OBJECT ORIENTED IMAGE SEGMENTATION

Dietrich W. R. Paulus

The following paper has appeared in the
Proceedings Int. Conference on Image Processing
Maastrich, April 7–9, 1992 (ISBN 0 85296 543 5)

Version: 1.5

Proc. 4th Int. Conf. on Image Processing and its Applications, Maastrich, April 7–9, 1992 (ISBN 0 85296 543 5)

OBJECT ORIENTED IMAGE SEGMENTATION

Dietrich Paulus

Lehrstuhl für Informatik 5 (Mustererkennung), Universität Erlangen–Nürnberg

1 Introduction

Research and application of image processing at the present state of the art usually require exhaustive use of computer software. Various large programming environments are available for that purpose. These — like any other large programming systems — still suffer from what was called the "software crisis" in the sixites. Interdependencies of the parts of the programs are often poorly structured. Consequently, the reuse of existing software is often more difficult than rewriting the required algorithm. One possible cure is the standardization of image processing routines and data structures as currently done by the ISO.

Modern software technology aims for high modularity. Data encapsulation offers the mechanism for structuring the interdependencies of software modules. Object oriented programming combines encapsulation with inheritance. Inheritance is a mechanism that allows for the reuse of the code written for the base classes in the derived class in a highly structured way. The object oriented approach thus is another possible cure to the software crisis.

The basic elements of object oriented programming are classes, objects and inheritance. *Classes* roughly correspond to data structures in conventional programming, combined with the required operations. The data components of classes are called *attributes*. *Objects* are instances of classes, i.e. they may be seen like a piece of memory. Objects are accesed by *messages* which invoke *methods*. *Inheritance* of methods and attributes is specified for classes. At this level of detail we do not have to go into the more subtle differences between classes and types (see: [2]).

2 Related Work

Object oriented programming faciliates programming of image preprocessing as well as image segmentation and analysis. Up to now object oriented systems have been proposed either mainly for preprocessing (e.g. [3, 5]) or for image analysis (including preprocessing, segmentation and the knowledge based part of the analysis; e.g. [4, 1]).

Typical data structures for image preprocessing are *images* or *devices* (e.g. frame–grabber, mouse etc.). These are naturally modelled by classes in an object–oriented formalism thereby allowing for a configuration independent access of devices by the methods of the classes.

Image analysis requires for the representation of many other types of information; knowledge about the particluar task domain usually has to be represented. Object oriented systems provide classes for that purpose.

3 Data Structures for Image Segmentation

Various data structures and data types are required for image segmentation, including all those for preprocessing. Very simple examples are enumeration types for color channels (like *Red*, *Green*, *Blue*) or the type *Gray–Level*. Simple *image data structures* consist of the raster data and size and bookkeeping information.

During image segmentation simple constituents like *lines*, *regions*, *vertices*, *surfaces* will be extracted [9]. These have to be represented in data structures as well. The segmentation stage usually results in a *set* of objects belonging to these classes, together with some features and *relations* between them. Examples of relations are groupings of lines, collinearity, or parallel lines.

Data structures for a general representation of the results of the segmentation stage can be found in the literature; they may serve as an interface to knowledge based symbolic processing and are listed in table 1; a more complete list can be found in [10]. Any of them uses sets or relations, which therefore have to be represented by data structures as well.

In general the segmentation leads to *segmentation objects*. They consist of simple constituents together with relational information and associated feature lists. We allow for a recursive definition of a segmentation object SO:

$$\begin{aligned} \mathrm{SO}: \quad & ((A : (T_A, R \cup V_T))^*, \\ & (P : SO)^*, \\ & (S(P) : R)^*, \\ & CF : R) \end{aligned}$$

A segmentation object consists of:

- a list of attributes or features (A), each represented as a pair of type (T_A) and value. The

Data Structure	Author
Sketches	[8]
Iconic–Symbolic Data Structure	[13]
RSE–Graph	[7]
Line Adjacency Graph	[11]
Region Adjacency Graph	[11]
Spatial Data Structure	[12]
Segmentation Objects	[10]

Table 1: Some candidates for an interface to knowledge based processing

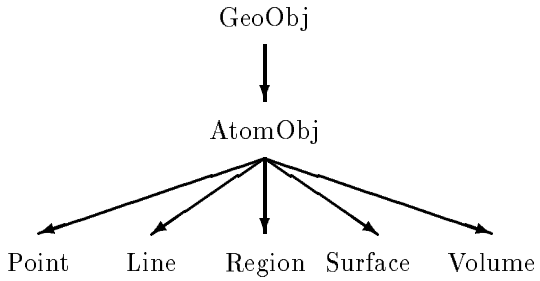


Figure 1: Hierarchy of atomic objects; arrows indicate the relation "specialization" and the technical property "inheritance".

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 structural (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 elaborate definition of the segmentation object see [9].

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.

An overall measure of the quality of the segmentation object may be placed in the certainty factor (CF). This simplifies further knowledge based processing which often has to rank competing results.

4 A New Object Oriented System

The data structures listed in section 3 can naturally be extended to classes by adding methods to them and imposing a hierarchical structure on the classes. The system "hippos" (**Hi**erarchy of **P**icture **P**rocessing **O**bjects, in greek letters written as ἵππος) provides a large number of classes for image preprocessing and segmentation, including classes for the data structures mentioned in section 3.

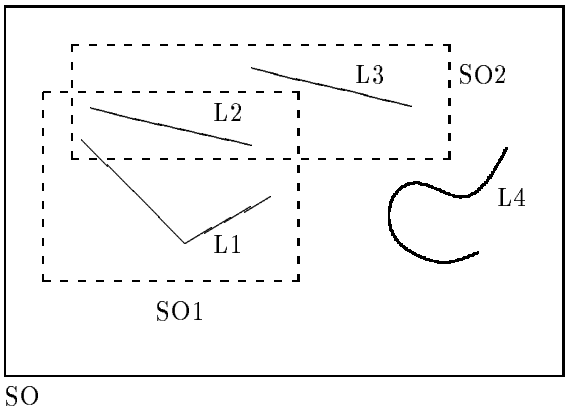
Points, lines, regions, and surfaces are specializations of the general concept 'atomic object' (AtomObj) which is in turn a special case of a geometric object (GeoObj). This hierarchical relation is mapped to a class inheritance tree (figure 1) which is part of the ἵππος–class–hierarchy.

Geometric objects may be either atomic or segmentation objects (SegObj), which serve as compound objects. Segmentation objects consist of parts and relations between them. By allowing *geomentric objects* as parts of a segmentation object we accomplish an expressive power similar to the recursive description in section 3. (See figure 4 on the left bottem for this part of the ἵππος–class–hierarchy).

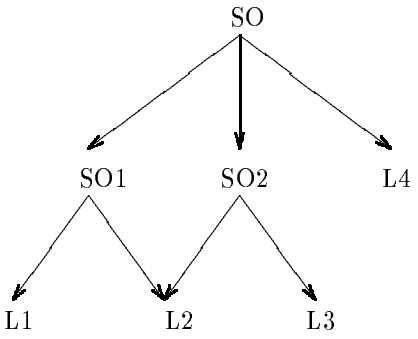
The relations in a segmentation object are restricted to its parts, i.e. we do not allow that parts of a segmentation object refer to objects outside this object via relations. Another restriction is put on the parts; it is forbidden that a segmentation object contains itself, even transitively. The parts of a segmentation object thus form a directed acyclic graph. It is however possible, that a segmentation object is part of several objects at the same time. This makes it possible to represent competing alternatives already in the segmentation stage. The relevance of the objects may be weighted in the certainty factor. These restrictions are checked by the methods that add parts and relations to the segmentation objects. These properties are depicted in figure 2.

5 Geometric Representations

Atomic objects (figure 1) may have two or three dimensions, depending on the application. Rather than attatching the dimensionality to the classes we attach it to the objects. Every atomic object contains a reference to an inidividual representation which can be chosen out of various representation classes at the time of instantiation. Chain codes, polygons and splines are examples of representations for lines. Any of these representation may be specified in two or three dimensions. Regions may be represented as quad trees, binary images or by their contours. This part of the ἵππος tree is shown in figure 3.



a.) Scene



b.) Hierarchy of parts

Figure 2: Example for a hierarchy of parts in segmentation objects. Arrows indicate the relation "part–of".

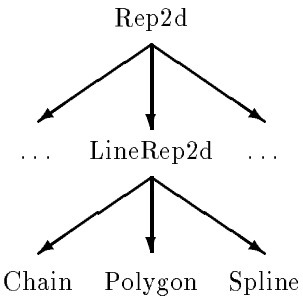


Figure 3: Representation classes for geometric objects

6 External Representation of Image Objects

Standards for image representation currently cover pictorial data only (e.g. tiff). Segmentation results however require for the representation of symbolic data as well as numeric and relational information. The de–facto standard XDR (SUN) for the binary representation of arbitrary data has be extended to objects in ἵππος. Every class posesses the method `xdr` which interprets resp. creates an external binary representation which is independent of particular machine properties.

XDR is used for the widely used SUN–rpc (remote procedure call) as well. Distributed image processing has been incorporated in ἵππος by the use of the xdr–methods in combination with rpc. An example is a frame–grabber device which can be accessed in the network transparently.

7 Results

The system ἵππος is implemented in C++ using the NIH–Class library ([6]). Up to now, all results of 2–D segmentation could be represented in the system. The classes for volumes and surfaces in figure 1 are however not yet completely implemented. Three–dimensional segmentation is under development. An overview of the classes is given in figure 4. In total, approximately 70 classes have been implemented and tested successfully.

According to common prejudice object oriented programming tends to be very time consuming. The system ἵππος however performs as well as a conventional programming system in the areas, where a comparison to conventional programs has been made. The frequent and thus time consuming access to matrix elements — altough handled by objects — can optionally be mapped *directly* to memory addresses. When compiled and linked with the appropriate options, a matrix operation (e.g. an edge–operator) performs exactly as fast as a programm written in conventional C.

Adding an element to a large segmentation object (over 100 parts) requires approximately 5 ms on a 2 MIPS computer. XDR–representation may be time consuming when large images of structured data have to be stored. Writing a 512^2–gradient–image, i.e. an image consisting of edge directions and edge strength, with the xdr–method requires 44 seconds instead of 1.9 seconds for writing the raw data. We do however have to sacrifice this time since it is essential to have machine independent storage in a local area network of heterogenous computers. Furthermore the time for reading and writing the information is normally small compared to the time for the real processing of images.

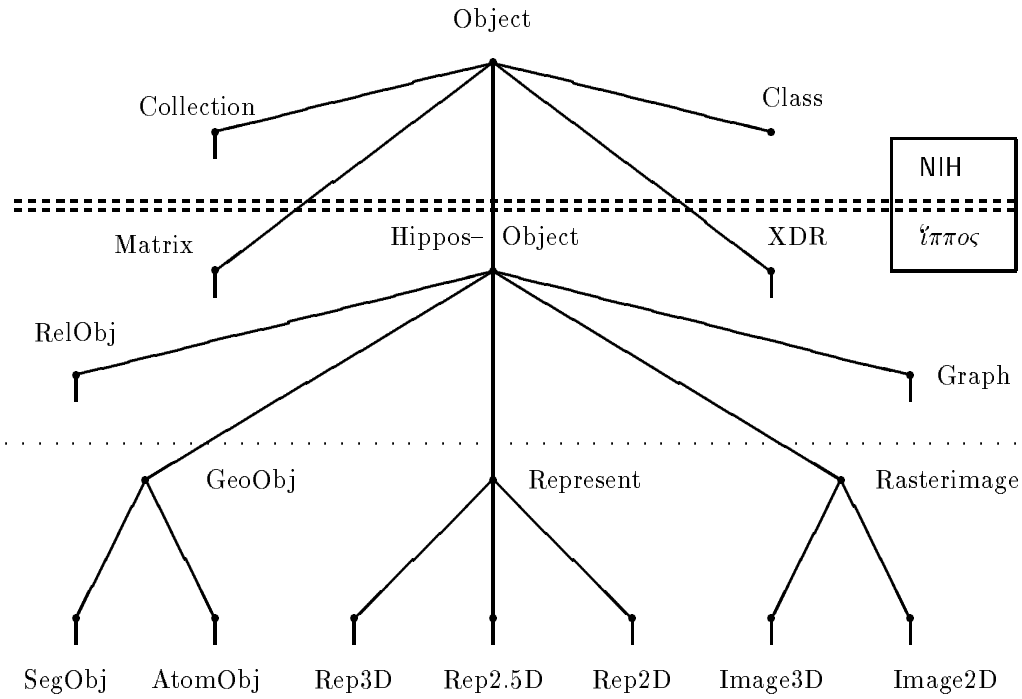


Figure 4: The ἵππος–class hierarchy **Updated in respect to the print!**

References

[1] 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.

[2] L. Cardelli and P. Wegner. On understanding types, data abstraction, and polymorphism. *Computer Surveys*, 17(4):471–522, 1985.

[3] M. Dobie and P. Lewis. Data structures for image processing in C. *Pattern Recognition Letters*, 12:457–466, 1991.

[4] M. Flickner, M. Lavin, and D. 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.

[5] 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.

[6] K. E. Gorlen, S. Orlow, and P. S. Plexico. *Data Abstraction and Object–Oriented Programming in C++*. John Wiley and Sons, Chichester, 1990.

[7] A. Hanson and E. 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.

[8] D. Marr. Representing visual information. In A. Hanson and E. Risemann, editors, *Computer Vision Systems*, pages 61–80, Academic Press, New York, 1978.

[9] H. Niemann. *Pattern Analysis and Understanding.* Springer, Berlin, 1990.

[10] D. Paulus. *Objektorientierte Bildverarbeitung.* PhD thesis, Technische Fakultät, Universität Erlangen–Nürnberg, Erlangen, 1991.

[11] T. Pavlidis. *Structural Pattern Recognition.* Springer, Berlin, 1977.

[12] L. Shapiro. Design of a spatial data structure. In H. Freeman and G. Pieroni, editors, *Map Data Processing*, pages 101–117, Academic Press, New York, 1980.

[13] S. Tanimoto. An iconic/symbolic data structuring scheme. In C. Cheng, editor, *Pattern Recognition and Artificial Intelligence*, pages 452–471, Academic Press, New York, 1976.