VISUAL ROUTINES: INGREDIENTS, DESIGN, AND CONTROL

Marc H.J. Romanycia

Department of Computer Science
The University of British Columbia
Vancouver, B.C., Canada, V6T 1W5

ABSTRACT

The concept of Visual Routine is introduced. A description is given of an implemented computer system which can correctly compute in images of simple 2-D geometric shapes eleven common properties and relations. A visual routine programming language is outlined. Issues relevant to the control of visual-routine-based search are discussed. The results of testing the system are reported.

KEYWORDS: visual routines, visual attention, non-local image properties and relations, computational vision.

1 Introduction

Visual routines are proposed by Ullman (1984) as a means of accounting for how abstract shape properties and spatial relations can be derived from early visual representations. Abstract shape properties like *convex, closed,* and *symmetric*, and abstract spatial relations like *inside, outside*, and *parallel* cannot be efficiently computed with simple local computations. To compute them efficiently we must attend to specific points or regions, and apply operations selectively at these places. Visual routines are sequences of such operations appropriately applied.

Visual routines are ideally suited to deducing the computational restrictions surrounding the task of visual perception. They form a natural language for dissecting visual tasks into their component tasks, and at the same time, they translate naturally into standard computer terminology. This latter feature makes them easy for us to build and work with on conventional computers. It allows us to draw on our large body of programming experience to suggest possible algorithms and approaches, and it allows us to make direct use of existing theoretical results when we analyze the complexity of the visual routine algorithms.

In this paper we describe an implemented computer system which makes concrete Ullman's visual routine proposal. This system computes eleven shape properties and relations: *centred-in, closed, connected, convex, horizontal, inside, outside, parallel, part-of, touching,* and *vertical.* Examples of these routines are given, and the visual routine language in which the routines are written is outlined.

The domain in which the system searches for properties and relations is the 2-D world of simple geometric straight-edged shapes, such as squares, triangles, and line segments.

The visual routines and their control logic are embedded in a Question-and-Answer system which can correctly reply to such queries of an image as "Find all the triangles inside squares" and "Find any three instances of a vertical bar outside a simple closed curve." The Question-and-Answer system serves as a visual routines development testbed. With it we are able to conveniently test the correctness and efficiency of the visual routines and the control logic.

The work described in this paper presents a fourfold contribution to intermediate level Computational Vision:

- It provides a concrete instantiation of Ullman's visual routine proposal.
- It argues for the importance of certain basic operations out of which visual routines can be composed.
- It provides a set of working visual routines for eleven common properties and relations.
- It identifies eight key issues for the control of visual routines.

This work follows the tradition of Marr (1982) in attempting to give a computational account of vision. Thus, our system is not built as an engineering project to suit some practical needs, and, although an effort is made to maintain a correspondence with what is currently known about the human visual system, the system is not presented as a model of the human system.

2 The Visual Routine Framework

Ullman (1984) presented visual routines as a framework for undertaking the study of intermediate level vision. This framework has several components, as illustrated in Figure 1.

Two components of the visual routine framework are the base maps and the parallel methods which create them. The base maps (also called "base representations") are uniform retinotopic maps describing the most basic properties of points or small patches in the image: orientation, colour, intensity, stereo disparity, and motion. These representations are typically derived directly from the image without taking into account any high level information about what may be in the scene. Hence they are created by "bottom-up" processes. This notion of base map feature is closely related to the idea of *preattentive* or "pop-out" feature that perceptual psychologists have developed (Julesz, 1983, 1984; Treisman, 1985, 1986).

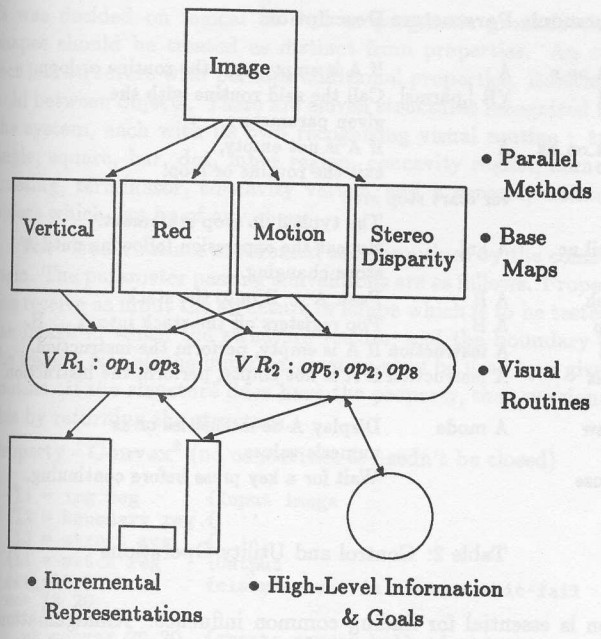


Figure 1: The Visual Routine Framework

A third component of the visual routines framework is the visual routines themselves. These are defined as sequences of elemental operations (also called "basic operations"). Examples of such basic operations are region colouring, curve tracing, shifting the focus of attention, and marking a point in some map.

For their operation visual routines may make use of parallelism. For example, bounded activation (colouring-in a region starting from a seed point and working out to any boundary) is a basic visual routine operation which is likely to be implemented using a form of parallel spreading activation. However, bounded activation is most likely to be initiated from only one or at most a few seed points at any one time. So, visual routines themselves are primarily applied sequentially to at most a few points in an image at any one time.

Although visual routines are ultimately composed of elemental, non-decomposable operations, if we wish, we may define visual routine subroutines and incorporate them in the definition of higher-level routines.

A fourth component is the set of intermediate data structures created by applying visual routines to the base representations. Ullman calls these intermediate maps "incremental representations." They are both output from visual routines and potential input to visual routines. Ullman anticipates that, by assembling sets of routines and/or operations and by using incremental representations, the visual system can compute an unbounded variety of shape-properties and spatial relations.

A fifth component of the visual routine framework is the input of goals and higher level knowledge to influence the routines. That is, top-down information is accommodated as well.

3 A Visual Routine Testbed

The idea of decomposing the computation of visual properties and relations into constituent elemental operations is a good one. Several issues immediately present themselves to anyone attempting such a decomposition. What elemental operations are to form the basis set? What sequences of operations will work? Where in the image and when are the operations to be applied? And, what kinds of structure are to be input and output from these sequences?; that is, what do the base and incremental representations look like? These are the same types of problem faced by designers of new computer systems. These problems arise in the design of programming languages and computer architectures, and in software engineering. Accordingly, we view the problem of devising visual routines as a combination of a system design and a programming problem. The way to approach it is to evaluate different elemental operations, try and test different sequences of these, and to test different architectures and control strategies. We do not see any easy way to deduce an optimal set of visual routines. What is needed are inspired design and test experiments. We need to acquire a body of experience from which sound judgments can be made.

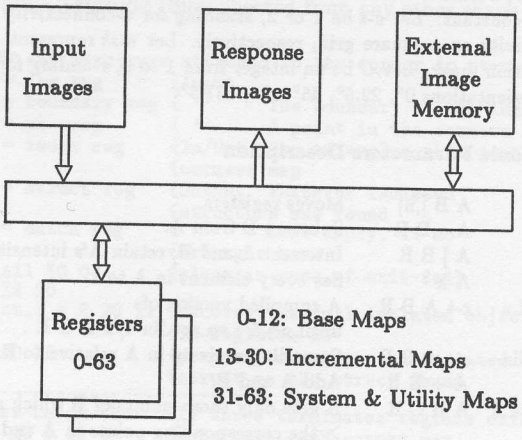


Figure 2: The Testbed Architecture

Accordingly, we have developed a visual routine testbed to facilitate and speed up this design process. The testbed is interactive. The user works at a video display station, entering commands at the keyboard and watching the results on the screen. He can easily create, edit, save, restore, and apply basic operators to images. He can ask questions about the image using a simple query language. He can save intermediate results, edit a visual routine, and then restore the results to test the revised routine. In short, the development of visual routines is greatly facilitated with this testbed.

The testbed architecture is given in Figure 2.

Each map, be it an image, a base representation, or an intermediate representation, is composed of 32x32 pixels using 64 shades of grey. This is not so small as to preclude sufficient detail, and not so large as to impede response time. Maps are treated as units, and can be thought of as registers in a conventional computer, indexed by location in high speed memory. Thus, we can copy maps from register to register, apply an operator to a register, and so on.

4 A Visual Routine Language

The basic operations from which visual routines are composed, as well as the procedural instructions which allow one to string together the basic operations and to otherwise manage their execution, can be thought to comprise a computer programming language. We call such a language a *visual routine language* (VRL).

Let A, B, C, and R stand for $32x32$ image arrays. Let k be an arbitrary integer constant. Let c-t be 1 or 2, standing for 4-connectivity or 8-connectivity on a square grid, respectively. Let *msk* represent a 3x3 convolution mask. Let O be an integer from 1 to 8, standing for each of the orientations 0°, 22.5°, 45°, ..., 157.5°.

Mnemonic	Parameters	Description
mov	A B [,n]	Moves registers.
bop	A – B R	Subtract B from A.
cbop	A] B R	Intersect A and B; retain A's intensities.
set_all	A k	Set every element in A to k.
sp_act_l_a	c-t A B R	A compiled version of until_nc A : sp_act_lin 1/2 A B R .
sp_act_lin	c-t A B R	Spreading activate in A relative to B.
bop	A + B R	Add A and B.
compete_3	A B C R	R gets only those points of B which are ≥ the corresponding points in A and C.
mk_convex	A R	A single step partial fill of any local concavities.
conv3	msk A R O	Convolve with the given mask indexed by the given orientation (if necessary).
oddman_out	A R	Let R have only a single most active point in A.
spread	c-t A R	Fast single width smoothing.
most_actv	A B R	*If* $\sum A > \sum B$ *then* $R \leftarrow A$ *else* $R \leftarrow B$
detotal	c-t A R	Remove all totally surrounded points.
deunit	c-t A R	Remove all isolated points (dots).
unitize	c-t A R	Reduce all connected structures to a single point.
centroid	A R	Compute the centroid of the contents of A.
kop	k + A R	Add k to each element of A.
kop	k < A R	Raise points < k to k.
kop	k > A R	Lower points > k to k.

Table 1: Image Manipulation Operations

4.1 The Basic Operations

Our VRL resembles an assembly language in that each program line contains an instruction mnemonic and assorted register numbers and/or parameters. It also possesses some high-level language features such as looping and begin-end block constructions. All the basic operations are listed in Tables 1 and 2. A visual routine is simply a sequence of such basic operations. The operations in each Table are sorted by decreasing frequency of occurrence in the 36 active visual routines used by the system. Romanycia (1987) gives detailed definitions of all the operators.

The image manipulation operations are all parallelizable and would benefit from implementation on a parallel machine, such as the connection machine (Hillis, 1985; Little, 1986).

In our visual routines the most frequently used operations are these: moving registers, spreading activation, intersecting registers(cbop]), amalgamating registers(bop +), clearing registers, subtracting out parts from a register(bop –), and exiting a routine on some condition. We interpret the popularity of these operations to be a sign that they are intrinsically important.[1]

It is easy to see why these particular operations may be important. Data have to flow from stage to stage; hence moving maps is likely to be important. Spreading activation reflects the basic need to follow paths and respect boundaries. Intersection is essential for finding common influence. Amalgamation is needed for comparison and relating to take place. Clearing maps is important for separating tasks so that a former result does not confuse a new and unrelated assignment. Removing parts is useful in making it easier for us to focus attention on something else. And, conditional quitting is necessary to save ourselves wasted effort whenever it becomes clear that there is no point in continuing a routine.

[1] We must of course be cautious in drawing such a conclusion because it is not easy to separate out which operations are essential to a task and which are accidental features of a programmer's style or of early system design decisions. It would help to answer the question of which operations are truly important if other researchers were to build systems with their own brands of basic operations and then present them for comparison.

Mnemonic	Parameters	Description
exit_on_z	A	If A is empty, exit the routine or loop.
call	VR [,parms]	Call the said routine with the given parameters.
exit_on_nz	A	If A is not empty, exit the routine or loop.
do	var start stop step	The typical do loop statement.
until_nc	A [,n]	Repeat the expression following until A stops changing.
push	A B	Push A ... B onto the stack.
pop	A B	Pop registers off the stack into A ... B.
if_z	A instruction	If A is empty, perform the instruction.
if_nz	A instruction	If A is not empty, perform the instruction.
draw	A mode	Display A as intensities or as numeric values.
pause		Wait for a key press before continuing.

Table 2: Control and Utility Operations

It is instructive to compare our list of basic operations with Ullman's. Ullman (1984) proposed five basic operations: shifting the processing focus, indexing to a point of interest, bounded activation (region colouring), boundary tracing and activation, and marking features in maps. The two operations we more or less have in common are spreading activation and indexing to the odd-man-out.[2] Ullman makes no mention of such mundane operations as moving maps or clearing them. He also makes no mention of less mundane operations like intersecting or amalgamating maps. This may be a mistake on his part. The point of taking a computational view is to make clear the *exact* nature of the computations involved. We should therefore specify every detail of a visual routine's computation. If two features are to be compared, then we will need operations for bringing them together. If a new map is to record markings, then it must first be cleared of previous markings. And so on. If we are someday in position to test whether visual routines operate in the human visual system, then we will want to know exactly what basic operations we are trying to correlate with brain processes. Since the more complex operations may rely on simpler ones, it will be very valuable to know what these simpler operations might be.

Let us now see how these basic operations are actually assembled to perform interesting tasks.

4.2 The Visual Routines

There are 36 active visual routines. Space does not permit us to describe all of them here. (All are given in Romanycia, 1987.) We will give three routines: one property, one relation, and one structure routine. Structures are defined as features or shapes.

[2] Our single form of spreading activation can be applied to curves as well as regions. In conjunction with a few other basic operations, we are thus able to simulate Ullman's bounded activation and boundary tracing operations.

It was decided on logical as well as pragmatic grounds that shapes should be treated as distinct from properties. An object is a structure with perhaps additional properties. Relations hold between objects. There are eleven structures recognized by the system, each with its own recognizing visual routine : triangle, square, bar, dot, inner region, concavity region, corner, crossing, terminator, concavity vertex, and a general, isolated object which can be of any shape.

The visual routines are treated as subroutines by the control logic. The parameter passing conventions are as follows. Properties receive as input the structure or shape which is to be tested, the image in which the structure resides, and the boundary of the image. Not all the input parameters need be used by a given routine. If the structure does have the property, then we signal this by returning the structure.

Property - **Convex**[3] (no concavities & needn't be closed)

```
{%1 = img reg       {Input image
{%2 = boundary reg  {  ''
{%3 = struct reg    {  ''
{%4 = match reg     {Output
set_all %4 0        {clear answer in case of exit-fail
mov %3 20
until_nc 20
  mk_convex 20 20   {create convex hull of object
detotal 1 20 21     {remove interior of hull
bop 21 - %3 22      {remove object, leaves edge of
                    {hull in concavity
exit_on_nz 22       {exit if such an edge exists
mov %3 %4           {return the object in %4 to
                    {indicate success
```

Relations are similar to properties, except that two structures are passed in a specific order. If the two structures in that order bear the relation to one another, then we signal this to the control logic by returning the union of the two structures.

Relation - **Inside** ("entirely within the convex hull of")

```
{%1 = img reg       {Input
{%2 = struct1 reg   {  ''
{%3 = struct2 reg   {  ''
{%4 = union reg     {  ''
{%5 = match reg     {Output
set_all %5 0        {clear answer in case of exit-fail
mov  %3 29          {store struct2 in 29
until_nc 29
  mk_convex 29 29   {make struct2 into a convex blob
cbop 29 [ %2 28     {intersect with struct1
exit_on_z 28        {if nothing in common, then
                    {not inside
bop %2 - 28 27      {o.w., compare the intersection
                    {with the original
exit_on_nz 27       {if any dif, then st1 is not
                    {inside convex st2
mov %4 %5           {o.w. return the union as
                    {the answer
```

The visual routines for properties and relations are simply asked to return a yes or no reply for the objects they are passed. In the case of a structure, a visual routine's task is to determine whether that structure can be found somewhere attached to a given index point. To this end, the routine needs the image in which to search for the structure, and it may need the image's boundary and the register in which we were indexing. If it finds an instance of the structure attached to the point, it returns this structure. Also, regardless of whether or not it finds the desired structure, it endeavors to return a structure attached to the given point.

[3]The reader may disagree with an interpretation that is given here to a particular property, relation, or structure. There are many definitions of 'inside', 'outside', 'touching', and so on. In order to keep things manageable, one popular definition was chosen to represent each term, and a clue to the specific meaning is given in parentheses. One difficult problem for Visual Routine Science will be accounting for the many subtle and context-dependent shades of meaning that visual properties and relations can have.

Structure - **Triangle** (disconnected from any other shape)

```
{FIND any Structure attached to the corner in pt-reg
{%1 = img reg       {Input
{%2 = boundary reg  {  ''     The boundary of the image
{%3 = pt  reg       {  ''     A point in the corners map
{%4 = index reg     {In/Out   For triangles this is the
                    {corners map
{%5 = struct reg    {Output   Whatever isolated
                    {structure was found
{%6 = match reg     {  ''     Left empty, if no
                    {triangle here
set_all %6 0        {clear in case of exit-fail
mov %3 20
sp_act_1_a 2 20 %1  {recover the full isolated object
                    {in register 20
mov 20 %5           {prepare to return the isolated
                    {struct as the struct found
{--------------------------
{First check that only three terminator regions exist
cbop 20 ] 11 21     {intersect with corner map
unitize 2 21 21     {make each term rgn a single point
oddman_out 21 22    {pick one corner
bop 21 - 22 21      {remove it
oddman_out 21 22    {pick another
bop 21 - 22 21      {remove it
exit_on_z 21        {exit, if there are only 2 corners
oddman_out 21 22    {pick another
bop 21 - 22 21      {remove it
exit_on_nz 21       {exit, if there are > 3 corners
{--------------------------
mov 20 27
call closed %1 %2 27 28 {verify that the object is
                        {simple closed.
exit_on_z 28        {if nothing returned,
                    {then obj is not closed
{--------------------------
mov 28 %6           {return the triangle
```

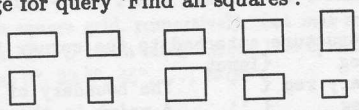
Our application of visual routines to representing shapes is unorthodox. One novel aspect of our work is the experimentation with visual-routine-based object representation. For simple geometric shapes defined in terms of simple properties and relations between their parts, it is feasible to define the shapes procedurally as the set of operations used to confirm their presence. For example, a parallelogram is hard to define prototypically, but it is relatively easy to define in terms of four corners, its being a simple closed curve, and its having two instances of parallelism. Likewise, there are many triangles, but all have three corners and are simple closed curves. So, for such cases as these, no matter what object representation we use, it is likely to have "hooks" into the visual routine calling mechanism. By doing visual-routine-based object recognition one can explore the interface between object representations and visual routines.

5 Attention and the Control of Visual Routines

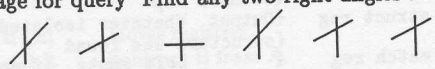
The psychological literature on the subject of visual search and attention is vast. Hurlbert and Poggio (1985, 1986) review from the computational perspective some of the recent psychophysical and physiological work in this area. Their review suggests that the role of attention in human vision is not well understood although there are many interesting theories. Our work does not present a theory of visual attention. Our contribution to attention research is the discovering of eight issues which we believe are intrinsic to the problem of applying visual routines while searching images. We have also contributed several solutions to these problems. These problems are as follows.

1. The problem of deciding which search strategy to use when searching for an object – We could search randomly; or we could search with guidance from heuristics, such as the proximity and similarity heuristics mentioned by Koch and Ullman (1984); or

a. Raster scan search is suited to the following image for query 'Find all squares':



b. Proximity search is suited to the following image for query 'Find any two right angles':



c. Random search is suited to the following image for query 'Find any one square':

Figure 3: Situations warranting different search strategies

For the query 'Find all vertical bars' we are better off searching the vertical map over the terminator map.

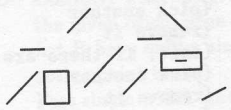


Figure 4: Example illustrating the value in choosing the right map to search

we could search by following a specific search pattern, such as a raster scan pattern or a pattern that spirals out from the centre. Figure 3 illustrates situations where different search strategies would be advantageous. Raster scan search suits Figure 3.a because we want to find *all* instances of an object in an image of evenly spaced shapes. Proximity search suits Figure 3.b because right angles occur back-to-back in images of crossing lines. And, random search suits Figure 3.c because we only have to find one instance of an object in an image where the objects are plentiful.

The problem of deciding on a search strategy may be faced repeatedly during a search and not just upon initiating the search. Portions of an image may benefit from different strategies. Also, we may want to change strategy mid search if we find we are making no progress.

2. The problem of spotlight diameter – The reason why we need to focus attention at all is that we cannot afford to apply visual routines uniformly across an image. Even when we can do this, we may want to restrict the area of application in order to reduce the time taken to complete an operation. For example, operators like spreading activation work more quickly on smaller areas. On the other hand, the larger the area processed is, the fewer shifts of the spotlight may be needed to get the work done. So the problem becomes this: for a given situation, what is the optimum size area within which we can apply a given routine?

3. The problem of which feature map we should be indexing within – For example, in the query of Figure 4 we would be best off searching the vertical bar map rather than the terminator map (which is associated with bars). Ullman (1984) also mentions this problem.

4. The problem of different search formats when trying to find a single object as compared to finding a pair of related objects – This problem arises because relations and objects are intrinsically different sorts of thing. Finding an instance of a relation requires finding objects first; whereas, in the 2-D geometry world at least, objects have existence independent of any relations they enter into.

a. For the query 'Find all triangles outside squares,' we may want to find triangles and squares first before computing the outside relation.
b. For the query 'Find all right angle triangles inside vertically oriented squares,' we may want to compute the inside relation first.

Figure 5: The advantages of different task schedules

If we are searching for isolated triangles and we find p1 is not attached to one, then we may legitimately remove p2, p3, and p4 from the index map we are searching.

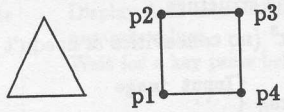


Figure 6: The removal of unneeded points from an index map

5. The problem of task scheduling – For example, when trying to find a relation, should we first look for the objects, verify their properties, and then see if they are related? Or, should we first look for general objects, then confirm the relation between or among them, and finally confirm that these general objects indeed have all the desired properties? The answer will depend on the relative costs of computing the objects and the relation. Since we want to minimize wasted effort, we should perform the least expensive computations first. In Figure 5 we see examples of where it may be advantageous to use each strategy. The scheduling problem also arises when trying to find an object with several properties. In what order should we test for the properties? The task scheduling problem is addressed by Ullman with his suggestion of "skeletal guidelines" (Ullman, 1984).

6. The problem of what to remove from a map in which we are indexing, after we have found or failed to find something at the indexed point – Provided that we are not working with a reflexive relation, we can certainly remove the indexed point; but if we are clever, we can also remove some nearby points that are no longer worth indexing to. For example, in Figure 6, when we find there is no triangle attached to point p1, we can remove points p2, p3, and p4 as well.

7. The problem of duplicate examinations – No matter how carefully one trims the search maps after each search step, situations can still arise where the search sequence brings you back to a previously examined object. In the case of relations, it is common that after having considered R(obj1,obj2), one will then encounter R(obj2,obj1). If R is symmetric, this is a waste of time. In order to avoid duplicating effort, one must somehow remember those portions of the image one has previously examined.

8. The problems of what to do when a search pass fails – Should we just quit, try a new search strategy, use a new resolution scale, rotate the image, lengthen lines, or otherwise modify the image?

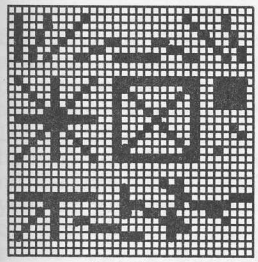
Romanycia (1987) gives details on how each of these problems is handled in the implementation.

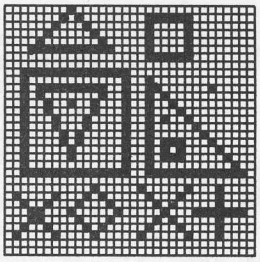
6 System Performance

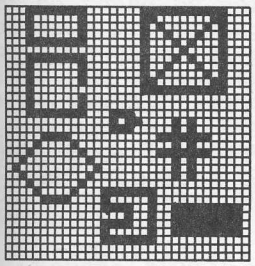
The system consists of about 4000 lines of PASCAL and runs on an IBM pc/AT clone. Run times for queries range from a few

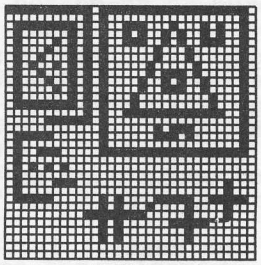
seconds to a few minutes. The system has undergone extensive testing. Here we summarize the test results.

Numerous images were employed in the course of development. Examples are as follows:


Test Image 1


Test Image 2


Test Image 3


Test Image 4

For test image 1, examples of the computed base maps are as follows:

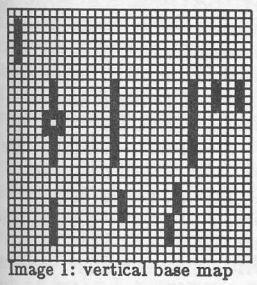


Image 1: vertical base map

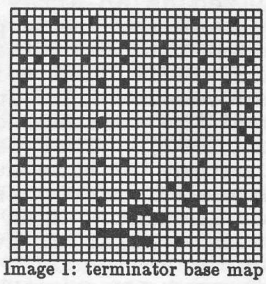


Image 1: terminator base map

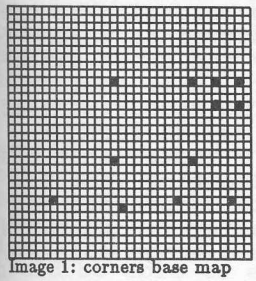
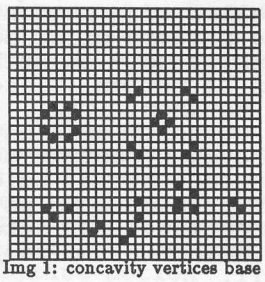


Image 1: corners base map


Img 1: concavity vertices base map

The purpose of the base map algorithms (see Romanycia (1987) for their specification) is to give us sufficiently accurate base maps with which we can carry on the business of testing visual routines. To this end the base map computations work well enough, even though they do make the odd error.

The system answers **all** queries correctly, with the exception of a few cases where errors in the base maps provide faulty input. Sample queries range from the basic (e.g., "Find all squares") to the challenging (e.g., "Find all vertical bars connected to vertical bars" and "Find all convex inner-regions inside closed shapes").

The control logic successfully manages the search. The implemented solutions to the control issues mentioned in the previous section substantially improve the search performance. Still, the search is inefficient at times. There remain situations where the system reevaluates the properties and structures of regions it has already examined. We could improve the control logic by first having the system create a high-level semantic map of what objects are in the image and then conducting the search in this much simpler map rather than in the image.

7 Conclusion

We outlined a working visual-routine-based question-and-answer system. A visual routine language was described and sample routines written in that language were given. Arguments were made for the importance of certain basic operations in that language. Eight issues intrinsic to the control of visual-routine-based search were explained. The implemented system was thoroughly tested on images of simple 2-D geometric shapes and, except for small errors introduced by the base map computations, it worked flawlessly and with considerable intelligence.

The discovery of the basic operations, the visual routines, the control logic, and the control issues constitute the principle contribution of this work to vision science.

References

[1] Hillis, W. Daniel, *The Connection Machine*, Cambridge, MA: The MIT Press, 1985.

[2] Hurlbert, Anya, and Poggio, Tomaso, "Spotlight on Attention", MIT AI memo #817, April, 1985.

[3] —— , "Visual Attention in Brains and Computers", *Nature*, Vol. 321, #12, June 1986, pp.651-652.

[4] Julesz, Bela, and Bergen, J.R., "Textons, The Fundamental Elements in Preattentive Vision and Perception of Textures", in *The Bell System Technical Journal*, Vol. 62, No. 6, July-August 1983, pp. 1619-45.

[5] Julesz, Bela, "Toward an Axiomatic Theory of Preattentive Vision", in *Dynamic Aspects of Neocortical Function*, ed. by G.E. Edelman, W.E. Gall, and W.M. Cowan, 1984, pp.585-612.

[6] Koch, Christof, and Ullman, Shimon, "Selecting One Among the Many: A Simple Network Implementing Shifts in Selective Visual Attention", M.I.T. A.I. Memo 770, 1984.

[7] Little, James J., "Parallel Algorithms for Computer Vision", M.I.T. A.I. Memo 928, November, 1986.

[8] Marr, David, *Vision*, San Francisco, CA: W.H. Freeman & Co., 1982.

[9] Romanycia, Marc H.J., "The Design and Control of Visual Routines for the Computation of Simple Geometric Properties and Relations", M.Sc. thesis, available as technical report 87-34 from The University of British Columbia, Department of Computer Science; October, 1987.

[10] Treisman, Anne, "Preattentive Processing in Vision", in *Computer Vision, Graphics, and Image Processing*, Vol. 31, 1985, pp.156-177.

[11] ——, "Features and Objects in Visual Processing", in *Scientific American*, November 1986, pp.114B-125.

[12] Ullman, Shimon, "Visual Routines", in *Cognition*, Vol. 18, 1984, pp. 97-159.