

Iconic–Symbolic Interfaces

Dietrich Paulus, Heinrich Niemann

A previous version of the following paper has appeared in
SPIE's Conference on Electronic Imaging
(Conf. 1659), San Jose, CA, Feb. 9–14, 1992
please cite as follows

Changes in regard to the published paper are:

- New versions of class hierarchies where appropriate
- Changed references from PhD–Thesis²¹ to Book²²

Iconic–Symbolic Interfaces

Dietrich Paulus, Heinrich Niemann
Lehrstuhl für Informatik 5, (Mustererkennung)
Universität Erlangen–Nürnberg, Germany

ABSTRACT

A uniform interface for the data exchange between image segmentation and high–level image analysis will be presented, which I will call an ‘iconic–symbolic interface’. The interface will be specified as a class in an object–oriented programming environment.

The term ‘iconic processing’ will be contrasted to ‘iconic data structures’. Symbolic processing will be separated from iconic processing by the use of explicitly represented knowledge about the task domain. Many segmentation algorithms may be performed independent of the task domain. I will point out that the same holds for the recovery of depth and surface information by shape from shading or stereo and for the detection of motion.

Several data structures for the representation of the results of segmentation will be compared. The new class ‘segmentation object’ (i.e., the data structure and the required operations on it) will be defined as a superset of the other proposed data structures. It allows for a uniform representation for 2–D and 3–D image segmentation and for motion detection. The interface to symbolic processing will be defined by a machine–independent external representation of the segmentation object. Compactness will be obtained by binary storage.

International standardization of low–level image preprocessing and of an image interchange format is in progress. I will indicate how a future standard can cooperate with the external representation of segmentation objects.

1 INTRODUCTION

Image processing software is presently used in many areas of application. The complexity of the systems and of the results depends highly on the particular application. A simple example is the automatic recognition of postal addresses of letters. More complex classification problems arise in the inspection of X–ray images. Three–dimensional (3–D) reconstruction is required for Computer Tomograph (CT) and Magnetic Resonance (MR) images or in robotics. Other examples can be found in medical, industrial, military or geographical applications.

Whereas simple *classification* problems can be solved with statistical methods, the automatic analysis and understanding of complex patterns generally requires knowledge about the particular task domain. The parameters for statistical classification are usually set automatically in a training stage using a representative sample. In general however, the problem of automatic acquisition of knowledge for pattern analysis using a sample of the expected scenes is still unsolved. Moreover, knowledge–based image analysis systems are mostly designed exclusively for a small number of possible applications. But even the knowledge–based image analysis systems usually have some components which operate on images independently from the specific task. These components are commonly referred to as the image preprocessing stage. Typical operations are a variety of filters, geometric transformations and corrections, extraction of features, like e.g. grey–level histograms.

Preprocessing is not the only part of an image analysis system which can be designed without knowledge about the task domain. The analysis of complex scenes usually requires that the image is split into simpler components (e.g. points, lines, vertices, regions). This process is called the *segmentation* of an image. Segmentation may in some cases be directed by the knowledge–base, but may in many cases as well be initially performed independently of the specific task.

Image preprocessing and image segmentation are part of most of the image analysis systems. Many popular algorithms for preprocessing and segmentation exist which may be specified abstractly without referring to details of a possible implementation. In the following, we will inspect the concrete data structures used in the implementations of these processes.

2 PREPROCESSING

Digital image preprocessing mainly deals with pictorial data structures, which are usually represented in matrices. Typical operations transform images into images, e.g., by filtering. The concept of an image is however more complex

than a simple matrix. Images may consist of a single matrix representing intensity information in a spatial domain. They may as well result from recordings of several channels, i.e., an image may have parts being images. Time series result in sequences of images. Volumetric images may be seen as a fixed length sequence of two-dimensional image slices or as a 3-D image cube. All of the above may be combined, e.g., resulting in a time sequence of volumetric multi-channel images. Images may furthermore be transformed from the spatial domain into some feature domain, e.g., by a discrete Fourier Transformation (DFT). Feature-images are regarded as images as well. Furthermore, some images make sense only with additional information e.g., a list of the camera parameters or parameters of the recording device. The image data structure has to provide slots where this — mostly symbolic — information can be filled in.

Any of these concepts for images has its own set of operations. There have to be operations to access individual elements (pixels) of images. Other operations extract areas of interest or subpictures. Further operations allow the access to the above mentioned extra information attached to the image data structure. The sizes of the matrices or the length of the image series are fixed parameters of the image.

Data type	Application in image processing
Enumeration	Colour channels: 'red', 'green', 'blue'
Linked list	Polygon
Tree	Quad tree
Graph	Region Adjacency Graph
Vector	Histogram
Set	Results of segmentation (e.g. Set of Lines)
Matrix	Image

Tab. 1: Examples for basic data types in image processing

In addition to image data structures, various general data types are required for image preprocessing, as can be seen in Tab. 1; further examples are given in e.g. the book of Niemann²⁰ and the article of Shapiro³⁰. A closer look will reveal that the data structures in the left row represent internal programming details and not concepts peculiar to image processing. The applications on the right use the structures on the left as possible representations. All modern programming languages provide mechanisms for the representation of these data types.

3 SEGMENTATION

Image segmentation is a process that transforms the image into a set of simpler constituents. The types of the objects detected depend highly on the algorithm used. The two major classes of algorithms are the region-based segmentation, which searches for objects fulfilling a certain homogeneity criterion, and the line-based systems, which detect discontinuities in the image function. The objects detected in the segmentation may be represented in various ways as indicated in Tab. 2. One Object in the scene may be represented by more than one representation at the same time. An example is a line which is detected as a chain code and is approximated by a spline.

Geometric object	Representation
Region	Contour-Line Characteristic Function Quad tree Run length code
Lines	Chain Code Polygon Spline
Point	Coordinate pair

Tab. 2: Representations of results of the segmentation

Region based segmentation as well as line based segmentation starts up with a preprocessed image and usually applies an operator which creates a new image in the feature domain. Edge-operators create gradient or line images.

Some possible paths for the segmentation of gray level images are shown in Fig. 1. Starting with a preprocessed graylevel image (GrayImg, arc 1) either operations based on the first derivative (Sobel, Prewitt, etc., arc 2) or on the second derivative (Laplace, arc 7) may be used. The former will create a gradient image (EdgeImg) consisting of edge strength and edge orientation. Repeated application of the first derivative will also result in an approximation of the second derivative (arcs 5,6). Other edge detection algorithms result in enhanced contours without information about the direction of the edge (arc 3), e.g., by special rank order operations.²⁰ By Hough Transformation (arc 8) an accumulator array will be created and a *set* of lines represented as polygons (arc 11) will be generated. Line thinning (arcs 9,10) and line following (arc 12,13,14) will also result in a *set* of lines, mostly represented as chain codes. We now call these sets a *Segmentation Object* (SegObj).

Region operators first create a region image (LabelImage, arc 4). Well known methods are region growing or 'Split and Merge'.¹⁸ The result will be a set of regions, possibly represented as quad trees (arc 15). This set will also be a segmentation object. The generation of the segmentation object is called the *initial segmentation*.

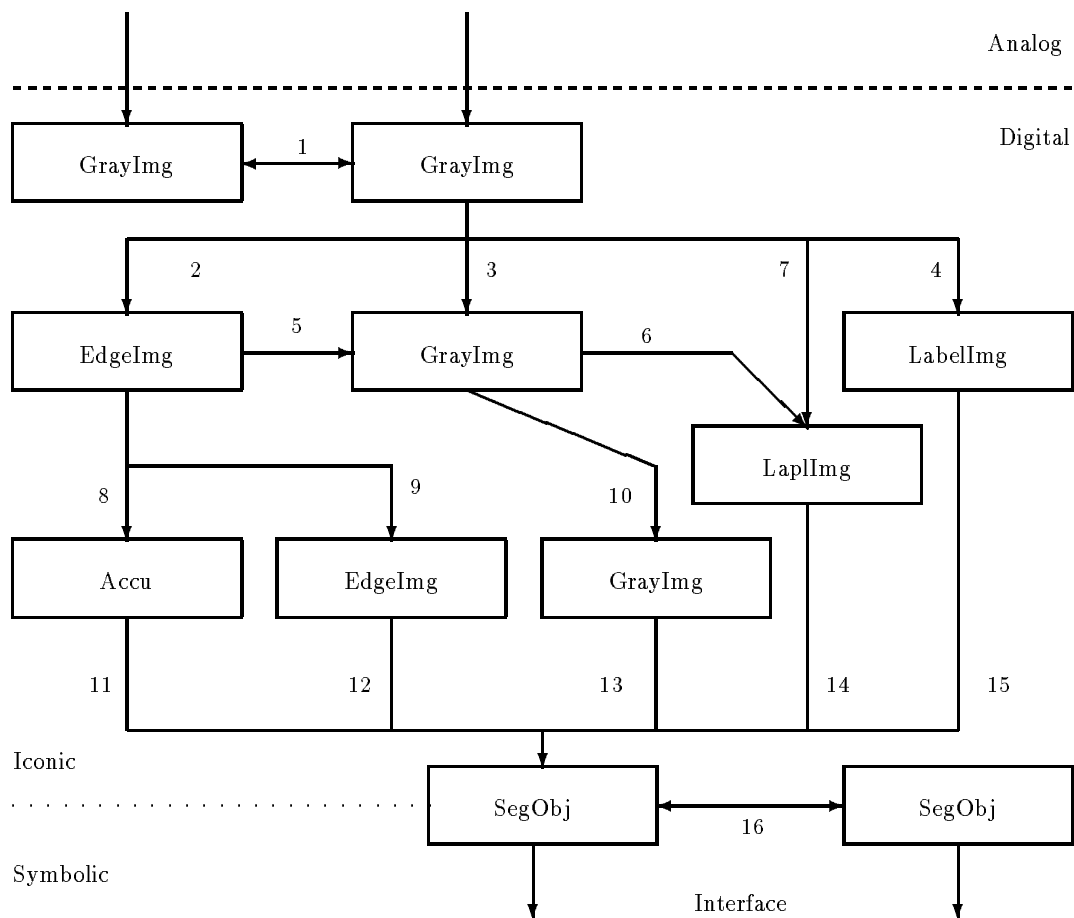


Fig. 1: Some paths from the camera to a segmentation object with line based segmentation of 2-D gray level images

Various structural relations can be detected during the segmentation process in addition to the primitive objects.²⁰ Lines can be grouped by collinearity or parallelism. Arcs or circles may be related, if they are concentric. During region segmentation relations showing the inclusion or neighbourhood are naturally detected. These relations are of course meaningful for the two-dimensional projection of the scene to the image plane. The representational mechanisms may however be used for relations of three-dimensional objects in later stages of the analysis.

Other operations on the primitive objects detected perform further processing of the initial segmentation, e.g., by

smoothing the edges or contours. These changes as well as new relations yield a new segmentation object (arc 16). Steps 1 (preprocessing) and 16 (enhancement of the segmentation result) may be repeated.

4 ICONIC AND SYMBOLIC IMAGE PROCESSING

Pictorial information which may as well be called *iconic*.²⁸ In image segmentation primitive picture elements are extracted from iconic data. Eventually, image analysis will transform the iconic data to a *symbolic* description. During this transformation various types of data structures will be needed as listed in Tab. 1, 2 and 3. An example is the 'iconic-symbolic data structure'³² which explicitly names its purpose. Algorithms manipulating iconic data are now called the *iconic processing* stage. A typical property of algorithms manipulating images is that no special knowledge about the contents of the pictures is needed to perform the algorithm. Iconic algorithms operate on the data independent of the task domain.

Various segmentation algorithms make no assumptions about the scene as well. A 'Split-and-Merge'-algorithm for example will produce regions no matter whether the image material contains X-ray images or satellite images. Transformations of the quad tree generated by such an algorithm into other region representations — e.g., the contour line with appropriate relations — are also performed without reference to the particular analysis problem. All the transitions shown in Fig. 1 may thus be performed data driven.

Any algorithm that operates data driven on pictorial data is called an iconic algorithm. This applies for image preprocessing as well as data driven image segmentation. We can conclude, that *symbolic processing* differs from iconic processing by the use of explicitly represented knowledge about the task domain.

5 ICONIC DETECTION OF MOTION AND DEPTH

The goal of many image analysis systems is the recovery of the third dimension out of a two-dimensional picture. Active or passive methods may be used for that purpose. Two popular approaches belonging to the passive methods are the class of algorithms called 'Shape from ...' which tries to calculate a surface map using some fairly general assumptions about the reflectance of the surfaces in the scene.²⁵ The other is stereo vision where more than one camera is used to obtain different aspects of the scene. When the camera parameters and the orientation of the optical axes are known, the depth of an object can be computed by triangulation using corresponding points in both images. The main problem is finding the corresponding primitive objects. A general stereo algorithm using straight line segments works as follows:²⁷ first the two images are normalized using the parameters of the cameras. Then lines are detected in either image using conventional edge detection operators. Every line in the first image will be compared to all lines in a region of interest in the other image. Lines fulfilling some similarity criterion will be potential correspondences. This process is facilitated by the use of a resolution pyramid. Using relaxation the best matching lines are extracted from the potential correspondences. The result is a set of three-dimensional straight lines. No problem specific knowledge is used in these processing steps.

Other stereo based systems use optical flow or block matching. Neither of these operations uses a knowledge base. Depth recovery may thus be an iconic operation. However, in many cases guidance by a knowledge based system may reduce the search space for the corresponding primitives.

The same holds for the detection of motion, which is also mostly based on optical flow, block matching or the correspondence of some primitive parts. The recovery of surface information for example by 'shape from shading' also requires iconic material only.

We can conclude that depth and surface information as well as motion can in many cases be computed using iconic processing.

6 INTERFACE DATA STRUCTURES

The results of data driven segmentation have to be passed to an interface which provides the transfer of the information to the symbolic processing stage. The iconic-symbolic interface is represented by the dotted line on the bottom of Fig. 1. Data structures for the representation of the results of the segmentation process were presented by

Data structure	Author
Primal Sketches	Marr ¹⁹
2½-D Sketches	Marr ¹⁹
Recursive Structure for Line Drawings	Shapiro ³⁰
Iconic-Symbolic Data Structure	Tanimoto ³²
RSE-Graph	Hanson & Riseman ¹⁶
Line Adjacency Graph	Pavlidis ²⁴
Region Adjacency Graph	Pavlidis ²⁴
Region Graph	Fisher ⁹
Spatial Data Structure	Shapiro und Haralick ³⁰
Object-Oriented Data Base	Goodman et al. ¹³
Segmentation Objects	Niemann ²⁰ , Paulus ²²

Tab. 3: Candidates for an interface to knowledge based processing

various authors as indicated in Tab. 3. In contrast to the left row of Tab. 1 the data structures listed here are concepts typical for image processing. We now call any result of segmentation a segmentation object.

A general data structure for the interface must have the expressional power to represent all kinds of information resulting from any segmentation process. Surfaces, lines, regions, vertices and various structural and temporal relations between any two of these objects may be detected during segmentation and have to be stored in these data structures. Objects of a higher dimension than two must be represented in the interface between iconic and symbolic processing, as indicated in section 5.

Formally, a segmentation object may be expressed as

$$\text{SegObj} : ((A : (A_T, R \cup V_T))^*, (P : \text{SegObj})^*, (S(P) : R)^*, CF : R).$$

A segmentation object consists of:

- a list of attributes or features (A), each represented as a pair of type (A_T) and a value. The value may be a real number (R) or some symbol out of a terminal alphabet (V_T); (e.g., the pair (`colour`, `'red'`));
- a set of parts (P) which are in turn instances of segmentation objects;
- a set of fuzzy relations between these parts ($S(P)$);
- a measure of the certainty for the whole object (a real number CF).

Segmentation objects without further parts are called 'atomic objects'. Lines, vertices, and regions are examples of atomic objects. For a more elaborated definition of the segmentation object see the book of Niemann.²⁰ Attribute-value pairs (A) allow for the representation of features. Lines with individual values for mean contrast may serve as an example. Relations ($S(P)$) are used to represent structural properties like neighbourhood or collinearity. A fuzzy value is used since these relations tend to be uncertain or inaccurate in image analysis due to segmentation errors or noise. Relations may be used for the representation of correspondences in stereo images (section 5). An overall measure of the quality of the segmentation object may be placed in the certainty factor (CF). This simplifies further symbolic processing which often has to rank competing results. Any of the data structures listed in Tab. 3 conforms to a subset of this formal specification.²² No dimension is specified in the definition of segmentation objects. They may therefore be used for the representation of objects with arbitrary dimension.

7 AN OBJECT-ORIENTED INTERFACE

In the previous sections we mentioned a variety of possible concepts for images and related data. These data structures may be combined with their typical operations yielding abstract data types. Using inheritance and classes we arrive at an object oriented description of the problem. Objects being instances of classes are accessed by messages

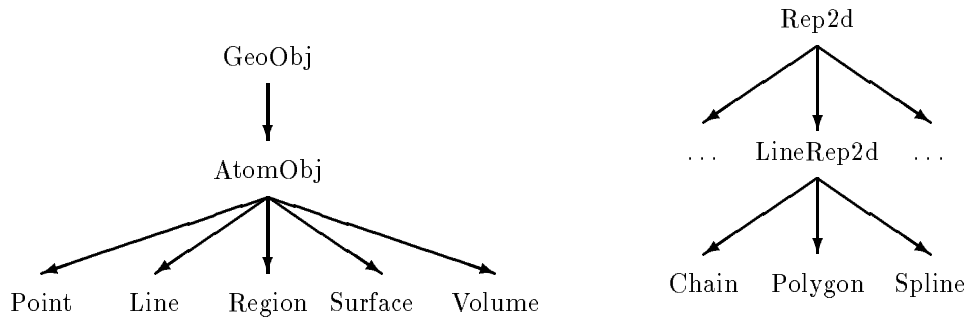


Fig. 2: Hierarchy of atomic objects (left) and their representations (right); arrows indicate the relation ‘specialization’ and the technical property ‘inheritance’.

invoking methods. The internal structure of the object is hidden. The differences between classes and types are treated in the paper of Cardelli and Wegner.³

Starting with the general concept of an image, a hierarchy of image classes may be formed.^{12, 11} An object oriented approach to image preprocessing is thus somewhat natural and was recently proposed by several authors.^{2, 5, 7, 8, 26, 29} Peripheral devices for image processing and special hardware often used in image preprocessing and image display may as well be formulated using objects. Object oriented systems have also been proposed for image analysis systems covering iconic and symbolic processing.^{4, 10}

Any of the concepts listed in Tab. 2 may be specified as a class. We discriminate between geometric objects (Fig. 2 left) and their associated representations (Fig. 2 right). The former classes are independent of the dimension. The latter have a fixed dimension. The figure shows representation classes for two-dimensional lines. An atomic line object may be used in a two-dimensional context. Possible representations are chain codes, polygons and splines. The dimensionality of an atomic object is therefore attached to the object by its representation rather than to the class. Some methods for the classes *Chain* (chain code) and *Line* are listed in Tab. 4.

The segmentation object as formally introduced in section 6 consists of parts (being segmentation objects) and relations between them. Objects without further parts are called atomic objects. This structure can easily be achieved by a specialization hierarchy of classes as shown on the left bottom of Fig. 4: segmentation objects as well as atomic objects are derived from geometric objects. Parts of segmentation objects may be geometric objects, i.e., either segmentation objects or atomic objects.

Class	Message	Description
Chain	noOfLinks	returns the number of links in the chain
	StartPoint	returns a <i>Point</i> -Object
	EndPoint	returns a <i>Point</i> -Object
	IsOnLine	Checks whether a <i>Point</i> lies on this Line
	XDR	stores (resp. reads) an external representation
Line	StartPoint	uses the corresponding method of the representation
	length	returns the length of the line
	getRep	returns a representation (e.g. a <i>Chain</i>).
	display	displays the line on the frame buffer
	XDR	stores (resp. reads) an external representation

Tab. 4: Examples of methods for geometric objects and representations

Relations are specified as separate classes. The relations in a segmentation object are restricted to its parts, i.e., we do not allow that parts of a segmentation object refer to objects outside of this object via relations. Another restriction is put on the parts; it is forbidden that a segmentation object contains itself, even transitively. The parts of

a segmentation object thus form a directed acyclic graph. It is possible however, that a segmentation object is part of several objects at the same time. This allows competing alternatives already in the segmentation stage. An example of these properties is shown in Fig. 3. These restrictions are checked by the methods that add parts and relations to the segmentation objects. Some of these methods are listed in Tab. 5.

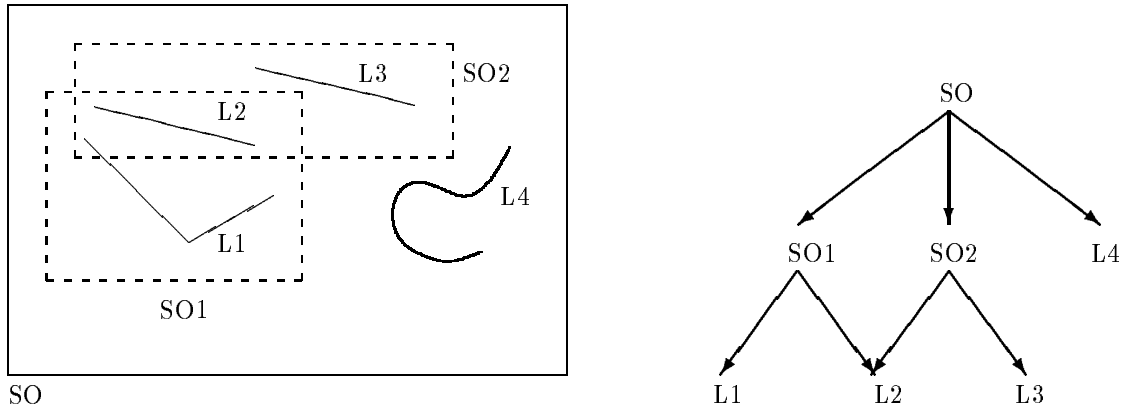


Fig. 3: Example for a hierarchy of parts in segmentation objects. The arrows on the right side indicate the relation 'part-of'.

Message	Description
addPart	adds a geometric object
getParts	returns a set of all parts
addRel	adds a new relation between parts
getRel	returns a set of all relations
display	displays the object on the frame buffer
XDR	stores or retrieves the object

Tab. 5: Some methods of the class 'SegObj'

8 THE $\zeta\pi\pi\sigma\varsigma$ SYSTEM

A complete implementation of a class hierarchy for iconic processing has been implemented in a system called 'hippos' (**H**ierarchy of **P**icture **P**rocessing **O**bjects, in greek letters written as $\zeta\pi\pi\sigma\varsigma$; the author of the system likes playing around with characters, icons and symbols and was inspired by Hofstadter's book¹⁷).²² It is implemented mostly in C++ and uses the 'National Institutes of Health' (NIH) class-library.¹⁵ Currently about 70 classes for iconic processing are implemented and tested successfully, including those for iconic three-dimensional image segmentation. An overview of the top of the hierarchy is given in Fig. 4. The hierarchies in Fig. 2 are two examples of a further refinement. The messages listed in Tab. 4 and 5 are subsets of the messages understood by the actual objects in $\zeta\pi\pi\sigma\varsigma$.

The abstract segmentation object (section 6) may have a variable number of attributes and a certainty factor. These facilities are introduced for the Hippos-Object (Fig. 4) which is the top of the hierarchy of image processing classes. Every class in the subtree may thus have an attribute-list and a certainty factor.

Using the general segmentation object this system allows the uniform implementation of preprocessing and segmentation algorithms in two or three spatial and temporal dimensions. Integrated computer vision is thereby possible as proposed in the book of Aloimonos.¹

9 EXTERNAL REPRESENTATION

The process of segmentation is usually divided into several intermediate steps (Fig. 1), often implemented as separate processes. It is therefore necessary to have interfaces for sharing intermediate results among successive processes. These results will normally be external representations of $\iota\pi\pi\sigma$ -objects. Another interface must be provided to give access to the results of existing programs not written in C++.

The different processes may be executed on different machines, since the special resources needed for image processing like stereo cameras or frame buffers may not be available at every machine on the network. Thus, the interfaces have to deal with different internal data formats. Because of the large amount of data to be transmitted (images of 512^2 points for example) the data have to be in a compact format.

Every $\iota\pi\pi\sigma$ -object (indeed every object in the system, including those imported from NIH) provides an external representation based on XDR (eXternal Data Representation).³¹ Generation and interpretation of the machine-independent binary representation is simply performed by a method obligatory to any class. Special care has been taken to include version and diagnostic information in the external representation with a minimum of storage overhead. The representation scheme is general enough to fulfill all the requirements of object oriented programming and — for selected classes — simple enough to be interpreted by conventional programs. Conventional programs for line detection and region segmentation (arcs 1-10 in Fig. 1) may thereby be combined with object oriented segmentation algorithms by a common representation for images. The external representation of the segmentation object provides a general interface for further symbolic image analysis.

The data representation scheme also provides the means for easy implementation of remote procedure calls (rpc) which can easily be programmed since the SUN-rpc³¹ and XDR are closely related. This allows for transparent access to remote image processing hardware (for an example refer to the method 'display' in Tab. 4 and 5; the call will invoke a remote procedure if the hardware addressed is not locally available and a local call otherwise).

Writing a 512^2 -edge-image, i.e., an image consisting of edge directions and edge strength, with the XDR-method requires 19 seconds on a 3 MIPS workstation instead of 1.9 seconds for writing the raw data. The XDR-function is called for every edge element. Presently, we sacrifice this time since it is essential to have machine independent storage in a local area network of heterogeneous computers. In future applications optimized XDR-routines with fewer function calls will reduce the overhead. Furthermore the time for reading and writing the information is normally small compared to the time for the real processing of images. (The computation of the above mentioned edge image with a Sobel-operator requires 100 seconds).

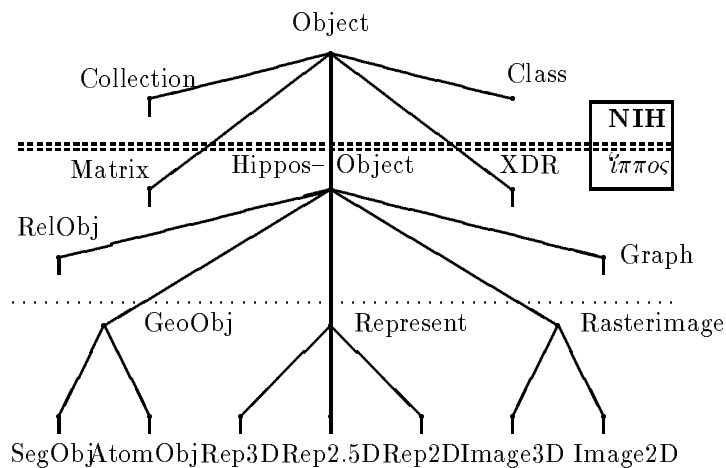


Fig. 4: An overview of the $\iota\pi\pi\sigma$ -class hierarchy based on the NIH class library¹⁵

10 STANDARDS AND OUTLOOK

International standardization of image processing and image file formats is in progress.^{6, 11} In addition many de-

facto standards for image data exist. The external representation in $\zeta\pi\pi\sigma$ is a union of XDR and the structure imposed by the NIH class library. An incorporation of standard data compression schemes is currently implemented. Simple filters convert image objects to any other data format and vice versa. The same holds for the external representation of segmentation objects.

Provided that the image analysis community agrees on a standard for image preprocessing, the standardization of modules for segmentation could be the next step. This would require a standard segmentation object which may be based on the abstract definition in section 6 and the book of Niemann.²⁰ A language binding for the segmentation object will also have to specify an external format for the objects. A concrete implementation may use XDR as in $\zeta\pi\pi\sigma$ or may as well be based on the 'Abstract Syntax Notation' (ASN.1).¹⁴

REFERENCES

1. John (Yiannis) Aloimonos and David Shulman. *Integration of visual modules*. Academic Press, Boston, 1989.
2. V. Cappellini, A. Del Bimbo, and A. Mecocci. Object oriented system for image processing. In V. Cappellini, editor, *Time-Varying Image Processing and Moving Object Recognition: Proc. of the 3rd Int. Workshop*, pages 69-74. Elsevier, Amsterdam, 1990.
3. L. Cardelli and P. Wegner. On understanding types, data abstraction, and polymorphism. *Computer Surveys*, 17(4):471-522, 1985.
4. Huey Chang, K. Ikeuchi, and T. Kanade. Model-based vision system by object-oriented programming. In *Robots: Coming on Age. Proceedings of the International Symposium and Exposition on Robots (19th ISIR)*, pages 295-313, Sydney, Australia, 1988.
5. R. Checcini, A. Del Bimbo, and P. Nesi. Ipoos: An advanced system for automatic classification. In V. Cappellini, editor, *Time-Varying Image Processing and Moving Object Recognition: Proc. of the 3rd Int. Workshop*, pages 75-81. Elsevier, Amsterdam, 1990.
6. Adrian F. Clark. An international standard for image processing and interchange. *Technical Committee Newsletter, PAMI*, 14:5-6, 1991.
7. J. M. Coggins. Integrated class structures for image pattern recognition and computer graphics. In Keith Gorlen, editor, *Proceedings of the USENIX C++ Workshop*, pages 240-245, Santa Fe, New Mexico, 9.-10. November 1987.
8. M.R. Dobie and P.H. Lewis. Data structures for image processing in C. *Pattern Recognition Letters*, 12:457-466, 1991.
9. Robert B. Fisher. *From Surface to Objects*. Wiley and Sons, Chichester, 1986.
10. Myron Flickner, Mark Lavin, and Das Sujata. An object-oriented language for image and vision execution. In *Proceedings of the 10th International Conference on Pattern Recognition (ICPR), Volume II*, pages 561-571, Atlantic City, 1990.
11. P. Gemmar and G. Hofele. *Empfehlung für ein Ikonisches Kernsystem IKS*. FIM Informationsverarbeitung und Mustererkennung), Karlsruhe, 1989. Unter Mitarbeit von: L. Dreschler-Fischer, H. Faasch, D. Haaks und D. Paulus, Bericht der Fachgruppe ITG im VDE, Fachgespräche IKS 1987-1989.
12. P. Gemmar and G. Hofele. An object oriented approach for an iconic kernel system IKS. In *Proceedings of the 10th International Conference on Pattern Recognition (ICPR), Volume II*, pages 85-90, Atlantic City, 1990.
13. Andrew M. Goodman, R. M. Haralick, and L. G. Shapiro. Design of an integrated programming language and data management system for knowledge-based computer vision. Technical Report 89-8-3, University of Washington, Department of Computer Science, Seattle, Washington, 1989.
14. W Gora and R Speyerer. *ANS.1*. DATACOM-Verlag, Pullheim, 1987.

15. Keith E. Gorlen, Sandy Orlow, and Perry S. Plexico. *Data Abstraction and Object-Oriented Programming in C++*. John Wiley and Sons, Chichester, 1990.
16. A.R. Hanson and E.M. Risemann. Representation and control in the construction of visual models. Technical report, A Progress Report on Visions, University of Massachusetts, Amherst, Mass., 1976. TR 76-9, Dep. of Computer and Information Science.
17. Douglas R. Hofstadter. *Gödel, Escher, Bach*. Vintage Books, New York, 1980.
18. S.L. Horowitz and T. Pavlidis. Picture segmentation by a tree traversal algorithm. *J. Assoc. Comput. Mach.*, 23:368–388, 1976.
19. D. Marr. Representing visual information. In A.R. Hanson and E.M. Risemann, editors, *Computer Vision Systems*, pages 61–80. Academic Press, New York, 1978.
20. H. Niemann. *Pattern Analysis and Understanding*. Springer, Berlin, 1990.
21. D. W. R. Paulus. Objektorientierte bildverarbeitung. Technical report, Dissertation, Technische Fakultät, Universität Erlangen–Nürnberg, Erlangen, 1991.
22. D. W. R. Paulus. *Objektorientierte und wissensbasierte Bildverarbeitung*. Vieweg, Braunschweig, 1992.
23. Dietrich W. R. Paulus and Heinrich Niemann. Iconic–symbolic interfaces. In K. Pratt Ronald. B. Arps, William, editor, *Image Processing and Interchange: Implementation and Systems*, pages 204–214, San Jose, 1992. Proc. SPIE 1659.
24. T. Pavlidis. *Structural Pattern Recognition*. Springer, Berlin, 1977.
25. A.P. Pentland. *From Pixels to Predicates*. Ablex Publishing Co., Norwood, 1986.
26. J. Piper and D. Rutovitz. An investigation of object-oriented programming as the basis for an image processing and analysis system. In *Proc. 9th Int. Conf. on Pattern Recognition (ICPR)*, volume 2, pages 1015–1019, Rome, 1988.
27. S. Posch. *Automatische Bestimmung von Tiefeninformation aus Grauwert-Stereobildern*. Deutscher Universitäts Verlag, Wiesbaden, 1990.
28. V. Roberto and R. Milanese. Matching hierarchical structures in a machine vision system. In *Proc. of an Int. Conf. Intelligent Autonomous Systems 2*, pages 845–852, Amsterdam, 1990.
29. H. Sato, H. Okazaki, T. Kawai, H. Yamamoto, and H. Tamura. The view-station environment: Tools and architecture for a platform-independent image-processing workstation. In *Proceedings of the 10th International Conference on Pattern Recognition (ICPR), Volume II*, pages 576–583, Atlantic City, 1990.
30. L. Shapiro. Design of a spatial data structure. In H. Freeman and G.G. Pieroni, editors, *Map Data Processing*, pages 101–117. Academic Press, New York, 1980.
31. Sun OS 4 Manuals, Network Programming, Part 2, Mountain View, CA. *RPC eXternal Data Representation Standard: Protocol Specification*, Revision B, 1986.
32. S.L. Tanimoto. An iconic/symbolic data structuring scheme. In C.H. Cheng, editor, *Pattern Recognition and Artificial Intelligence*, pages 452–471. Academic Press, New York, 1976.