A PARALLEL-SERIAL "RECOGNITION CONE" SYSTEM FOR PERCEPTION: SOME TEST RESULTS

by

Leonard Uhr and Robert Douglass

Computer Sciences Technical Report #292

March 1977

A Parallel-Serial "Recognition Cone" System for Perception: Some Test Results*

Leonard Uhr and Robert Douglass
University of Wisconsin

Abstract

This paper presents some results of tests of specific "recognition cone" systems for probabilistic parallel-serial recognition and description of two-dimensional scenes of objects. Simula and Fortran encoded systems were given particular sets of transforms, and examined for their ability to handle scenes that contain 1) letters, 2) "place-settings" of several pieces of silverware and china, and 3) a "real-world" outdoor scene, in color. The same parallel-serial flow through structured layers of variable-resolution probabilistic transforms serves to detect edges, find features, characterize, recognize and describe, without any sharp dividing lines between different types of processes. Thus, a wide variety of diverse sources of information contextually interact, in a relatively simple and general way.

Keywords: perception, pattern recognition, scene description, recognition cones, resolution pyramids, parallel-serial systems.

*This research was partially supported by the National Science Foundation (Grant MCS76-07333) and the University of Wisconsin Graduate School.

Introduction

This paper presents and examines tests in which a "recognition cone" system was asked to recognize objects and describe scenes of several different sorts. A number of variant systems have been coded in Snobol and in EASEy, to explore the large number of possible configurations. But these are too slow and expensive to allow for testing. So the present tests were made using much faster systems coded in Simula and in Fortran.

"Recognition cones" are being developed by Uhr, 1972, 1974, and similar "cone" or "pyramid" systems are being developed by Hansen and Riseman, 1975, Douglass, 1977; Tanimoto, 1976; Klinger, 1974; and Levine, 1976. They are all systems that apply a parallel set of operations (here called "transforms") to the raw transduced scene that is input to the program's input array (called the "retina"). Then a second layer of parallel transforms is applied to the output of this first layer, and then a third layer is applied, and so on.

Figure 1 about here

Recognition Cones Described

Each layer of transforms outputs its set of implications (in effect, what these transforms have found and conjecture) into a buffer array that is smaller than is the array into which the transforms looked. So the system converges, forming a cone or pyramid, from its base, the retinal input array, to a final output array, its apex, that contains only a single cell.

Figure 1. Recognition Cone Structure. Layers of transforms (dotted lines) look at and imply things into buffer stores, transforming from retina to apex.

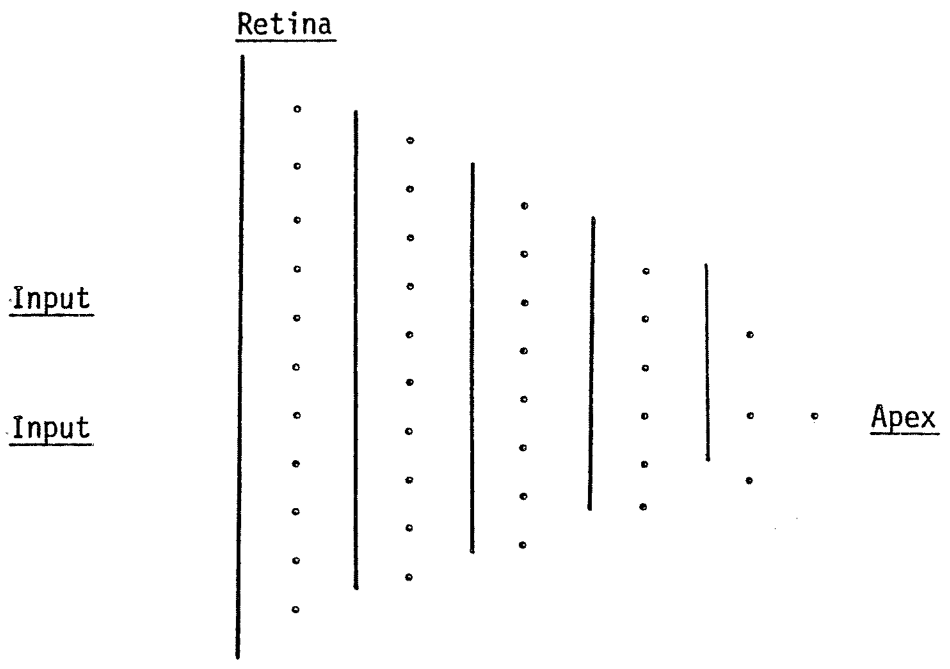


Figure 2. Letters (a check indicates the program correctly recognized the letter).

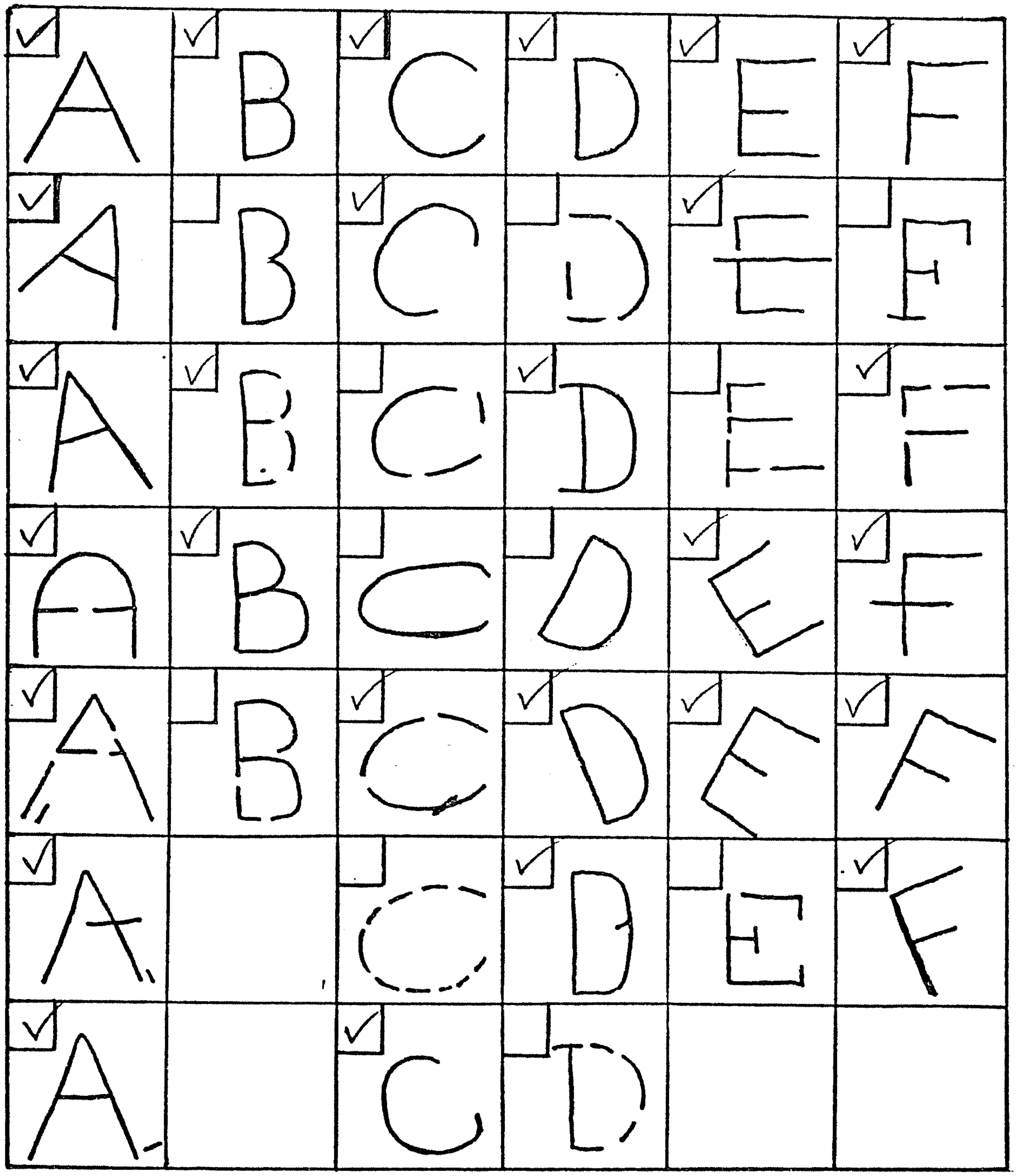


Figure 3. Place-settings.

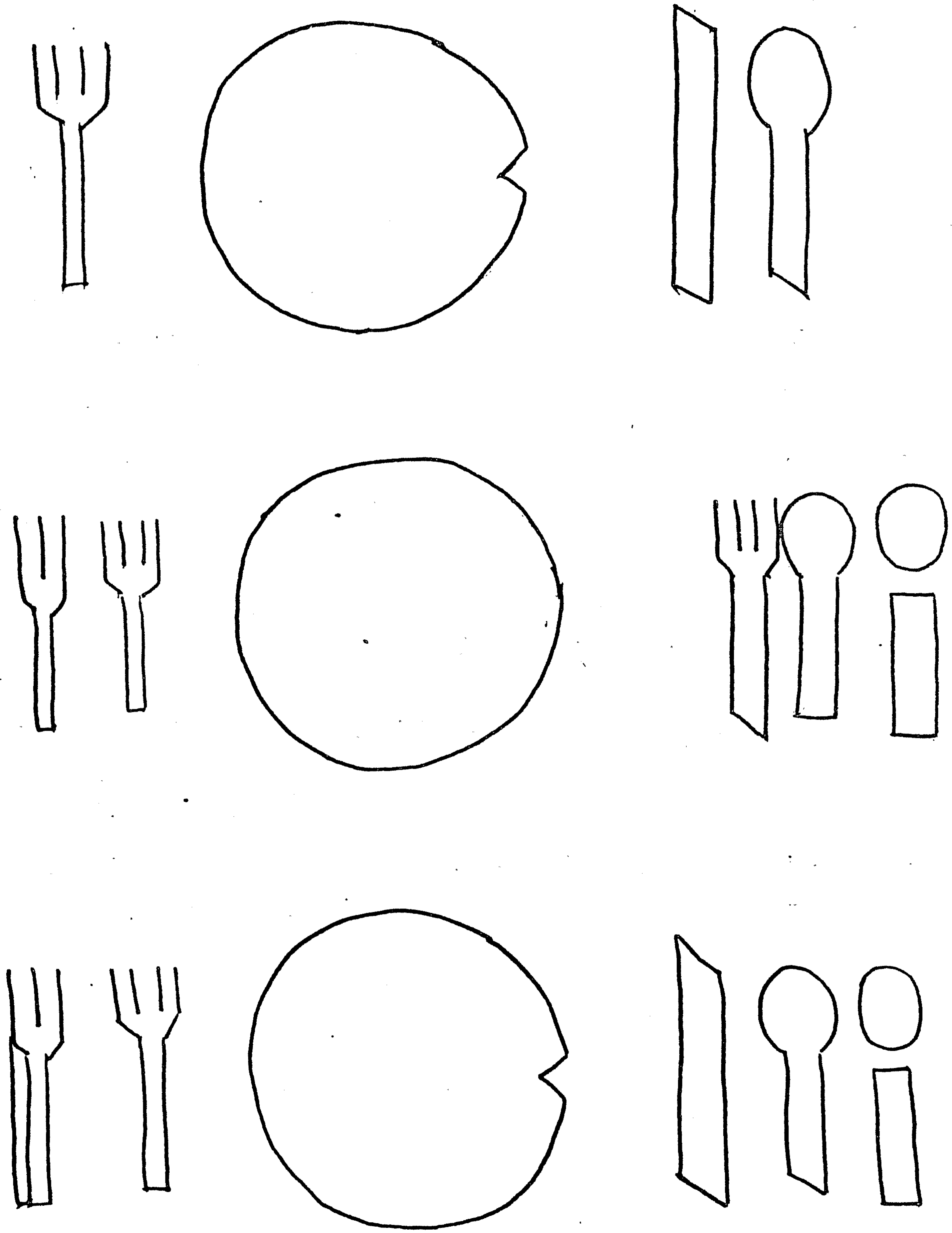


Figure 4. Shows the original scene (unfortunately much degraded and in grey-scale; see Ohlander's thesis for a good reproduction). But note that the scene has been cropped, so that none of the grass remains, and only the left 2/3ds of the house is used.

Each transform resides at a particular location in its transform layer, which is sandwiched between that transform layer's input buffer layer (into which its transforms look) and its output buffer layer (into which its transforms - if they succeed in finding what they are looking for - merge their implieds). So the location of the transform determines the location of the cell in its layer's output buffer array into which it will merge its implieds (that is, it will fire into the cell lying directly "behind" when looking at the cone's retina, as though looking "straight into its eye" along a straight line that ends in the cone's apex and goes through the retinal cell being looked at). The transform's location also determines the (relative) location of the parts of its layer's input buffer array that it looks at. A transform can be coded to look at any number of parts, at any relative locations, and to imply any number of things, of any sort (qualities, features, internal names, external names, of parts, sub-wholes, wholes, groups, or aspects). But the whole parallel-serial structure of many interlaced transforms is designed so that each particular transform can look relatively locally, and can serve as a stepping-stone in the recognition of more and more global and more highly structured things.

A transform can imply several additional important kinds of things, which serve a crucial purpose in giving flexibility and power to the system.

First, it can imply additional transforms to apply, and particular things, and classes of things, to look for (which in their turn imply additional transforms to apply - the transforms that would, if successful, imply them). This means that the system now combines a "bottom-up" "environment-driven" flow of processes with a "top-down" "internal attention mechanism-driven" flow of processes, where each is constantly calling upon the other. This also serves to focus, and to shift attention, as indicated. In the present runs this feature is heavily used in the deeper layers of the cone, as the particular higher-level wholes come into prominence (that is, are emerging as the more highly implied things).

Second, transforms can imply that a "trigger" be fired. The trigger is simply another transform that, when its combined weight has finally exceeded its threshold, fires, leading to the system's making a choice, in the cell where the trigger fires, among the things implied (the trigger can, optionally, specify a particular class of things to be chosen among). In this way the system can choose things in sub-regions of the scene (for the cell in which this choice is made is the apex of a sub-cone whose base is a sub-region of the retinal input scene).

This appears to be an important mechanism for handling scenes of several different objects, without an exhaustive tracing of the boundaries of each object. But the adjustment of the weights of the variety of transforms firing into the trigger, which should reflect the variety of cues that suggest that the system's picture of that region has crystallized to the point where a decision should be made (before the continuing reduction in resolution coarsens things too much), is an extremely subtle matter that needs a good bit of empirical testing. So this mechanism has not been used in the present runs.

The system is designed so that any number of transforms can reside at each particular cell location of a transform array. Any desired function over the set of parts the transform looks at can be used (though we tend to use simple partial matches, since these seem most consonant with what probably goes on in living visual systems). And any desired evaluation function for determining whether the transform has succeeded and therefore will fire its implieds can be used (though we tend to use a simple combining of the success of each part, which must exceed a specified threshold for success, again to mirror living firings of neurons into and across synapses).

The "recognition cone" system, then, serves as an overall structure, one that attempts to embody the overall structure of living visual systems (with their synapsing neurons, organized in converging layers in the retina, lateral geniculate and visual cortices). For a particular computer run, one must specify the number of layers in the system and the size (and therefore the converging cone structure) of each layer, and also the specific individual transforms used.

The Potential Efficiency and Power of Such Systems, Given the Suitable Hardware Embodiment

Such a system is not at all suitable for today's serial computers. Its highly parallel applications of layers of transforms need long serial loops through each layer array, to simulate the parallel processes. But this kind of cone/pyramid system suggests an architecture for parallel-serial computers that today for the first time seems feasible and economical, given today's very cheap large scale integrated circuitry on tiny chips. We already see the precursors of such computers in Duff's CLIP-4 (1976) parallel array, and in Kruse's (1976) similar parallel computer.

Given true parallel processes in hardware, such a parallel-serial system will become extremely fast.

It is designed to give the great efficiencies in time inherent in parallel processing when they are appropriate (as they certainly are in perceptual systems whose input is a very large parallel array), along with the efficiencies in space from a serial structure that allows for a natural relatively local successive combining of parts into more and more global sub-wholes, as needed. Thus, as Cordella, Duff and Levialdi, 1976, have shown, parallel processes can give enormous increases in speed. In contrast, as Minsky and Papert, 1969, have shown, a single process can lead to overwhelming inefficiencies in space with extremely global patterns.

But parallel-serial processes can very effectively arrive at trade-offs that give quite attractive efficiencies in both space and time. Thus, for example, with an input retina of 10^6 units, a purely serial recognizer will need 10^6k (k is a relatively small constant) moments of time, and 10^6 storage cells (for the retina) plus some negligible additional space for its program (say 10^5). A purely parallel processor will (in the extreme when it is asked to handle an entirely global pattern) need only 10^0 moments of time (literally, 1 single moment), but

$10^6(2^{10^6}+1)$ pieces of information stored. But parallel-serial systems will need only kL moments of time (where L is the number of Layers) and, roughly, $2S(10^6)$ storage cells (where S is the average number of storage cells needed to store the average number of transforms plus implieds at each cell in each layer). They also offer great promise of being powerful and efficient. For the total set of transforms can build up arbitrarily complex and arbitrarily global procedures. And they can be loosely coupled, with different sets of transforms serving different purposes, in assessing different types of things, and different aspects of the same class of things or even the same individual object.

Loose and Flexible Organizations of Transforms

This means that instances of patterns do not have to be carefully analyzed and described, or, even worse and even more common, restricted to a particular small set of objects, each varying in only very restricted, almost always linear and perfectly known ways.

For example, we human beings can recognize an object like a chair or a face from a) a colored scene that contains nothing but gradients and smoothly varying shades of color and intensity; b) a grey scale reduction of that scene (as one turns the color down on a color TV set), c) a line drawing of the scene (as when one superimposes a photo and its negative, and prints a new positive that contains only the contours, or when an artist or cartoonist makes a line drawing). But then we can throw noise, fog, or other kinds of fuzzings at such a scene. Even more drastic, we can ask a mad eraser to cut random swathes through the scene, making any lines or areas gapped, dotted, or missing. And a human being will have no trouble; in fact often we will not even notice such distortions, since all of our perceptual experience has been with just such distorted scenes, and worse.

This argues strongly for a system that does not rely on any particular operations for the "right answer" but rather uses a collection, some of which will be useful in some situation, but some others in other situations. The set of independent, each relatively weak and fallible, transforms serves that purpose. On the other hand, since transforms look at the outputs of other transforms, larger and larger structures, reflecting more and more global and contextually interrelated things, can be built. And they are built efficiently, since parts that are in common to several higher-level transforms can be handled by a single lower-level transform that serves as a building block to them all. And, in general, such a system decomposes all its needed complex functions into a hierarchical network of successively simpler building blocks.

Recognition of Simple Letters and Symbols

A Fortran recognition cone system was tested for its ability to recognize letters and symbols over a variety of severe distortions (rotations, gaps, fragmentings, stretchings, etc.; see Figure 2 for approximate copies of the letters used). Letters were input into a 20 by

Figure 2 about here

20 retinal array, and transformed as follows (see Table 1a): Layer 1 got local edges. Layer 2 got features (short line segments and curves). Layer 3 compounded several features together into a characterizer. Transforms from Layer 3 implied the possible output names, with weights chosen to be appropriate. This set of transforms was developed by looking at one example of each letter A through F (the top row in Figure 2). It took roughly 7 hours to formulate the 40 transforms, 7 hours to describe them in Fortran code for the program, and less than 2 seconds of 1110 CPU time to recognize one letter.

The same system was also used for one example of each of the five symbols star, triangle, circle, square, and plus. The same edge and feature-detectors in Layers 1 and 2 were used, except that these new symbols were added as implied (each with the estimated appropriate weight), when appropriate. And the appropriate compound characterizers were put into Layer 3.

All the original (known) examples of letters, and the symbols, were recognized. Some mistakes were made on the variant letters. But they include a number of quite unusual and difficult distortions, and there was little effort to adjust weights, or to develop a really good set of transforms. The purpose of these tests was primarily to demonstrate that the system is designed to work over unknown, highly non-linear and strangely distorted instances of the same pattern set - something that is crucial for real-world vision but has not received much attention in recent "scene analysis" systems.

Scenes of Eating Utensils, Forming "Place-Settings"

A Simula version was tested for its ability to recognize simple objects in a scene of several objects, and also the larger whole into which they formed. Scenes of line-drawings of forks, knives, spoons, and plates were input to a 20 by 48 retina, and transformed as follows (see Table 1b): Layer 1 got local edges. Layer 2 got features (line segments and curves). Layer 3 got compounds of features, to imply the individual objects. Layer 4 got compounds that implied place-setting, and different kinds of place-settings.

Transforms were coded so that a variety of variant objects would be recognized, including broken, gapped and noisy objects. But little effort was made to handle unknown non-linear distortions (though some unknown number of them will be handled, since the threshold operators accept variants). For these tests were designed to examine how the system can handle simple scenes that contained several simple objects that interact with one another to form a larger whole. It took about 2 hours to formulate and describe in code the roughly 30 transforms used, and about 8 seconds of Univac 1110 CPU time to recognize and describe one place-setting.

Figure 3 gives some rough drawings of place settings that were success-

Figure 3 about here

fully recognized (that is, all the correct objects, and also the whole were more highly implied than any possible alternatives). The bottom example shows a knife partially under the left fork - everything was recognized correctly. It would be easy to add additional transforms to handle more objects (e.g. cups, ladles), and to distinguish among different types of objects (e.g. salad vs. dessert fork), and to distinguish among different types of place-settings (e.g. French, breakfast).

Description of an Outdoor Scene in Color

The Simula program was used, in conjunction with Fortran routines, to handle the first two layers, to recognize and describe a detailed outdoor scene, containing a large house (with windows, walls, roof, etc.) fronted by trees and bushes, with grass below and sky above. The picture used was from the set first used by Ohlander, 1975, and originally digitized at USC. This picture is of interest because it has been widely used by others, e.g. Hanson and Riseman, 1976, and Schacter, Davis and Rosenfeld, 1976.

The original color picture was digitized into a 600 by 800 array, each cell containing 3 8-bit numbers, one for each of the primary colors. This is more data than the Simula program can conveniently handle, so the first two layers were processed by Fortran routines, and the rest by the Simula program (see Table 1c).

The following transforms were used:

Layer 1: Averaging is effected by looking at each 4 by 3 local array of cells, and outputting the sum of the intensities for each of the 3 primary colors into the cell in the output buffer layer corresponding to

Table 1. Specifications of the Cones Used for the Test Runs

A) For simple letters and symbols:

	Size of Array	Type of Transform Applied	Shrinkage
	20 by 20	1. Local edge detectors	1/2
	10 by 10	2. Feature detectors	1/2
	5 by 5	3. Compound characterizers	1/5

B) For place-settings:

	Size of Array	Type of Transform Applied	Shrinkage
	20 by 48	1. Local edge detectors	-
	20 by 48	2. Feature detectors	1/2
	10 by 24	3. Compound characterizers	1/2
	5 by 12	4. Compound characterizers	1/5, 1/12

C) For outdoor scene:

	Size of Array	Type of Transform Applied	Shrinkage
Fortran	800 by 600	1. Average	1/4, 1/3
	200 by 200	2. Hue, saturation, intensity	cropped
Simula	120 by 120	3. Gradients	1/2
	60 by 60	4. Short edges, texture	1/2
	30 by 30	5. Long edges, compounds, textures of short edges	-
	30 by 30	6. Higher-level compounds	-
	30 by 30	7. Higher-level compounds	-
	15 by 15	8. Average	1/2
	8 by 8	9. Average	1/2
	4 by 4	10. Average	1/2

the center of the 4 by 3, thus converging from an 800 by 600 to a 200 by 200 array.

This was done chiefly to reduce the very large amount of data in the 800 by 600 tv image. Averaging is probably usually a reasonable thing to do on most scenes. But if the scene might contain any tiny details of importance, then the system cannot take the chance of averaging such information out of existence. Rather, it should start with something like local differencing, to get gradients and edges.

Layer 2: The primary colors are combined, giving a) hue (the single combined color), b) saturation of that color, and c) intensity of that color. This combining is effected by a transform with 3 parts to its Conditions, where all 3 parts look at the same cell, giving a 200 by 200 output array.

The first two layers of transforms are effected by a Fortran program. The 200 by 200 image is now cropped to 120 by 120 and the Simula program takes over. (The Simula program is now being modified so that it will handle arrays larger than 120 by 120, so that the Fortran program will no longer be needed.)

Layer 3: Local gradients are computed, using a transformation that looks at the 4 parts in a 2 by 2 array, and sums the absolute values of the difference between the Northwest and Southeast pair of cells, plus the difference between the Northeast and Southwest pair of cells.

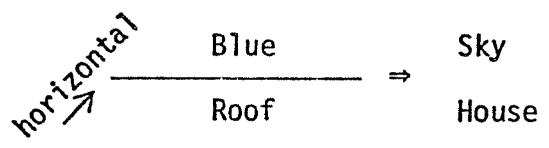
Layer 4: Short local edges are searched for in a 4 by 4 array, using 4 edge detectors (one for each of the slopes 45°, 90°, 135°, 180°). A texture detector fires if more than 6 simple gradient points above a threshold of 16 (from layer 3) are found in a 4 by 4 local array.

Layer 5: Long edges, angles, curves and textures are compounded together, using transforms that typically fire if about 3 out of 5 parts are found.

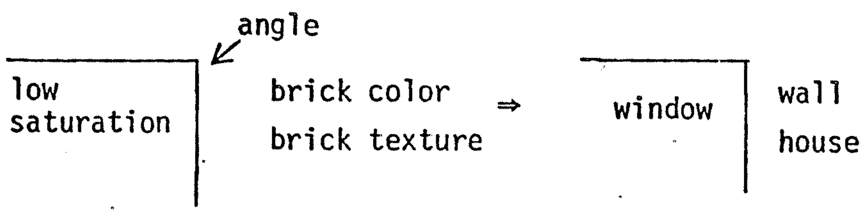
Two additional textures are got, by counting the number of edges in a 4 by 4 window, and by getting the principal orientation of edges in a 4 by 4 window.

Layer 6: A number of different compounding transforms look for configurations of edges (e.g. vertical edge, slope) and region elements (e.g. wall, sky) and already-implied objects (e.g. roof, window, house). Three examples follow:

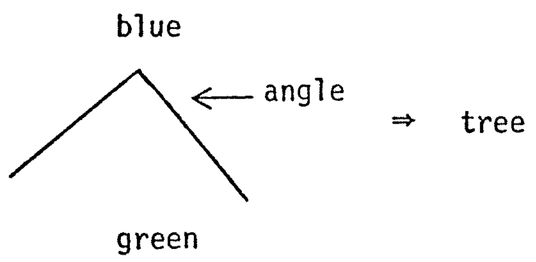
a) Blue above a long horizontal edge above the previously implied object roof implies sky (above) and house (below):



b) A low saturation region, an angle of long edges, and brick color and brick texture on the other side of the edges implies house, with window in the low saturation region and wall in the brick region:



c) Blue above two long sloped edges giving an upward pointing angle with green below implies trees below:



<u>Layer 7</u>: Still more higher-level compounds that are combinations of previously-implied names are looked for.

<u>Layer 8</u>: Each implied name is averaged over a 2 by 2 array.

<u>Layer 9</u>: Each implied name is averaged over a 2 by 2 array.

Layer 10: Each implied name is averaged over a 2 by 2 array.

Figure 4 shows the original scene (actually a good photo, in color, but poorly reproduced here). Figures 5 through 10 show various stages in the program's processes.

Figure 5 shows the results of the first three transformations layers through the cone (only the results of local gradient detectors are indicated, by the asterisks; many other things have been implied by this time).

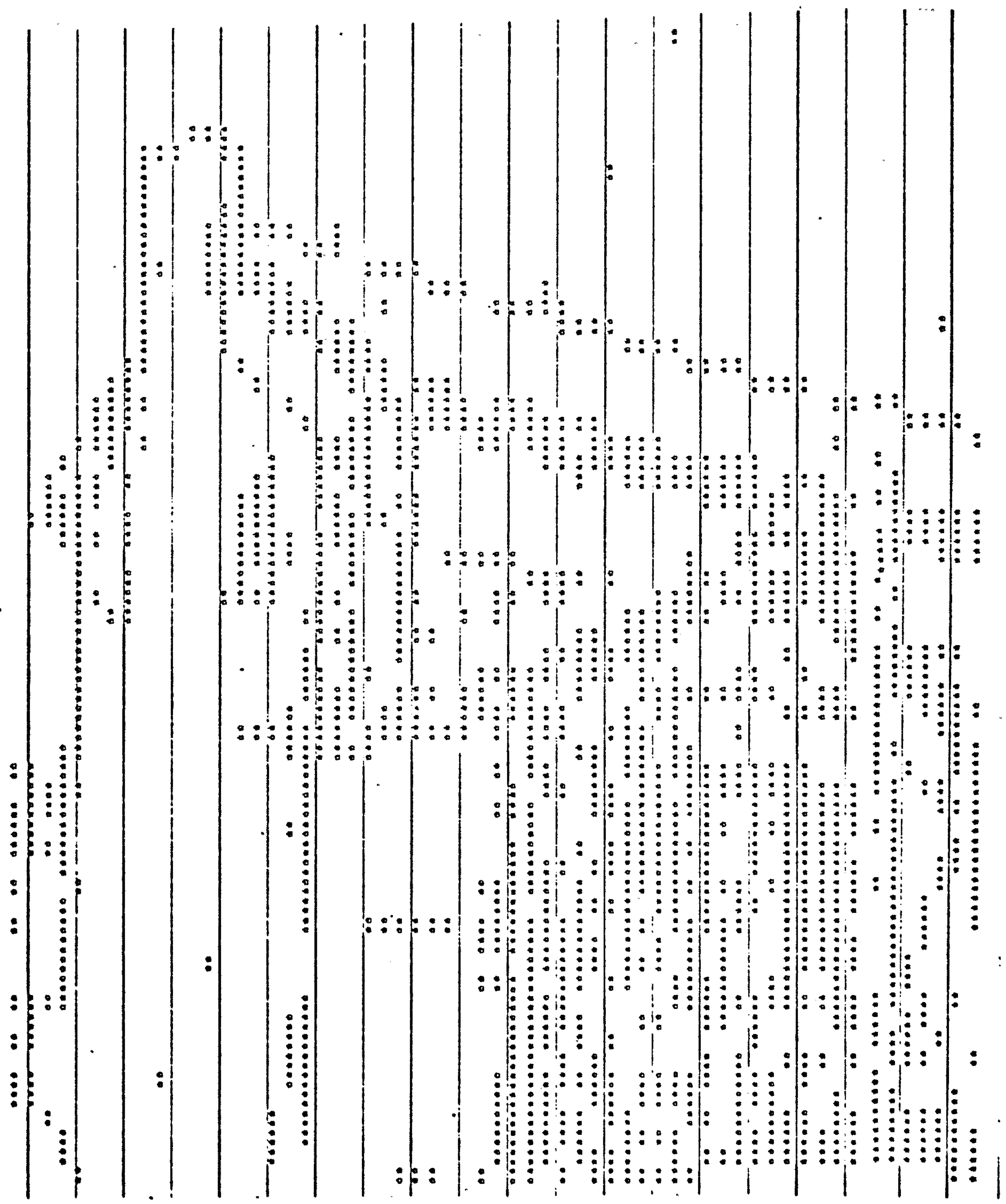


Figure 6 shows the single most highly implied thing output to each cell by Layer 6 (F = Window, G = Grass, H = House, S = Sky, T = Tree, W = Wall).

```
S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S
S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S
S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S
S S S S T T W T S S S S S S S S S S S S S S S S S S S S S S
S S S S T W W T S S S S S S S S S S S S S S S S S S S S S S
S S S S T W H H H H H S S S S S S S S S S S S S S S S S S S
S S S S H W H W H H T H H H S S S S S S S S S S S S S S S S
S S S S T W W T H H H G W G S H H H S S S S S S S S S S S S
S S S S T W W W W H H W W W T S G T T H H H S S S S S S S S
S S S H H W W W W W W H H H T H H T T G G T F H H H H S S S
S S H H W W W W W W W T T H H H H H T H H     S   T S H S S
S H H H W W W W H H R H H W W W H T H H H H H H H H T S S T
S H W W W W W H W H H H H W W W G W W W W H H H H H H H T T
S T H W W W W H H H H W H W W W W W W W W W H H H H H H F F
S S H H W W W F H H H W H W W H W W W G H T R H H S H H G T
S S W H W W W W W H H W H W W H H H H R H H H H H H H H W W
S S T W W W W W W W H W H W H W H H H H H H W W H H H F W W
S S H W W W W W W W W T H H H H R H H H H W H W H H H H F W
S S H W W W W W H W H H H T H H H H H H H H H H F F H H T F
F H W W W W W W W H H W W W W H H H H H H F F F F C H H W T
F H W W W W W W W W G W W W W H H W H T T H H H H H H H W T
T H W W W W W W W W W W W W W H H W H T T T S H H S F H W W
S H W W W W W W W W W W W W W H H H H T T T H S H S F H W T
S H W W W W W W W W W W W W H T H T H T T H H S H S H H W T
S H W W W W W W W W W W H W W H H H T H T H H S H H H H T T
T T W W W W W W W W W W W W W H T T G T H H H H H H H H T T
W W W W W W W W W W W W W W W W H T T W H H H H H H H H T G
W W W W W W W W W W W W W W W H W H H T T T T T T H H T G G
W W W W W W W W W W W W W W W T W H T T T T H T T T T T T T
G T W W W W W W W W W W W W W W T W T T T T W T T T T T T T
```

Figure 7 shows only those cells in which House (H) was most highly implied (this differs slightly from Figure 6, since ties are shown here but not there).

```
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
.                                                             .
.                                                             .
.                                                             .
.                                                             .
.         H     H                                             .
.         H   H H H H H                                       .
.         H   H   H H   H H H                                 .
.         H   H   H H H         H H H                         .
.         H         H H                 H H H                 .
.       H H     H H   H H H H H H H             H H H H       .
.     H H           H     H H H H H H H H H             H     .
.   H H H         H H   H H       H   H H H H H H H H         .
.   H H H       H   H H H H               H H H H H H H H     .
.   H H H       H H H H   H                 H H H H H H H     .
.     H H         H H H   H     H H H     H     H H   H H     .
.     H H           H H   H     H H H H   H H H H H H H H     .
.     H             H H H H   H H H H H H H H   H H H H       .
.     H               H   H H H H   H H H H   H   H H H H     .
.     H           H   H H H   H H H H H H H H H H     H H     .
.   H               H H       H H H H H H H         H H H     .
.   H               H           H H H H     H H H H H H H     .
.   H               H           H H   H     H   H H     H     .
.   H               H           H H H H     H H   H     H     .
.   H               H     H   H H H   H     H H   H   H H     .
.   H                           H H H H H   H H   H H H H     .
.   H                           H         H H H H H H H H     .
.                                 H       H H H H H H H H     .
.                               H   H H           H H H       .
.                                   H         H               .
.                                                             .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
```

Figure 8 shows the most highly implied things in Layer 7, the next layer.

```
S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S
S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S
S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S
S S S S T T H T S S S S S S S S S S S S S S S S S S S S S S
S S S S H H H H S S S S S S S S S S S S S S S S S S S S S S
S S S S H H H H H H H S S S S S S S S S S S S S S S S S S S
S S S S H H H H H H T H H H S S S S S S S S S S S S S S S S
S S S S H H H T H H H G H G H H H H S S S S S S S S S S S S
S S S S H H T H H H H H H H T S G T T H H H S S S S S S S S
S S S H H H H H H H H H H H H H H T T G G T H H H H H H S S
S S H H T H H H H H H T H H H H H H H H H     S   T H H S S
S H H H T H H H H H T H H T T T H T H H H H H H H H T S S T
S H H H H H H H H H H H H H H H G H H T H H H H H H H H T T
S H H H H H H H H H H T H H H H H H H H G H H H H H H H T T
S S H H T H H H H H H H H H T H H H H G H T H H H S H H G T
S S H H T H H T H H H H H H T H H H H H H H H H H H H H H H
S S H H H H H H H H H H H T H H H H H H H H H H H H H H T T
S S H H H H H H H H H T H H H H H H H H H H H H H H H H T T
S S H H H H H T H H H H H T H H H H H H H H H H T T H H T T
T H T H H H H H T H H T H H H H H H H H H H H H H H H H T T
T H T H H H H H H H G H H H H H H H H T T H H H H H H H T T
T H T H H H H H H H H H H H H H H H H T T H S H H S H H H H
S H T H H H H H H H H H H H H H H H H T T H H S H S H H T T
S H T H H H H H H H T T H H H H H T H T T H H S H S H H T T
S H T H H H T H H H H H H H H H H H H H T H H S H H H H T T
T H T H H H H H H H H H H H H H T T G T H H H H H H H H T T
T H H H H H H H H H H H H H H H H T T T H H H H H H H H T G
T H H H H H H H H H H H H H H H T H H T T T T T H H H T G G
T T H H H H H H T T H H H H H T H H T T T T H T T T T T T T
G T T H H H H H T T H H T T H H T T T T T T T T T T T T T T
```

Figure 9 shows (for Layer 7) only those cells into which House (H) is most highly implied. Note how the higher-level compound for house has turned many of the local "Wall" areas into "House."

```
...............................................................
.                                                             .
.                                                             .
.                                                             .
.           H H                                               .
.         H H H H                                             .
.         H H H H H H H                                       .
.         H H H H H H   H H H                                 .
.         H H H   H H H   H   H H H H                         .
.         H H H H H H H H H H           H H H                 .
.       H H H H H H H H H H H H H H           H H H H H H     .
.     H H H H H H H H H H H H H H H H H H H           H H     .
.   H H H   H H H H H H H H H H H H   H H H H H H H H         .
.   H H H H H H H H H H H H H H H   H H H H H H H H H H H     .
.   H H H H H H H H H H H H H H H H H H H   H H H H H H H H H .
.     H H H H H H H H H H H H H H H H H   H H H H H   H H     .
.     H H H H H H H H H H H H H H H H H H H H H H H H H H H H .
.     H H H H H H H H H H H   H H H H H H H H H H H H H H H H .
.     H H H H H H H H H   H H H H H H H H H H H H H H H H H H .
.     H H H H H H H H H H H   H H H H H H H H H H H H H H   H .
. H H H H H H H H   H H H H H H H H H H H H H H H H H H H H   .
. H H H H H H H H H H   H H H H H H H H   H H H H H H H H H   .
.   H H H H H H H H H H H H H H H H H H     H   H H   H H H H .
.   H H H H H H H H H H H H H H H H H H     H H   H   H H H   .
.   H H H H H H H H H H H H H H H H   H     H H   H   H H H   .
.   H H H H H H H H H H H H H H H H H H H   H H   H H H H     .
. H H H H H H H H H H H H H H H H   H     H H H H H H H H     .
. H H H H H H H H H H H H H H H H H H     H H H H H H H H     .
. H H H H H H H H H H H H H H H H H H H H   H     H H H       .
. H H H H H H H H H H H H H H H   H H         H H             .
.     H H H H H H H H H H H H H H   H         H H H   H H H H .
...............................................................
```

Figures 10a, 10b, and 10c show the outputs from Layers 8, 9 and 10.

```
S S S S S S S S S S S S S S S
S S S S S S S S S S S S S S S
S S H H S S S S S S S S S S S
S S H H H H H S S S S S S S S
S S H H H H H H G G H S S S S
S H H H H H H H H H H H H H S
S H H H H H H H H H H H H H T
S H H H H H H H H H H H H H T
S H H H H H H H H H H H H H T
S H H H H H H H H H H H H H T
H H H H H H H H H T H H H H T
H H H H H H H H H T H H H H T
H H H H H H H H H H H H H H T
H H H H H H H H H T H H H H G
T H H H T H H H T T T H T T T
```

Figure 10a

```
S S S S S S S S
S H H S S S S S
S H H H H H H S
H H H H H H H T
S H H H H H H T
H H H H H H H T
H H H H H H H T
T H H H T T T T
```

Figure 10b

```
S S S S
H H H H
H H H H
H H H H
```

Figure 10c

It took (very roughly) about 120 hours to formulate the 70 transforms used for this run, plus 40 hours to code them in Simula. Processing one scene takes less than 2 minutes of 1110 CPU time (roughly 30 seconds for the Fortran routines and 60 seconds for the Simula program).

Using Ad Hoc Information (e.g. Color) to Improve Performance, When Useful

A number of additional runs were made, on the house picture and also the Ohlander pictures of a car and an interior, using only color and a local color edge detector. These gave rather good results (as judged subjectively, as is always the case) in bringing out the outline, with results comparable to Figure 5, and to the results shown by Schacter, Davis and Rosenfeld, 1976. But such results depend entirely on the (typical expected) color of a thing - grass is green, sky is blue, houses are red,.... The larger test did not use the color edge detector, and gave colors only low weights. For houses are known to be grey and even sky blue, skies can be rosy-fingered and grass purple.

So we can expect that the addition of color edge detectors, to imply the standard things with a low weight (so that they will be overriden by other transforms except in unusual situations) should substantially improve performance. And this is but one small example of how diverse sources of information can be captured in such a set of different characterizing transforms, to work in concert toward the decisions of the whole.

The Need for Learning by Discovery

It is tedious to describe, code and adjust the weights and thresholds of these transforms. Much preferable (and in the long run inevitable) is to have the system learn - generate and discover - a good and sufficient set of transforms. So we hope that the addition of learning routines, as in Uhr, 1978, can replace such human pre-specification. And learning is absolutely necessary for other reasons. We cannot assume that environments will never change; on the contrary, in the real world new things, and new variations on old things, are constantly being created, by nature and by human beings. And interestingly difficult things (and that means most real-world things) are too complex for us humans to be able to analyze and describe them completely.

On Testing and Generality

It is very hard to evaluate the results of a program for perception. When a leafy tree partly obscures a house should the program output "tree" or "house" or both? If it uses red-brick-color as a feature that makes it do well on a brick house, what should we say when it fails on a very similar green clapboard house - that it is an "expert" on red-brick, or too "ad hoc" for the "real" world? These are just a few of the problems that plague all of today's systems for scene description. Clearly, much larger test runs are needed, on an ever-wider variety of different kinds of scenes, with variants on each individual type of object, different interactions among objects, and more different kinds of objects.

Far more extensive tests are needed than those reported here (or, indeed, reported for any other scene description system of which we are aware). At the very least, one must test such a system on scenes that were not known to the people who designed the system (including any particular transforms or other embedded knowledge), as a guarantee that ad hoc information was not programmed in just to handle these scenes as special cases. One should also test the system on several examples of the same type of scene, and also on several different types of scenes. One should try to develop some measures of the complexity of scenes, in terms of the number and difficulty of the things they can contain, the complexity of variations among different members of the same object class, and the complexity of interactions among different objects. And different systems must be compared with one another (and with human perceptual systems), which means that standard tests should be developed, and used in a standard way by different researchers.

So the test results reported in this paper are meant to be just first demonstrations. The worth of this system (along with any other system for pattern recognition and scene description) can be determined only by a wide range of tests that explore its range, generality, and power.

But the recognition cone system has been designed not at all for a particular scene, or type of scene. Rather, the general structure makes the encoding of a particular set of transforms as simple as possible. The probabilistic structure of transforms, and the parallel-serial structure of the layered cone, are designed to make perception as general, powerful and efficient as possible. And the whole structure is designed to make the learning - the generation and discovery - of a good set of transforms as easy as possible, to ultimately remove the tedious burden of analyzing and pre-programming.

A Note on Problems of Speed, and How Parallel-Serial Computers Will Solve Them

A few interesting comments can be made about the timings. Many researchers in pattern recognition of individual letters would find 8 seconds of 1110 CPU time for the place-settings, and 90 seconds for the outdoor scene excessively long. On the other hand, programs for scene analysis typically appear to take far longer (Ohlander's takes 9 hours on the PDP-10). This program will slow down as more fixed transforms are added (remember, a transform is iterated throughout its layer). But dynamic transforms do <u>not</u> add to processing time (though they need storage space), since they are applied only when dynamically implied. So the hope is that not too many additional transforms will be needed but, if they are, time can be kept within reasonable bounds by making more transforms dynamic.

But these are problems <u>only</u> when such a system is run on a serial Von Neumann computer. Once we have suitable parallel-serial systems, with a sequence of layers of processors that directly embody the layers of the cone, processing time is only the time taken for a single transformation (and that is only a few machine cycles, since most everything is done in parallel) times the number of layers in the cone. E.g. a 10 layer cone taking 1 msec per transform (a very slow figure) will process 100 scenes a second. (It would pipeline continuing scenes, as input from a TV camera, of 1000 scenes a second!). Adding new transforms will take more space (but on today's chips this is already cheap, and on tomorrow's it will be negligible), but absolutely no more time.

On the Use of One Vs. Many Sets of Transforms

The three tests reported in this paper were designed to examine different aspects of the program's performance. They were made at different times, extending over more than a year, and on two different encodings (one in Fortran, one in Simula) of the recognition cone system. So they used different numbers of layers (which to some extent reflects the differing presumed complexities of the scenes being handled) and different sets of transforms.

To a great extent the transforms at the lower layers, for local edge detection and feature extraction, were quite similar (but we made no effort to keep them exactly the same). And Layers 1 and 2 for the place-setting runs used almost (but not entirely) the same set of transforms as Layers 3 and 4 of the outdoor scene run.

In the future the hope is to use a single set of transforms, no matter what type of scene. This will allow for exact comparisons as to the system's success on different kinds of scenes, and will force us to develop general rather than ad hoc sets of transforms. There will almost certainly be some overhead cost, as there always seems to be for generality. But our conjecture is that this cost will be small (and, when embodied in parallel hardware, the time costs will be nill), and the total number of transforms needed will be far smaller (and easier to code, or learn) than the total number of transforms needed if different sets are used for each type of problem.

Summary Discussion

The particular recognition cone configuration used, with the particular transforms programmed into it, was able to handle several different kinds of scenes, containing a) letters, b) place-settings, and c) outdoor color photos. The system handles scenes for which it is _not_ specifically preprogrammed. And it makes use of whole sets of transforms that gather diverse kinds of information, to arrive at its decisions. Transforms can easily be added or replaced, no particular transforms are crucial, and widely different representations of the same object can be handled, by one or another subset of the transforms.

The present tests show how the system does indeed develop rich and reasonable-appearing arrays of descriptive information, as it applies its transforms to the scene, to recognize as well as characterize and segment. But the final choices are not yet made, or are made in an over-simple fashion. Once the triggering transforms for making intermediate decisions are added to the program, we can expect interesting new improvements in forming a well-tailored final description.

These tests show how information about diverse aspects of the scene, including low-level edges, gradients, colors and textures, and higher-level features, parts, characteristics and wholes, work in concert to improve performance. Thus not only do the lower-level transforms serve to compose into the higher-level characteristics, but also information can usefully be got from any and all levels where it is uncovered.

Recognition cones have several central features designed to make them general, robust and efficient: their parallel-serial structure, probabilistic transforms that compound and interact, the ability to reconfigure the overall architecture and replace and modify the individual transforms, the way transforms can flow in a combined out-in and in-out fashion, and the ability to learn. All this means that such systems should work on a variety of different kinds of scenes, and should not be limited by ad hoc programming to particular types of scenes and representations. Only when given a sufficient set of transforms can this kind of system be adequately judged. But its development can proceed in a relatively smooth small-increment manner, since changes need only be made to an individual transform, or a few transforms, without having any effect on the larger program.

Finally, this kind of parallel-serial system offers great promise for future increases in power, simply by using parallel-serial hardware (which has today finally become feasible, and economical). Thus we can expect a four to six orders of magnitude increase in speed, even with very large sets of transforms, at relatively reasonable costs. This kind of architecture would seem to offer great promise for perceptual systems that are powerful, general, flexible, efficient and adaptable.

References

Cordella, L., Duff, M.J.B. and Levialdi, S. Comparing sequential and parallel processing of pictures, Proc. IJCPR-4, 1976, 4, 703-707.

Douglass, R., Recognition and spatial organization of objects in natural scenes, Proceedings of IJCAI-5, Cambridge, 1977.

Duff, M.J.B., CLIP 4: A large scale integrated circuit array parallel processor, Proc. IJCPR-4, 1976, 4, 728-733.

Hanson, A., Riseman, E. and Williams, T., Constructing semantic models in the visual analysis of scenes, Proc. Milw. Symp. Auto. Comp. & Contr., 1976, 4, 97-102.

Klinger, A. and Dyer, C., Experiments on picture representation using regular decomposition, TR Eng. 7497, UCLA, 1974.

Kruse, B., The PICAP picture processing laboratory, Proc. IJCPR-4, 1976, 4, 875-881.

Levine, M.D., A method for non-purposive picture segmentation, Proc. IJCPR-4, 1976, 4, 494-498.

Minsky, M. L. and S. Papert, Perceptrons, Cambridge: MIT Press, 1969.

Ohlander, R. B., Analysis of Natural Scenes, Unpubl. Ph.D. Diss., Carnegie-Mellon Univ., Pittsburgh, 1975.

Schacter, B. J., Davis, L. S. and Rosenfeld, A., Scene segmentation by cluster detection in color spaces, SIGART Newsletter, 1976, 58, 16-17.

Tanimoto, S. L., Pictorial feature distortion in a pyramid, Comp. Graphics and Image Proc., 1976, 5, 333-352.

Uhr, L., Layered 'recognition cone' networks that preprocess, classify and describe. IEEE Trans. Computers, 1972, 21, pp. 758-768.

Uhr, L., A model of form perception and scene description. Computer Sciences Dept. Tech. Rept. 231, Univ. of Wisconsin, 1974.

Uhr, L., Learning by Discovery in Parallel-Serial Layered 'Recognition Cones,' in preparation, 1978.