MVA'94 IAPR Workshop on Machine Vision Applications Dec. 13-15, 1994, Kawasaki

GENERATING SYMBOLIC DESCRIPTIONS OF TWO-DIMENSIONAL BLOCKS WORLD

A. Sowmya and K. W. E. Lee
School of Computer Science and Engineering
University of New South Wales

Kensington, NSW 2052, Australia
email on internet: sowmya@spectrum.cs.unsw.oz.au

ABSTRACT

This paper considers three-dimensional (3-D) scene analysis and explores the role of models for this task. Both object and task models are included and the issues of *model design* and the *mapping from image to model* are raised. The imaging process produces a two-dimensional (2-D) image of a real-world scene, while in the modelling process, a model of the real-world objects and tasks is built. The 2-D image is then analyzed using image processing and computer vision tools, in the light of the model built.

In this paper, we propose a methodology to build symbolic descriptions of images which addresses these issues. A major goal is to use non-pictorial means to specify a model of the world and the application. Another goal is the separation of domain-dependent and application-dependent aspects of the problem from the independent aspects. To illustrate the methodology, a case-study of a 2-D blocks world is presented, for which a model specification using a Symbolic Description Language (SDL), based on symbolic logic, is presented. The results of an experiment in building an object recognition application in this domain, using a user-designed symbolic object model, are discussed.

INTRODUCTION

Computer vision must produce a "useful" description of a scene depicted in an image, whose initial representation is an array of image intensity values. At the *low-level vision* stage, the early processing of the image takes place, and domain-independent image processing algorithms deliver more generalised image representations to the *higher-level vision* stage (Marr 1982, Ballard and Brown, 1982). To cope with the changes in lighting and viewpoint, the effects of shape and shading, variations in the imaging process such as in camera angle and position, and noise at the lower level, we need rich representations of the world at the higher level.

Of the many high level representations used for three-dimensional (3-D) scene analysis, model-based methods have been very popular (Pope, 1994, Zhang et al., 1993), though other approaches that rely on knowledge of the context or function have also been reported (Strat and Fischler 1991, Stark and Bowyer 1991). Further, the term *model* often seems to refer to object models, as in object recognition systems, and only occasionally to models appropriate to the application, as in task planning applications. Models may be geometric, quantifying shape information, or relational, specifying relationships between components in the scene. While building a model, issues such as what features to include in the model, choosing an appropriate model representation, the method of model acquisition and the generality of the model have to be addressed. Till now, most models have been acquired manually using common sense knowledge of the scene or by consulting a human expert familiar with the application, and this is a very time-consuming and error-prone process. Methods for automatic model building for scenes and their images are not commonly available nor used. Finally, general-purpose models have proved difficult to build, and the models that were built were highly domain-specific.

There appear to be two issues here, namely *model design*, and the *mapping from image to model*. In any recognition problem, it is accepted that the selected features and their representation are crucial for success (Connell and Brady, 1987). Thus in model design, high level features which are scene- and application-specific are defined, often by hand. The relationships between these features may impose structure on the model, and the model representation must be capable of representing this structure. Finally, given an image, the high-level features of the model must be mapped into low-level image features that may be derived using image processing techniques.

In this paper, we propose a methodology to build symbolic descriptions of images, which addresses both these issues. The major focus of our proposal is a shift in paradigm for representing images. As opposed to conventional pixel matrices and image-level properties such as edges and their derivatives, we propose a shift to the symbolic domain where multiple levels of image properties and features are represented symbolically and uniformly. A second focus is the separation of domain-dependent and application-dependent aspects of the problem from the independent ones. To illustrate the methodology, we undertook a case study of a two- di-

mensional (2-D) blocks world for which symbolic descriptions were generated automatically and tested on an object recognition system successfully (Lee, 1993). The paper describes this study in detail.

THE METHODOLOGY

The proposed framework accommodates a variety of domains and applications in a uniform way, by separating the domain- and application independent image processing from the model design and mapping phases. Image features are obtained by applying low-level image processing techniques and thereafter represented using a symbolic feature language.

In a separate phase, the domain- or application dependent aspects may be added in a number of ways. For example, as in Figure 1, a model may be designed, including model features suitable for a specific task in the domain, and the mapping from model features to image features is also designed. Significantly, both the image and model features are represented in the same symbolic language. We choose *symbolic logic*, which is a well-studied and analysed knowledge representation technique in Artificial Intelligence, and there is a huge body of analysis tools to draw upon. Thus, a variety of problems in computer vision may be handled using a single toolkit which is readily available, without the distraction of lower level image processing issues. Another advantage is the modular extensibility of the model, since in a symbolic representation, new features may be appended without affecting the earlier features. Finally, in comparison to frames and semantic networks, logic provides a flat structure which is an advantage when we do not know the relative importance of features and their relationships. This is also an advantage when we try to *learn* structure rather than supply it ourselves by building it into the representation; this will be a focus of future research.

We assume that only first order symbols would be used, so that a first order symbolic language should be sufficient to represent model features. We choose Prolog, which is a logic programming language based on Horn clause logic, a subset of first order predicate logic. A Prolog program works on a database of Prolog clauses, and attempts to prove theorems using the resolution principle. Our plan is to represent the model features using Prolog clauses, so that the description of an image would be a collection of Prolog clauses. The image may then be analysed by writing Prolog programs which work on the clauses representing the image.

CASE STUDY

To illustrate the approach and the methodology outlined so far, we studied the problem of automatically building symbolic descriptions of 2-D images of a constrained scene, namely the blocks world. This paper reports on this experience and the conclusions reached.

Overview

The domain chosen is the two-dimensional blocks world, whose objects may include two-dimensional polygons of any size, and also circles and ellipses of any radius. Besides being an oft-studied, and hence "benchmark" domain, we think that polygonal blocks are sufficient to provide a good approximation of 2-D shapes in many constrained domains. In our methodology, both the domain and the application in the domain ought to be specified beforehand. We chose object recognition as our application, based on an object model specified by the user. A working 2-D symbolic model building system has been implemented and tested successfully for object recognition.

The major effort was spent on designing a symbolic description language suited to the domain and the application. This language is the vehicle that captures the appropriate model for the domain-application pair. Then, a mapping from low-level image features, obtained by image processing techniques, to the high level features in the symbolic description language, had to be designed. A number of images of the 2-D blocks world were then run through this system and symbolic descriptions of the scenes obtained. Finally, these descriptions were fed into an object recognition program written in Prolog. The remaining subsections describe the symbolic description language and the experiments conducted.

Earlier Work

On a search of earlier models of the 2-D blocks world, we found that lines-junctions analysis due to Huffman and Clowes is one of the earlier models, and describes objects in an image hierarchically using lines, vertices and their relationships. Lines and vertices are directly obtainable from an image and hence the mapping between model features and image features is simple. Badler presents a hierarchical model of objects based not only on their physical attributes such as location and orientation, but also more abstract ones such as their mobility (Badler, 1975). Our main inspiration was the paper by Ambler and Popplestone (Ambler and Popplestone, 1975), in which they use object features to describe objects. These features include spatial relationships expressed in mathematical form.

Symbolic Description Language (SDL)

A symbolic model of the 2-D world is embedded in

a **S**ymbolic **D**escription **L**anguage (called SDL, hereafter). When we set out to design this language, our requirements included the following:

- to facilitate the representation of an image in a form amenable to symbolic processing
- to include at least the primitive features of objects in the domain, so that more sophisticated features can be built using them
- to allow flexibility in further processing and interpretation
- to be able to plug into existing algorithms and methods for applications in the domain.

We chose to embed SDL in Prolog, which is a logic-programming language suitable for symbolic processing. This decision enabled us to build applications quickly and cheaply, and also to hook into other image processing and vision systems easily.

The basic features included in SDL are of three types:

1. *locational features* which represent location information of an object or part of an object
2. *geometric features* which represent geometric properties of an object or part of an object
3. *statistical features* which represent statistical measures derived from the image.

Locational Features: Locational features describe the location of an object part or the locational relationships between parts, between objects and parts, and between objects. *Absolute* locational features describe locational information in absolute co-ordinates with respect to the whole image, whose origin is chosen at the upper left corner of the image by convention. For example, the centroid of an object is an absolute locational feature. Table 1 shows some of these features.

Relative locational features indicate locational relationships between objects and parts. For example, *toRight (A, B)* indicates that object *B* is to the right of object *A*. Table 2 shows more of these features.

Geometric Features: Geometric properties of objects are captured by geometric features. A geometric feature may be quantitative, statistical or compound in nature. *Quantitative* geometric features describe measurable geometric properties indicated quantitatively. For example, *rawArea* indicates the area occupied by the object in the image. Table 3 lists some of these features.

Statistical geometric features describe statistical geometric properties of an object, such as major variance and skewness. This statistical information cannot be measured directly from the image but must be computed. Table 4 lists some of them.

Compound geometric features are constructed from the quantitative and statistical geometric features. They describe specific geometric properties which are derivable mathematically from the others. An example is the perimeter of a closed object. Table 5 lists a few.

Finally, wholistic *image level features* such as average brightness of the image, frame dimensions, the number of objects in the image and a label for each object are also computed or assigned.

As evident from the syntax of the features, SDL employs Prolog-like relations to specify the features. SDL is simple to use and interfaces cleanly with a Prolog environment.

The environment

We now describe the experimental setup and the hardware/software resources before describing the image-model mapping.

A high resolution CCD camera is used to acquire monochromatic images of 2-D blocks. A frame grabber attached to an IBM-compatible PC gets the image and transmits it to an Apollo Unix workstation. All further processing takes place on the Unix platform.

Simple image operations, such as baseline correction, histogram generation and image averaging, may be done on the PC itself, using a package called DT-IRIS, which is designed to be used with the frame grabber card DT2851 from Data Translation. On the Unix platform, we use the image processing package HIPS which has a rich set of library functions to perform advanced and complex image analysis using C-calleable functions. Finally we used UNSW-Prolog programming environment for the object recognition phase of the project.

Image-Model Mapping

We now define mappings from image features to the model level features. The image level features are acquired from the image using standard image processing techniques. From these features, the model level features are built using the pre- defined mappings. Note that both sets of features are represented as Prolog clauses and hence are compatible for symbolic processing.

The sequence of processing steps is as follows:

1. image format conversion
2. pre-processing operations

3. thresholding to obtain binary image

4. noise removal, and stretching to remove distortion due to different aspect ratios

5. extraction of object features

6. generation of symbolic descriptions.

The output is a symbolic description of the scene in SDL, which may be further analyzed by symbol manipulation software.

Figure 2 illustrates a set of sample outputs for a 2-D blocks world scene. Figure 2(a) is the input image. Figure 2(b) is created after extraction of image features.

The Application

The application chosen to illustrate the utility of this approach is object recognition. For this purpose, an object model was built by hand, after careful study of published models cited in the review. The model was encoded in Prolog and symbolic descriptions of the 2-D blocks world images were fed into the object recognition program. Figure 4 is Prolog dialog window, illustrating outputs of the object recognition application for the image in Figure 2(a). The program accepts queries about objects in the image and answers them after symbolic processing of the corresponding symbolic description.

CONCLUSION

We propose a symbolic framework for describing images and illustrate it by applying the methodology to a 2D-blocks world and building an object recognition application on top of it. The methodology is general and applicable to a variety of scenes and applications. The symbolic description language presented for the case study illustrates the methodology, and is not meant to be definitive. Currently, an application in the 3-D domain is being developed, which we we think will establish the extensibility of the methodology for 3-D.

REFERENCES

Ambler, A. P. and Popplestone, R. J. (1975) , Inferring the positions of bodies from specified spatial relationships, Artificial Intelligence, p. 157-174.

Badler, N. I. (1975), Temporal Scene Analysis: Conceptual descriptions of object movements, Tech. Rep. No. 80, Department of Computer Science, The University of Toronto, Canada.

Ballard, D. H. and Brown, C. M. (1982), Computer Vision, Prentice Hall Inc., En glewood Cliffs, New Jersey.

Connell, J. H. and Brady, M. (1987), Generating and generalizing models of visual objects, Artificial Intelligence 31, p. 159-183.

Lee, K. W. E (1993). Generating symbolic descriptions of images, B. E. Project report, School of Electrical Engg, University of New South Wales.

Marr, D. (1982). Vision, W. H. Freeman and Co., New York.

Pope, A. R. (1994), Model-based object recognition- a survey of recent research, Tech. Report 94-04, Dept of Comp. Sci., University of British Columbia, Vancouver, BC Canada.

Stark, L. and Bowyer, K. (1991), Achieving generalized object recognition through reasoning about association of function to structure, IEEE Trans. Patt. Anal. Machine Intell. 13 (10), p. 1097-1104.

Strat, T. M. and Fischler, M. A. (1991), Context-based vision: recognizing objects using information from both 2-D and 3-D imagery, IEEE Trans. Patt. Anal. Machine Intell. 13 (10), p. 1050-1065.

Zhang, S., Sullivan, G. D. and Baker, K. D. (1993). The automatic construction of a view- independent relational model for 3-D object recognition, IEEE Trans. Patt. Anal. Mach. Intell., Vol. 15, No. 6, p. 531-544.


```
%sp a1
Input prolog database file : a1.prolog
Producing prolog program file : a1.sympro
Loading UNSW Prolog

UNSW - PROLOG V4.2
: rectangles?
Object 5 is a rectangle
** yes
: triangles?
Object 3 is a triangle
Object 6 is a triangle
** yes
: circles?
Object 7 is a circle
** yes
: ellipses?
Object 1 is an ellipse
** yes
: pllobj(4)?
** yes
: squares?
** no
```

Figure 3: Dialog window of object recognition system

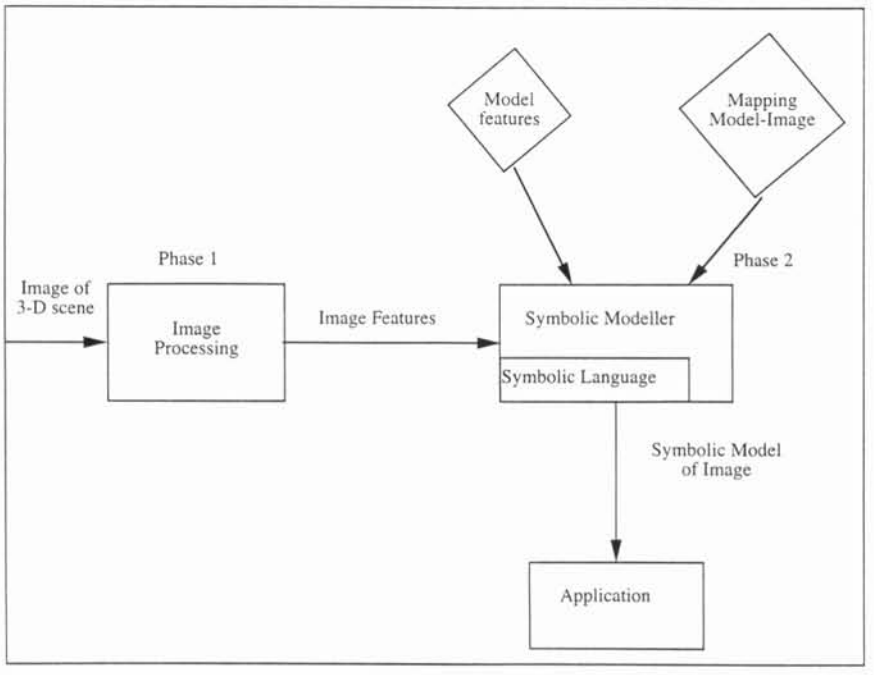
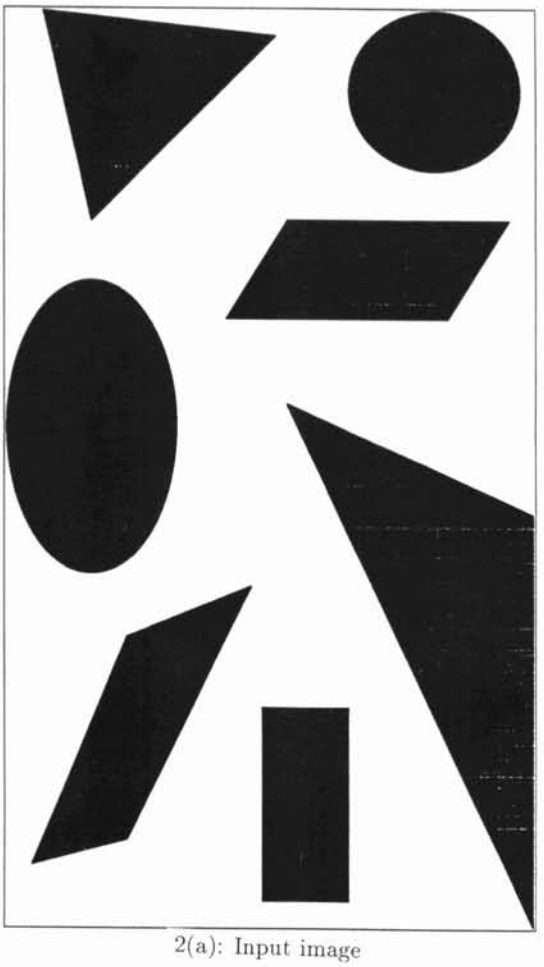
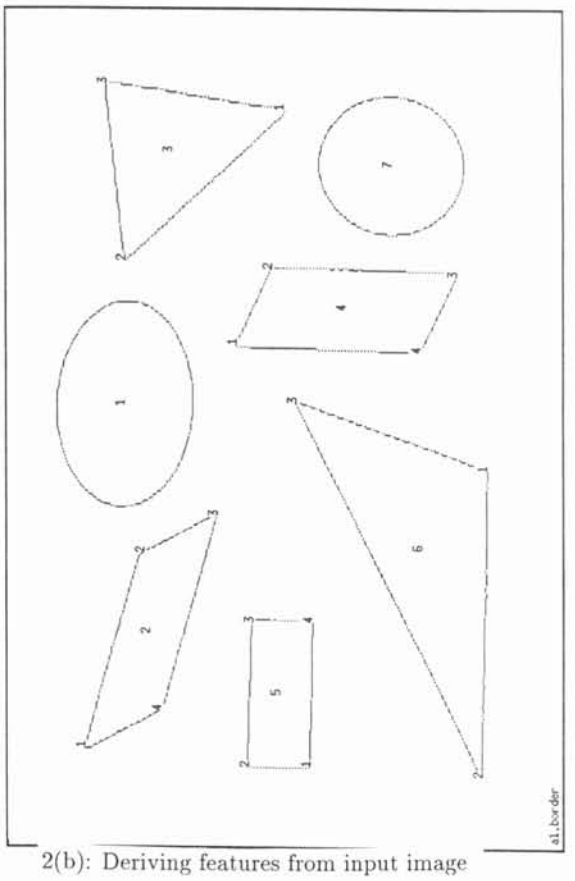


Figure 1: Schematic diagram of a symbolic modelling system



2(a): Input image



2(b): Deriving features from input image

Figure 2: Generating symbolic model of 2-D blocks world

1.	start(objectnum,xstart,ystart). The first pixel of the object num has the coordinates (xstart,ystart). This is the first pixel found when scanning the image from the origin of the image, i.e. from left to right, and from top to bottom of the image.
2.	centroid(objectnum,xcen,ycen). The coordinates (xcen,ycen) is the centroid of the object num.
3.	vertex(objectnum,vn,x,y). Object num has a vertex with vertex number n at (x,y). The value of n starts from 1. If the object has more than one vertex the value of n will be increased accordingly in each vertex statement.
4.	boundbox(objectnum1,x1,y1,x2,y2). Object num has a bounding box with top-left corner at (x1,y1) and bottom-right corner at (x2,y2).

Table 1: Absolute locational primitives

1.	toRight(objectnum1,objectnum2). Object num1 is to the right of the object num2.
2.	toLeft(objectnum1,objectnum2). Object num1 is to the left of the object num2.
3.	atAngle(objectnum1,objectnum2, θ, length). Object num2 is at an angle *theta* measured from object num1 and they are length units apart.
4.	above(objectnum1,objectnum2). Object num1 is above object num2.

Table 2: Relative locational primitives

1.	rawArea(objectnum,area). The area of the object num is area unit square.
2.	majDir(objectnum,mdir). The direction of the major axis of the object num is mdir.
3.	minRad(objectnum,minrad). The minimum radius of the object num is minrad.
4.	maxPer(objectnum,maxp). The maximum perimeter of the object num is maxp.

Table 3: Some quantitative geometric primitives

1.	majVar(objectnum,majvar). The variance along the major axis of the object num is majvar
2.	majSkew(objectnum,majs). The skewness along the major axis (3rd order moment) of the object num is majs.
3.	majKurt(objectnum,majk). The kurtosis along the major axis (4th order moment) of the object num is majk.

Table 4: Some statistical geometric primitives

1.	ratioRad(objectnum,rrad). The ratio of the minimum radius and maximum radius of the object num is rrad.
2.	shrwidth(objectnum,swidth). swidth is the shrink width of the object num (half the number of shrink steps needed for the shape to disappear).
3	circularity(objectnum,circ). circ is a measure of the circularity of the object num.
4.	gyration(objectnum,gyr). gyr is the radius of gyration of the object num.

Table 5: Some compound geometric primitives