The University of Wisconsin
Computer Sciences Department
1210 W. Dayton Street
Madison, Wisconsin 53706
U.S.A.

Received: November, 1974

A MODEL OF FORM PERCEPTION AND SCENE DESCRIPTION

by

Leonard Uhr

Computer Sciences Technical Report #231

November 1974

A MODEL OF FORM PERCEPTION AND SCENE DESCRIPTION

Leonard Uhr

ABSTRACT

This paper examines a computer-programmed model for human recognition and description of scenes of objects that extend over time as well as space.

When the scene contains several objects, the model will first output a single most salient name - which may be the name of a high-level whole, like "couple," or of a salient lower-level object, like "stick-girl." It will then describe that salient object, and go on to mention other objects in the scene.

The basis of the system is a parallel-serial memory network of interrelated transforms, or characterizers, which are themselves complex sets of probabilistic n-place relations over "things" (where a "thing" can be a value, an internal or external name of an object, part, quality, class, or function, a characterizer to apply, or an act to effect) - a network of simplified neuron-like elements interacting at synapse-like junctions.

This research has been partially supported by grants from the National Institute of Mental Health (MH-12266), the National Science Foundation (GJ-36312), (NGR-50-002-160) and the University of Wisconsin Graduate School.

INTRODUCTION

This paper presents the perceptual system of a wholistic model of cognition (called SEER[1], see Uhr, 1974, 1975a, 1975d).

The SEER system performs a variety of functions that are normally thought of as "perceptual." These functions will be examined in some detail after the model has been described. They include such things as a) recognition over non-linear as well as linear transformations and distortions, b) handling of shaded, textured, and colored scenes, c) the use of internal sets, expectations, needs and goals to help focus perception, d) contending with changes (e.g. movements) that occur over time, e) "glancing about," over time, as a function of internal expectations, f) developing descriptions of scenes, and g) outputting appropriate descriptions as a result of dynamic conversational interactions with the hearer - the recipient - of these descriptions.

The system described here is actually the perceptual "front end" (Uhr, 1972, 1973e) of the complete cognitive model which, along with conversational interaction and learning capabilities, will be described more fully in subsequent papers.

[1]SEER might stand for something like Semantic learner, or See and err, or:

Scene			
Sensor	Environment	Extractor	Recognizer
Semantic	Encoder	Educer	Recoder
			Responder

. Its chief virtue is as a mnemonic space-saver.

I must emphasize that this paper describes a running computer program. Therefore the model it embodies is precise and consistent. And when running it exhibits its own behavior, internal processes, and consequences. It has been given only very limited "toy" tests (in common with all large programs to date). It has been given only a few of the many characterizing transforms it would need to handle a wide variety of scenes. This is much like the situation for a parser, where it is quite tedious to get and then give it the entire grammar of a natural language.

The Necessity of Learning to Cope with Potentially Infinite Environments

With effort we could enlarge the set of transforms. But it seems to me crucial that we instead give the system learning routines, so that it can discover good transforms for itself. This not only saves us an enormous amount of tedium, it is absolutely necessary for language and objects, which in a very real sense are "potentially infinite." New words and objects, and structures over them, are constantly being created, and the organism must learn to cope with them. In theory we might pre-program the grammar for a dead language like Latin, or the transforms for a finite world, like a robot's "blocks-world" (as in the major robot projects; see Nilsson, 1969; Feldman et al., 1971; Winston, 1972).

But the real world is always changing, and for all practical purposes infinite. Any intelligent organism must learn about that part of the world with which it must cope.

A System for Exploring Other Variant Systems

This system can be thought of as a simulation of a general-purpose parallel-serial computer, one that might be used much like a nerve-net system, for specifying and running a particular configuration of transforms. From that point of view, this paper specifies what a prospective user must do to develop, test and run a set of characterizing transforms that he is interested in exploring.

A BACKGROUND OVERVIEW OF SYSTEMS FOR FORM PERCEPTION (PATTERN RECOGNITION AND SCENE DESCRIPTION)

The enormous amount of research in pattern recognition and scene description is examined in some detail in Uhr, 1973a, 1975b and will only be mentioned here (for other reviews, see Rosenfeld, 1969; Duda and Hart, 1973). For collections of papers, see Uhr, 1966; Kanal, 1968; Watanabe, 1969, proceedings of the Internal Joint Conferences on Pattern Recognition and on Aritficial Intelligence, and the journals Pattern Recognition, Computer Graphics and Image Processing and IEEE Transactions on Computers.

Problems Tackled

Briefly, people have tried to develop computer-programmed systems to a) recognize and <u>name</u> single objects (e.g. the letters of the alphabet, spoken words, chromosomes, tables and chairs, simplified faces), b) extract features (e.g. a break in a bone) or simple objects (e.g. a silo) from a "scene" (e.g. an X-ray, or an aerial photo), and c) describe simple scenes of objects (e.g. a TV picture of blocks in a room, a spoken sentence, or other pictures containing more than one object).

As the problem is made more complex, the "scenes" are made cleaner and more abstracted. E.g. single letters are handled in all the real-world distortions given them by different people who write them, and the wavery, smudged, irregular media of pencil, carbon, and paper. In contrast, scenes are usually very carefully composed, and often quite severely simplified. E.g. the utterer of the sentence is trained to speak carefully; only three or four simple flat-surfaced blocks are used in research on "robot vision," each block remains exactly the same, and differs drastically in shape from the others (e.g. cubes vs. pyramids).

Alternate Approaches to the Assessment of Structure in Patterns and Scenes

Six major (but rough) distinctions can be made as to the approaches taken:

A) Local spots and collections of spots are assessed, and their implications combined into a decision (e.g. Rosenblatt, 1962; Bledsoe and Browning, 1959). Such an approach cannot handle scenes unless a prior step subdivides the scene into regions where each contains only one object. But that is very hard to do, and is best done as an integral part of the recognition process, so that it can make use of information about partially recognized objects to segment.

B) Structure is built into the individual feature-detector. E.g. a single feature might be an "L"-shaped angle, or a loop (e.g. Doyle, 1960; Munson, 1968) or even a general curve-follower (Sherman, 1960).

C) "Syntactic" structure is superimposed over the individual features (e.g. Narashimyan, 1966; Evans, 1971; Ledley, 1972; see Miller and Shaw, 1968; Fu and Swain, 1971). To do this, linguists' "rewrite rules" (which talk about strings of objects concatenated in the 1-dimensional sentence) must be generalized to handle 2- or N-dimensional scenes. A number of generalizations have been suggested (e.g. Shaw, 1969; Pfaltz and Rosenfeld, 1969) but there are problems - chiefly in combining parts from real-world scenes, where anything can be broken, gapped, bent, or missing (see Clowes, 1969, 1971; Uhr, 1971).

D) Some kind of "list-structure" or graph-like model is superimposed over the individual features (e.g. Guzman, 1968, Brice and Fennema, 1970, Waltz, 1972).

The problem now becomes one of converting the scene into a list-structure, and then trying to find matches between some of the list-structure representations of objects stored in memory and sub-regions (that is, sub-list-structures) of the scene. This bears some resemblance to the finding of paths through problem-solving nets (e.g. to prove theorems or find good moves in a game like chess), and people are attempting to develop perceptual systems that use "heuristic search" techniques developed for problem-solving (e.g. Slagle and Lee, 1971).

E) Successive types of information can be assessed in stages. Thus in speech recognition, the feature, word, syntactic structure, and semantic structure; in "robot vision" the gradients, edges, surfaces, and solids, can be examined at different stages. Some systems (e.g. Guzman, 1968) do this from smaller structure to larger; others use parallel co-routines for each stage. (e.g. Reddy et al., 1973); still others try to get other varieties of interaction among stages (e.g. Duda and Hart, 1970; Tappert and Dixon, 1973).

F) Finally, probabilistic relational characterizers, whose inputs are (lower-level) features, or the implications of features, are used to build up a hierarchical structure of features. This can be done in a variety of ways, with a variety of different organizations -

in a parallel-serial layering of transforms (e.g., Uhr, 1972; Riseman and Hanson, 1974; Hanson and Riseman, 1974) in a strictly-serial discrimination net (e.g. Unger, 1959; Hunt, 1962), or in a hierarchical structure without any well-delineated layers or levels (e.g. Uhr and Vossler, 1961; Uhr and Jordan, 1969; Zobrist, 1971; Williams, 1974).

The Importance of Structure, Parallel-Serial Processes, and Probabilities

There are many similarities between these different approaches. Ultimately they may turn out to be alternative ways of realizing much the same architecture and flow of information. The crucial point is that this architecture (except for A) is one that uses structural information in order to cope with the structure that exists in patterned objects (indeed the word "pattern" is probably best defined as a structure of interacting parts). But there are important differences, chiefly related to issues of parallel (A) vs. serial (C,D,E,B) vs. parallel-serial (F,E), probabilistic (A,B,F) vs. deterministic (C,D,E), conjectural (A,F,) vs. analytical (C,D,E) processes, and vagueness and ambiguity (A,F) vs. certainty (C,D,E). (These are briefly discussed below, and also in Uhr, 1975b and Minsky and Papert, 1969.)

The SEER Model Is a Dynamic Parallel-Serial Probabilistic Network of Structural Transforms

The SEER model presented in this paper is a parallel-serial layered network of structural characterizers. Characterizers are probabilistic threshold functions that attempt to cope with uncertainty and unknown variations, rather than deterministic tests that can only handle things anticipated in advance. The system makes directed searches, dynamically deciding which transforms to apply next, that is, what to look for, as a function of internal expectations and external information that has been found so far.

OVERVIEW OF THE SEER MODEL

We will now examine the SEER system:

A) The overall architecture of the parallel-serial, converging "recognition cone."

B) Its basic structural details-the characterizing transforms that it uses, the connections among structures of these transforms, and the variety of functions that they serve.

C) The overall flow of information through a SEER network - how it characterizes, finds parts and qualities, builds wholes, implies what transforms to apply next and whether to make intermediate decisions, merges successive moments of time into its dynamic memory stores, and, in general, develops a description of the scenes it has seen in its recent past.

After the model has been described, we will examine some of its behavior from the point of view of criteria that might be posed for the evaluation of a system that purports to be a "model of form perception."

OVERALL ARCHITECTURE OF SEER "RECOGNITION CONES"

The perceptual system is organized as a serial sequence of layers of parallel processes (see Figure 1). Information from the external environment is input (sensed) into the first layer, the "retina." Each layer contains a set of characterizing transforms, which are all applied in parallel (this parallel aspect must be simulated when a serial digital computer is used). A "transform" specifies a structure of related things to be looked for and a criterion for success. If the transform succeeds, its set of implications is merged into the cell in the next layer that lies directly behind that transform's center, except for transforms, which are merged into CHARACTERIZE,[2] and triggered choices, which are merged into FOUND. A "shrink-size" can be specified for each layer. Thus layers get smaller and smaller as information is averaged, extracted, and coalesced, and the system has the form of a cone.

Insert Figure 1 about here

Parameters and Specific Transforms of the Recognition Cone

The overall structure of the recognition cone allows for a wide variety of structural modifications, and of parameter adjustments, including the size of the input layer, shrink-size at each layer, number of layers, and number

[2]Caps indicate actual lists in the SEER program.

and type of transforms. The specific transforms used can also be varied, since they are read in as tabled data, or learned.

This seems extremely desirable at this early stage of model-building, since it means that we can vary details, both as to general structure and specific transformations, and thus attempt to draw the system's behavior closer and closer to the domain of perceptual behavior being modeled. We can also particularize specific models to a variety of different purposes - to model different species, and also perceptual systems that use different basic material building-blocks or that are designed for different practical problems. Thus the general structure is a framework within which we can examine "natural" and "artificial" perceptual systems side by side.

Sensing and Inputting Information into the First Layer

Information is input to the Retinal buffer of the recognition cone, using as fine detail of resolution as is needed for the kind of scene being processed. The first layer of characterizing functions then transforms this information, outputting into the next buffer, which is input to the second layer of transforms. And so on.

Types of Transforms in Each Layer

Each layer contains a mixture of a variety of different kinds of functions that can be computed by nets of neurons, that converge into threshold elements. We will examine these later, but they include: a) pre-processors (e.g. averagers and differences to smooth the raw input and bring out gradients and edges), b) local feature-extractors (e.g. edge, curve, angle, and color detectors), c) structure-detectors, that compound lower-level features or structures into larger structures (e.g. lines into angles, strokes into letters, letters into words); d) impliers of transforming characterizers to apply (e.g. to find the rest of an object, or to "glance about"); and e) triggers to decide among alternatives implied in that region (e.g. to choose an object-name, or a color, for that region).

Ordering of Layers

The placement and ordering of these different types of transforms is one of the parametrized details that can be specified for each different model, and varied until behavior satisfies. A simple architecture might be: 1) an averaging layer, followed by 2) a differencing layer, 3) a first layer of low-level feature-detectors, 4) a layer of structural characterizers, and, possibly, 5) a second layer of structural characterizers (see Figure 2a).

A more realistic (and potentially more powerful) architecture is shown in Figure 2b. The first few layers of this cone do pre-processing operations --1) averaging, 2) differencing, 3) averaging, and 4) differencing. Then 5) primitive features (e.g. sloped lines, and, possibly, very simple curves and textures) are detected, followed by

Insert Figure 2 about here

several layers of structure-detectors to build compounds such as 6) angles, 7) enclosures, 8) articulated figures, 9) coherent sets of several figures (e.g. a "family" or a word) and 10) higher-level groups or wholes (e.g. a "forest" or a phrase or other meaningful verbal unit). As we will see later, parts, qualities, whole objects, symbolic words and phrases, and internal things (e.g. concepts, images or ideas) stored in memory can all be mixed together and processed in the same layer.

The set of 10 layers I have just specified seems a <u>minimum</u> for the realistic handling of scenes of objects. Probably several more layers are required to handle compounding adequately. And feature- and structure-detecting layers are a good bit more complex than this simple list suggests, since they must also use averaging, differencing and deciding transforms to keep information available and in order (this will be discussed later).

<u>Convergence and Shrinkage to Reflect Extracting, Abstracting and Coalescing Information</u>

Most of these layers' transforms reduce the amount of information being processed. Averaging coalesces local information in the raw input scene, and eliminates tiny bits of noise. Differencing and feature-detection focus upon gradients and local regularities, leaving homogeneous and non-feature areas unprocessed. Structure-detectors output internal names of patterned sub-parts that they are successful in finding, and ignore everything else, as without "meaning" to them. Here we have the general process of convergence from sensory surface as information is successively transformed back deeper into the cognitive system - the convergence that is an obvious empirical fact of the sensory systems of animals and has been noted by Barlow (1959, 1969) and others as an important general principle of information-processing in perceptual systems.

The recognition cone handles this by allowing for a "shrinkage" in the size of a buffer as a function of the transform layer being applied. The simplest way of handling this would be to have a standard shrink-size - e.g.divide X and Y size each by 2 - at each layer. The present system allows the user to specify a separate shrink-size at each layer. So he can have a uniform shrink-size, or 2, 3, 7, or anything he desires, or he can use different shrink-sizes for each layer that are appropriate to the different kinds of transforms at that layer. To complete the specification, giving a reasonable first approximation to a sensible model, let's assume that the shrink-size for all averaging layers is 3, for all differencing layers is 1 (i.e. no shrinkage), and for all feature- and for all structure-detection layers is 2. (Note that a simple variant - which has also been coded - allows for separate X and Y shrink-sizes.)

This shrinking means that layers grow steadily smaller in size, until after a sufficient set of shrinkages the output from a transform layer will merge into a single cell, which now contains all the information extracted, abstracted, and coalesced about the input scene. This is the "apex" of a "recognition cone" whose "base" is the Retinal input buffer that contains the raw scene, and whose body is the successive Internal output buffers from the successive layers of Transforms (see Figure 1). The height and width of the input can be got by multiplying all the shrink-sizes together. Thus if shrinkage is the same (S) at all N layers, height = S^N (e.g. $2^{10} = 1024$; $3^8 = 6,561$).

Some Size Estimates for Model and For Living Visual Systems

Let's use these shrink-sizes to help specify the total size of the example 10-layer cone in Figure 2b. Working from apex to base, layers 10,9,8,7,6,5 each have outputs whose X and Y coordinates are each 1/2 as large as their inputs. Layers 1 and 3 each have a shrinkage to 1/3d the size. This gives a total shrinkage of $2^6 \times 3^2$ to the apex, whose X and Y lengths are each 1. Thus the base of such a cone will be 1052^2, or over 1 million cells - a rather hefty size for a program, but still small if we compare it to the 7 million cones or 125 million rods of the human eye.

According to Polyak (1957) the "central bouquet" of the human fovea, contains only 2000 of the thinnest and most one that is designed to handle a) memory search, question-answering and conversation, b) deductive inference, c) language processing, and d) the construction and execution of actions-sequences, as well as perception (Uhr, 1975a). Linguistic rewrite rules, semantic nets, productions for deductive problem-solving, concept formation trees, and associations for memory searches can all be expressed, as relatively simple nodes, using the basic SEER transform.

In order to handle this variety of types of transformation in a simple, general, unified way a number of important modifications had to be made to the typical structure of a recognizer. These are described, discussed, and examined in detail in Uhr,1972, 1973c. Briefly the distinctions between "pre-processing," "feature-extraction," and "naming" have been erased, so that all of these are handled by sets of transforms, and they all output their results into the same kind of next-layer memory buffers. This is in sharp contrast to most of today's systems, where separate subroutines are used to handle the separate stages, and possible names to be output are merged into a special list, to be chosen among when the system switches to its decision routines.

Insert Figure 5 about here

The Parts of a Transform

A transform specifies: A) A set of If-Conditions to be satisfied, B) a Threshold test for success or failure, C) A set of consequences, the Implieds, D) a Region within which the transform is to be applied, and E) a set of Expectations tightly packed cones, and subtends a 20' arc of the visual field. Each cone is 1 to 1.5 microns thick, subtending a 12" to 18" arc. Cones, which are the discriminators of fine detail (and of color), grow 4 to 6 times as thick, and much less tightly packed, away from the center of the retina. The central rodless fovea (subtending 1^o 50' of arc) contains about 35,000 cones, and the 1.75 sq. mm. central fovea about 100,000, along with some rods. The entire retina contains over 7 million cones and 125 million rods. But a pattern like a typed letter is recognizable when it subtends as little as a 5' or 10' arc (look at typed letters, moving back to about 4 or 6 feet, where they become hard to recognize). So a 20^2 array should just barely be sufficient for a single letter, as argued by Uhr and Vossler (1961), and a 200^2 roughly simulates the 1^o 50' rodless central fovea, where we might pack several lines of 10 or 20 letters each. Thus the range of cone-size suggested here, although it does not approach the entire retinal system, is quite commensurate with central foveal cone vision of articulated scenes.

Figure 3 shows several other configurations of layers, from among the very large number that can be specified and tested. The relatively simple 5-layer system which, although it is probably too shallow to serve as a good model for complex real-world scenes, is already pushing the economics of computers to their limits, in terms of space and time needed to make realistic runs.

Insert Figure 3 about here

But note how easy it is to specify a cone whose base is as large as the fovea of the human eye, as shown in Figure 4.

We need only add a few more layers, and/or increase the shrink-size of some of the already present layers from 2 to 3 or 5. An interesting aspect of this playing-around with the details of the structural aspects of convergence is how small our possible range is - from a minimum of 6 to 8 layers if shrinkage is large, to a maximum of 12 to 15 layers if shrinkage is small (albeit with the further possibility of adding more layers with no shrinkage - though I suspect that is pointless, both because convergence appears to be everywhere and uniform, and also because the transforms in a non-shrinking layer can just as well be intermingled with the transforms of the subsequent layer - that is, the transforms may well be useful, but there is no need to give them a special layer of their own).

The nicest thing about these numbers - from 6 to 15 - is their close correspondence to the picture of the depth of transforms in the visual systems of man and higher vertebrates, as got from firm anatomical and physiological evidence as to the retina (with its 2 to 4 synaptic layers), the lateral geniculate (1 layer) the primary visual cortex (about 2 to 6 layers) and the several (see Kaas, 1974) secondary visual cortexes (about 2 to 6 layers each).

In fact this whole converging structure of the cone begins to reflect the overall architecture of the visual system. Information impinges upon the parallel set of cones that serve as the retinal sensors. It is then successively transformed back, with nicely layered synapses across which transformations from tightly packed neurons are computed, to input to the next synaptic layer. In living systems this proceeds in an orderly parallel-serial layered system, from retina into cortex, with successive convergences of information (see Konarski 1967, for an informal verbal model).

We have examined the overall structure. Let's look now at the fine details, and then at the ways in which information flows and is transformed by this system.

Insert Figure 4 about here

THE DETAILED STRUCTURE OF TRANSFORMING CHARACTERIZERS

The Generality of the Basic Transforms

The SEER system uses a very general and powerful kind of thresholding synapse-like transform that allows it to handle the wide range of pre-processing, feature-detection, structure-compounding, and naming needed for recognition and description. In fact the transforms handle several other kinds of processes also, as will be discussed below. They are, further, general enough to serve as the basic building blocks for unified memory structure that is used in a whole integrated cognitive system, (see Figure 5). (For our present purposes we will we will ignore E) Expectations, since they become important for learning and for directed planning, both of which are beyond the scope of this paper.)

A) The If-Conditions include a set of parts, where each part is an internal name of a thing, or a <u>class</u> of things that might be stored in the memory buffer the transform is looking at. Associated with each part is, optionally, a weight (or a probability), a value, and a location relative to the overall location at which the transform is to be applied. (See Figure 6 for some detailed examples.)

Insert Figure 6 about here

Figures 7-9 give some examples of sub-arrays of cells that transforms' If-Conditions look at. For simplicity, only very simple examples are given. For example, 3 by 3 arrays are used for pre-processing, but a realistic system might average or difference over a much larger area. (But note that successive 3 by 3 averagings, using extra layers for each, could compute the same results as a single averaging over larger areas.)

The region at which the transform is to be applied is also specified in the computer program. But an actual hardware embodiment of the parallel system would have another instance of the same transform wired-in at each location within the region. Note that the sub-arrays can be thought of as

Insert Figures 7-9 about here

wiring diagrams of the equivalent parallel computer or nerve net.

B) The Threshold test for success or failure of the transform is built into the present system in the following way: When each thing among the conditions looked for is found, its associated weight is added to a sum of weights for that transform. If that sum of weights exceeds a "threshold" stored for the transform, the transform is considered to have succeeded.

It is quite easy to modify the system, by adding simple 2- or 3-statement subroutines that compute different functions, for example multiplying the inverses of the weights, or tallying the parts found (several such alternatives have been coded). It is also easy to use multi-valued rather than 2-valued (succeed vs. fail) functions, and this too has been coded as an option. But note that a set of 2-valued functions can be used to handle any multi-valued function - simply by having the first insist upon the highest threshold, the 2d insist upon the next highest, and so on.

Insert Figure 10 about here

C) The set of Implieds indicates what is to be done if the transform succeeds (see Figure 10). An implied can be either 1) a name to be merged into the corresponding cell in the next layer, where this name can be either a) internal or b) external, of i) a part, ii) whole, or iii) a quality, 2) a transform characterizer to apply, or 3) an act to effect.

Associated with each implied is, optionally, a weight, and a relative location. The weight is used in the merging, and the relative location is used to project where to glance when the implied is a transform to apply, or where to merge the Implied Name.

Implying Transforms to Apply

Each Implied thing that is a transform to apply is added to the list of transforms to look for now (as opposed to the transforms that are built permanently into the net). These are then looked for at the specified relative locations in the next layer, along with the fixed set of transforms. This gives the system a capability to dynamically "glance about," looking for things that have been suggested by things found so far - something that I have called "flexibility," and described elsewhere, for hierarchical, rather than layered, systems (Uhr, 1973a, ch. 8; 1973b). It also allows for feedback loops from central to more peripheral layers.

A number of variants are possible, several of which have been coded. Rather than apply the dynamically implied transforms at the _next_ layer, they can be added to the list of transforms applied at the _present_ layer. But both of these are unrealistic in a system that is handling scenes that are input over time, since the implication of a transform and its subsequent application are serial operations that themselves take time. So it seems better to have implied transforms applied at the _next moment_ in time (or, alternately, after whatever interval of time needed). Here again, several alternatives are possible: The transform can be applied at the next layer, the same layer, the layer specified along with the transform, prior layer(s) or every layer.

This depends upon and interacts with the mechanisms used to store information over time, letting some things gradually fade away, but reinforcing some things, as a function of change and newness, motion, and having been "noticed" by transforms - so that they become salient and stay around-as discussed below.

Merging Implications Using Averaging Transforms

Implied names are merged into the corresponding next-layer cell. But as the cone collapses, neighboring implications of the same thing should merge together. This is handled by the same "averaging" kind of transform used in the first few layers to smooth over local noise. So routinely, at each layer the system is given a transform that averages over all things. Alternately, information can be culled by setting a threshold, so that weakly implied things are eliminated. (This is probably crucial for realistic tests, since an enormous amount of information can be accumulated and passed along by such a system. Here is one of the major parameters for adjustment. Probably the best way to handle it is to have a fixed storage capacity, C, at each cell, and keep only the C most highly implied things. But such a sorting process is costly, either in computing time or the extra comparator networks needed.)

Triggering Choices in Sub-Regions

Among implieds can be the action of deciding among the various possibilities in the cell for which this action is triggered. Along with this action is, optionally, the name of a class among which the decision is to be made. (If no class is named a grand decision is made, among all the things in that cell.) Thus if the class is e.g. EXTERNALNAME it will choose among all such names, if it is COLOR it will choose among all implied colors, if LETTER it will choose among all names that belong to the letter class, and so on.

This choosing action is itself implied, and can be implied by a variety of kinds of information. Among these are the following (all of which can be expressed as characterizers): the fact that several members of the class have been implied, and/or some of them have been implied above some value, or sufficiently above the value of the other members of the class; 2) the fact that an object or members of its class have been implied in this cell, but not in adjacent cells (suggesting that a region containing a figure has now been collapsed, and is surrounded by background) can be got by differencing for that object, or for the whole class; 3) the fact that a sufficient number of transforms have been applied, or that a sufficient layer-depth has been reached. All of these can imply a trigger-transform with a certain weight. The trigger will go off, implying the decision-transform when a second simple transform looks at it and finds its combined weight has now exceeded threshold.

When such a choice is made, it is tagged as chosen, and its location stored. (Alternately, chosen objects are put on a FOUND array, rather than being passed back through the cone.) A sub-cone is then projected, using this location as its apex, giving a sub-region of the larger cone as its base, where the chosen object is inferred to lie.

D) At each layer, the system applies the set of transforms indicated for that layer. But each transform specifies a sub-region in which it is to be applied.

Many transforms will be looked for everywhere, as though there were copies of them everywhere in a parallel system. Averaging and differencing transforms will be of this sort, looking everywhere. Simple gradient and edge detectors, and other low-level feature-detectors, will probably be of this sort.

At the other extreme, the transform can be specified to look at only one spot. (Note that this does not mean that the transform looks into only one cell, for each of its parts has a location specified along with it, which tells where to look for that part <u>relative</u> to the cell location where the transform as a whole is being applied.)

Looking at only one spot makes sense only when the system has good indications exactly where something might be - for example deep into the cone, where a projected part of a large whole, e.g. the head of a stick-boy, can be fairly precisely specified. More typically, the characterizer will be applied within some small region, so that slight variations will be ignored. This is effected very simply, by specifying the bounds of a small rectangle through which the transform will be iterated, rather than through the entire layer. Note that an Implied transform (e.g. those glancing about to look for the rest of a partially recognized figure) is projected to a region relative to the location of the transform that implied it.

E) The structure of the characterizer is actually more complex than the present description indicates. But these complexities will not be described here, but deferred for a subsequent paper, because they are of importance for the learning and interaction with more central cognitive processes that go on in a more complete whole, integrated system. Briefly, the actual form of a transform is what I will call an Hypothesis, which contains Antecedent conditions (which are sets of related things), Evaluation of success, Consequences, and Expectations.

Expectations are useful when the consequences are themselves high-level _acts_, where an act is performed upon the external world as a succession of specific _actions_. Now expectations 1) monitor that the step-by-step unrolling of the act is followed by anticipated consequences - e.g. the object being moved toward indeed appears to be getting closer, and 2) await reinforcement or other feedback-related indications that the act was indeed successful, and thus lead to appropriate learning.

Handling Color, Shades of Gray, and Other Qualities

This system can handle black-white, shaded, or colored scenes. Information about color and intensity is input in the form THING-VALUE, (E.g. RED-5, BLUE-4, YELLOW-0 all in the same cell might indicate PURPLE. INTENSITY-9; INTENSITY-8; INTENSITY-5; INTENSITY-2 in _adjacent_ cells might indicate a local fragment of an edge). Now the standard characterizing transform can be used just as easily (albeit with more transforms needed) for colored as for black-white scenes.

To handle shades of gray, the system can use the same transforms as for black-on-white scenes, since it must already handle a range of values after the first layer. The basic averaging and differencing operations turn black-white scenes into internal representations with shades of gray, and they can just as well handle gray-scales in the retinal buffer also.

To handle color we need only give the system transforms that refer to three parts - the three color components - in the <u>same</u> cell. (This is in contrast to all the transforms we have examined so far, which look in <u>different</u> cells, for spatially separated parts; now the transform looks for several dimensions stored in the <u>same</u> cell.) E.g. a transform that looked for RED in the range of 5 and YELLOW in the range of 5, and BLUE in the range of 0 would, when it succeeded above threshold, output "ORANGE" with some fairly high weight. Alternately we can assume 3 types of cones, as seems to be found in living systems, and have a transform combine them into a resulting color - possibly using a separate layer.

Other qualities - e.g. ROUGHNESS, FUZZINESS, BOLDNESS, or even BEAUTY, might be names implied by transforms that assessed appropriate characteristics (e.g. texture, or the value with which a thing is implied, or its relative balance or unusualness). They could then be used as parts of higher-level structural characterizers. E.g. ROUGH and BOLD might imply FORCEFUL.

or any characterizer reported in the literature or desired, whether or not it can conveniently be effected by the basic nerve-network-compounding mechanism that is basic to the system.

Often this will give us a function that could be computed, albeit more cumbersomely, without this capability - e.g. a contour can be detected by a serial curve-follower, or by a hierarchy of edge features. Even though such nerve-net computations are less efficient and more cumbersome, they seem desirable, since they make use of the basic neuron- and synapse-like building blocks of our model, and keep things plausible. But we always have the option of additional types of functions, if only to improve efficiency, often a crucial practical factor. And when a compelling need for a new function is felt, this can be the beginning of a new search for brain capabilities and mechanisms needed to compute it.

N-Place Relata as Transforms

Rather than have If-Conditions that are compounds of several parts, we can specify a relata, which is a relation over N sub-parts, and specify a whole set of such relata for a single Transform. This allows for the convenient building of graph-structures of any level of complexity as the description of the If-Condition to be looked for. But, since weights will now be associated with the basic parts of the relata, and also with the relation

Relata and Functions as Characterizers

Three additional levels of power have been given these transforms.

Dynamic Adjustment of Characterizers' Descriptions

First, the parts of the characterizer's description can be functions of one another. When one part is found, its specific value and attribute are temporarily stored, and used to adjust what subsequent parts to look for. For example, if a faint (in that it has a low intensity value) vertical stroke is found, other faint strokes are looked for. If the color in the cell where the part is found is dark red, dark red is favored as the color for other objects.

This dynamic adjustment seems quite desirable and powerful. But it violates the parallel nature of the system, for it would take serial time to signal and effect such modifications in a characterizer. The same thing can be done without this mechanism, by implying new characterizers to apply, at a slightly later moment in time, and adjusting these characterizers accordingly.

Arbitrary Functions as Parts of Characterizers' Descriptions

We can designate any part of any characterizer as being of a special type, so that the program will branch to a special subroutine that computes that function. E.g. we can introduce a curve-follower, or a color-compounder, and the parts as part of the relata, the system can still specify thresholds that allow for success of the characterizer even though not all of its relata succeed. This gives flexibility, since not all parts of such a description need be present.

Further, the implied things can now be relata. This is crucial for the larger SEER system, where implieds must often be acts to be effected, and an act is coded as a relata of the action to be effected and its objects. But for scene description the system can make use of a few simple types, to trigger the built-in acts of merging the implied into the next layer, or onto the list of new characterizers to apply. This can also be combined with the dynamic adjustment capability, so that the parts of the relata can be adjusted as the relata is matched. And the implieds can be adjusted as a function of what has been found in the matching of the description. This generalizes what is already being done when relative locations are projected.

OVERALL FLOW OF INFORMATION THROUGH THE SYSTEM, OVER TIME

Fixed and Dynamic Characterizing Transforms

Information is successively transformed from the base, to the apex of the cone. This is done chiefly by characterizing and "pre-processing" transforms that are built into the layers of the cone, many of them iterated as though in parallel throughout the layer.

It is also done by a dynamic set of transforms to be applied, where these transforms, and the relative location at which they should be applied (and thus used to "glance about") are implied by previously successful transforms. Note that these dynamically implied transforms are hard to interpret anatomically: They are not a part of the cone; rather, they must be merged into dynamically built-up and faded lists, one for each layer of the cone. But such dynamic lists are certainly necessary for more central cognitive processes, and it is interesting to find them indicated and helpful even at the earliest layers of perceptual processing.

The Many Uses and Types of Transforms

Averaging transforms serve, especially in the first few layers, to smooth out noise. But they also serve the crucial function of merging together the implications of the same object in adjacent regions, as these regions shrink from layer to layer. Differencing transforms serve to find gradients and edges, and second-order changes in them, in the earlier layers. But they also serve to help in triggering choices among alternatives in local apexes of sub-regions.

Feature extractors can work at a low-level concrete thing (e.g. building up a long line or simple angle), and also at high-level compounds (e.g. face, stick-man, family, or a sentence). For each part they look for can be any kind of thing - an intensity, a gradient, an internal name of a part, a structure, an object, or an abstract thing like a class or a quality.

The Numbers of Transforms in Each Layer, and at Each Point

The number of transforms applied in each layer, and at each point, can be varied, and this is one of the prime questions for experimental exploration. Overall architectural considerations probably suggest that the total number of processes (neurons and transforms) remain roughly constant. (The fact of moving from the surface-skin of the system toward the center suggests there may be less and less space, so that the total number in each layer, moving inward, should lessen. On the other hand, it is reasonable to assume that as they become more precious components can be miniaturized - as appears to be the case in the cortex. So, we might make a rough first approximation that these two factors cancel one another.)

A reasonable set of values seems the following: The first averaging and differencing layers need only <u>one</u> transform per layer - the single averaging or differencing transform - iterated at every point. The first feature-detecting layer, if it detects something like short line-segments, might use about 4 to 16 detectors at each point, one for each of 4 to 16 slopes, or other primitive features. It seems reasonable to limit the number of transforms at any point to 5, 10, or at the most 20. This suggests assigning each to a smaller region, as the possible number increases and, probably most important, flexibly implying them dynamically. See Figure 11 for a range of examples.

The more cells each transform looks at, the less space there will be for transforms, as wires begin to crowd one another. So we can think of a trade-off between number and complexity (in terms of input and output wires) of transforms. But deep-layer transforms can be quite complex with few wires, because the things they look for are themselves extremely complex.

Insert Figure 11 about here

Time, and the Fading and Reinforcing of Memory

Up to this point I have described the behavior of the recognition cone as it processes a single moment of time, for example a single movie frame, the single 30 milli-second scan of a tv camera, or a human glance. Now let's assume that a new scene - a next moment of time, in the next movie frame or tv scan - is sensed into the input memory buffers taking, let's say, another 30 msec. Let's further assume that it takes the same amount of time, say 30 msec, to perform each set of parallel operations needed to transform information from each Ith to the I+1th buffer.

(This seems a plausible assumption, since each of these represents one serial transformation of the many parallel elements in that layer, across something equivalent to a synapse - and it seems reasonable that all such 1-layer synapse crossings would take about the same amount of time.)

Now, rather than replacing what was in each buffer by the new information got from the new scene, the model must merge this new information in. This is done in a very simple way (see Uhr, 1973b, 1975c for details).

Merging takes place as follows:

The first time a new thing is implied, it is entered into its output cell with a relatively <u>high</u> initial weight. This serves to make salient both new things and moving things (since they will also be new to that particular location). Whenever a thing is implied (e.g. from parts, context, or higher-level wholes) its weight is up-ed. A general tendency toward forgetting is given the system, by having everything fade away slightly at each moment of time. (The initial weight, fade, and up weights are all parameters.)

Things can be implied from many sources. Characterizers can imply possible things. Things can imply related things. And internal sources can make things salient. For example, an internal need like hunger can imply a class of objects, like food, that will satisfy it; food might imply banana, apple, and cracker; banana might imply yellow and certain curve features, etc. A transform can focus the system's attention, by implying characterizers to apply to the input scene, and also imply, probably with only a small weight, the presence of suggested objects, slightly enhancing and making salient any such implications that might exist from partial recognition. Thus, depending upon the content of the specific memory net given to, or learned by, the model, salience can become a function of a variety of different external and internal sources.

Thus the salience of something in memory, and hence the length of time it remains in memory depends upon how recent it is, how long it has remained present in the sensed external world, how much it has moved, and how much it has been implied by other things, both external and internal. Note how this applies not just to the Retina or to the Apex of the cone, but also to the Internal memory buffers at every layer.

Transforms that are merged into the lists of characterizers to apply next are also faded over time, and made more salient to the extent that they are repeatedly implied. Here, as elsewhere, it is interesting to play around with the fade and up-weight parameters. (Learning of the Long-Term Memory also follows much the same algorithm, with different up-weight and fade functions, so that information is not forgotten so fast.)

Depending upon the specific up-weight and fade factors, information will be forgotten faster or more slowly. It must remain at least long enough to a) allow for recognition of objects and scenes that unroll over time - for example a spoken sentence - and b) keep enough of the recent past in memory so that when feedback is recognized, as referring to an act in response to some prior scene, the system still remembers that scene and that act, so that it can learn from this episode.

Time to Characterize, Time to Glance About

This leads to rather subtle, and rather nice, time effects. When a characterizer implies the possible existence of an object in a certain direction, the system determines where to glance next, and what further transforms to apply, to confirm or deny that possibility. This process will take several moments of time. (When perception is embedded in a larger system that actually glances by moving the eye, head, or body about, rather than simply moving the "mind's eye," this will take a good bit longer still.) Thus the new characterizers will be applied to a scene that is slightly changed, as a result of the passage of time.

To be successful, characterizers must be implied fast enough so that they will still be pertinent. Fading of the recent past must be slow enough to hold whole episodes "in mind." The external world must not change overly fast with respect to the system's internal processes and internal clock. Here we begin to see a subtle and intimate coupling of internal and external time. We should expect and be able to count upon such a coupling. For after all living systems evolve processes that respond to externals fast enough to derive some benefit - e.g. by catching and eating the fly before it dies of old age, and even before it buzzes away.

Outputting a Description

The apex of the cone and the FOUND list contain a rich variety of descriptive information, including all the triggered choices, along with their locations and sub-parts, qualities and characteristics, and all partially-implied things. We could simply have all this output, as a "complete" description, or have the 1,2, or N most highly implied things output. But what is really needed is a pertinent description, and that entails developing a concept of what the hearer of the description is interested in finding out (see Uhr, 1973c, 1973e).

For the moment, SEER gives a hearer a simple "conversational" capability to make stylized requests for more objects, and/or more information about the parts or qualities of already-named objects.

PSYCHOLOGICAL AND PHYSIOLOGICAL PLAUSIBILITY OF THE SEER MODEL

Many aspects of the physiology and psychology of the visual system have already been discussed, as part of the rationale for SEER's overall architecture, transforms used, and flow of processes. It is still premature to talk about details, since they depend upon specific features and parameter values. But the overall fit of model to data, and the variety and breadth of phenomena SEER at least begins to handle, seem to me to make it a reasonable first-approximation toward a model of form perception in a changing environment.

A model for so complex a system, where the model itself has the large number of degrees of freedom of a complex computer program, should predict to a very large number of data points, of a wide variety. Such a model does not have the clean simplicity that we are used to in the physical sciences. But it must be _relatively_ simple - as simple and elegant as possible, given the complexity of the domain being modeled. This must ultimately be shown as exhibiting its _relative power_. That is, as the model becomes less simple it must exhibit relatively more power. Otherwise we can fall into _ad hoc_ models with as many degrees of freedom as data-points predicted.

Two general types of data should be fitted. First, the system should perform the variety of tasks for which it was built (in our case, perception of objects and description of scenes). If we want it to model perception in humans or a specific animal, it should do this only as well as that animal, and also exhibit the confusions and peculiarities (e.g.illusions) of that animal.

Second, it should look like that animal's perceptual system, in overall architecture and in details.

The preceding discussion has been directed to the second point. Extensive testing and tuning will be needed to examine how well this system can handle a wide variety of perceptual tasks. But I can only note that it is designed for flexibility and generality - two prime attributes of human perception - and has a minimum of built-in procedures or transforms. (In fact the hope is that as many as possible of the transforms will be learned.)

Let's look now at some criteria for animal-like behavior that have been posed in the literature.

N.S. Sutherland's Criteria for a Model of Form Perception

N.S. Sutherland (1968) suggested 12 criteria for a model of form perception, as follows (see Uhr, 1975b for tables evaluating other research on these and other criteria):

1) Invariance over Size

The shrinking layers of the cone handle size invariance in an especially simple way. We can think of each layer as another step in trying to focus the image of the object so that it fits the high-level structure-characterizing transforms that imply that object. Assume that these transforms are repeated in several layers. If they do not succeed in the first layer, because the object is too large, they will succeed in some later layer, where the object is the right size. Thus the layer in which they succeed gives a direct estimate of the object's apparent size.

2) Invariance over Retinal Position (Translation)

This system (like many others) recognizes and describes objects no matter where they lie in the larger field. This is very simply effected, by having the characterizing transforms iterated throughout the field, much like Hubel and Wiesel's (1962) "simple" features. The successive shrinking of the cone, and convergence-by-averaging then merges the implied object into the apex, from whatever position, giving the effect of Hubel and Wiesel's (1965) "complex" and "hypercomplex" cells, without programming them explicitly. Alternately, higher-level transforms can look for the simple features everywhere, acting like complex cells directly.

3) Invariance over Brightness.

The differencing transforms are sensitive to gradients, and to changes in gradients, rather than to absolute intensities. Thus forms are recognized no matter what the brightness of their parts, so long as there is sufficient contrast.

4) Equivalence of Outline and Filled-in Shapes.

This depends upon the differencing transforms used, and also upon the structural transforms that build up higher-level shapes from edges. An outline will give both an outer and an inner edge; a filled-in shape only an outer edge. The higher-level structural transform can specify combinations of either or both.

5) Invariance over Small Rotations (but not Large Rotations)

The basic configurational characterizer used, where each part of the configuration is a relatively sketchy representation of a segment (e.g. line, curve, angle) of a figure, appears to have roughly the right amount of invariance over local noise and jitter, and also over small rotations, as indicated by living systems. Hubel and Wiesel (1962, 1965) find edge detectors sensitive to slopes within around 10^o of change. In humans recognition appears to begin to fall off around 25^o of slope change (but here we are talking about recognition of things like letters or pictures that are made up of a whole complex of simple line segments). Depending upon how schematic the basic parts are made, this range, from, say, 10^o to 45^o seems just about what structural threshold transforms will give.

On the other hand, with learning, an alternate set of transforms will be developed to handle the different rotations. In a learning experiment they might be developed over a short period of time. More realistically, they would have been learned in prior experience, but have low weights and rarely be used (this again needs a more central mechanism that has access to and dynamically implies transforms from a much larger set than those built into the basic recognition cone) but are available for fairly quick up-weighting and use when indicated, as when we adjust ourselves to reading a book upside-down. Note that these are structure-characterizers at a high level, ones that make use of the same relatively small set of basic strokes, so that there is no need to store very many additional transforms for rotations.

This fits well with the whole spirit of parallel redundant threshold transforms for recognition over distortions and noise. The size of the transform set needed to handle the very difficult non-linear variations is already large, and will be expanded only a relatively small amount more to handle the simpler linear rotation.

6) Confusions Between Different Shapes Should Mirror those of the Animal Being Modelled.

This will depend upon the specific set of transforms used, and that in turn should depend upon the specific animal being modelled. But to the extent that edge, curve, angle, loop, etc. feature-detectors and structure-characterizers are plausible we should expect reasonable similarities.

At some point we must tune (or teach) our model, by finding where it goes astray and using this information to help improve upon the details of the transforms.

7) Invariance over Local Distortions and Jitter.

This has already been discussed under 5) above, and seems to be a basic strength of the probabilistic thresholding structure, where specifications are loose, and can be satisfied by a variety of fragmentary, loosely-positioned information.

More important is invariance over more global, non-linear and unknown distortions (e.g. all the different A's, or expressions of the same person's face), which will be discussed below.

8) Segmentation of the Same Picture in Different Ways.

This is done quite naturally when partially-overlapping structures are built up. SEER will at some point decide among these, and often it will be wrong. When the decision is close we might if we wished call it an ambiguous situation. Much empirical testing would be needed, with a specific set of transforms, to determine how similar SEER's difficulties and ambiguities are to those of people.

9) Recognition of Complex Scenes should be Possible, but More Difficult.

This seems to be the case for SEER, as well as for most systems that attempt to recognize objects in scenes - which is the more difficult task - as well as when isolated. But SEER is able to make fairly nice use of information from a variety of contextual sources, and can direct its attention toward different aspects of the scene. These are the sorts of things that seem part of a reasonable scene-describing system.

10) Perceptual Learning.

SEER has fairly sophisticated learning capabilities, but these will be described elsewhere. (Uhr, 1975c; for related work see 1973a; Uhr and Jordan, 1969). In a word, it tries to discover and learn the feature- and structure-detectors at the later layers, as well as inductively learn good weights for them. (The earlier layers are assumed to be a result of much slower "evolutionary learning.")

11) Redundancy Should be Taken Advantage of.

This is basic to SEER's use of compounding probabilistic structures. It is expected that objects will be varied, noisy, and fragmented. Alternate, partially overlapping and redundant feature- and structure-detectors are used, so that no specific parts or features need be present.

This is where the more traditional syntactic and structural recognizers fall down, since they must have everything in place. A basic assumption behind SEER is that a large part of any object (apparantly for real-world-recognizable objects, well over half) can be distorted, fuzzed-up, or eliminated entirely, without having much effect on recognition.

12) The Model Must Satisfy Physiological as Well as Psychological Evidence.

This again seems to me one of the strengths of SEER, as discussed throughout this paper (see also Uhr, 1975a).

In contrast, systems that use curve-followers, long chains of serial processes, or directed deductive inferences seem quite implausible as perceptual models.

The anatomy and physiology of living visual systems is clearly parallel-serial, with 3 to 5 layers in the retina and lateral geniculate, another 3 to 6 in the primary visual cortex, and then several more additional projections in subsequent visual cortex areas. The basic transform in the brain is the synapse, which appears to be a threshold element into which tens, hundreds, or even thousands of neurons fire. The human eye has the fovea of about 7 million shape- and color-sensitive cones at the center of the retina surrounded by over 100 million change- and motion-sensitive rods. And information from the two eyes merges in the cortex to give binocular depth perception. There appear to be a number of feedback loops, and lateral connections within single layers.

The recognition cone does not capture all the details of the visual system. But it attempts to abstract the essentials.

Additional Criteria, for Color, Time, Descriptions, Context

Several additional criteria should be examined, including:

13) Invariance over Unknown Non-Linear Transformations

This is the most complex and difficult type of distortion, and also the kind that has motivated much of the structure of this model, which, as the entire paper indicates, is designed to handle real-world variations, in all their unknown richness.

14) It Should Handle Shaded, as well as Black-White Scenes

15) It Should Handle Colored Scenes

16) It Should Handle Scenes that Extend over Time

These are all extensions to higher-dimensioned scenes that recognition cones are especially designed to handle, as discussed elsewhere in this paper.

17) It Should Handle 3-Dimensional Scenes

The third spatial dimension, giving depth, is the extra dimension that "robot vision" research has focused upon, because of the need for the robot to interact with solid objects in a room. Very powerful techniques have been developed for inferring straight-edge solids from information about homogeneous regions, even when solids overlap one another. They use elaborate serial projective geometry computations that seem very different from anything a living visual system appears to do. Recognition cones are weaker at this. But they can be given structure-characterizers sensitive to gradients and perspective in the depth dimension.

And it is not clear how much depth perception we should expect from a single eye, especially if we want to model living systems. It seems more reasonable to use <u>two</u> cones, one for each eye, converging, as is done in living systems, into a single apex that makes use of binocular cues for depth, rather than trying to infer everything from the single monocular 2-dimensional projection.

18) <u>It Should Give Appropriate Descriptions</u>

It is not at all clear what we mean by a "description" (for examinations of this complex concept, see Uhr, 1973c, 1973e; Firschein and Fischler, 1971; Fischler, 1969). But recognition cones can give the sort of "standard description" of the whole objects and their structures, given a suitable set of transforms with suitable weights assigned to their implications. Or the cone can be put into an interactive conversation with the "hearer" of the description, who now can demand those aspects that he deems pertinent (Uhr, 1973e). This, I would argue, is only a first step. Good descriptions, much like good question-answering or conversation, will depend upon larger systems that are capable of building models of the hearer - the other system in the conversation - and the import of his demands and the pertinence of the descriptive information.

19) <u>Recognition Should be Directed from Within as Well as From Without</u>

As discussed elsewhere, back-linking from needs and expectations to lower-level structures and transforms that might imply them allows this system to mix inner- and outer-directed activities. This is effected in a natural way because recognition continues over time.

20) <u>The System Should Handle a Variety of Contextual Interactions</u>

Any transform can imply anything, including other transforms to apply. Thus, e.g., a structure-characterizer that implies the letter "D" can also imply (with a lower weight) a larger whole, e.g. the word "DOG" and/or a separate entity, e.g. the letter "O-to-the-right." The word "DOG" can back-link to imply the letter "O" which further back-links to the perceptual transforms that imply it. Moving deeper into the memory net, beyond the perceptual transforms, still other words in the immediate context might be "FOOD" or "SAUSAGE" (or simply imply them), and these in turn would imply "HOT DOG" as the meaning of "DOG," so that the system would search for the word "HOT" to the left of "DOG." And so on.

Generality, Efficiency, Power, Predictability

There are several important overall criteria:

21) How well does the Model Work?

That seems to me a crucial question. Being able to name and describe well is as pertinent as being made of the proper kinds of nerve nets, or exhibiting some of the human's peculiarities.

This model can handle a great variety of perceptual tasks. It must be tested extensively, using a good set of transforms. But it seems at least as likely as more deterministic and less flexible systems to give interesting results.

22) There Should be no Theoretical Limits on the Model's Generality

Minsky and Papert (1969) have shown that Rosenblatt's "1-layer perceptrons" and Selfridge's "1-layer pandemonia" can become grossly inefficient if asked to recognize global characteristics, like "connectivity." They have further implied, and many researchers have mistakenly thought that they had proved, that multi-layered systems, or indeed any kind of network systems, had similar grave problems. But this implication is clearly false - just look at the multi-layered anatomy of the eye-brain network! In any case, as they mention themselves (e.g. "a universal computer could be built entirely out of linear threshold elements," p. 231), Minsky and Papert are only raising issues of efficiency, not of generality. Their results have been widely misunderstood to suggest that any kind of hierarchical network of any kind of characterizing transform suffered from the proved limits of "1-layer perceptrons." But they proved only that strictly parallel systems are inefficient in the size of the threshold element needed to compute a complex function <u>at a single step</u>. In contrast, a strictly serial system will give the extreme of inefficiency in terms of the <u>time</u> needed to compute that function. Mixed parallel-serial systems are needed, whenever it is desirable to try to optimize over both space and time, to decompose a single complex function into a hierarchical network of simpler functions, where the depth of the hierarchy determines the amount of time needed. This is a commonly used technique, whether in parsing sentences, building complex switching circuits, or computing complex functions. It also appears to be the way brains do things - as evidenced by the architecture of parallel layers that serially converge toward the cortex, and by our subjective intuition that parts hierarchically compound into larger and larger wholes (e.g. strokes into letters into words into phrases into ideas).

Despite Minsky and Papert, there is no reason to think such systems are inefficient. In contrast, the strictly serial extreme that they propose suffers from grave time inefficiencies, just as the strictly parallel system suffers from grave space inefficiencies.

In any case, inefficiency and not generality is the issue. A network of threshold elements with input and output is equivalent to a Turing Machine, and therefore has all the potential power of a general-purpose computer.

The hierarchical parallel-serial structure of recognition cones is designed with efficiency a primary consideration. The building of larger structures from smaller is probably the best (and also nature's) way to get complexity.

Minsky and Papert suggest as the alternative "serial algorithms." Most of the recent "robot vision" research has been of that sort, where long necessarily-serial chains of computations are needed to match complex graph-like structures. As pointed out elsewhere (Uhr, in preparation), the advantages of serial processing can be got in a mixed parallel-serial system - but without paying the terrible price of the excessively long times and careful processing needed for strictly serial processing. In nature the perceiver must act as fast as possible, or die; and it must contend with all of nature's distortions and dissimulations.

PARALLEL VS SERIAL VS PARALLEL-SERIAL SYSTEMS

This is a parallel-serial model. The perceptual part of SEER is organized into the layered cone. Central memory does not have such well-delineated layers. But central processes are also parallel in their associative search out into a hierarchically organized memory network. This search continues serially over time, as directed by a melding of what is expected and desired, and what is implied by the things that have been found so far.

Parallel processes give robustness and speed. The large number of partially redundant transforms can recover from the errors of any particular transform. As scenes become large and complex it becomes increasingly important to apply transforms in parallel rather than one after another, in order to recognize things fast enough to manipulate them. And living animals with well-developed perceptual systems are extremely parallel, especially at the surface of the body.

Serial processes give direction, and reduce the total number of computations. Things implied so far can imply other things to look for next. Rather than make a single parallel search for a complex structure and all its possible real-world variants, the necessary transforms can be decomposed into a much more efficient set of hierarchies of simple transforms. Implications can come from any kind of contextual source, including any aspect of the scene, the recent past, the memory store, or higher-level possibilities. The mixing together of parallel and serial processes thus gives a number of advantages.

PROBABILISTIC VS. DETERMINISTIC SYSTEMS

The use of weights and probabilistic decisions allows for a simple and realistic system. Many partial implications of each possible thing can be merged together, and then the most highly implied among alternative possibilities chosen. This appears to be what goes on in the thresholding elements, the synapses, of living brains.

In contrast, deterministic systems must make very careful serial searches in order to try to match portions of the scene with descriptions of structures stored in memory. This can take a great deal of time, and necessitate the use of complex algorithmic or heuristic searches. All possible variations of real-world patterns must be anticipated with additional tests and intermediate decisions in the matching process.

Most researchers have tried to handle perception in a probabilistic way, and deductive problem-solving in a deterministic way. The determinists are beginning to try to handle perception, and the probabilists are similarly moving into problem-solving. It will be interesting to see which is the more suitable approach, or whether they must both be used, and if so how they can best be combined.

The SEER model is, of course, an attempt to use a probabilistic, parallel-serial approach.

A QUICK COMPARISON WITH OTHER PROGRAMMED MODELS

The parallel-serial probabilistic architecture of recognition cones stands in sharp contrast to the serial, usually deterministic structure of most programs that attempt to do scene description by using "syntactic" or "graph-matching" techniques. (For a detailed examination of different systems, see Uhr 1973a, 1975b.)

Briefly, the recognition cone builds up hierarchies that give higher and higher level structures. It expects redundant and fragmented objects, so it uses weights and thresholds, rather than insisting that everything be present, exactly as specified. It uses the same network-flow of the same basic type of transform to let a wide variety of sources - including what has been found so far, and also contexts and internal needs and expectations - imply, in parallel, what might be there, and what to look for next. There is thus a complex flow of processes, simultaneously from outside and inside, that is organized by the overall parallel-serial converging architecture.

Processing over time gives the serial depth needed for longer chains of transforms. The ability to decide when to choose among possibilities allows the system to assign descriptions to sub-regions of the larger scene. Decisions are expected to be less-than-perfect, but the system tries to do as well as possible, under a basic assumption that there will be error, and even ultimate ambiguities. "Specific knowledge" is built-in (or learned) as the specific-transforms - just as specific "rewrite rules" give a parsing system a grammar for a specific language.

In contrast, most scene describers build "syntactic"-like or "graph-like" structures over the scene. They therefore must assume either extremely clean, simple scenes of objects that do not vary, as in the robot research, or powerful pre-processing routines that have turned noisy and distorted real-world scenes into regular graph-like drawings of nicely connected line-segments, as in the "syntactic" systems. These systems use "parsing," "graph-matching" or "heuristic search" techniques to perform long deductive sequences of processes that attempt to match parts of the scene with parts of internal graph-representations. It seems far harder to interpret this latter approach in terms of the neurophysiological architecture of nature's eye-brain systems, or in terms of the living eye's ability to make use of, or do without, redundancy, in the way it handles noisy, distorted, and grossly fragmented scenes.

A COMPARISON WITH MORE TRADITIONAL PSYCHOLOGICAL AND NEUROPHYSIOLOGICAL MODELS

Black-box models like Hunt's (1973) or Norman and Rumelhart's (1970) (see Figure 12), which are among the most detailed, are far too vague to be useful in specifying mechanisms. They encompass all of pattern recognition and scene description in the box that extracts features and the next box that assigns names. But that is so general as to include all pattern recognizers, and is nothing more than putting the statements, "Input the pattern, extract features, name the pattern" into separate boxes with arrows between them. Worse, it obscures the crucial issues of the overall structure of features, so that we get no idea of the possible virtues of hierarchical parallel-serial systems, or why they are needed, or how they can be organized.

It is also dangerous to ignore the issue of what the features are. That is a reasonable tactic for awhile since, especially when we concentrate on the more central processes, we can always simplify our inputs to the point where only a few obvious features are needed (e.g. if we input only one perfect square, circle and triangle, always using exactly the same pictures, then a few edge, curve and angle detectors or even whole-templates will be enough). But that sweeps the most difficult part of the pattern recognition problem under the rug - the recognition that an object is the same over widely different and unknown transformations and under all kinds of noise conditions. And it seems crucial to handle that problem at the same time as our more central problem, since the two interact intimately, control and direct one another, use the same mechanisms, and throw light upon one another.

Figure 12 about here

Similarly, the Anderson-Bower model (1973) ignores all the actual scene recognition, assuming that this is done, as though by magic, so that the recognized letters are now in short-term memory. It then very carefully traces out the graph searches needed in order to match them with representations in memory. But the amount of transformation needed just to recognize that scene, even if it is done in as parallel a manner as possible, is almost certainly longer - probably 5 or 10 times as long - as the final match that they so carefully trace out. So timing estimates will almost certainly be quite unrealistic.

More traditional neurophysiological models, like Young's (1964), Konorski's (1967), and Eccles' (1973) fill in many interesting and important details at the level of individual neurons and synapses, and make provocative suggestions as to overall structure. But they still seem to suffer from the inevitable sketchy and incomplete nature of verbal descriptions. Deutsch, 1955, 1962, Sutherland, 1968, and Dodwell, 1957, 1970 have made major steps in filling in gaps and stating things fully, and begin to approach the precision, self-consistency and completeness of actual computer-programmed models. (I should note that Stevens, 1961, programmed her version of Deutsch and the result was, I think it is fair to say, a fairly typical pattern recognition program for naming a single isolated object.)

We might better think of these verbal discussions as general frameworks, or points of view. It is almost impossible to judge whether, or how well, they will work on the great variety of patterns that must be recognized. And they say virtually nothing about the mechanisms for handling moving or changing objects, or for perceiving scenes of several interacting objects.

The computer-programmed models can begin to fill in the details of the sketch, and grapple with the problems of the actual mechanisms, their interactions and overall structure, and the ways in which they fit and work together. They can also begin to bridge the gap between neurophysiological networks and psychological functions. It is here that a model that uses parallel-serial structures of synapse-like transforms to perform the perceptual functions of recognition and description appears especially appropriate.

ICONIC, SHORT-TERM, AND LONG-TERM MEMORIES

The present model was designed holding in mind primarily 1) the perceptual problems of naming and appropriately describing objects that can vary widely from instance to instance, 2) the functions that seemed necessary and desirable to handle naming and describing as efficiently and as elegantly as possible, and 3) the physiological macro-structure of the layered parallel-serial visual system of retina, geniculate, and cortex, and the physiological micro-structure of neurons synapsing on other neurons, giving threshold elements that have the properties of lateral inhibition, edging, and the local feature-detectors that have been found in living eyes.

There was no special desire to designate functional black-boxes like Iconic Memory (IM), Short-term Memory (STM), or Long-Term Memory (LTM) of the sort information-processing psychologists have posited, and examined experimentally (see Reed, 1973).

Memory was kept as simple and homogeneous as possible, with complexity, if needed, forced upon the model by the problems that must be handled. That is, I wanted to add mechanisms only as compelled by the problem, rather than superimpose a priori conceptions upon the model.

I should point out though that from the problem it follows quite simply and obviously that a) inessential information from the present input is discarded during transformation into a description, b) information from the immediately past inputs must rather quickly fade away and be got rid of, it only to keep from clogging up memory stores, and c) some information must be kept relatively permanently, in order to characterize and recognize. So this varying durability of information will surely lead to more of less transient memories to which, if we are so inclined, we can give names like STM, LTM, 2TM, 3TM,.... (My inclination was and is to have a continuum of some sort.)

But it turns out that, post hoc, we can fairly easily designate where IM and LTM might reside and further, I think, get some interesting insights into STM which, indeed, appears to be a graded hierarchical structure, dispersed through the many layers of the recognition cone.

Iconic Memory

Iconic Memory (IM) is, pretty clearly, the first Input Buffer, where all the details of the present sensed environment are stored. (Alternately, it might be the first and/or the next layer or two, since they serve to average, to clean and smooth the raw input a bit, and difference, to heighten its gradients.) When the next moment of time is input the new sensed data merge in with the old, and all gradually fade away. But if we flash one picture for one input moment, as in Sperling's (1960) or Eriksen and Collins' (1968) experiments, this detailed representation of the flashed picture remains, albeit rapidly fading (giving much the phenomenon of Sperling's Iconic Memory). It is during this fading period of the unchanging picture that we have the IM. Masking by disparate stimuli will occur because of the choices made in cells, or if we limit the size of the store in each cell.

But notice that this IM fades away exactly as does the memory stored at every layer of the cone, and in that sense it is a part of the STM.

Short-term Memory

The locus of Short-Term Memory (STM) presents a much tougher problem. Obvious homes are the Apex of the cone and the FOUND list, where the external names and qualities that form the raw material of a description reside. We can, if we wish, now limit its size, following the results of the many experiments that suggest relatively small numbers like 7, or at the most 15 or 50, things held in this memory.

Cognitive and Perceptual Aspects of STM

I think it would be more reasonable to worry about how this perceptual model fits into a larger cognitive model that can transfer information into STM from internal as well as external sources. Think of a memory search for the names of all people talked with during the past month, or a deductive search for some set of objects, e.g. all the integers divisible by 17. The apex of the cone isn't a bad place for such a store, which could be handled by having memory searches transfer information into it, much as the perceptual search transfers information into it.

But I am inclined to put such a store in a common FOUND list one or two transform layers back from the cone's apex, so that a convergence from these very different sources of information can be made more cleanly - with each source having its own apex. (When we remember that a complete perceptual system will have several sense modes - e.g. ears and skin as well as eyes, this becomes more compelling, since each separate mode should, at least for awhile, converge separately, as though in its own cone - though an attractive alternative is to merge into the same cone only part-way through, or, as a compromise, do a bit of merging or cross-talking.)

Glimmerings of "Consciousness"

STM implies several functions, and has several aspects. We tend to think of it as a temporary store that we have some control over, and choose to use. There is an aspect of conscious awareness - we know information is there in STM - and of conscious control - we put that information there, in order to use it in a certain way - whether to scan new scenes, search memory, or perform deductions. I would prefer to call such a store "awareness" or even "consciousness" - except that such terms make many people terribly uneasy, and I am inclined to reserve "consciousness" for the more complex system that, I am sure, will ultimately have to be programmed, where it is clear that a "self" is consciously aware of and therefore is able to exercise greater processing power over that body of data that has been transferred to the central "consciousness."

I also feel that it is quite premature to assume that there is one and only one such STM or "awareness." It seems just as likely that there might be several, or even many, such temporary buffers (at the least one for perceived and another for remembered information) but that the still-higher-level conscious controller has only limited access to these and, when it pulls information into consciousness (or maybe I should say when consciousness throws its limited searchlight over information) we notice it. All that may merely be saying that a central conscious STM is something like a super-apex that has access to many apexes in perception and in memory, and that this super-apex is quite limited in its capacity (why? - probably because its processes are so sophisticated as to be quite costly) and quite active and choosey in what it takes in.

Dispersed Loci of STM

But much else is STM. Each layer in the cone must store information temporarily, and must let this information fade away relatively quickly, in order to be able to handle change. Similarly, there must be temporary transfers in associative memory searches, and temporary scratch-pads for deductive computations (unless the latter can only be handled by the central consciousness - which would be a good explanation as to why we humans are so bad at deductive inference). I would be inclined to call this all STM. In doing so, we radically change our conception of STM.

Long-Term Memory

Long-Term Memory (LTM) is quite clearly the whole structure of characterizing transforms. Remember, we are talking only about a model for perception. When the rest of memory is added, to handle facts and ideas stored from the past, to do deductive inference, and to understand and use language, we add what people normally think of as "memory." But this will only be more of the same kind - except that specific content will differ, and the overall structure will not follow the nicely layered converging processes that we see in perception. Thus LTM does not reside after, more centrally than STM. Rather, it is _everywhere_, and it is especially needed just in order to convert raw sensed information into understood and meaningful information of the sort the STMs and the conscious awareness contain.

Another crucial issue, one that is beyond the scope of this paper, is that LTM must itself be learned (which also means modifiable and un-learnable). Thus almost anything in LTM is grist for forgetting, and therein similar to STM. Much in LTM might be wired in solid (this I'd be inclined to say is what has been learned by evolution of the species, rather than by discovery and induction over the much shorter lifespan of the individual), and we might want to restrict our definition of LTM to refer just to this.

Note that this elevates the perceptual characterizers to prime place in LTM, since much of the peripheral structure of the sensory systems does indeed appear to be wired in, and turns what we more typically think of as LTM - e.g. remembrances of childhood scenes, names, and memorized poems - into relatively permanent STM. Maybe it is best to think of a Hard or Certain LTM and a Soft or Probabilistic LTM, noting that a probability can reach almost 1 for all practical intents and purposes.

Transfer from STM to LTM

STM is thought to be used as a buffer for transfer of information into LTM, and indications have been found that there may be different electro-chemical processes involved (See Gurowitz, 1969; Deutsch, 1973, for both evidence and controversy.) STM may be erased by shock or drugs, which might suggest a temporary chemical state, or a circulating train of impulses. Mechanical change to make synapse crossings easier, or the laying down of template-like molecules that store memory - both apparently slow processes with relatively permanent results - appear to be strong possibilities for LTM. But why should a single "aware" STM be necessary? The present model suggests other possibilities.

It seems a bit strange that the same STM holds a phone number to be dialed, and then forgotten, and also a Latin word to be memorized forever. We try to exert "effort" to stamp the Latin word in (and maybe that's why we usually don't do very well). It seems at least as plausible to think of <u>any</u> of the temporary STM stores as possible sources of transfer of information to LTM. This suggests rich interconnections from many of the layers of the cone, and not merely the Apex. But more important, it suggests that motivating context - which indicates salience, importance, and significance - is crucial in telling the system what it ought to start remembering. It further suggests that information transferred into LTM is likely to be transient, shortly to be forgotten. Only if future experiences reinforce its importance, so that the bonds connecting it into memory are both strengthened and made more numerous, will it remain.

Thus LTM is a dynamic, ever-changing store. It might best be thought of as just a (very large) extension of STM, where the principles of salience are emphasized, as a counter-force to the inevitable fading over time. Thus learning, using the fuels of feedback and motivation, is the fire that burns memory into place, and also burns it over.

Overall Architecture of a Unified Memory

There may well be a central consciousness store, limited in size because of the great cost of its enchanced processing power (e.g. for simultaneous scans and deductions, by looking at everything at the same time and from a variety of points of view). And there almost certainly are hard-wired memory nodes, especially in the relatively peripheral perceptual and motor systems. There might be two major substances for carrying memory, one that is relatively fast-acting, another that is relatively permanent. But "shortness" in STM, and "longness" in LTM almost certainly vary and merge. An obvious observation is the ever-lessening amount of information we can remember about 10 seconds, 10 minutes, 10 hours, 10 days, 10 weeks, 10 months, and 10 years ago.

This suggests very dynamic LTM as well as STM. Fading as a function of time and disuse is fastest in IM, slowest in LTM. Memory is reinforced from a variety of sources, but especially because of inferred usefulness for future processing. Immediate episodes must stay around long enough so that the relevant antecedents are still remembered when feedback indicates that they should be learned. But "immediate" can mean seconds, hours or days.

THE EFFICIENCY OF PARALLEL SYSTEMS

Parallel Systems Embodied in Hardware

We use general-purpose computers to simulate cognitive systems because it is so convenient to write and rewrite "software" code, and because "general-purpose" means that any other possible computer can be simulated, with the appropriate programs. So we can easily build, test, modify, and rebuild our models.

But once we have a model worth testing and using extensively, we can effect enormous savings in speed and costs by using a special-purpose computer built with that model in mind. Today's general-purpose computers are extremely slow at handling any parallel model, like the SEER recognition cone, for they simulate parallel sequences of operations by iterating through them one at a time. Thus, for example, if we have a 1000 by 1000 input buffer on which the system must perform only 1 averaging operation, where each such operation must look at a center cell and its 24 neighbors, approximately (ignoring the borders) 24 million operations must be computed serially. (Actually, each operation takes several times longer because of the need to compute and move to the next location of each iteration.)

But these operations could all be computed at the same time by a parallel system that had a 1000 x 1000 input array (e.g. made up of tiny optical fibers) and parallel hardware for the averaging transform - which would simply be connections going from each cell in the input array and also its 24 neighbors to the corresponding cell in the next layer back, with appropriate thresholding.

Similarly, a later layer, for convenience let's say one with a 50 by 50, or 2500 cell, input buffer, and a shrink-size of 2, with 10 different transforms, each looking at an average of 10 cells in the input, where some iterate over the entire input, but others look only locally, so that the average iterates over 1000 rather than all 2500 cells, would use 2500 cells for its input buffer, 625 cells for its output buffer, and 10 connections for each of the 10 transforms, in each of its average of 1000 locations, or 100,000 connections. With this amount of parallel hardware, one moment of time would be needed to replace the

100,000 serial steps.

Thus, with a great deal of hardware, we can reduce the time needed for such a parallel-serial system to its serial depth alone - which, inveitably, will be a small number like 15, or 5. (I should note that this large amount of hardware is just what we see in the brain-eye visual system. It would also be quite cheap using current LSI technology, except for major problems in connecting and building up the overall structure - essentially because connections in a plane are cheap, but crossings and connections from one plane to another are expensive and hard to effect in large quantities. Micro-programmed parallel computers (see Kruse, 1973) may be an interim compromise.)

Shortening and Simplifying Runs on the Serial General-Purpose Computer

Without such parallel systems, we are complelled to keep runs small - chiefly by cutting down on the size of the input buffer (which means resolving the scene more coarsely, and/or using smaller scenes). Thus a 20 by 50 scene, which is big enough to resolve several articulated forms, needs only 1000 input cells, as opposed to the 1,000,000 needed by the more realistic - but still skimpy - 1000 by 1000 input buffer, and will therefore reduce processing time roughly a thousand-fold.

Similarly, we can make significant savings by cutting corners and paring down here and there. Rather than average in a 5 by 5 or larger neighborhood, looking at 25 or more cells we can look only at the immediate surround, thus cutting 25 down to 9 cells. (Note that a realistic system would look at many more than 25 cells in the surround, according to the probability distributions that have been found in living eyes.) We can similarly cut down the number of cells looked at by differencers and feature-extractors. But I should emphasize that once we have the proper parallel hardware there will be _very little_ extra cost in components, and _absolutely no_ extra cost in processing time, in using connectivities as rich as seems desirable.

Cutting Costs by Using Faster Languages and Specific Models

The use of a high-level list processing and pattern matching programming language also introduces inefficiencies - roughly, a program coded in SNOBOL, LISP, or a similar language might run from 2 to 10 or 20 times as long as the same program efficiently coded in assembly language. And the use of general characterizers and overall structural features in the present recognition-cone program makes the running of each specific cone somewhat slower, maybe 2 to 5 times as slow.

Thus significant savings can be effected by coding a specific model in a fast and efficient programming language. This is worthwhile only at the point where large tests are needed, since it is far more difficult to code this kind of complex program in assembly language, and especially to rethink, tear apart, modify, and restructure, and do this over and over again - as is always the case in developing such a model. And it further seems more efficient, in terms of research strategy, to run short tests on different specific models using the general system, in order to determine which specific model is worthy of more extensive tests, and therefore of recoding as a special-purpose program.

Although the savings from faster languages, more efficient code, and/or less general models may only be on the order of 2, 10, or 50-fold, compared to the thousand or million-fold savings in using parallel computers, they may well be crucial at any moment in time from the point of view of how much research and testing can be bought with each available dollar. A $1000 budget for computer time looks very different if each run costs $2 rather than $100. On the other hand, we must take into account the significant but hidden extra costs in coding and using a more efficient but more rigid program.

SUMMARY DESCRIPTION OF THE MODEL

Architecture: A Parallel-Serial Converging Cone

The SEER cognitive model's perceptual "recognition cone" is a parallel-serial system that converges as successive transforming layers successively extract, abstract, combine, coalesce, and generalize. A scene is "sensed" at the Retinal Input buffer I_R that forms the base of the cone. The first transform layer looks at I_R and merges the outputs of successful transforms into the next Internal buffer, I_1. This continues until the apex of the cone is reached.

A General Type of Transform that Handles a Variety of Tasks

The early layers will do what we tend to call pre-processing operations - smoothing, eliminating local noise, averaging, and differencing to find gradients and the beginnings of edges. Then first-level feature-detectors are applied, followed by several layers of higher-level structure-characterizers. Transforms can imply other transforms to apply, so that the system can explore possibilities to adduce more evidence, glance about over the scene, and dynamically decide what to do next.

Transforms can trigger the decision to choose among alternative possibilities. Each such choice defines the cell in which it is made as the apex of a sub-cone whose base is the sub-region of the larger scene in which the chosen object is assumed to lie. The various parts, qualities, and other descriptive information about that object will have been merged into and therefore be stored in that same cell.

Thus all the various processes are handled in a single homogeneous way.

Changes and Motion Over Time

Information fades gradually over time, with new implications merged in with the old. Objects are highly weighted to the extent that they are implied by transforms (because of features, structures, contexts, needs, or larger wholes that probably contain them - these depend upon the specific transforms given the system), and also as a function of change - both newness and motion. This means that the system can handle scenes of objects that move and change over time.

Internal Needs and Expectations

Transforms to apply also endure over time, before they gradually fade away, so that they look for and "expect" things into the future. Internal needs, sets, ideas and expectations similarly imply transforms to apply, and thus help direct and focus the perceptual process. Thus the system flows in a "top-down" (from internal processes) as well as a "bottom-up" (from external sensed data) direction.

Shaded and Colored Objects

Shaded and colored objects can be handled by the same general type of transform (albeit the specific transforms must be given to, or learned by, the system - just as the specific rewrite rules must be given to a parsing system). Shaded scenes have Intensity ranges reflecting the possible shades of gray, rather than 0 and 1 (for white and black). Colored scenes are input with the 3 primary colors as qualities, and transforms are used that look at the same cell for these three color-qualities, and imply the appropriate color-mixture - just as other transforms look for three stroke-features and imply the likely shape.

Iconic, Short-Term, and Long-Term Memories

It seems reasonable to equate the Input buffer with Iconic Memory, and the Apex of the cone, and possibly also the

FOUND list and internal buffers, with the Short-Term Memory. The Transform layers are clearly part of Long-Term Memory. In fact the early transform layers do processes that are almost certainly built into living eyes by evolution, so that they are fixed at birth. And in general the perceptual transforms are the most long-lasting nodes in memory. The more complete SEER model uses the same general type of transform for all aspects of memory, and it is important to note that the perceptual transforms are an integral, and especially permanent, part of Long-Term Memory.

Possible Loci of "Awareness" and "Consciousness"

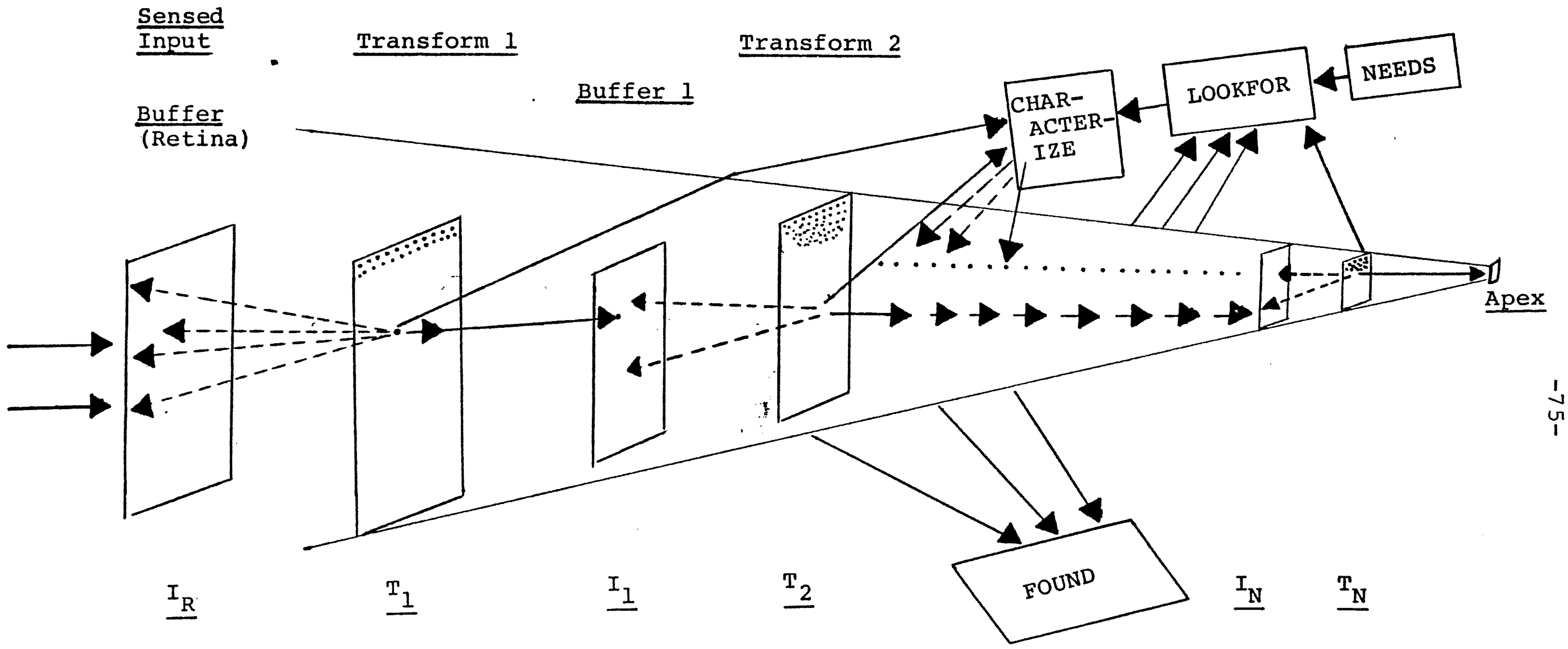
We might loosely equate the many memory buffers with nodes of "awareness" and the buffer into which information from external scenes and from internal memory and needs finally merges as the possible germ of "conscious awareness." It might be best to think of all of these temporary buffers as Short-Term Memory, and the transform nodes as Long-Term Memory. Transforms are strengthened by being used and being implied; but all memory gradually fades away, with fading proceeding more slowly the more central the transform.

Plausibility as a Model for Perception

This system's parallel-serial probabilistic structure, with its intermingling of pre-processors, feature-detectors, and ever-higher-level structure-characterizers, appears to satisfy many of the criteria suggested for a model of form perception. It can handle a wide variety of patterns, over large amounts of noise, distortion, and unknown and non-linear distortions. It builds and can describe objects that are complex structures of sub-parts and qualities. It handles a number of the problems posed by increasing the dimensionality of scenes, and in particular to shaded and colored objects, and to objects that change over time. It makes use of contextual information, and of internal needs, desires and expectations, to help direct and focus its perceptual processes.

It is by no means guaranteed to give the correct responses, as a result of built-in analytic algorithms that have been coded to anticipate all eventualities. Rather, it makes probabilistic choices among alternative possibilities. It attempts to learn,as a function of feedback and needs, ever-better structures of transforms for doing this. Its converging parallel-serial layered structure seems to be an effective technique for gaining the advantages of parallel processes (speed and robustness) and of serial processes (directedness and economy of computation). Such a structure also appears to be strongly indicated by living visual systems.

Figure 1. The overall architecture of a SEER's "recognition cone."



A scene from the external environment is transduced (e.g. by a TV camera, photocells, or reading in punched cards) into the Retina, the Input buffer, I_R. The first layer of Transforms, T_1, is applied to I_R and, when any transform succeeds, its implications are output and merged into the cell in the next Internal buffer, I_1, that lies directly behind (that is, to the right in the figure). Transforms to apply are merged into CHARACTERIZE; the choices of triggered decisions are merged into FOUND. A layer can contain many transforms. Each transform resides at a specific location in its layer, but it specifies any number of relative locations in the buffer that it looks at for its parts. Any number of transforms can reside at any specific location. Each layer has an associated shrink-size, which determines the relative size of the next layer (if desired, shrink-size can be set to 1, so that no shrinkage occurs). Shrinkage means that the sequence of layers will eventually collapse into a single cell, the apex. The whole system then forms a (jagged) "recognition cone."

FIGURE 2. Some Examples of Layer-sequences in recognition cones.

a) A simple System

I_R (Sensed Input)

T_1 Average

I_1

T_2 Difference

I_2

T_3 Feature-Detect

I_3

T_4 Structure-Characterize$_1$

I_4

T_5 Structure-Characterize$_2$

I_5 (Apex)

b) A more realistic, but still simple, system.

I_R (Sensed Input)

T_1 Average

I_1

T_2 Difference

I_2

T_3 Average

I_3

T_4 Difference

I_4

T_5 Primitive Feature-Detect

I_5

T_6 Structure-Characterize$_1$ (e.g. Angles)

I_6

T_7 Structure-Characterize$_2$ (e.g. Enclosures)

I_7

T_8 Structure-Characterize$_3$ (e.g. Figures)

I_8

T_9 Structure-Characterize$_4$ (e.g. Sets)

I_9

T_{10} Structure-Characterize$_5$ (e.g. Groups)

I_{10} (Apex)

c) A minimal System

I_R (Sensed Input)

T_1 Feature-Detect

I_1

T_2 Structure-Characterize

I_2 (Apex)

FIGURE 3. Some plausible shrink-values for the 5-layer cone of Figure 2a, and the resulting sizes of input and subsequent layers. The Height (and also Length) of each layer is got by multiplying the Height of the next layer (below it) by the shrink-size of that layer's Transforms starting with the Apex (at the bottom). The size of the Sensing Retinal buffer (at the top) determines the size of the Scenes that are input to the system.

Structure	Alternate Shrink-Configurations							
Layer	a) Shrink-Size	Height	b) Shrink Size	Height	c) Shrink Size	Height	d) Shrink Size	Height
I_R Sensed Input		24		108		264		32
T_1 Ave	3		3		3		2	
I_1		8		36		81		16
T_2 Diff.	1		2		3		2	
I_2		8		18		27		8
T_3 Feature	2		3		3		2	
I_3		4		6		9		4
T_4 Structure	2		2		3		2	
I_4		2		3		3		2
T_5 Structure	2		3		3		2	
I_5 Apex		1		1		1		1

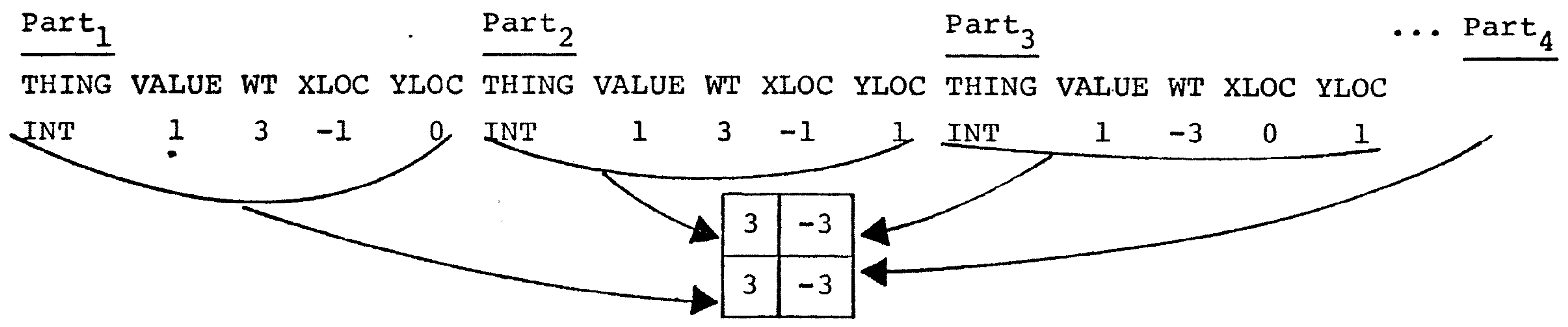
FIGURE 4. Some recognition cones with Input Buffers as large as the Fovea in the Human Eye's Retina. The fovea (a circle roughly 5° in diameter) contains roughly 7 million cones (the receptors for detailed color vision). This is roughly equivalent to a $2{,}700^2$ array. The following are several resulting cones:

		Example A		Example B		Example C	
		Shrink	Height	Shrink	Height	Shrink	Height
Retina			2700		2700		2700
T1	Average	3		3		5	
			900		900		540
T2	Difference	1		2		2	
			900		450		270
T3	Average	3		3		5	
			300		150		54
T4	Difference	1		2		2	
			300		75		27
T5	Feature	2		3		3	
			150		25		9
T6	Structure	2		3		3	
			75		9		3
T7	Structure	2		3		3	
			38		3		1
T8	Structure	2		3			
			19		1		
T9	Structure	2					
			8				
T10	Structure	2					
			4				
T11	Structure	2					
			2				
T12	Structure	2					
			1				

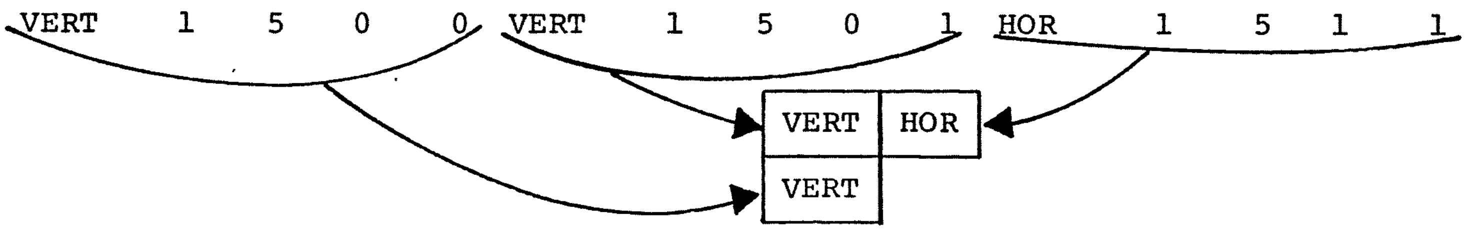
FIGURE 5. The Overall Structure of Transforms (used for Feature-Detection, Structure-Characterizing, Pre-Processing, Implying Transforms to Apply, and Triggering Choices).

A	B	C	D	E
If-conditions	Threshold	Implieds	Region	Expectations

FIGURE 6. Examples of the Structure of the "If-Conditions" in a Transform that are applied to the Transform's Input buffer.

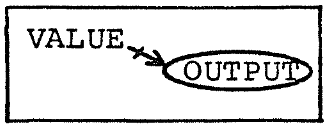

Part$_1$
Part$_2$
Part$_3$
... Part$_4$
THING VALUE WT XLOC YLOC THING VALUE WT XLOC YLOC THING VALUE WT XLOC YLOC
INT 1 3 -1 0 INT 1 3 -1 1 INT 1 -3 0 1
3
-3
3
-3

a) a gradient detector



b) a characterizer for the symbol GAMMA (Γ). (this would be deeper in the cone)

FIGURE 7. Some examples of Pre-processing Transforms, to a) Average, b) Difference, c) Smooth and Eliminate local noise. A weight or Value is shown in each cell, as indicated. Output is the combined weights of the Parts, except when a Threshold is used, indicated



1	2	1
2	3	2
1	2	1

1) center = 3, diags = 1

1	1	1
1	2	1
1	1	1

2) center = 2, neighbors = 1

1	1	1
1	1	1
1	1	1

3) all cells have same weight

a) weights for cells looked at by local averaging transforms (for a 3 by 3 neighborhood)

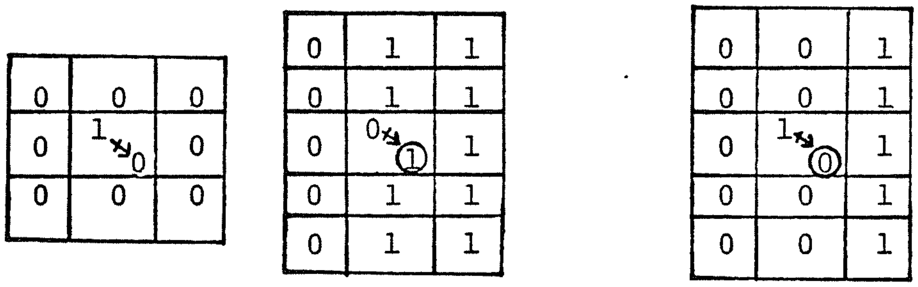
-1	-4	-1
-4	20	-4
-1	-4	-1

1) center = 20, diags = -1

-1	-1	-1
-1	8	-1
-1	-1	-1

2) center = 8, neighbors = -1

b) weights for cells looked at by local differencing transform.



1) Changes 1 to 0 when surrounded by 0's.

2) Changes 0 to 1 on an edge

3) Changes 1 to 0 next to an edge

c) expected values for cells looked at by noise-eliminating transforms

FIGURE 8. Some Examples of Primitive Feature-Detectors

-1	-1	1	1
-1	-1	1	1
-1	-1	1→ VERT	1
-1	-1	1	1
-1	-1	1	1

a) Vertical edge detector

-1	-2	-2	-2	-1
2	3	4→ HOR	3	2
-1	-2	-2	-2	-1

b) Horizontal line detector

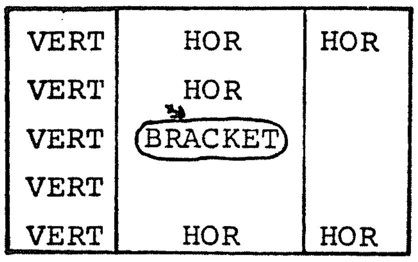
1	1	1	
-1	1	2	
-1	-1	1	2
-2	-1	1→ CURVEA	2
-1	1	1	
-1	1	1	
1	1		

c) Curve detector (note that a complete rectangle need not be specified)

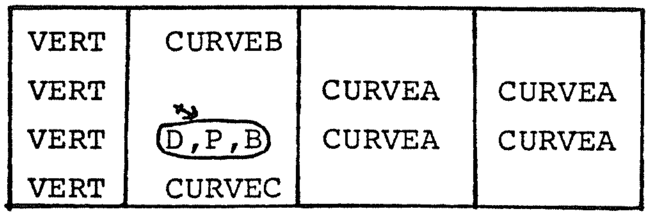
1	2	3	2	1
		2		
-2		2→ TJOIN		-2
-2	-1	1	-1	-2
-2	-1	1	-1	-2
		1		
		1		

d) T-join detector

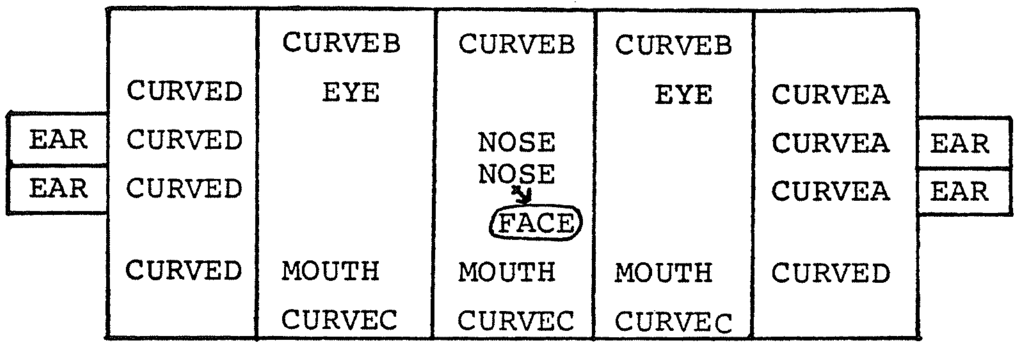
FIGURE 9. Some Examples of Structure-Characterizers



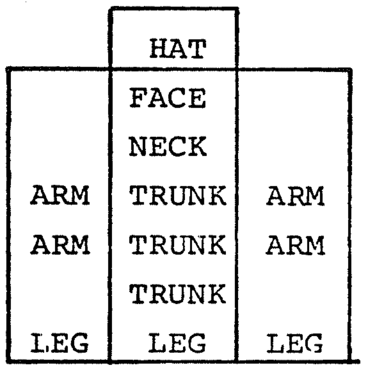
a) Strokes imply the symbol BRACKET ([)



b) Strokes imply several objects (the letters D,P,B)



c) Strokes and parts imply FACE



d) Parts imply PERSON

FIGURE 10. <u>Details of the Structure of the "Implieds" in a Transform</u>. An Implied may be either 1) a Thing (to be merged into the next layer's input Buffer), 2) a Transform to apply (to be added to the CHARACTERIZE list of dynamically implied Transforms), or 3) a trigger to choose among the implieds in that cell (the choice put into FOUND.)

$$(\text{Type Implied}_1\ \text{Wt}_1\ \text{Rel Loc}_1 \,|\, \text{Type Implied}_2\ \text{Wt}_2\ \text{Rel Loc}_2 \,|\, \ldots \qquad)$$

$$\underbrace{N \quad B \quad 7 \quad 0 \quad 0}\ \underbrace{N \quad R \quad 5 \quad 0 \quad 0}\ \underbrace{N \ P \ 4 \ 0 \ 0}\ \underbrace{T \ \text{CHAR}_{17} \ 9 \ 0 \ 1}$$

FIGURE 11. A Few Examples of Cone Specifications

I) Minimal

Layer	Transform Type	Number Transforms	Shrink Factor	Width & Height	Shrink Factor	Width & Height	Shrink Factor	Width & Height
				162		32		405
1)	Average	1	3	54	2	16	5	81
2)	Difference	1 to 4	2	27	2	8	3	27
3)	Slopes	4 to 16	3	9	2	4	3	9
4)	fixed structures (2-tuples, or 3-tuples + flexibly implieds	5	3	3	2	2	3	3
5)	fixed structures + flexibly implieds	5 to 20	3	1	2	1	3	1

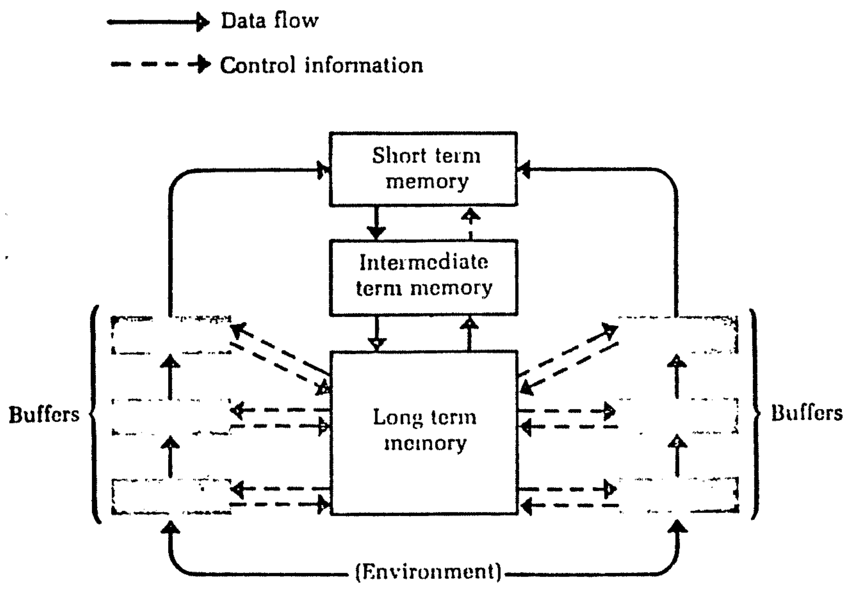
II) Sub-Minimal

	Transform Type	Shrink Factor	Width & Height
			27
1)	Average	3	9
2)	Difference & slopes	3	3
3)	Structures	3	1

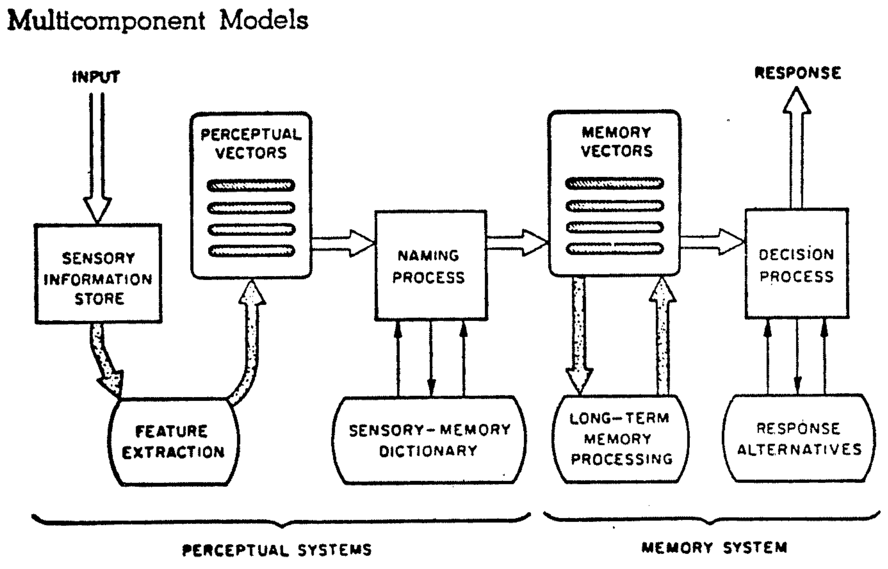
III) Adequate

	Transform Type	Shrink Factor	Width & Height
			256
1)	Average	2	128
2)	Difference	2	64
3)	Average	2	36
4)	Difference	2	16
5)	Slopes	2	8
6)	Structures	2	4
7)	Structures	2	2
8)	Structures	2	1

FIGURE 12. Two Examples of "Black Box" Information Processing Models of Perception and Memory



a) Hunt's (1973) Distributed Memory Model



b) Norman and Rumelhart's Multi-component Model (1970) of Perception and Memory (as adapted by Reed, 1973).

Allman, J. M. & Kaas, J. H., A crescent-shaped cortical visual area surrounding the middle temporal area (MT) in the owl monkey (aotus trivirgatus), Brain Res., 1974, 81, 199-213.

Anderson, J. R. and Bower, G. H. Human Associative Memory, Washington: Winston, 1973.

Barlow, H. B. Sensory mechanisms, the reduction of redundancy, and intelligence. In The Mechanization of Thought Processes (D. V. Blake and A. M. Uttley, Eds.) London: HMSO, 1959, pp. 535-539.

Barlow, H. B. Trigger features, adaptation and economy of impulses, In: Information Processing in the Nervous System, (K. N. Leibovic, Ed.) New York: Springer-Verlag, 1969, pp. 209-226.

Bledsoe, W. W. and Browing, I. Pattern recognition and reading by machine. Proc. East. Joint Comput. Conf., 1959, 16, 225-232.

Brice, C. R. and Fennema, C. L., Scene Analysis Using Regions, Artificial Intelligence, 1970, 1, pp. 205-226.

Clowes, M. B. Transformational grammars and the organization of pictures. In A. Grasselli (Ed.), Automatic interpretation and classification of images. New York: Academic Press, 1969.

Clowes, M. B. On seeing things, Artificial Intelligence, 1971, 2, 79-116.

Deutsch, J. A. A theory of shape recognition, Brit. J. Psychol., 1955, 46, 30-37.

Deutsch, J. A. A system for shape recognition, Psychol. Rev., 1962, 69, 492-500.

Deutsch, J. A. (Ed.) The Physiological Basis of Memory, New York: Academic Press, 1973.

Dodwell, P. C. Shape recognition in rats, Brit. J. Psychol., 1957, 48, 221-229.

Dodwell, P. C. Visual Pattern Recognition, New York: Holt, 1970.

Doyle, W. Recognition of sloppy, hand-printed characters, Proc. IFIP WJCC, 1960, 17, 133-142.

Duda, R. O. and Hart, P. E. Experiments in scene analysis, Proc. 1st Natl. Symp. on Industrial Robots, 1970, pp. 183-202.

Duda, R. O. and Hart, P. E. Pattern Classification and Scene Analysis, New York: Wiley, 1973.

Eccles, J. C. The Understanding of the Brain, New York: McGraw-Hill, 1973.

Eriksen, C. W. & Collins, J. F. Sensory traces versus the psychological moment in the temporal organization of form. Journal of Experimental Psychology, 1968, 77, pp. 376-382.

Evans, T. G. Grammatical inference techniques in pattern analysis, in: J. T. Tou (Ed.) Software Engineering, Vol. 2, New York: Academic Press, 1971, pp. 183-202.

Feldman, J. A. et al. The Stanford hand-eye project. Proc. 2nd Int. Joint Conf. on Artificial Intell., 1971, pp. 521-526.

Fu, K. S., Swain, P. H., On Syntactic Pattern Recognition, in Software Engineering, Vol. 2, (J. T. Tou, Ed.) New York: Academic Press, 1971.

Gurowitz, E. M. The Molecular Basis of Memory, Englewood-Cliffs: Prentice-Hall, 1969.

Guzman, A., Decomposition of a Visual Scene into Three-Dimensional Bodies, Proc. AFIPS FJCC, 1968, 33, 291-304.

Hanson, A. R. and Riseman, E. M. Preprocessing cones: a computational structure for scene analysis. Comp. Sci. Tech. Rept. C-7, Univ. of Mass., Sept. 1974.

Hubel, D. H. & Wiesel, T. H. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. Journal of Physiology, 1962, 160, pp. 106-154.

Hubel, D. H. and Wiesel, T. H. Receptive fields and functional architecture in two nonstriate visual areas (18 and 19) of the cat, J. Neurophysiol., 1965, 28, 229-289.

Hunt, E. B. Concept Formation: An Information Processing Approach, New York: Wiley, 1962.

Hunt, E. B. The memory we must have. In R. Schank and K. M. Colby, Eds. Computer Models of Language and Thought, San Francisco: Freeman, 1973, pp. 343-371.

Kanal, L. (ed.) Pattern Recognition, Washington: Thompson, 1968.

Konorski, J. Integrative Activity of the Brain, Chicago: University of Chicago, Press, 1967.

Kruse, B. A parallel picture processing machine, IEEE Trans. Comput., 1973, 22, 1075-1086.

Ledley, R. S., Analysis of Cells, IEEE Trans. Comput., 1972, 21, 740-753.

Miller, W. F. and Shaw, A. C. Linguistic methods in picture processing - a survey. Proc. Spring Joint Comput. Conf., 1968, 33, 279-290.

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

Munson, J. H. Experiments in the recognition of hand-printed text. Proc. Spring Joint Comput. Conf., 1968, 33, 279-290.

Narasimhan, R. Syntax-directed interpretation of classes of pictures, Comm. ACM, 1966, 9, 166-173.

Nilsson, N. J. A mobile automaton: an application of A. I. techniques, Proc. 1st Int. Joint Conf. on Artificial Intell., 1969, 509-520.

Norman, D. A. & Rumelhart, D. E. A system for perception and memory. In D. A. Norman (Ed.), Models of human memory. New York: Academic Press, 1970.

Pfaltz, J. L. and Rosenfeld, A. Web grammars, Proc. 1st Int. Joint Conf. on Artificial Intell., Washington, 1969.

Polyak, S. L. The Vertebrate Visual System, Chicago: Univ. of Chicago Press, 1957.

Reddy, D. R., Erman, L. D., Fennell, R. D., and Neely, R. B. The hearsay speech understanding system. Proc. 3d Int. Conf. on Artificial Intell., 1973, 185-193.

Reed, S. K. Psychological Processes in Pattern Recognition, New York: Academic Press, 1973.

Riseman, E. M. and Hanson, A. R. Design of a semantically directed vision processor, Tech. Rept. 74C-1, Computer Science, Univ. of Mass., Jan., 1974.

Rosenblatt, F. Principles of Neurodynamics, Baltimore: Spartan Press, 1962.

Rosenfeld, A. Picture Processing by Computer. New York: Academic Press, 1969

Shaw, A. C., A formal picture description scheme as a basis for picture processing systems," Info. and Control, 1969, **14**, 9-52.

Sherman, H. A quasi-topological method for the recognition of line patterns, in Information Processing (IFIPS Conf. Proceedings), Paris: UNESCO, 1960, pp. 232-238.

Slagle, J. R. and Lee, R.C.T., Applications of game tree searching techniques to sequential pattern recognition, Comm. ACM, 1971, 14, 103-110.

Sperling, G. The information available in brief visual presentations. Psychological Monographs, 1960, 74, No. 11 (Whole No. 498).

Stevens, Mary E. Abstract shape recognition by machine, Proc. Eastern Joint Computer Conf., 1961, 20, 332-351.

Sutherland, N. S. Outlines of a theory of visual pattern recognition in animals and man, Proc. Royal Society, B, 1968, 171, 297-317.

Tappert, C. C. and Dixon, N. R. A procedure for adaptive control of the interaction between acoustic classification and linguistic decoding in automatic recognition of continuous speech. _Proc. 3d Int. Conf. on Artificial Intell._, 1973, 173-184.

Uhr, L. and Vossler, C. A pattern recognition program that generates, evaluates and adjusts its own operators, _Proc. AFIPS WJCC_, 1961, _19_, 555-570. (Reprinted, with additional results, in E. Feigenbaum and J. Feldman, Eds., _Computers and Thought_, New York: McGraw-Hill, 1963.)

Uhr, L. (Ed.) _Pattern Recognition_, New York: Wiley, 1966.

Uhr, L. and Jordan, S. The learning of parameters for generating compound characterizers for pattern recognition. _Proc. 1st Int. Joint Conf. on Artificial Intell._, 1969, 381-415.

Uhr, L., Flexible linguistic pattern recognition, _Pattern Recognition,_ 1971, _3_, 363-384.

Uhr, L. Layered "recognition cone" networks that preprocess, classify and describe, _IEEE Trans. Computers_, 1972, _21_, 758-768.

Uhr, L. _Pattern Recognition Learning and Thought_, Englewood-Cliffs: Prentice-Hall, 1973(a).

Uhr, L. The description of scenes over time and space, _Proc. AFIPS NCC_, 1973, _42_, 509-517(b).

Uhr, L. Describing, using "recognition cones," _Proc. 1st Int. Joint Conf. on Pattern Recognition_, 1973(c). (Also _Computer Sci. Dept. Tech. Rept. 176_, Univ. of Wisconsin, Madison, 1973.)

Uhr, L. A wholistic cognitive system (SEER-2) for integrated perception, action and thought. _Computer Sci. Dept. Tech. Rept._, Univ. of Wisconsin, 1974.

Uhr, L. Toward integrated cognitive systems, which _must_ make fuzzy decisions about fuzzy problems, In L. Zadeh et al, Eds., _Fuzzy Sets_, New York, Academic Press, 1975. (Also _Computer Sci. Dept. Tech. Rept. 222_, August, 1974.) 1975(a).

Uhr, L. Form perception and scene description: Toward a theoretical and experimental science of complex structures, 1975(b).

Uhr, L. "Recognition cones" that perceive and describe scenes that move and change over time, _Computer Sci. Dept. Tech. Rept. 235_, University of Wisconsin, 1975(c).

Uhr, L. A wholistic integrated cognitive system (SEER-T2) that interacts with its environment over time, _Computer Sci. Dept. Tech. Rept. 236_, Univ. of Wisconsin, 1975(d).

Unger, S. H. Pattern detection and recognition, _Proc. IRE_, 1959, _47_, 1737-1752.

Waltz, D. L. Generating semantic descriptions from drawings of scenes with shadows, Unpubl. Ph.D. Diss., MIT, Cambridge, 1972 (AI TR-271).

Watanabe, S. (Ed.) _Methodologies of Pattern Recognition_, New York: Academic Press, 1969.

Winston, P. H. The MIT robot, In: _Machine Intelligence 7_ (B. Melzer and D. Michie, Eds.) Edinburgh Univ. Press, 1972.

Young, J. Z. A Model of the Brain, Oxford: Oxford Univ. Press, 1964.

Zobrist, A. L., The organization of extracted features for pattern recognition, Pattern Recognition, 1971, 3, 23-30.