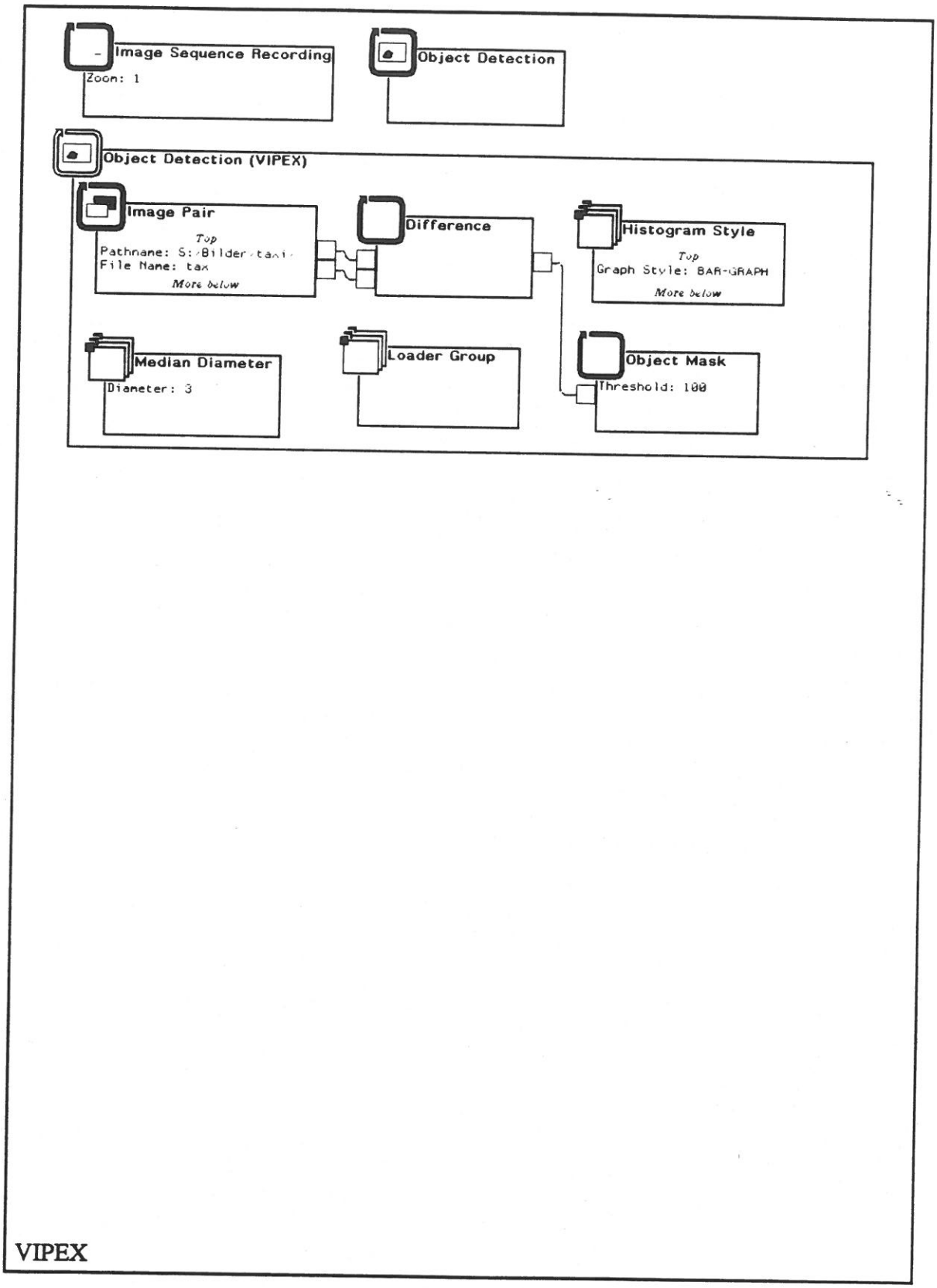


Abbildung 5.12

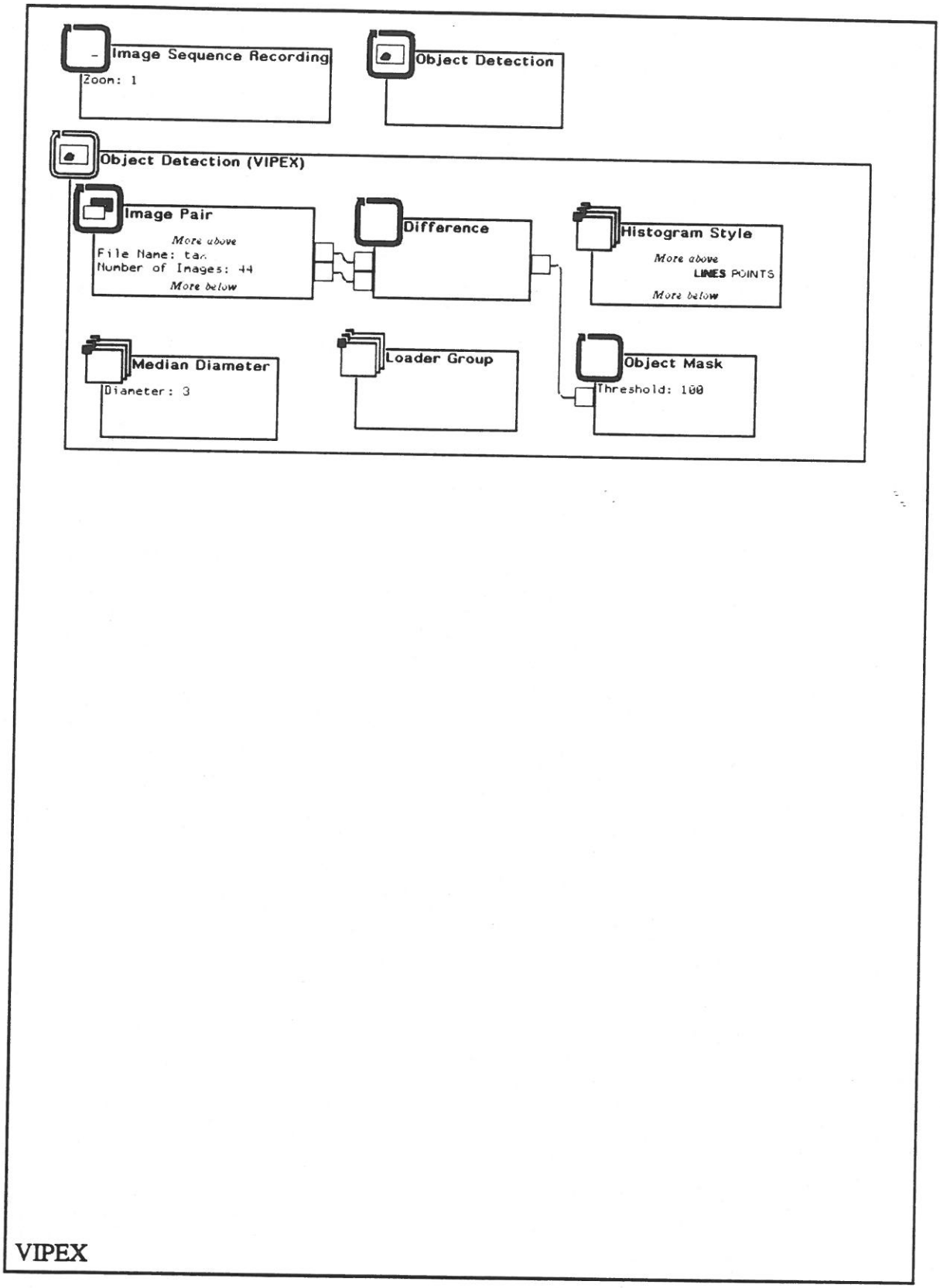


Abbildung 5.13

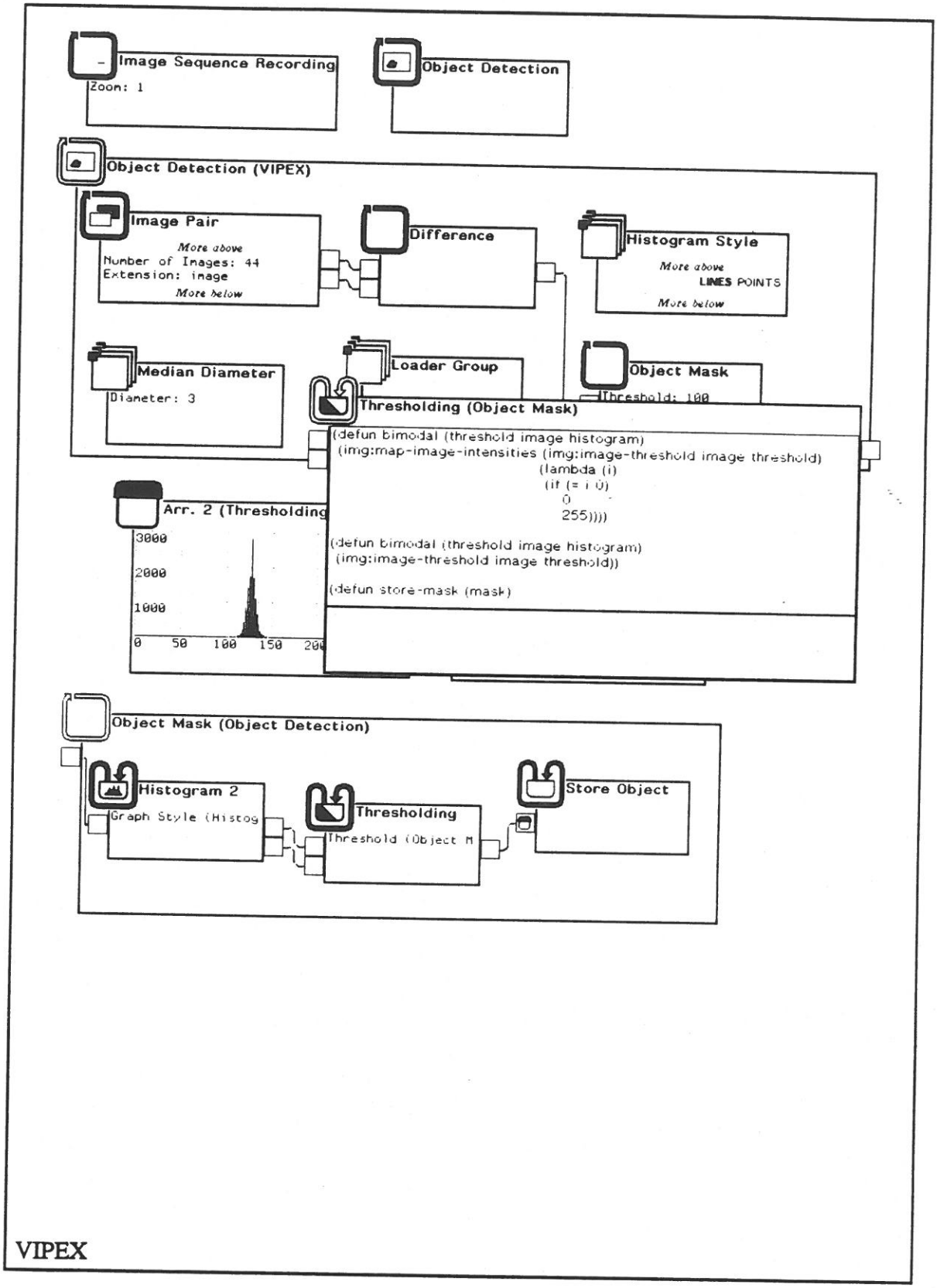


Abbildung 5.14

Kapitel 6

Erweiterungen

In Kapitel 4.1 wurde erläutert, daß zyklenfrei verschaltete Systeme aus Verarbeitungs- und Leitungsobjekten eine Teilklasse der Pr/T-Netze bilden (markierte Kausalnetze). Dieses rührt daher, daß jede Stelle genau eine Eingangs- und Ausgangstransition besitzt, da bisher pro Anschluß nur die Ankopplung höchstens eines Leitungsobjekts erlaubt ist. In den nachfolgenden Abschnitten werden Erweiterungen geschildert, die von dieser Einschränkung abweichen. Die Wirkung dieser Erweiterungen verdeutlichen jeweils äquivalente Netzdarstellungen.

6.1 Aktivierungsbedingungen von Abstraktionsobjekten

Alternativ (oder zusätzlich) zu der im Abschnitt 4.1.1 dargestellten Aktivierungsbedingung für Abstraktionsobjekte (verfeinerte Transitionen), die identisch ist mit der für Verarbeitungsobjekte (Transitionen), könnte man auch ein transparentes Verhalten der Abstraktionsobjekte fordern. Das bedeutet z.B., daß Datenobjekte, die an einem beliebigen Eingangsanschluß eines Abstraktionsobjektes ankommen, sofort nach "innen" weitergeleitet werden. Das vergröberte Netz ist dann jedoch kein Pr/T-Netz mehr.

6.2 Ankopplung von mehreren Leitungsobjekten an einen Anschluß

Läßt man die Möglichkeit der Ankopplung von mehreren Leitungsobjekten an einen Anschluß zu, so könnten Objekte gemäß Abbildung 6.1 verschaltet werden. Die "Bedeutungen" ergeben sich aus den nebenstehenden Netzdarstellungen. Im ersten Fall kann nur ein Leitungsobjekt den (rechten) Anschluß bedienen. Falls im Falle b) und c) beide Leitungsobjekte ein Datenobjekt aus dem (linken) Anschluß abholen wollen, so könnte eines per Zufall ausgewählt werden (siehe Fairneß von Netzen). Nichtdeterministisches Verhalten scheint zumindest in Bildfolgenanalysesystemen wenig sinnvoll zu sein. Eine denkbare "Vervielfältigung" von Datenobjekten im linken Anschluß zur anschließenden Weiterleitung auf allen Leitungen (nicht durch das nebenstehende Netz ausgedrückt) ist speicherintensiv und wird daher nicht berücksichtigt.

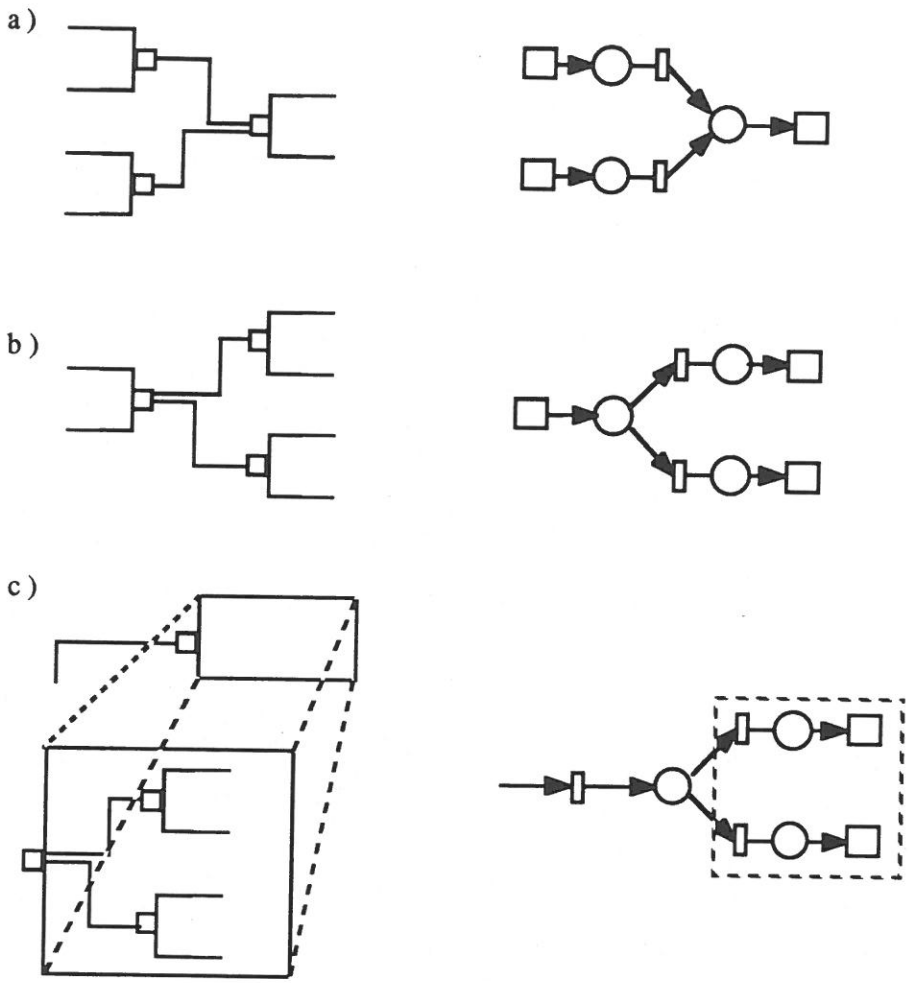


Abbildung 6.1: Ankopplung von zwei Leitungsobjekten an einem
a) Einganganschluß
b) Ausgangsanschluß
c) internen Eingangsanschluß eines
Kompostionsobjekts
jeweils mit äquivalenten Netzdarstellungen

Ist der Fall a) noch unkritisch, so wird im nächsten Abschnitt ein Beispiel aufgezeigt, an dem deutlich wird, daß das Ankoppeln von mehreren Leitungsobjekten an einen Anschluß wie im Falle b) oder c) einer Erweiterung bedarf, um ein verwendbares Konzept darzustellen.

6.3 Transportbedingungen

Versucht man ein Abstraktionsobjekt zu konstruieren, welches Multiplexerverhalten zeigt, so müssen mehrere Leitungsobjekte an einen Anschluß angekoppelt werden können.

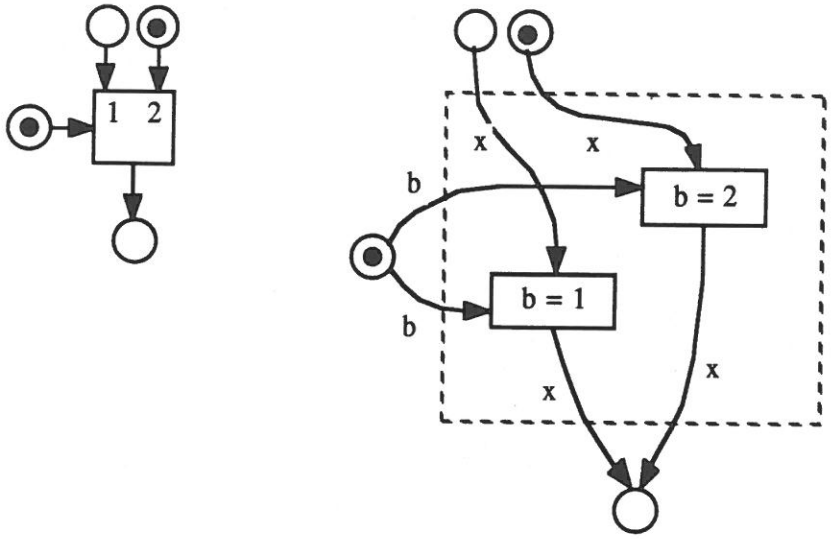


Abbildung 6.2: Multiplexer in Netzdarstellung (nach *Jessen & Valk 87*)

Ein Multiplexer verarbeitet seine Eingangsobjekte gemäß dem (von *Jessen & Valk 87* übernommenen) Netz aus Abbildung 6.2. Je nach Art des links ankommenden Datenobjekts wird Eingang 1 oder 2 "durchgeschaltet".

Der Versuch einer Nachbildung mithilfe eines Abstraktionsobjekts ist in Abbildung 6.3 skizziert. Sei das verwendete Abstraktionsobjekt transparent nach Abschnitt 6.1. Leider ist

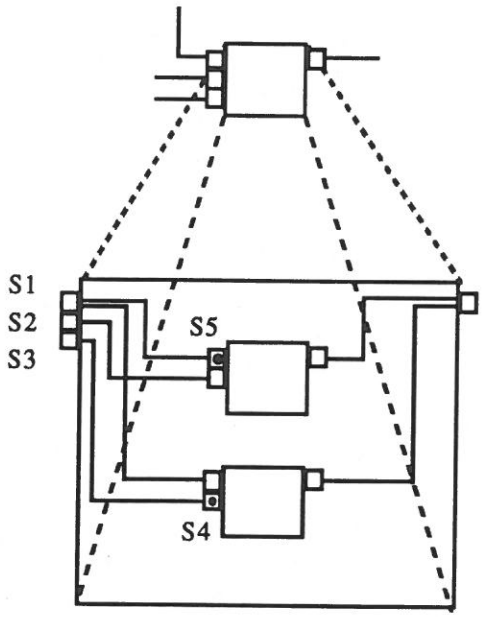


Abbildung 6.3: Versuch der Konstruktion eines Multiplexers

die Konstruktion nicht äquivalent mit dem Vorbild aus Abbildung 6.2. Hier ergibt sich die in Abbildung 6.4 aufgezeigte Netzdarstellung.

Gelangt ein Datenobjekt nach S_3, so kann es direkt über T_2 nach S_4 weitergeleitet werden. Ein nach S_1 kommendes Objekt könnte nun "unglücklicherweise" sofort über T_1 nach S_5 transportiert werden. Man erkennt die in Abbildung 6.4 entstandene Verklemmungssituation, welche jedoch nicht im Multiplexer-Netz aus Abbildung 6.2 auftreten kann. Aufgrund der zwischengeschobenen "schmalen" Transitionen kann zu früh in die "falsche" Richtung geschaltet werden. Andererseits läßt sich das Netz aus Abbildung 6.2 nicht mit bisherigen

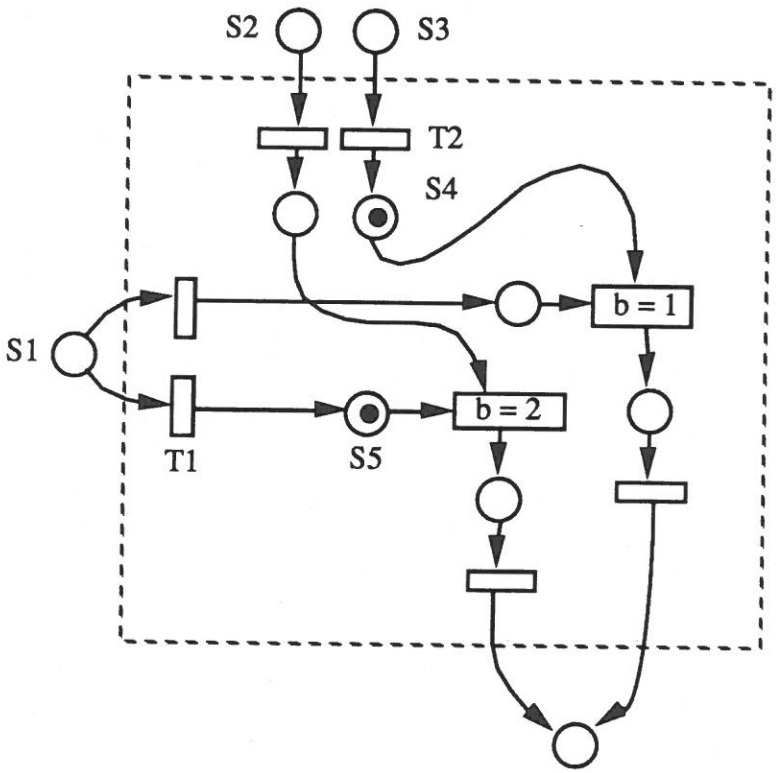


Abbildung 6.4: Netzdarstellung zu Abbildung 6.3

"Bauelementen" nachbilden. Hierzu wäre es nötig, die gemeinsam an einen Anschluß angekoppelten (unidirektionalen) Leitungen je mit einem ***Prädikat*** zu versehen (z.B. Lisp-Funktionen, die entsprechend die Bedingungen b=1 oder b=2 berechnen), welches bestimmt, ob überhaupt ein Transport erfolgen soll. Es stellt sich sofort die Frage nach einer entsprechenden (graphischen) Darstellung des Prädikats.

In der erstellten Implementation sind diese Vorschläge bisher nicht eingeführt worden. Daher haben bisher (zyklenfreie) Verschaltungen von Verarbeitungs- und Leitungsobjekten nur die Mächtigkeit von (markierten) Kausalnetzen, verfügen aber über ein einfach zu durchschauendes Kommunikationsschema.

Die erwähnte Zyklenfreiheit ist keine besondere Situation, da Verschaltungen mit Zyklen (ohne die hier erwähnten Erweiterungen) aufgrund der Aktivierungsbedingungen für Verarbeitungsobjekte nach Abschnitt 4.1.2 nicht "lebendig" sind. Eingangsdatenobjekte eines Verarbeitungsobjekts, die über einen Zyklus von diesem selbst erzeugt oder beeinflußt werden, stehen beim ersten "Durchlauf" noch gar nicht zur Verfügung, so daß die Verarbeitungsabfolge schon hier "abstirbt".

Literatur

Bewley et al. 83: *Human Factors Testing in the Design of Xerox's 8010 "Star" Office Workstation*, W.L. Bewley, T.L. Roberts, D. Schroit, W.L. Verplank, Proceedings Human-Computer Interaction CHI'83, pp. 72–77.

Faasch 87: *Konzeption und Implementation einer objektorientierten Experimentierumgebung für die Bildfolgenauswertung in ADA*, H. Faasch, Dissertation November 1987, Universität Hamburg, Fachbereich Informatik.

Foley et al. 84: *The Human Factors of Computer Interaction Techniques*, J.D. Foley, V.L. Wallace, P. Chan, IEEE Computer Graphics, Vol. 4, No. 11, pp. 13-42.

Haarslev 86: *Interaktion in Systemen zur Bildfolgenauswertung basierend auf einem objektorientierten Ansatz*, V. Haarslev, Dissertation Juli 1986, Universität Hamburg, Fachbereich Informatik, auch erschienen als Bericht Nr. FBI-HH-125/86.

Haarslev 87a: *Eine ergonomische Benutzerschnittstelle für den Anwendungsbereich der Bildfolgenauswertung*, V. Haarslev, Software Ergonomie '87, Berlin, 27.-29. Apr. 1987, Berichte des German Chapter of the ACM, W. Schönpflug, M Wittstock (Hrsg.), Teubner-Verlag, Stuttgart, 1987, pp. 176-186.

Haarslev 87b: *Human Factors in Computer Vision Systems: Design of an Interactive User Interface*, V. Haarslev, In: Second IFIP Conference on Human-Computer Interaction - INTERACT '87 Stuttgart, F.R.Germany, 1-4 September, 1987, Proceedings, H.-J. Bullinger, B. Shackel (eds.), North-Holland, Amsterdam, 1987, pp. 1021-1026.

Hightower 69: *A Solution to Line-Routing Problems on the Continuous Plane*, D.W.Hightower, Proceedings 6th Design Automation Workshop 1969, pp. 1-24.

Hutchins et al. 85: *Direct Manipulation Interfaces*, E.L. Hutchins, J.D. Hollan, D.A. Norman, Human-Computer Interaction, 1985, Vol. 1, pp. 311-338, Lawrence Erlbaum Associates, Inc.

Image Calc 84: *Image Calc Programming Guide*, SRI International, 1984.

Jessen & Valk 87: *Rechensysteme*, E. Jessen, R. Valk, Springer Verlag, 1987.

Preece et al. 87: *Towards a Structured Approach to Specifying User Interface Design*, J. Preece, M. Woodman, D.C. Ince, R. Griffiths, G. Davies, Human-Computer Interaction - INTERACT'87, H.–J. Bullinger, B. Shackel (Editors), Elsevier Science Publishers B. V. (North Holland), pp. 415–421.

Reisig 82: *Petrinetze*, W. Reisig, Springer Verlag, Berlin, 1982.

Smith et al. 82: *Designing the Star User Interface*, D.C. Smith, C. Irby, R. Kimball, B. Verplank, Byte, Vol. 7, No. 4, April 1982, pp. 242–282.

Steele 84: *Common Lisp - The Language*, G.L. Steele jr., Digital Press, 1984.

Symbolics 85: Handbücher zur Symbolics Programmierumgebung, Symbolics Inc., 1985.

Anhang A

Linienführungs-Algorithmus

Als "Piktogramm" für ein Leitungsobjekt dient ein Linienzug, der verkoppelte Anschlüsse verbindet. Offensichtlich kommt es in dieser Anwendung (VIPEX) nicht auf möglichst kurze Wege, sondern auf ein ansprechendes Erscheinungsbild an. Für eine interaktive Benutzerschnittstelle sollte die Wegeermittlung außerdem recht schnell erfolgen.

Die Aufgabe eines Linienführungsalgorithmus' besteht also in der Berechnung einer Verbindung zwischen zwei Punkten unter Berücksichtigung einer Menge von nicht zu überschreitenden Liniensegmenten (Grenzsegmente), die die Umgebung und die gezeigten Darstellungsobjekte abgrenzen. Der von *Hightower 69* übernommene Algorithmus arbeitet nach folgendem intuitiv erscheinenden Grundschema:

- Von beiden zu verbindenden Punkten ausgehend werden unabhängig voneinander Pfade expandiert, die versuchen einander zu treffen.

- Die Pfade werden expandiert, indem abwechselnd versucht wird, "im Weg liegende" Grenzsegmente zu umgehen.

- Ping-Pong-Effekte müssen erkannt werden.

- Ist eine Umgehung nicht direkt möglich, wird versucht, den Pfad zunächst in Richtung eines Grenzsegmentes auszudehnen, um von dort aus zu versuchen, ein (anderes) Grenzsegment zu umgehen (einstufiges Backtracking).

- Kreuzen sich zwei Pfade, so wird der Verbindungslinienzug berechnet. Die Segmente der Verbindung werden so verschoben, daß sie in der Mitte zwischen ihren benachbarten Grenzsegmenten zu liegen kommen.

Durch dieses "Suchverfahren" (der Suchbaum ist radikal beschnitten) wurden die Linienzüge aus Kapitel 5 berechnet. Die Berechnungsdauer ist dabei gemäß den Anforderungen einer interaktiven Umgebung sehr kurz.

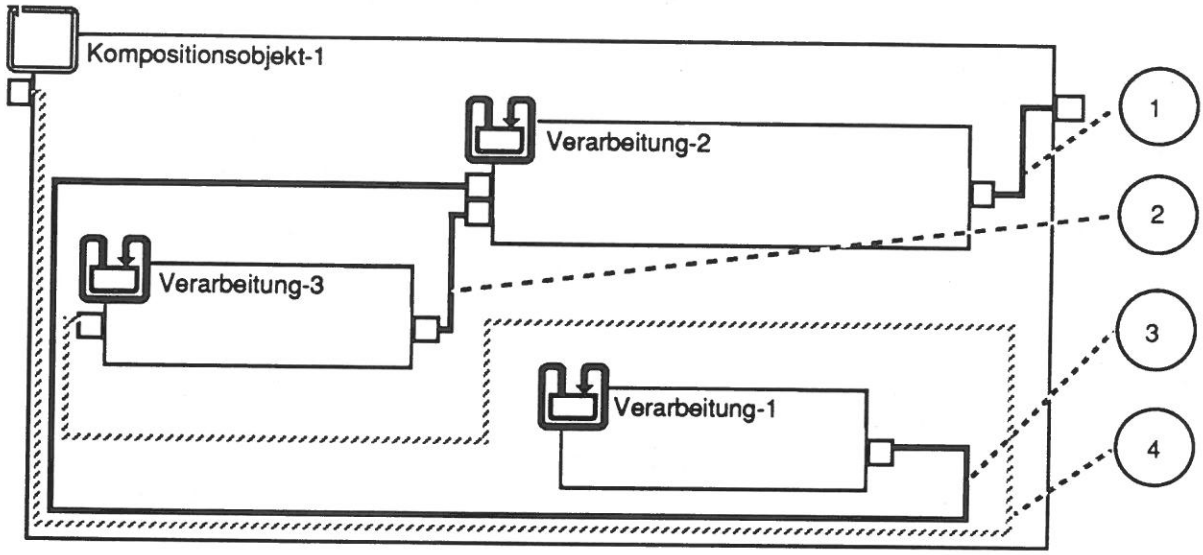


Abbildung A.1: verschlungene Leitungen

Innerhalb von "VIPEX" hat es sich gezeigt, daß die zunächst sinnvoll erscheinende Forderung nach Vermeidung von Kreuzungen berechneter Linienzüge zu kuriosen Verbindungen führt (Abbildung A.1). Für eine menschlichen Betrachter ist es wesentlich schwieriger, ineinander verschlungene Linienzüge zu verfolgen als solche, die sich schneiden. Die schraffierte Verbindungslinie sollte in jedem Fall durch eine kürzere Leitung, die Linie 3 schneidet, ersetzt werden. Die Problematik der Reihenfolge von Wegeermittlungen (in der Zeichnung A.1 durch Nummern angedeutet) zeigt sich in einigen Fällen besonders kraß. Leitungsobjekte können jedoch in beliebiger Reihenfolge von einem Benutzer hintereinander erzeugt werden. Eine ständige "Umverlegung" schon vorhandener Leitungen erscheint ebenfalls nicht ratsam.

Anhang B

Erzeugung von Interaktionsobjekten

Die Darstellungsweise von Datenobjekten in Anschlußdarstellungen hängt von den Typen der betreffenden Anschlüsse ab. Die Typen der Anschlüsse werden bei der Erzeugung der entsprechenden Objekte festgelegt (Angabe einer Liste von Symbolen, s.u.). Die Ausgabefunktion wird unter der "Property" :`print` eines Typsymbols eingetragen. Ausgabefunktionen haben zwei Parameter: ein Fensterobjekt sowie das (in diesem Fenster) darzustellende Datenobjekt.

```
(defun draw-histogram (window histogram)
  (send window :draw-graph histogram
               :graph-style :bar-graph)

(setf (get 'histogram :print) #'draw-histogram)
```

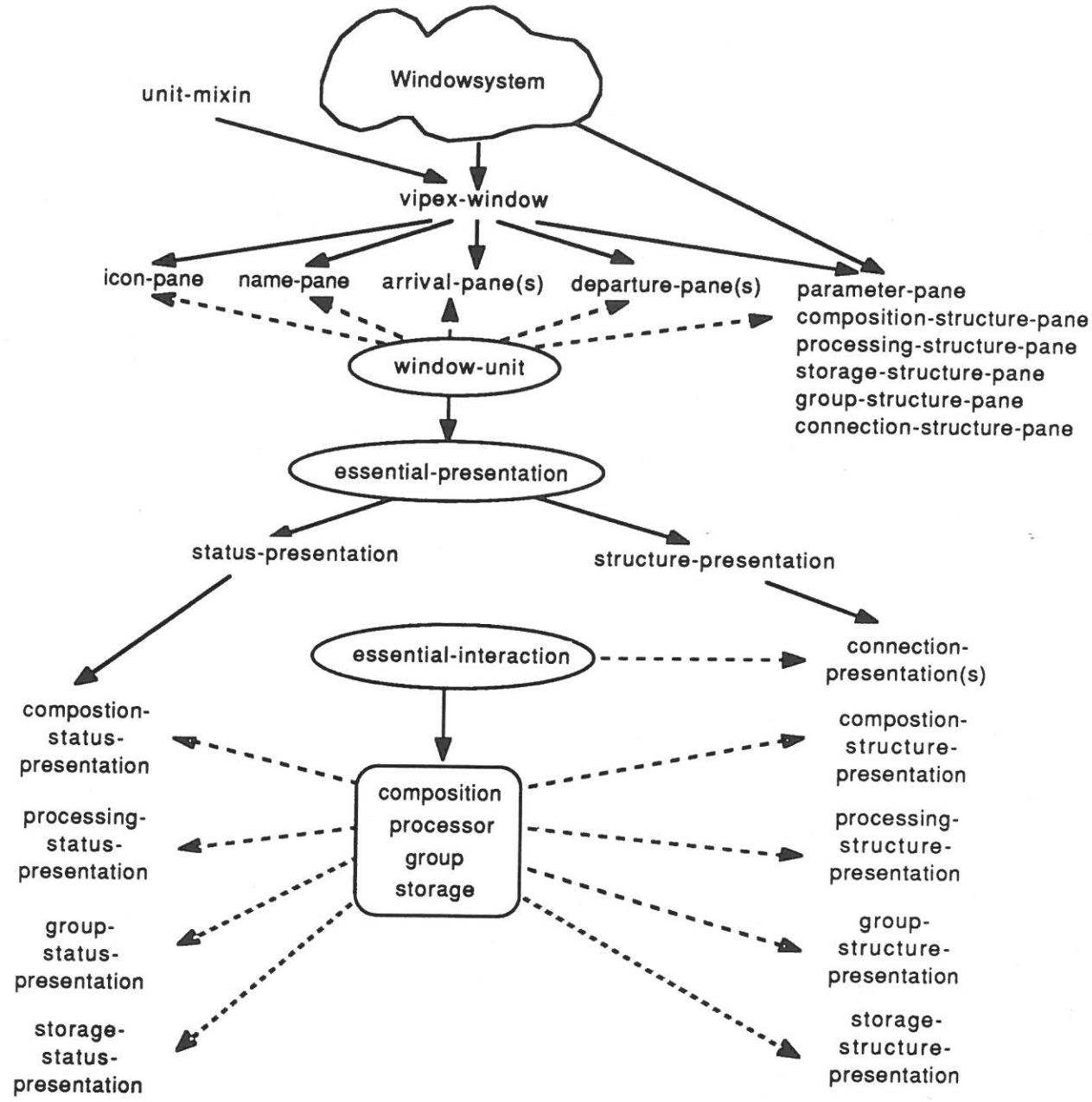
Für die Nachricht :`draw-graph` wird durch das Bildverarbeitungsprogrammsystem ImageCalc™ [*Image Calc 84*] eine geeignete Methode implementiert. Wird einem Typsymbol keine Ausgabefunktion zugeordnet, so können Datenobjekte dieses Typs nicht dargestellt werden (die Anschlußdarstellung bleibt leer).

Interaktionsobjekte sind mithilfe einer objektorientierten Erweiterung von "CommonLisp" auf der Symbolics [*Symbolics 85*], dem "Flavor-System", realisiert. Eine grobe Übersichtseinteilung der erstellten Objektklassen findet sich in Abbildung B.1. Das Klassensystem teilt sich in drei Hauptkomponenten auf.

Die Klasse "window-unit" stellt die Schnittstelle zum vorhandenen ebenfalls objektorientierten Fenstersystem dar. Es wird ein Verbund von (Teil-)Fenstern erzeugt (siehe Abbildung 3.2 ff.). Je nach Verwendung kann ein Verband verschiedene Darstellungen annehmen: "parameter-pane", "compostion-structure-pane" usw.

In der nächsten Schicht geht es um die Darstellung von VIPEX-spezifischen Attributen (Container, Verarbeitungsmaß, Piktogramme, etc.). Die Verwaltung der Prozesse, der Kommunikationspuffer, Mausklicks u.a.m. wird durch die Interaktionsklasse "essential-interaction" geleistet.

Zur Erzeugung von Interaktionsobjekten dienen die Klassen "composition", "processor", "group" und "storage", deren Instanzen jeweils entsprechende Status- und Strukturdarstellungsobjekte enthalten. Zur Interaktionsobjekterzeugung wurde eine Reihe



Die Pfeile sind zu lesen als:

Vererbung (Unterklasse)

enthält ein (oder mehrere) Objekt(e)
dieser Klasse als Wert von Instanzvariable(n)

Abbildung B.1: Objektklassenstruktur (siehe Text)

von Makros bereitgestellt (s.u.). Doch zunächst noch eine Erläuterung zu den erwähnten Parametern:

Parameterobjekte können durch folgende Spezialform erzeugt werden:

```
defparam name print-name initial-value                    [Macro]
         type &rest type-args
```

Durch `defparam` wird ein Parameterobjekt *name* erzeugt. Der Parameter wird innerhalb von Objekten, in denen er auftritt, durch die Zeichenkette (String) *print-name* dargestellt. Durch *initial-value* wird der Anfangswert bestimmt. Der Wertebereich richtet sich nach den nachfolgenden Argumenten, die denen der Variableneingabe-Menüs der Symbolics-Programmierumgebung (siehe *Symbolics 85, Vol. 7, Kap. 22.2*) entsprechen.

```
(defparam diameter "Diameter" 3 :number)
(defparam style
   "Graph Style" :bar-graph
   :choose '(:bar-graph :lines :points :filled-curve))
(defparam pathname "Pathname" "S:>Bilder>taxi>" :pathname)
```

```
make-param print-name initial-value                       [Macro]
           type &rest type-args
```

Wie `defparam`, das Parameterobjekt wird als Wert geliefert.

Interaktionsobjekte werden durch folgende Spezialformen erzeugt:

```
defcomposition name                                       [Macro]
               &key :name            (symbol-name name)
               :status-icon          #\2
               :structure-icon       #\3
               :superior             *vipex*
               :operating-status'    *cycle*
               :arrival-types        '()
               :departure-types      '()
               :free-parameters      '()
               :bound-parameters     '()
```

Durch `defcomposition` wird ein neues Kompositionsobjekt *name* erzeugt. Die `&key`-Parameter sind optional - die Standardwerte sind hinter den Schlüsselworten angegeben. Die Zeichenkette `:name` gibt den das Objekt kennzeichnenden Namen an. Piktogramme für Status- und Strukturdarstellung können als Zeichen (des Fonts `SYS:Fonts;vipex-icons`) angegeben werden. Das Schlüsselwort `:superior` gibt an, in welchem Kompositionsobjekt sich das Objekt befinden soll (`*vipex*` steht für das Wurzelobjekt der Kompositiosob-

jekthierarchie). Die Verarbeitungsart der zu erzeugenden Objekte wird durch Angabe eines der Symbole `*step*`, `*stop*` oder `*cycle*` (mit entsprechender Bedeutung) festgelegt. Die Typen der Ein- und Ausgänge (`:arrival-types, :departure-types`) sind durch Angabe einer Liste von Typsymbolen, denen möglichst vorher unter der "Property" `:print` eine Darstellungsfunktion zugeordnet werden sollte (s.o.). Die freien und gebundenen Parameter (`:free-parameters` bzw. `:bound-parameters`) sind durch Angabe einer Liste von mit `defparam` oder `make-param` erzeugten Parameterobjekten festzulegen.

```
(defcomposition image-pair
        &key   :name "Image Pair"
               :status-icon #\c
               :structure-icon #\d
               :superior experiment-2
               :departure-types '(geo-image geo-image)
               :free-parameters (list pathname filename
                                      n-images extension
                                      enhance-factor))
```

```
defprocessor name                                          [Macro]
      &key  :name                (symbol-name name)
            :status-icon         #\0
            :structure-icon      #\1
            :superior            *vipex*
            :operating-status    '*cycle*
            :arrival-types       '()
            :departure-types     '()
            :free-parameters     '()
            :bound-parameters    '()
            :advance-indication  nil
```

Ähnlich wie Kompositionsobjekte können Verarbeitungsobjekte durch `defprocessor` erzeugt werden. Die zugehörigen Funktionen werden mit `defun` definiert. Dabei gilt folgende Konvention:

```
(defun name (free-param_1 ... free-param_n
             bound-param_1 ... bound-param_m
             input_1 ... input_p)
        ...
        (values  output_1  ...  output_q))
```

In der Parameterliste treten also zunächst die freien und dann die gebunden Parameter auf. Nachfolgend werden die Eingangsdatenobjekte aufgeführt. Die Ausgangsdatenobjekte werden durch das "Multiple-Value"-Konstrukt (CommonLisp : [*Steele 84*]) geliefert. Der Name *name* der mit `defun` definierten Funktion muß identisch mit dem zugehörigen durch

`defprocessor` definierten Verarbeitungsobjekts sein. Durch eine Zusatzfunktion könnte eine Plausibilitätsprüfung der Parameter erfolgen.

Der zusätzliche Schlüsselwortparameter `:advance-indication` in `defprocessor` gibt an, ob durch das definierte Verarbeitungsobjekt eine Anzeige des Verarbeitungsmaßes nach Kapitel 3.6 unterstützt wird. Eine verarbeitende Funktion hat dann folgende Form:

```
(defun name (free-param_1 ... free-param_n
             bound-param_1 ... bound-param_m
             input_1 ... input_p
             advance-function)
    (do ((n 0.0))
        (nil)
      ...
        (funcall advance-function n)
        ;; n gibt prozentual ein Maß für den
        ;; Verarbeitungsfortschritt an
        (return (values output_1 ... output_q))))
```

Die Funktion *advance-function* übernimmt einen Parameter (float).

```
defgroup  name                                              [Macro]
    &key  :name                    (symbol-name name)
          :status-icon             #\6
          :structure-icon          #\7
          :superior                *vipex*
          :operating-status        '*cycle*
          :free-parameters         '()
          :bound-parameters        '()
          :grouped-objects         '()
```

Gruppenobjekte werden durch `defgroup` erzeugt. Als `:grouped-objects` wird eine Liste von Symbolen erwartet, deren Wert jeweils durch eine der obigen Objektdefinitionen erzeugt werden muß.

```
(defgroup histogram-style
          :name "Histogram Style"
          :free-parameters (list style)
          :grouped-objects '(histogram))
```

```
defpipe (sender-object   [:internal] connection)      [Macro]
        (receiver-object [:internal] connection)
```

Erzeugung eines Leitungsobjekts durch Angabe zweier durch `def...` erzeugter Objekte (*sender-object*, *receiver-object*) und jeweils einer Anschlußnummer

`connection` (von oben mit 1 beginnend). Der Schlüssel `:internal` kennzeichnet ggf. einen "inneren" Anschluß eines Kompositionsobjektes (s. Strukturdarstellung von Kompositionsobjekten).

Im folgenden werden die zur Erzeugung der Beispielkonfiguration aus Kapitel 5 verwendeten Definitionen aufgeführt. Nach ihrer jeweiligen Erzeugung können die Objekte mit der Maus in den entsprechenden Kompositionsobjekten plaziert werden. Für einige Verarbeitungsfunktionen wurde die Definition angegeben. Die Beispiele zeigen, wie Funktionen des Bildverarbeitungssystems "ImageCalc" (`img: ...`) von der Benutzerschnittstelle verwaltet werden können.