Abstract:

An "Observer" of natural scenes is described. The domain of the Observer includes single, color, monocular views of outdoor scenes. This domain is further restricted to exclude man-made objects such as houses and roads. The Observer may be divided into a preprocessing perceptual component and a more cognitive recognition component. The perceptual component is based upon primitive region-growing in an opponent-colors space. The recognition component uses a production system which allows for the partial match of rule patterns and conflict resolution based on the information content of the matched patterns.

The preparation of this paper was supported in part by the Defense Advanced Research Projects Agency, monitored by ONR under Contract Number N00014-78-C-0164.
1. Introduction

Computer vision systems usually consist of two major sub-systems: low-level operators which provide sensory information and other measurements, and a higher-level program which provides the strategy of recognition. The low-level operators (filters, edge finders, texture operators) are properly represented as subroutines, available when needed by the higher-level program. In general, they do not change with time and may be coded once and used without modification. The higher-level program, however, is at once much more complicated and much more subject to change than the low-level operators. It has been recognized that this subsystem should be based upon a World Model (e.g., Tidhar, 1974; Lieberman, 1974), but procedures for creating the higher-level program from the World Model have not been made clear.

The main goal of this research was to build a computer vision system in which the higher-level recognition strategy followed more or less directly, and automatically from the data and the World Model.

1.1 Domain

The scenes in our world are medium to long range scenics. The point of view of the Observer is assumed to be from above, looking in a direction parallel to the ground, or slightly downward. Thus, the Observer is assumed to be "properly" oriented with respect to the ground.
Therefore, the scenes can be roughly divided into two portions, "sky" and "ground", separated by a "horizon". An example of such a scene is shown in Figure 2.

The sky is assumed to be at the top of the picture and is required to be in the scene. The sky may be clear (blue), or cloudy (white), or there may be individual clouds in the sky. Trees and mountains are allowed to extend into the sky but must be connected to the ground. All other objects found in the sky are considered unidentifiable. The description of the sky is limited to a characterization of the weather (clear, overcast, cloudy).

The area of the picture identified as ground is the most interesting and complex area in these scenes. The objects to be considered here are slopes, hills, mountains, peaks, ridges, trees, bushes, lakes, islands, rocks, and snow patches.

1.2 Methods

The overall block diagram of our approach is shown in Figure 1.

The input data are digitized color photographs of outdoor scenes. The spatial resolution that we are working with is rather coarse (120 x 128 pixels).

The pictures are segmented into regions based on color separation. The color model used in this work follows the opponent theory (Hurvich and Jameson, 1957; Hurvich, Jameson and Krantz, 1965), where instead of having primary colors blue, green, red, we have a 3-dimensional color space with opponent colors, red-green
(R-G), blue-yellow (B-Y), white-black (W-B). We have chosen this model primarily for its convenience in naming colors (Miller and Johnson-Laird, 1977; Sloan and Bajcsy, 1975). Each region is described by its color values in the R-G, B-Y, W-B space, crude shape and location descriptors (the [TOP, LEFT, BOTTOM, RIGHT] furthest pixels belonging to the region, the size of the region, and the (X,Y) coordinate of the seed point of the region. More detailed shape descriptors are calculated for some regions using a method similar to that described in (Maruyama, 1972). The texture measure implemented in this work uses the borders of the regions found by the region grower as an indication of the "busyness" of an area (see Figure 3). This approach is similar to one reported by Rosenfeld (1975).

The World Model and the control structure are implemented using a production rule system which allows for the partial match of rule patterns and conflict resolution based on the information content of the matched patterns (Rosenschein and Joshi, 1977).

2. **The World Model**

First, we shall briefly describe a particular Production System, then we will present the representation of the knowledge of our Observer in this particular framework. We have chosen a production system for the representation of knowledge and high level control because it offers the flexibility needed in representing the complex knowledge used during recognition of outdoor scenes. Similar methodology is under development in the context of outdoor scenes by Bullock (1978).
2.1 Production Systems

Production Systems are a general class of computation mechanisms. They have been applied to many different problems, with many variations in implementation detail and style. For a good overview of the unifying themes involved in these variations on the Production System idea, see (Davis and King, 1975).

A Production System is characterized by the existence of three basic components: a Data Base, a set of rewrite Rules, and an Interpreter which applies the rewrite Rules to effect changes in the Data Base.

Here we describe a particular Production System (Rosenschein, 1976; Joshi and Rosenschein, 1977). The key difference between this system and other Production Systems is in the performance of the Interpreter. Rules may be chosen on the basis of partial matches and conflict resolution is done by comparing the information content of the Rules' Left Hand Side.

The Data Base for this system consists of an unordered set of assertions, called FACTS. FACTS are assertions about the world. Such assertions may also be stored in a long term memory (LTM) which is not immediately accessible to the Interpreter.

Rules are of the form (LHS, RHS), where LHS stands for left hand side and RHS stands for right hand side. Both the LHS and RHS are either constant or variable assertions (Forms). In our extended system there are two special Forms; (MUST <x>) and (EXEC <x>). The FORM (MUST <x>) may appear in the LHS and means that the Form <x> is
absolutely essential to the success of the matching of the LHS with FACTS. (EXEC <x>) may appear in the RHS, and it specifies that the Form <x> is to be executed for effect in LISP when the Rule is invoked.

The Interpreter seeks to find a minimal set of Rules which cover the assertions in FACTS. Rules are partially ordered (implicitly and dynamically) according to specificity.

Often there does not exist a single minimal Rule which covers all of FACTS (note: especially when FACTS contains everything currently known by the system). This is a case of "multiple Rule response". The Interpreter selects a set of Rules which together cover as much as possible of the FACTS.

Another problem arises when two (or more) incomparable Rules cover the same subset of FACTS. This is known as "ambiguous response". The current Interpreter resolves this conflict by randomly selecting one of these Rules for inclusion in the set of Rules to invoke at this iteration. We do not attempt to try all of the possibilities (e.g. by backtracking). This does not seem to cause any harm. After all, we are not interested in proving theorems about the scene; rather, we are interested in a first-order, reasonable impression of the scene. If errors are introduced by making arbitrary choices (where only arbitrary choices are possible), we prefer to wait for additional information (from the next scene, in time, say) rather than to puzzle out a definitive answer.
2.2 Representation of Knowledge of an Observer

Now that we have described the general framework, we can proceed to describe how we use this framework for representing knowledge which is available to an Observer during the process of recognizing outdoor scenes. This knowledge may be divided into two general classes: particular knowledge about the specific scene under analysis and general knowledge about the world. The latter type of knowledge includes information about the structure of the world, the expectations of the Observer, and the Observer's own perceptual capabilities. The former, on the other hand, includes information on what is happening now; what the Observer is seeing in the current scene.

These two classes of knowledge are used in different ways by the Observer. In fact, we may classify the knowledge according to its use as well as its content. The scene specific knowledge is used to direct the acquisition and use of the data; it is action oriented.

What follows will be a more detailed description of the scene specific knowledge and the Observer's general knowledge about the world.

2.2.1 Scene Specific Knowledge - The Data Base

Knowledge about the specific scene under analysis is stored in the form of assertions about the scene. These assertions are stored in the Data Base (i.e. FACTS and LTM).
2.2.2 Assimilated Knowledge - The Rules

Information about the structure of the world, the perceptual capabilities of the Observer, and the possible usage of low-level vision primitives is represented in RULES. This is information which has been assimilated by the Observer (in our case by direct instruction, i.e. manual construction of the RULES) and which is useful for the interpretation of the assertions in FACTS and LTM. Such information does not change with the scene, it is the accumulated knowledge about the world. This includes: conventional names for colors, expected locations of objects in the scene, and rules of inference for dealing with depth cues. For example, one of the Rules for color naming is:

\[(\text{R-G} \ ?X \ \text{PLUS})\]
\[(\text{Y-B} \ ?X \ \text{PLUS}) => (\text{COLOR} \ ?X \ \text{ORANGE})\]

Informally this rule says: If a region is more red than green, and more yellow than blue, then call it ORANGE. As another example, we present a rule for identifying sky:

\[(\text{MUST} (\text{VERY-HIGH} \ ?\text{REGION}))\]
\[(\text{MUST} (\text{W-B} \ ?\text{REGION PLUS})) => (\text{SKY} \ ?\text{REGION})\]

More informally: A region will qualify as a piece of sky if it is very high in the scene and it is also bright. Other rules are
similar; for full details see (Sloan, 1977).

3. Attention Sequencing - The Control Structure

An important part of the specification of the Observer model is a description of the manner in which its attention is directed to particular facets of the Scene Analysis problem.

This and the content of Memory (context) will determine the control structure of the recognition process.

In general, our Observer's attention is drawn initially to large regions in the scene. Smaller details are attended to only after the surrounding, larger regions have been used to establish a context.

Therefore, the outermost control structure (representing the action corresponding to no knowledge in FACTS) consists of the sequential consideration of regions ordered by size.

Once we have examined the first (largest) region, we have access to FACTS which may be sufficient to establish a context for more detailed investigation. For our Observer, establishing a new context involves two focusing operations. First, attention is focused on a particular area of the image (IMAGE focusing). Second, the FACTS already known about this area of the image are used to construct a new set of RULES (MODEL focusing). These new RULES are, in general, only applicable in this small part of this image. IMAGE FOCUSING is accomplished by abandoning the consideration of regions based on SIZE, and instead examining those regions very close to the region which established the context. Only these two iterators are used: a)
regions by size; and b) regions by location.

The Observer is allowed to use this mechanism to investigate (to whatever depth is required) a particular aspect of the current context before returning to consider the remaining information available at this level. While this investigation is taking place, previous versions of FACTS are not available and they do not affect the computation. However, the LTM is available to all levels and LTM is never copied or stacked. In this way it is easy to implement a depth first investigation of the scene with information gathered on one branch of the tree available for use elsewhere. However, we do not have a full-fledged "alternate context" mechanism. The Observer does not backtrack or consider hypothetical cases. An assertion placed in LTM will stay there and it will affect future processing. Of course, it may later be contradicted or access to it may be blocked by other assertions, but the fact that the assertion was thought to be true at some point in the analysis will be reflected in LTM.

Thus, our Observer has two distinct modes of thought: iteration (through lists of regions) and recursion. The third alternative, i.e. selection, is provided by the Production System.

The question is not whether the Observer has enough computational power, rather it is whether the use of this power is sufficiently disciplined to yield a structured model which can be understood and incrementally developed as well as be effective. Our solution is to severely restrict the use of these three types of control structure. That is, arbitrary recursion is discouraged except where it is used to introduce another context; iteration is used to step through lists of
regions (either the entire set or a smaller set limited by a property such as being the neighbor of a particular region); all selection is done by the Production System.

4. Implementation

The overall system is presented in Figure 1. While we have already explained its function, here we shall describe more detailed implementation considerations.

4.1 Memory

The structure of memory and the ways in which it can be accessed impose very real constraints on the performance of a system such as ours. For that reason, we feel that it is necessary to discuss these topics in some detail.

FACTS and the Long Term Memory (LTM) are both lists of assertions. While FACTS are unordered assertions, the LTM is structured so that related assertions may be retrieved together. This is implemented via TAGS and clusters.

A TAG is the name of a concept which is common to every assertion in a cluster. For example, consider a cluster with the TAG "BLUE". Each assertion in this cluster would be about some thing which has been asserted to be BLUE (e.g., (COLOR SKY BLUE), (COLOR (REGION 5) BLUE). Thus, the BLUE cluster would look like: "(BLUE ((COLOR SKY BLUE) (COLOR (REGION 5) BLUE) (COLOR (REGION 73) BLUE)...))". Note that each element of LTM is a list with two elements: the TAG and a
list of assertions.

Assertions are added to LTM by a LISP function REMEMBER (e.g., (REMEMBER TAG ASSERTION)). This function adds the ASSERTION to the cluster associated with TAG, creating the cluster when necessary. When the assertion is added to LTM two ordering operations are done. First, the cluster is treated as a stack, with new assertions being added at the front of the list. If the assertion is already in the cluster then the cluster remains unchanged. No checks are made for inconsistency or even outright contradictions. Second, the cluster itself is repositioned to the front of the list LTM. This provides for fast access to recent assertions and to assertions about concepts of current interest.

In general, each assertion in LTM is stored in several clusters, one for each concept in the assertion. A LISP function REMEMBER-ALL (e.g., (REMEMBER-ALL ASSERTION)) is provided which generates the appropriate TAGs from the assertion and stores the assertion under these TAGs.

Two LISP functions, RECALL and EXTRACT, are used to retrieve information from LTM (e.g., (RECALL TAG) or (EXTRACT TAG PREDICATE). RECALL finds the cluster named by TAG and puts all the assertions into FACTS. Thus, RECALL is used to acquire all the information in LTM about a particular concept. EXTRACT is more specific. EXTRACT returns a single assertion as a value. The assertion is found by searching the cluster names by TAG for an assertion which has PREDICATE as its first element. This function is used to check for specific knowledge. As a side effect, EXTRACT reorders the cluster of
assertions so that the returned assertion is at the front of the list, in addition to reordering LTM. If several such assertions are in the cluster, only the first one found is moved to the front of the cluster and returned.

Another function, KNOWN, is used to check for the presence of a particular assertion in LTM. KNOWN returns T if the assertion is in LTM and Nil otherwise. It depends on the fact that all assertions in LTM are in a cluster with a TAG identical to the first element in the assertion. As a side effect, LTM is reordered, with this cluster at the front of the list.

4.2 Rule Focusing

Just as we found it necessary to structure the Data Base, we also decided that the set of Rules needed to be partitioned. The need to structure RULES stems from two root causes. First, the computational complexity of the matching process requires that RULES be small in order to achieve reasonable response from the Interpreter. Second, not all rules are applicable in all situations. Certain contexts require the use of rules which are not relevant (or even correct) in other contexts. For a discussion of similar problems with a somewhat different emphasis, see the work on "Frames" (Minsky, 1975; Kuipers, 1975; Winograd, 1975). Although there are various ways to circumvent the problem, we decided to exercise direct control over the contents of RULES. EXEC clauses in some rules are used to invoke LISP functions which change RULES. For example, there is in our World Model a function called "GROUND-RULES" which activates (places in
RULES) those Rules which have meaning in the context of "GROUND".

There are distinct advantages and disadvantages to this approach. First, it is very handy to be able to invoke an arbitrary LISP function to change the RULES. RULES can be constructed from current knowledge, rather than simply selected from a pre-determined set. The construction of such RULES may be an arbitrarily complex task; this is a mixed blessing. There is always the temptation to solve hard problems by writing RULE-constructors which make subsequent analysis trivial.

4.3 Image Processing Interface

The digitized picture has been pre-processed and is represented by a region MAP and a vector of region descriptors. This data structure was produced and may be manipulated, by a set of FORTRAN programs. At the UNIVAC 90/70 at the University of Pennsylvania there is an interface between LISP and FORTRAN which allows for the transfer of information between FORTRAN and LISP programs. In our view, such interfaces are more promising than the search for a "universal" language.

The information about a region is translated from the FORTRAN data structure to assertions about the region by the LISP function FIRST-LOOK. FIRST-LOOK creates assertions such as:

(SIZE (REGION 5) 1200)
(R-G (REGION 5) MINUS)
(Y-B (REGION 5) PLUS)
(W-B (REGION 5) ZERO).
The attributes MINUS, PLUS, and ZERO are provided by a LISP function QUANTIZE, which divides the range of absolute intensity into rough qualitative judgements. This is in keeping with the guiding principle that what we want is judgement and impression, rather than calculation. FIRST-LOOK is not called directly by the World Model. Rather it is invoked by RECALL when that function is called upon to provide assertions about a region and none exist in LTM. As its name implies, FIRST-LOOK is called only once for a specific region. After that, all the information about that region has been provided to the Observer and must be preserved in some way. The Observer may choose to ignore some information, synthesize some of it into a more concise description, or simply store some of it in LTM. First though, RECALL places all of the information into FACTS where it is used to select Rules for invocation.

Other functions which directly use the information in the FORTRAN data structure (i.e. the low-level operators) are implemented in a similar fashion. There is a low bandwidth, descriptive interface between the FORTRAN image processor and the LISP analyzer.

5. A Concrete Example

This section is a commented condensation of a small portion of the output generated during the processing of the SLOPE1 scene (Figure 2). The reason for including the reports of the Observer during its analysis is to show the reader the actual performance. Especially we like to note how the recognition strategy is data and context driven.

*(look 100000 100 nil)
This is the invocation of the analysis phase. It directs the Observer to investigate all regions which fall in the size range 100-100000, in decreasing order of size. The regions are displayed in Figure 3. The LTM is initialized to NIL.

```
our first look at (region 11)
x y r-g y-b w-b size top bottom left right
35 2 24 29 6 2 2132 35 97 2 127
```

This report is produced by FIRST-LOOK, which will take the information in the FORTRAN data structure displayed here and produce assertions about (REGION 11). This region is shown highlighted in Figure 4.

```
--------Progress Report--------
time spent thinking: (0 195)
regions on this branch:
(11)
context stack:
nil
contents of short term memory:
((far-left (region 11)) (far-right (region 11))
(right (region 11) 127) (left (region 11) 2)
(bottom (region 11) 97) (top (region 11) 35)
(w-b (region 11) minus) (y-b (region 11) zero)
(r-g (region 11) zero) (size (region 11) 2132))
--------end of progress report--------
```

The display above is generated once after every iteration of the Production System Executive. We see here that (REGION 11) is being investigated; there is no known context for this investigation; the information about (REGION 11) created by the sensory apparatus has been placed in FACTS. The stack of region numbers is used to terminate looping; the context stack is for information only, and is not referenced by the analysis procedure.
Above, we see assertions moving from FACTS to LTM. They are reported because this is the first time they have been REMEMBERed.

-----progress report-----
time spent thinking: (9 124)
regions on this branch:
(11)  
context stack:
nil  
contents of short term memory:
«color (region 11) black»

-----end of progress report-----

The assertions created from the region data structure have been digested, and lead only to the conclusion that (REGION 11) is BLACK.

-----progress report-----
time spent thinking: (1 506)
regions on this branch:
(11)  
context stack:
nil  
contents of short term memory:
«ground (region 11)» «shadow (region 11)»

-----end of progress report-----

Things which are black are GROUND and SHADOW. These Facts are again stored in the LTM.

Having discovered a piece of ground, the Observer has activated the GROUND-RULES, and changed context. We now investigate the neighbors of (REGION 11).
our first look at (region 16)

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>r-g</th>
<th>y-b</th>
<th>w-b</th>
<th>size</th>
<th>top</th>
<th>bottom</th>
<th>left</th>
<th>right</th>
</tr>
</thead>
<tbody>
<tr>
<td>58</td>
<td>2</td>
<td>23</td>
<td>30</td>
<td>3</td>
<td>1833</td>
<td>57</td>
<td>119</td>
<td>2</td>
<td>127</td>
</tr>
</tbody>
</table>

(REGION 16) is a neighbor of (REGION 11), and we are trying to determine whether or not they are sufficiently similar, in both color and texture, to justify merging them.

------progress report------

time spent thinking: (15 480)
regions on this branch:
(16 11)
context stack:
(ground)
contents of short term memory:

[((texture-difference (region 16) (region 11))
(color-similar (region 16) (region 11))
(distance (region 16) near) (ground (region 16))
(color (region 16 black))

------end of progress report------

The Observer decides that (REGION 16) is ground because it is at the bottom of the picture.

The ground-growing process continues, until finally one of the neighbors of a ground region is a piece of sky. Our Observer will not immediately recognize this region as sky, but will see that a depth discontinuity exists and that the border between these two regions represents a section of 3-D skyline.

our first look at (region 8)

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>r-g</th>
<th>y-b</th>
<th>w-b</th>
<th>size</th>
<th>top</th>
<th>bottom</th>
<th>left</th>
<th>right</th>
</tr>
</thead>
<tbody>
<tr>
<td>27</td>
<td>2</td>
<td>13</td>
<td>13</td>
<td>33</td>
<td>394</td>
<td>15</td>
<td>38</td>
<td>2</td>
<td>57</td>
</tr>
</tbody>
</table>
-----progress report-----
time spent thinking: (9 304)
regions on this branch:
(8 13 16 11)
context stack:
(ground ground ground)
contents of short term memory:
((new-neighbor (region 800 (far-left (region 8)))
(right (region 8) 57) (left (region 8) 2) (bottom (region 8) 38)
(top (region 8) 15) (w-b (region 8) zero) (y-b (region 8) minus)
(r-g (region 8) minus) (size (region 8) 394))

-----end of progress report-----
texture descriptors for (region 8) are (54 50)
texture descriptors for (region 13) are (44 51)

Texture measurement is appropriate in the context of ground areas.

-----progress report-----
time spent thinking: (14 202)
regions on this branch:
(8 13 16 11)
context stack:
(ground ground ground)
contents of short term memory:
((texture-similar (region 8) (region 13)) (color-difference
(region 8) (region 13)) (color (region 8) blue-green))

-----end of progress report-----
(REGION 8) passes the texture similarity test, but fails the color match.

-----progress report-----
time spent thinking: (6 727)
regions on this branch:
(8 13 16 11)
context stack:
(ground ground ground)
contents of short term memory:
((darker (region 13) (region 8)) (brighter (region 8) (region 13))
(yellower (region 13) (region 8)) (bluer (region 8) (region 13))
(redder (region 13) (region 8)) (greener (region 8) (region 13))
(below (region 13) (region 8)) (above (region 8) (region 13))

-----end of progress report-----
i think that (checked-skyline (region 13) (region 8))
checking the border between (region 13) and (region 8)

-----progress report-----
time spent thinking: (7 935)
regions on this branch:
(8 13 16 11)
context stack:
(skyline ground ground ground)
contents of short term memory:
(((segments built) (skyline-segment ((117 42)) (region 13)
(region 8)) (skyline-segment ((14 40) (13 40)) (region 13) (region 8)))

-----end of progress report-----

-----progress report-----
time spent thinking: (3 376)
regions on this branch:
(8 13 16 11)
context stack:
(skyline ground ground ground)
contents of short term memory:
(((peak (14 40)) (peak (17 42))))

-----end of progress report-----

Two local maxima have been discovered in the skyline. On the basis of a depth judgment, these peaks are correctly identified as treetops.

The analysis continues until all the major regions have been analyzed. The sky, ground separation is shown in Figure 5 and skyline in Figure 6.

In most cases, complete analysis of the image follows from the context established by the first (largest) region. This implies that initial scanning of such scenes can be quite coarse, and very simple ideas about gross context are enough to get started. Once started, inferences about local surroundings lead the Observer's attention over the entire scene, often returning many times to the same part of the image, each time with a bit more knowledge.
We have analyzed five additional scenes similar to the one in Figure 2. In five scenes out of these six the horizon was correctly identified (among other things). The scene where the horizon was not found was a case of the sky being very dark, looking very much like water. The point is that the cause of false recognition is evident in the relatively small set of production rules and easily correctable. Similarly, the Observer had problems with a view of Mt. Rainier, in which the glacial peak closely resembles a bank of clouds. Correct analysis of such scenes clearly requires a broader knowledge base.

6. Conclusion

The main contribution of this work is the formulation of a methodology for high-level, experimental scene analysis. This methodology is centered on the use of a flexible Production System for high-level decision making. We believe that the specification of high-level knowledge as Production Rules enables one to think clearly about the problem. Details which are well understood are reflected by clear precise rules. Ideas which are less well thought out tend to produce Production Rules which seem to be attempts to "get around" the formalism. A convoluted, difficult to explain set of rules is usually a sign of a half-realized idea.
REFERENCES


Maruyama, K., "A study of visual shape perception", UIUCDCD-R-72-533, Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, Illinois, 1972.


Figure 1: The overall block diagram
Figure 2: Slope 1

Figure 3: Texture in the scene
Figure 4: Region 11 outlined

Figure 5: Sky, ground, separation
Figure 6: Skyline