# Representing and Learning Visual Schemas in Neural Networks for Scene Analysis

**Wee Kheng Leow and Risto Miikkulainen**
Department of Computer Sciences
The University of Texas at Austin
Austin, Texas 78712 USA
Email leow,risto@cs.utexas.edu

Using scene analysis as the task, this research focuses on three fundamental problems in neural network systems: (1) limited processing resources, (2) representing schemas, and (3) learning schemas. The first problem arises because no practical neural network can process all the visual input simultaneously and efficiently. The solution is to process a small amount of the input in parallel, and successively focus on the other parts of the input. This strategy requires that the system maintains structured knowledge for describing and interpreting the gathered information. The system should also learn to represent structured knowledge from examples of objects and scenes. VISOR, the system described in this paper, consists of three main components. The Low-Level Visual Module (simulated using procedural programs) extracts featural and positional information from the visual input. The Schema Module encodes structured knowledge about possible objects, and provides top-down information for the Low-Level Visual Module to focus attention at different parts of the scene. The Response Module learns to associate the schema activation patterns with external responses. It enables the external environment to provide reinforcement feedback for the learning of schematic structures.

## 1 Introduction

Consider the task of understanding simple scenes. A scene analysis system has to identify the objects in the scene (e.g., an arch and two trees, Fig. 1a) and determine what the scene depicts (e.g., a park). In designing a neural network system that performs this task, three fundamental problems are encountered:

1. How can a fixed-size neural network process indefinitely large amounts of information?
2. How can a neural network represent and use structured knowledge?
3. How can a network learn to represent structured knowledge?

To appear in *Proceedings of the IV Iberoamerican Congress on AI*, Caracas, Venezuela, 1994.

In fact, these problems are also encountered in many other neural network application areas, such as speech understanding and natural language processing. The goal of this research is to develop general solutions to these problems, using scene analysis as a concrete task.

Consider the first problem: limited processing resources. In practice, it is only possible to construct a neural network with a fixed number of input units and internal processing units. The weights and activities have finite precision and are bounded within certain ranges of values. Even if such a network can capture a large part of the scene at once, it may not be able to process all the information in parallel unless it has an exponential number of units and connections [7; 17]. The only viable option is to process a small amount of visual input in parallel, and successively focus on different parts of the scene. This strategy also seems to be in use in biological visual systems [12].

Since the network is fixed and finite, it may not have enough storage space for the input information. It will have to build and maintain a *partial interpretation* of the input gathered so far. It estimates the likelihoods that the input features form known objects, and as more information is received, it strengthens or weakens these estimates. It continues processing other parts of the scene until it has gathered sufficient information to build a consistent, complete interpretation.

A system that adopts this strategy requires an internal model, generally known as a *schema* in psychological research, for making the interpretations [1; 3; 8; 12]. Thus, the solution to the first problem requires addressing the second problem: the network needs to be able to represent structures in the input, including the spatial layout of the objects and the entire scene.

Although visual schemas have been extensively studied in the symbolic framework [5; 10], there has been very little work in neural networks in this area (see [2; 6; 16] for related approaches). Neural networks are not very good at manipulating symbolic structures explicitly. Instead, they are good at feature extraction, association, constraint satisfaction, pattern classification, and making other fuzzy decisions, based on cooperation and competition among units and networks. A successful schema implementation would have to be based on such subsymbolic, "neural" processes.

The third problem concerns the learning of structured knowledge. In practice, a lot of knowledge is required to describe the objects in the application domain. It is very difficult and tedious, if not impossible, to handcode all the knowledge required. Learning simplifies the process. The same system can be used in different applications after the domain-specific knowledge has been acquired. Such a system can also adapt to environmental changes.

The VISOR system (VIsual Schemas for Object Representation; [13; 14]) is designed to address the three fundamental problems in the domain of scene analysis. In this paper, we will describe the core parts of VISOR and illustrate its operation on "blocks world" scenes.

## 2 The VISOR architecture

The overall architecture of VISOR is based on the separation of the "what" and "where" pathways in low-level vision ([18]; Fig. 1). Its representation of hierarchical knowledge is

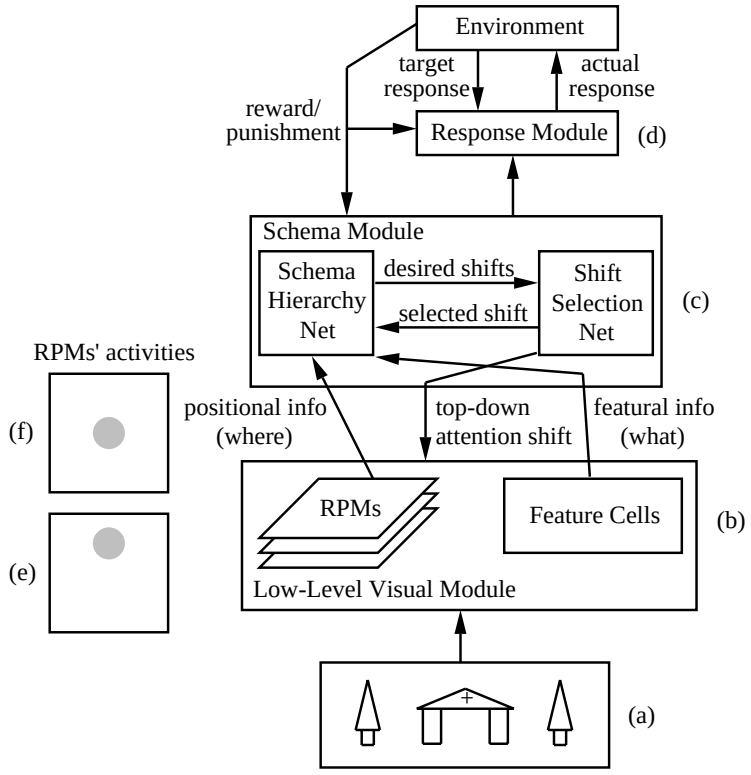


Figure 1: VISOR consists of the Low-Level Visual Module (LLVM, b), the Schema Module (c), and the Response Module (d). LLVM extracts "what" and "where" information from the scene (a). The Schema Module performs scene analysis and the Response Module produces responses expected by the environment. Figures (e) and (f) indicate the activities of fine-scaled and coarse-scaled Relative Position Maps (RPMs) when attention is focused at the position marked with "+".

similar to that of Hinton's part-whole hierarchies [11]. VISOR consists of three main components: the Low-Level Visual Module (simulated using procedural programs), the Schema Module, and the Response Module. The architecture and operation of the first two modules will be described below. The Response Module will be discussed with schema learning (Section 4).

The Low-Level Visual Module (LLVM, Fig. 1b) focuses its attention at one position in the scene at a time, and extracts the feature (e.g. line, rectangle, triangle) at that location. As its output, the Feature Cells indicate how strongly the LLVM believes a particular feature is present ("what" information; Fig. 2a). The Relative Position Maps (RPMs) encode the relative positions of the features at several scales (i.e. "where"). For example, suppose that part of the scene contains an arch and two trees (Fig. 1a), and attention is currently focused on the triangular roof of the arch. At a fine scale, the RPM locates the triangle above the two rectangles, and gives a peak response at the top part of the map (Fig. 1e). At a coarser scale, the RPM locates the blob of features constituting the arch in the middle of the blobs corresponding to the two trees, and forms a peak response at the center of the map (Fig. 1f).

The Schema Module (Fig. 1c) maintains the hierarchy of schemas, integrates successive

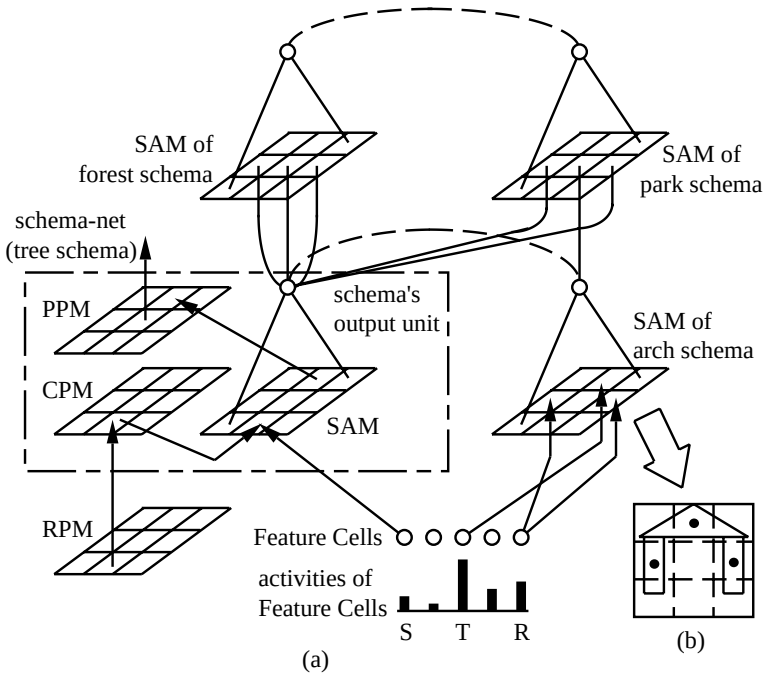


Figure 2: (a) The Schema Hierarchy Net encodes part-whole relationships among the schemas. Arrows represent one-way connections, solid lines represent both the bottom-up and top-down connections (which are different), and dashed lines indicate inhibition. The Feature Cell marked "S" is sensitive to squares, the one marked "T" to triangles, and "R" to rectangles. For simplicity, only the components of the tree schema-net are shown, where a square is expected at the bottom-center position, and the center position is a potential next position of attention. (b) The arch image encoded by the arch schema. The grid represents the SAM and the black dots denote the positions of the components in the SAM.

input information, and determines the next position of attention. It consists of two main neural networks: the Schema Hierarchy Net (SHN) and the Shift Selection Net (SSN). The SHN is a multi-layer network of schema representation nets, or schema-nets for short (Fig. 2). A schema-net consists of four main components: the output unit, the Subschema Activity Map (SAM), the Current Position Map (CPM), and the Potential Position Map (PPM). Before describing these components in detail, let us first look at how the schema hierarchy is represented in the SHN.

Each layer of schema-nets corresponds to a level in the schema hierarchy. The higher the level, the larger the scale of the schema-net. High-level nets represent scenes while low-level nets encode objects. A schema-net can simultaneously be a subschema of a higher-level schema-net and a super-schema of a lower-level schema-net. The first-level schemas take the Feature Cells' activities as inputs. The connectivity of the SHN encodes the part-whole relationships among the schemas. Consider, for example, the representation of an arch (Fig. 2b). The arch consists of three components: a triangular roof and two rectangular pillars. The grid superimposed on the arch represents a map called the Subschema Activity Map (SAM) in the arch schema-net. The black dots indicate the positions of the components in the map. For example, the triangle is located at the top-center of the arch map. Corresponding to

each black dot, there is a connection from a Feature Cell to a SAM unit. The connection indicates that the feature is a component of the arch schema at the position of the SAM unit.

The SAM units' activities indicate how strongly the subschemas are believed to be present in the scene. These activities may change as more information is extracted from the scene. In effect, SAM encodes a summary of current evidence for a schema. The evidence is added up in the activity of the schema-net's output unit (or schema-net's activity for short), which represents the certainty, or confidence, that the entire schema matches the input. In addition to bottom-up connections from the schema-net's own SAM units, the output unit receives top-down connections from the super-schemas' SAM units (Fig. 2). If a higher-level schema matches an input object with high confidence, then its subschemas are expected to match the object's components as well. The top-down feedback encodes such expectations and gives an activation advantage to subschemas that match the currently active high-level schemas. There are also mutually inhibitory connections among the schema-nets' output units to allow the schemas to compete in interpreting the input.

In addition to the dynamic information encoded in SAM, it is necessary to maintain information about the static structure of the schema, so that the system can decide where to focus its attention next. Such information is stored in the Potential Position Map (PPM). A high activity in a PPM unit indicates that a subschema is expected at the corresponding position.

The current position of attention is stored in the Current Position Map (CPM), coded by the location of a single active unit in the map. Each CPM unit connects multiplicatively to the SAM unit at the corresponding location (Fig. 2). If a CPM unit is on, the corresponding SAM unit's activity can be updated. Otherwise, the SAM unit's activity remains unchanged. In other words, only the activities of the subschemas that match the current position are propagated upwards.

## 3 VISOR operation

At the beginning of the scene analysis process, all the schema-nets are reset to their initial states: none of their CPM units are on (no current position of attention), and the activities of their SAM units are 0 (nothing has been found). At each focusing of attention, the Schema Module processes the featural and positional information received from the LLVM in four main stages:

**1. Setting Current Positions.** After the LLVM has shifted its attention to the selected location in the scene, the schema-nets update their current positions of attention. If a schema-net is not attending to anything, that is, none of its CPM units are on, its current position is set at the peak position in the RPM (Fig. 2). If one of the CPM units is on, the current position is shifted according to the shift vector received from the SSN (Fig 1c).

**2. Schema Activation.** At this stage, one of the CPM units is active indicating the current position of attention. The activity of the SAM unit at the corresponding map position is updated (e.g. bottom-center in the tree schema of Fig. 2). Other SAM units' activities

remain unchanged. The activity of the schema's output unit is also updated according to how well the schema matches the input. If it matches well, its activity increases as a result of increased SAM activity; otherwise, its activity decreases as a result of mutual inhibition among the schemas. The activity of a schema in turn feeds back to its subschemas and boosts their activities. The activities are updated asynchronously for several cycles until they stabilize.

**3. Selection of Desired Next Positions.** After the activities have stabilized, each schema chooses a position at which it would like the system to focus its attention. These are the positions where a schema expects to discover features that will contribute to increasing its activity, that is, where there are high PPM activities (e.g the center of the tree schema in Fig 2). The selected position is encoded as a shift vector (x-shift, y-shift, x-scale, y-scale) and is sent to the SSN.

**4. Selection of The Actual Next Position.** The SSN receives the desired shift vectors from all schema-nets as its input, and selects one of them to be adopted. It prefers a small shift desired by a highly active schema-net. This criterion favors the interpretation of the visual input in terms of the best-matched schema while minimizing the amount of attention shift. If VISOR has not finished processing the object that it is currently focused on, the SSN will favor shift vectors with the scales comparable to the object schemas, which allows VISOR to focus attention at other parts of the object. Otherwise, it will select a shift vector with scale comparable to the scene schemas. The selected shift vector is propagated to the LLVM and to all schema-nets.

The example in Fig. 3 illustrates the operation of VISOR. All schemas are handcoded in the SHN. The first level of the SHN consists of the arch, the house, and the tree schemas. Of these, the arch and the house schemas are very similar. Both have flat triangular roofs, and the rectangular pillars of the arch may be confused with the square windows of the house. The second-level schemas are forest, park, suburb and city. These schemas are very similar as well. For example, if the scene is either a forest, park or suburb, and VISOR scans from right to left, it will be unable to disambiguate until the object on the far left is identified.[1]

In this example, VISOR received a suburb image as the input, and was set to initially attend to the triangle of the rightmost tree. Note that this state is ambiguous because the forest, park and suburb schemas all have a tree as the rightmost component. At step 2, the rightmost tree was identified. At step 4, the middle tree was identified. At this time, detailed information of the middle tree was stored in the SAM of the tree schema, but detailed information of the rightmost tree was lost. The previous activity of the tree schema (corresponding to the rightmost tree) was stored only in the SAMs of the second-level schemas. Throughout the first 4 time steps, VISOR was unable to determine whether the input scene was a suburb, a park or a forest. At step 5, VISOR focused attention at the triangular roof of the house. It thought that the object on the far left was most likely an arch, and that the whole image was most likely a park. Final disambiguation occurred at step 10. After this time, the house schema became most active at the first level indicating that the last attended object was a house. Consequently, the suburb schema became the

[1]Note that these second-level schemas are not intended to be general representations of these scenes. They are conjured up to test the performance of VISOR in highly ambiguous situations.

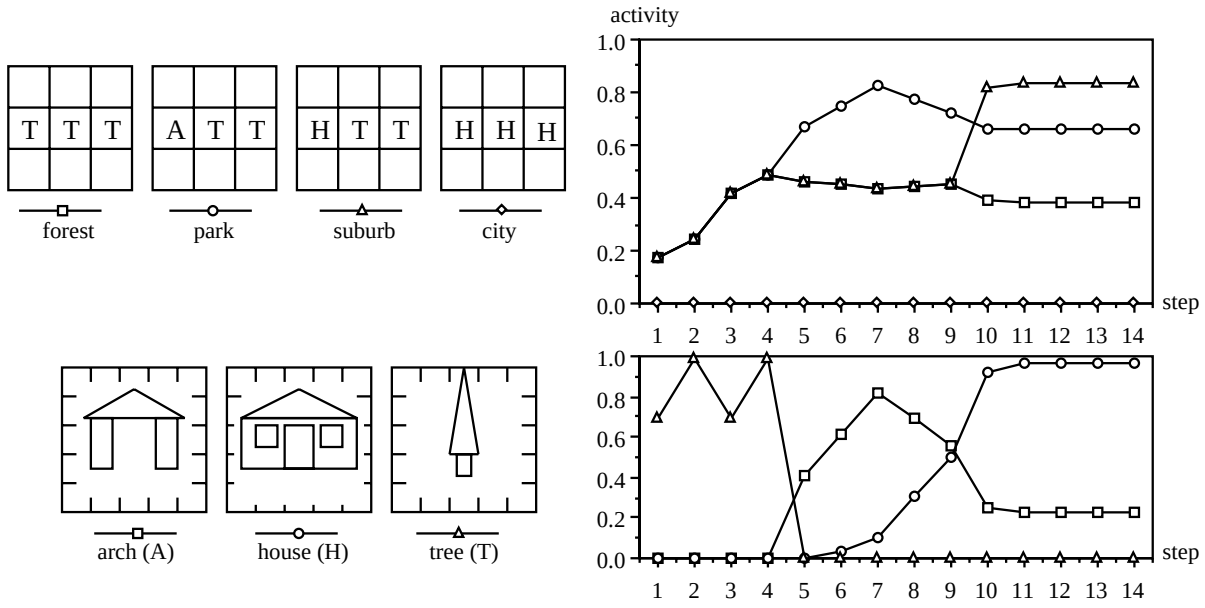


Figure 3: Processing a suburb image. The schemas were hand-coded as shown at left. The first-level schemas consisted of $5 \times 5$ units, and the second level schemas $3 \times 3$ units; "A" stands for arch, "H" for house, and "T" for tree. The time course of schema activation is shown at right. The first-level schemas are plotted in the bottom graph, the second-level schemas on the top graph.

most active second-level schema. After processing all the objects in the scene, the schemas' activities stabilized and the process terminated.

## 4 Schema learning

The Response Module serves as an interface between the Schema Module and the external environment (Fig. 1). It learns to associate the schema-net activations with the target responses (scene and object labels) that the environment expects as a result. At the same time, the Schema Module develops a hierarchy of visual schemas based on the environment's feedback. Therefore, learning is multi-modular,[2] consisting of interacting components that simultaneously learn different parts of the overall task.

Let us first consider how a single schema can be learned in the Schema Module. Initially the connection weights in the Schema Module have small positive random values. Given an input object and the target response, VISOR performs low-level visual processing and schema recognition as described in Section 3, and as a result, one of the schema-nets becomes most strongly activated. Its connection weights are then modified to encode part of the input object. As attention shifts to other parts of the object, the same schema-net remains most active and its weights are further modified to encode other parts of the object. After presentations of several different instances, the weights of the schema-net will gradually

[2] Multi-modular learning schemes have also been proposed by Grossberg and Kuperstein [9], and Miikkulainen [15].

converge to stable values that encode the essential structure of the object, and allow it to recognize further instances with minor variation. At the same time, the Response Module learns to associate the target response with the activation of this particular schema-net.

In order to learn to represent several schemas in the Schema Module, a reinforcement signal is required to distinguish among them. Suppose that VISOR has already learnt about the arch schema, and the environment is teaching it a house. A house image is fed to the Low-Level Visual Module and a *house* response to the Response Module. If the Schema Module finds the house to be significantly different from the arch, then a schema-net different from the arch schema-net will become most active. The Response Module will produce no response since nothing has yet been associated with this schema-net. The environment will deliver reward signals to both the Schema Module and the Response Module, and learning will proceed as in the single-schema case. On the other hand, if the Schema Module finds the house to be very similar to the arch, the arch schema-net will be most active and the Response Module will produce the *arch* response. In this case, the environment must deliver a punishment signal to the Schema Module and the Response Module. After receiving the punishment signal, the Schema Module suppresses the activity of the arch schema-net so that a different schema-net can become most active and correct learning can proceed as in the single-schema case. The punishment signal is very similar to the mismatch-reset signal in the ART network [4]. It tells the Schema Module to find a different schema-net to encode the house without specifying which one.

Learning begins with a number of schema-nets connected in a hierarchy, each with small random weights, and therefore encoding no spatial structure. The object schemas are learned first, and after that, the scene schemas begin to organize. Fig. 4 illustrates an example of learning to encode a suburb after mastering the arch, house, tree, forest and park schemas. At the first presentation of the suburb scene, the park schema-net was most active. The Response Module produced the *park* response which was found incorrect. The environment delivered a punishment signal, suppressing the activity of the park schema-net. At the second presentation, the forest schema-net became most active, and *forest* was produced as the response. This was again incorrect; a second punishment signal was delivered and it suppressed the forest schema-net's activity. At the third presentation, an unused schema-net (marked as "suburb" in Fig. 4) became most active. No response was produced since none was associated with it. The environment delivered a reward signal, allowing the most active schema-net to learn, and the Response Module to form the correct association. At subsequent presentations, the same schema-net remained most active, and its activity grew as its weights adapted to better encode the suburb scene.

## 5 Conclusions

The main goal of this research is to develop representation and learning methods for visual schemas in neural networks. The representation scheme supports integration of successive information so that scene analysis can be accomplished with limited processing resources. The multi-modular learning method allows the Schema Module to self-organize yet allows external intervention to correct recognition error. Learning new schemas does not erase previously established schemas, while relearning old schemas corrects representation errors.

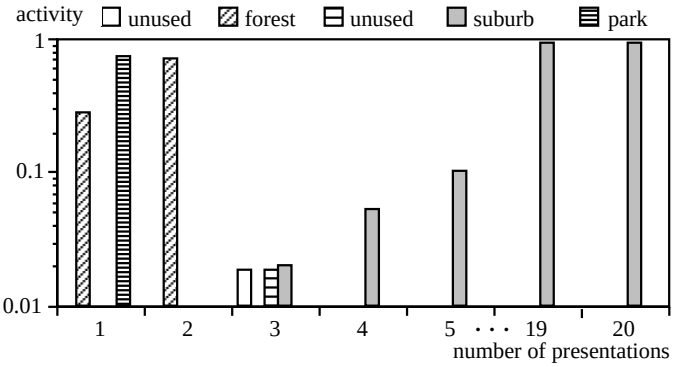


Figure 4: The process of learning to encode a suburb after learning forest and park scenes. At the first two presentations of the suburb scene, the park and the forest schema-nets were most active, and their activities were subsequently suppressed. Starting from the third presentation, the most active unused schema-net (marked as "suburb") began to encode the suburb schema.

We are currently extending VISOR to process more realistic scenes. Compared to the "blocks world" scenes in the above examples, real world scenes contain more variability: (1) In many cases, the relative positions of the components are not important in determining the scene. For example, a house can be anywhere among the trees in a suburb scene. (2) The components do not always have recognizable shapes. For example, the trees are usually merged into regions with irregular shapes and approximately uniform texture. Similarly, roads and rivers and patches of grass are just regions with uniform texture. Our first results are promising: by learning to encode knowledge about scene components and textural features, VISOR has successfully analyze certain classes of real world scenes as well.

## Acknowledgements

This research was supported in part by NSF grant #IRI-9309273.

## References

[1] Antes, J. R., and Penland, J. G. (1981). Picture context effects on eye movement patterns. In Fisher, D. F., Monty, R. A., and Senders, J. W., editors, *Eye Movements: Cognition and Visual Perception*. Erlbaum.

[2] Arbib, M. A. (1989). *The Metaphorical Brain 2: Neural Networks and Beyond*. New York: Wiley.

[3] Biederman, I. (1972). Perceiving real-world scenes. *Science*, 177:77–80.

[4] Carpenter, G. A., and Grossberg, S. (1987). A massively parallel architecture for a self-organizing neural pattern recognition machine. *Computer Vision, Graphics and Image Processing*, 37:54–115.

[5] Draper, B. A., Collins, R. T., Brolio, J., Hanson, A. R., and Riseman, E. M. (1989). The schema system. *International Journal of Computer Vision*, 2:209–250.

[6] Feldman, J. A. (1986). Connectionist models and parallelism in high level vision. In Rosenfeld, A., editor, *Human and Machine Vision II*. New York: Academic Press.

[7] Feldman, J. A., and Ballard, D. H. (1982). Connectionist models and their properties. *Cognitive Science*, 6:205–254.

[8] Friedman, A. (1979). Framing pictures: The role of knowledge in automatized encoding of memory for gist. *Journal of Experimental Psychology: General*, 108:316–355.

[9] Grossberg, S., and Kuperstein, M. (1989). *Neural Dynamics of Adaptive Sensory-Motor Control*. New York: Pergamon Press.

[10] Hanson, A. R., and Riseman, E. M. (1978). VISIONS: A computer system for interpreting scenes. In Hanson, A. R., and Riseman, E. M., editors, *Computer Vision Systems*. New York: Academic Press.

[11] Hinton, G. E. (1988). Representing part-whole hierarchies in connectionist networks. In *Proceedings of the 10th Annual Conference of the Cognitive Science Society*. Hillsdale, NJ: Erlbaum.

[12] Hochberg, J. E. (1978). *Perception*. Englewood Cliffs, NJ: Prentice-Hall. Second edition.

[13] Leow, W. K. (1994). *VISOR: Learning Visual Schemas in Neural Networks for Object Recognition and Scene Analysis*. PhD thesis, Department of Computer Sciences, The University of Texas at Austin.

[14] Leow, W. K., and Miikkulainen, R. (1993). Representing visual schemas in neural networks for object recognition. In *Proceedings of the IEEE International Conference on Neural Networks* (San Francisco, CA). IEEE.

[15] Miikkulainen, R. (1993). *Subsymbolic Natural Language Processing: An Integrated Model of Scripts, Lexicon, and Memory*. Cambridge, MA: MIT Press.

[16] Rumelhart, D. E., Smolensky, P., McClelland, J. L., and Hinton, G. E. (1986). Schemata and sequential thought processes in PDP models. In McClelland, J. L., and Rumelhart, D. E., editors, *Parallel Distributed Processing, Volume 2: Psychological and Biological Models*, 7–57. Cambridge, MA: MIT Press.

[17] Tsotsos, J. K. (1988). How does human vision beat the computational complexity of visual perception? In Pylyshyn, Z. W., editor, *Computational Processes in Human Vision*. Norwood, NJ: Ablex.

[18] Van Essen, D. C., and Anderson, C. H. (1990). Information processing strategies and pathways in the primate retina and visual cortex. In Zornetzer, S. F., Davis, J. L., and Lau, C., editors, *An Introduction to Neural and Electronic Networks*. New York: Academic Press.