Visual Schemas in Neural Networks for Object Recognition and Scene Analysis *

Wee Kheng Leow† and Risto Miikkulainen‡

†Department of Information Systems and Computer Science
National University of Singapore
Kent Ridge Road, Singapore 119260, Singapore
`leowwk@iscs.nus.sg`

‡Department of Computer Sciences
The University of Texas at Austin
Austin, Texas 78712, USA
`risto@cs.utexas.edu`

Abstract

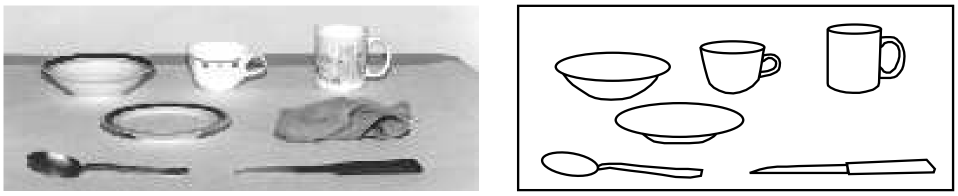
VISOR is a large connectionist system that shows how visual schemas can be learned, represented, and used through mechanisms natural to neural networks. Processing in VISOR is based on cooperation, competition, and parallel bottom-up and top-down activation of schema representations. Simulations show that VISOR is robust against noise and variations in the inputs and parameters. It can indicate the confidence of its analysis, pay attention to important minor differences, and use context to recognize ambiguous objects. Experiments also suggest that the representation and learning are stable, and its behavior is consistent with human processes such as priming, perceptual reversal, and circular reaction in learning. The schema mechanisms of VISOR can serve as a starting point for building robust high-level vision systems, and perhaps for schema-based motor control and natural language processing systems as well.

1 Introduction

Neural networks have been successfully applied to problems such as pattern recognition, time-series prediction, and process control. In these applications, typically the domain does not have an apparent or well-defined structure, and they can be solved based on correlations in the training data. Knowledge about the correlations can be encoded in the connection weights of a multi-layered neural network using a weight modification method such as backpropagation (Rumelhart et al. 1986a).

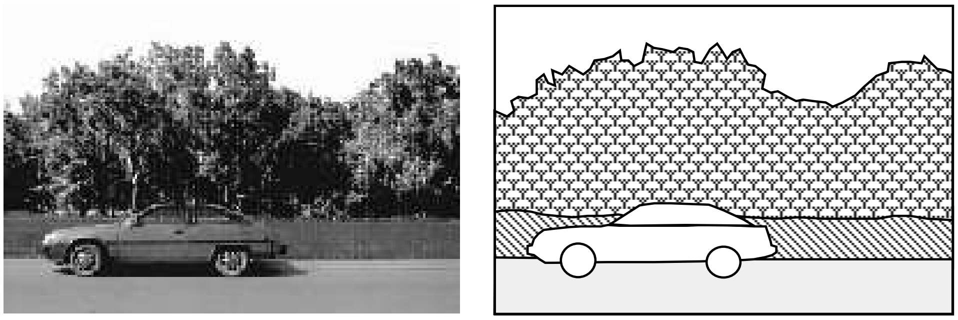
On the other hand, in applications such as object recognition, scene analysis, and natural language understanding, the domain has natural and well-defined structure. As an example, consider analyzing a simple scene such as a dining table of Fig. 1. It contains objects such as a knife, spoon, plate, bowl, cup, and a mug. These are rigid objects that consist of several regularly-shaped parts.

*To appear in *Connection Science*, 1997.



(a) (b)

Figure 1: **Unstructured input scene.** (a) The image of a dining table. (b) A visualization of the actual input to VISOR, which currently consists of a list of locations and shape descriptions of the components in the scene. The napkin has no rigid shape and is currently regarded as an unknown object.



(a) (b)

Figure 2: **Structured input scene** (a) The image of a road scene. (b) An outline of the road scene input to VISOR. The car is a rigid object which is segmented into four components: a trapezoid, a rectangle and two circles. The trees, the grass, and the road are represented as regions with approximately uniform texture.

For example, the knife has a long, rectangular handle and a long, pointed blade. The parts are connected at specific places to form an object. Since the objects may be placed anywhere on the table, the combination of these objects identifies the scene as a dining table, and the spatial arrangement of the objects is not important. On the other hand, the outdoor scene of Fig. 2 contains (in addition to rigid objects with regular shapes such as the car) also large irregularly-shaped regions with approximately uniform texture such as road, grass, and tree foliage. These scene components have a more specific spatial layout than those on a dining table. For instance, the grass lies on the ground, and the trees grow upward. In order to understand scenes such as these, it is not enough to store correlations between pixel values; one must be able to represent knowledge about spatial structure as well.

In general, in designing a neural network system for a structured task such as understanding scenes, three fundamental problems need to be addressed:

1. How can a neural network represent and use knowledge about structure, such as object and scene layouts?

2. How can a network learn such knowledge from examples?

3. How can a fixed, finite neural network process indefinitely large number of structures, such as those presented by scenes with many objects?

This article describes a neural network system called VISOR (VIsual Schemas for Object Representation; Leow 1994; Leow and Miikkulainen 1993, 1994a, 1994b) that was specifically designed to address these issues. These problems are difficult because neural networks most naturally process information based on unstructured correlations. The approach taken in VISOR is to develop solutions from the perspective of object recognition and scene analysis. Structure is represented in terms of schema hierarchies, implemented as laterally and vertically connected topological maps. Several experiments with VISOR are presented in this article, demonstrating the effectiveness, robustness, and psychological validity of this approach. Implications for schema theory and for building robust vision systems are also discussed.

2 Schema Theory

Schema theory has been used by many scientists to account for human perceptual and cognitive functions (Arbib 1975, 1987, 1989; Bartlett 1932; Head and Holmes 1911; Hochberg 1978,Piaget 1952, 1971; Rumelhart 1975, 1980;Rumelhart and Ortony 1977; Schmidt 1975). A schema is a package that contains stereotypical knowledge about an object or a process. Schemas serve as building blocks for cognitive functions, modeling cognition at an intermediate level between behaviors and their neural implementations.

In the visual domain, schemas describe visual objects in terms of the physical properties and the spatial arrangements of their components. Consider the schema for a deer for example. The real-world deer has several parts such as a head, a neck, a body, and four legs. Each part has a characteristic shape and size: the head is a small triangular block, the neck a long cylinder, the body a large rectangular block, and the legs are long and slim cylinders. The parts are arranged in more or less specific locations. The head is attached to one end of the neck, and its other end is attached to the body. The four legs are attached to the bottom of the body to support it. All such information must be represented in the schema for a deer.

Much of the schema theory has been developed in the symbolic framework (Arbib 1989; Draper et al. 1989; Hanson and Riseman 1978; Rumelhart 1980). Based on the symbolic approach, the three main functional characteristics of visual schemas can be summarized:

1. Visual schemas can be organized into a schema hierarchy, with scene schemas at the topmost level, object schemas at the next level, and so on.

2. Schema instances representing the components of an object cooperate to support the instantiation of the object schema.

3. Schema instances representing different objects (or scenes) compete to determine which one best matches the inputs.

These characteristics can very naturally be implemented in neural networks. However, although such schema system implementations have been proposed (Arbib 1989; Hinton 1990), and there are systems that simulate parts of the process such as hierarchical structures (Feldman 1985; Zemel et al. 1990), cooperation and competition (Rumelhart et al. 1986b), and dynamic bindings (Hummel and Biederman 1992), no complete visual schema systems have been built so far on neural networks.

The main motivation for neural network schema systems, besides an elegant implementation of the cooperative and competitive processes, is that several powerful learning mechanisms exists for neural networks. If a way can be found to implement schemas with neural networks, it might also be possible to develop mechanisms for learning schemas from examples. Schema learning has turned out a very difficult problem in the symbolic approach; perhaps with neural networks, this

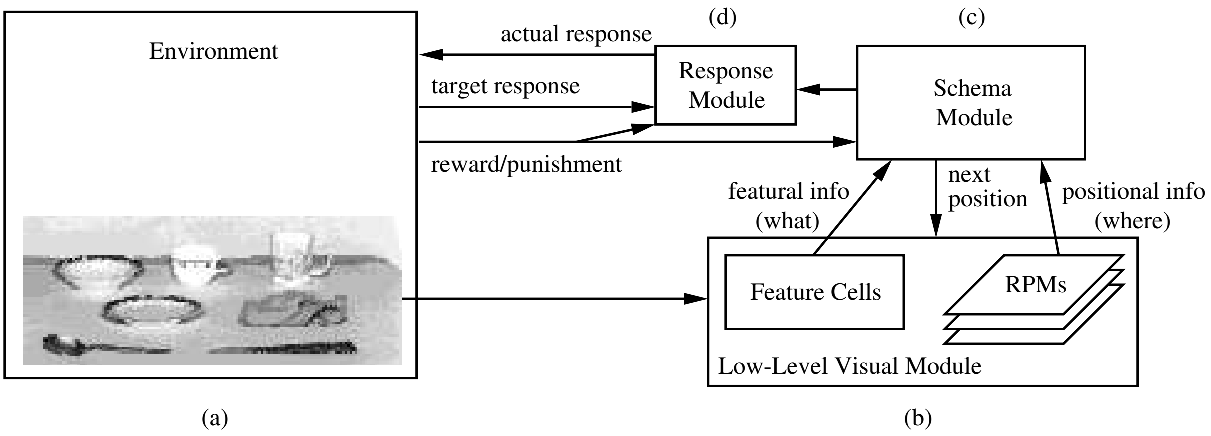


Figure 3: **The VISOR architecture.** VISOR consists of the Low-Level Visual Module (LLVM, b), the Schema Module (c), and the Response Module (d). LLVM extracts "what" and "where" information from the scene (a). The Schema Module performs scene analysis and the Response Module produces scene labels for the environment.

barrier can be broken. From the point of view of schema theory, this is the main contribution of VISOR.

3 The VISOR System

The core of VISOR research consists of methods for representing and learning visual schemas in neural networks. Simplifications have been made to render the task tractable and to focus on the main issues. Instead of actual bitmaps, the system processes symbolic descriptions of the input, like those visualized in figures 1b and 2b (and described in detail in Appendix B). The objects and scenes are treated as two-dimensional structures, and objects have fixed sizes and do not occlude each other. However, the objects in an unstructured scene (such as Fig. 1) may be located anywhere in the scene and tilted from their typical orientations. The outdoor scenes (such as Fig. 2) are divided into three sections: the sky, the middle level where most of the objects are located, and the ground level. The knowledge that describes objects and scenes involves only four positional relationships (left, right, above, and below) and one hierarchical relationship (part-of). With these simplifications, the research can focus on the main issues of representing, applying, and learning visual schemas for high-level vision.

In the following, VISOR's main modules and how VISOR represent schemas in maps and connections are first described. The cooperation and competition among schemas during the analysis process is then reviewed, followed by VISOR's method of learning to encode schemas from examples. The details of these processes are presented in Appendix A.

3.1 Architecture

VISOR consists of three main components: (1) the Low-Level Visual Module extracts positional and featural information from the scene, (2) the Schema Module matches schemas with inputs, and (3) the Response Module generates the object and scene labels expected by the environment (Fig. 3).

The Low-Level Visual Module (LLVM) has the task of providing input to the Schema Module. The LLVM is currently simulated procedurally. As its input, it receives a symbolic representation of a real-world scene that has been parsed into objects and their components (Appendix B). The LLVM focuses attention at one component at a time and produces three outputs: the shape of the component encoded as a distributed activity pattern over a set of shape feature cells, the position of the component represented in Relative Position Maps (RPMs), and the suggested next position of attention encoded in Next Position Maps (NPMs).

The distributed activity pattern over the shape feature cells represents the shape of the attended component in terms of attributes such as length, breadth, closure, vertical tilt, horizontal tilt, degree of expansion, and curvature (Biederman 1987b). These attributes are measured relative to the object's main axis, and are thus invariant with respect to the object's orientation. Each attribute is represented by a set of feature cells that are maximally sensitive to different ranges of values (e.g., small, medium, large). If the focused object is a region with approximately uniform texture but no specific shape (such as a grass patch), the texture of the region is extracted and encoded in the texture feature cells. Each of these cells corresponds to the output of a filter channel (Bovik et al. 1987) that is maximally sensitive to a particular type of texture.

The Relative Position Maps (RPMs) represent the position of the attended component at two different scales. At the scale level of individual objects, the RPM encodes the position of the component relative to the entire object. At the scene level, the RPM represents the position of the attended object relative to the entire scene.

The featural and positional information extracted by the LLVM is fed into the Schema Module (Fig. 3c), where it is matched with the schema representations. The Schema Module is a multi-layer network of schema representation nets, or schema-nets for short (Fig. 4). Each layer of schema-nets corresponds to a level in the schema hierarchy, with the scene schemas at the top and the object schemas at the bottom. Consider the representation of the spoon (Fig. 4a), which consists of two components: an elliptical part and a long, slim handle. The spoon's spatial layout is represented by a two-dimensional spatial map called the Sub-schema Activity Map (SAM) in the spoon schema-net. For example, the spoon's handle is represented by the right middle SAM unit. The weights of the connections from the shape feature cells to this unit have values such that the unit is most highly activated when the input at this location is a long, slim rectangle. Similarly, the weights of the connections to the left middle SAM unit encode a moderately large ellipse. The SAM activities are summed up by the spoon schema's output unit, indicating how well the schema matches the entire object.

The objects in a table scene, such as the dining table (Fig. 1), may be located anywhere in the scene. The SAM of such a scene schema encodes only one position representing the entire scene, but each position can contain more than one object (shown as a SAM column in Fig. 4). For example, the connection from the spoon schema's output unit to the corresponding dining table SAM unit indicates that the dining table may contain a spoon anywhere in the scene. Some objects, such as the knife, may appear e.g. on a dining table as well as on a workbench. Therefore, the knife schema's output unit connects to the SAM units of both the dining table and the workbench schemas (Fig. 4).

The objects and regions in an outdoor scene, such as a road scene (Fig. 2), are located in typical areas of the scene, but there may be multiple objects in the same area as well. The spatial layout of such a scene is encoded in a 3-D SAM which, for outdoor scenes, reduces to 3 vertical positions representing the sky, the middle portion of the scene, and the ground level (Fig. 4). Each position contains a SAM column that encodes the objects or regions that may appear in that portion of the scene. For example, figure 4 shows a road scene schema that has a road region at the bottom of the scene.

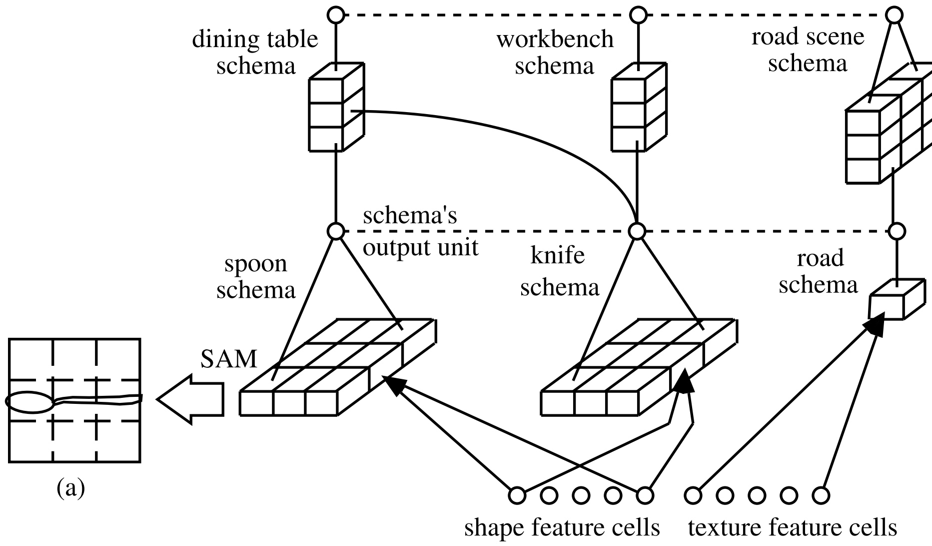


Figure 4: **The hierarchy of schema-nets.** Schemas' output units are represented as circles, and the units of the Sub-schema Activity Maps (SAMs) as cubes. Arrows represent one-way connections, solid lines represent both the bottom-up and top-down connections (which are different), and dotted lines denote the inhibition among the schemas' output units. (a) The SAM units represent the relative positions of the spoon's components.

3.2 Recognizing Objects and Scenes

This section presents a high-level description of how VISOR processes a scene such as Fig. 3a (see Appendix A.1 for details). Given an input scene, the LLVM focuses attention at one component of the scene, say the elliptical part of the spoon, and (through procedural programs) generates a feature representation of its shape in the shape feature cells. Next, the activation propagates up to the SAMs of the object schemas. Only those SAM units that match the current position in the Relative Position Maps are enabled (in this case the left middle SAM units), and they compare the featural inputs with their input weights by computing a weighted sum. The result is encoded in the SAM units' activities, which then propagate to the object schemas' output units. These units indicate how well the entire schemas match the inputs. In this example, the spoon schema is most strongly activated since its left middle SAM unit best matches the left component of the input object. In other words, VISOR believes that the input object is probably a spoon.

After processing the inputs at the current position, the schemas each suggest a next position of attention where another component is expected. The suggestions are encoded in Next Position Maps (NPMs) similar to those of the LLVM. Based on the schemas' and the LLVM's suggestions, the Schema Module selects one position to be adopted by the whole system. The selected position is sent to the schemas and the LLVM, which then shift attention to the new location, and the process repeats.

Schemas are activated both bottom-up and top-down. The spoon schema's activity is propagated upwards to activate the dining table schema. At the same time, the dining table schema propagates its activity back to the spoon schema to indicate that the spoon is indeed expected on the dining table. Receiving both bottom-up and top-down inputs, the spoon schema becomes more active and it tends to decrease the activities of other schemas through inhibitory connections between schemas' output units. This way the schemas cooperate and compete to determine which one best matches the inputs.

The environment does not have to peek into the Schema Module of VISOR to determine the recognition results. Instead, it receives the output response (a label) generated by the Response Module (Fig. 3d), based on the current activation of the schema hierarchy. The Response Module also plays an important role in learning new schemas.

3.3 Learning to Encode Schemas

Let us consider how VISOR learns to encode the spoon schema. The environment presents the image of a spoon to the LLVM and the *spoon* label to the Response Module as the target response (Fig. 3). As a result of the interactions among the schema-nets, one of them becomes most strongly activated. There are three possible outcomes, each indicated by a different response by the Response Module:

1. Suppose that the most active schema-net has not yet encoded any schemas. This means that there is no schema in the system that matches the input well (the schema nets are initialized to low weight values which makes them a poor match with anything). In this case, the Response Module will produce no output response, and the environment will deliver a reward signal to VISOR. The most active schema-net now modifies its weights to encode the spatial structure of the spoon, and the Response Module learns to associate the activation of the newly-formed spoon schema with the target *spoon* label. The weights are modified through variations of the Hebbian process (see Appendix A.2 for details).

2. If the most active schema-net happens to be the newly formed spoon schema, then the Response Module will produce the correct *spoon* label as the response. The environment will deliver a reward signal to VISOR and learning continues as in the first case.

3. Suppose VISOR finds that the input matches, say, the knife schema that has already been learned earlier. The Response Module will produce the *knife* label which is incorrect. In this case, the environment will deliver a punishment signal to inform VISOR of the error. This signal suppresses the knife schema-net's activation so that a different schema-net can become most active and correct learning can proceed as in the first case. In other words, the punishment signal plays a similar role as the mismatch-reset signal in the ART network (Carpenter and Grossberg 1987). It tells the Schema Module to find a different schema-net for the spoon without specifying which one.

The same process takes place when VISOR learns to encode a scene schema. When VISOR receives an input image that contains more than one object, it assumes that the image consists of a scene, and that the target label presented by the environment should be associated with the schema that describes the scene. On the other hand, if the image consists of only one object, then the label should be associated with an object schema.

In other words, VISOR makes use of both unsupervised adaptation (in learning the structure of a schema) and reinforcement signals (in deciding when to start a new schema). Such a learning scheme results in very robust processing properties, as will be discussed below.

4 Schema-Based Recognition in VISOR

VISOR's recognition process is based on cooperation, competition, and parallel bottom-up and top-down activation of schema representation networks, which results in several powerful properties. This section presents six experiments, demonstrating

dining table	workbench	desk	kitchen-top
cup	hammer	stapler	chopper
mug	chisel	letter opener	knife
plate	screwdriver	paper punch	turner
bowl	wrench	pen	rice scoop
spoon	pliers	stamp	ladle
knife	long-nose pliers	scissors	sauce pan

Table 1: **Typical objects in the table scenes.**

road scene	house scene
tree foliage	tree foliage
grass	grass
road	road
car	car
	house

Table 2: **Typical objects in the outdoor scenes.**

1. Competition among object schemas;
2. Representing confidence of the object recognition process;
3. Distinguishing between different objects that look alike;
4. Cooperation and competition among object and scene schemas;
5. Representing the importance of objects in activating scenes schemas; and
6. Contextual disambiguation.

Let us first describe the input data and the learning task in these experiments.

4.1 The Learning Task

In the first five experiments, four types of unstructured table scenes were used: dining table, workbench, desk, and kitchen-top, consisting of a total of 23 different objects (Table 1). The objects can be identified by the spatial arrangements and the shapes of their components, and the scenes by the combinations of objects. In the sixth experiment, two types of outdoor scenes, namely road scene and house scene, were used. The objects that appear in these scenes are listed in Table 2. To illustrate that VISOR can discriminate between ambiguous objects based on the spatial layout of a scene, the texture qualities of the tree foliage and the grass were chosen to be the same. Disambiguation of these regions is possible only by taking into account their spatial context: the foliage is in the middle section and the grass is in the bottom.

VISOR was trained to represent the four types of table scenes and the two types of outdoor scenes starting with 30 schema-nets at the first level, and 8 nets at the second level. The Subschema Activity Maps (SAMs) at the first level had 8×8 spatial positions and room for 1 component in each position. The schema-nets that learned to represent region schemas actually used only one SAM unit to encode the texture of the region. The SAMs of the second level schema-nets had 3 positions (corresponding to the top, middle, and bottom part of a scene) of 10 components each. The schema-nets that developed for unstructured scenes used only the bottom SAM column for the entire scene.

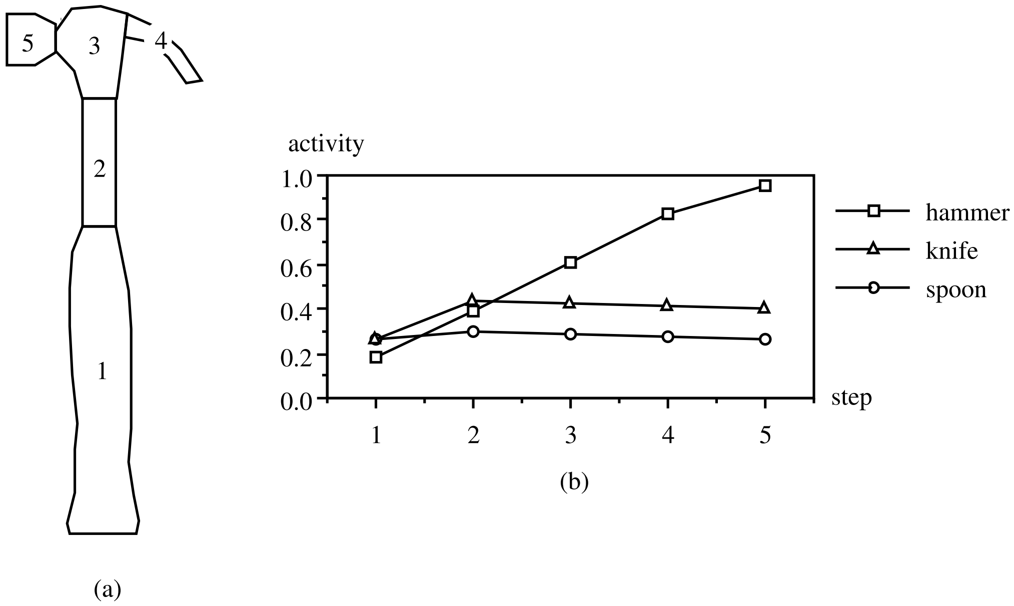
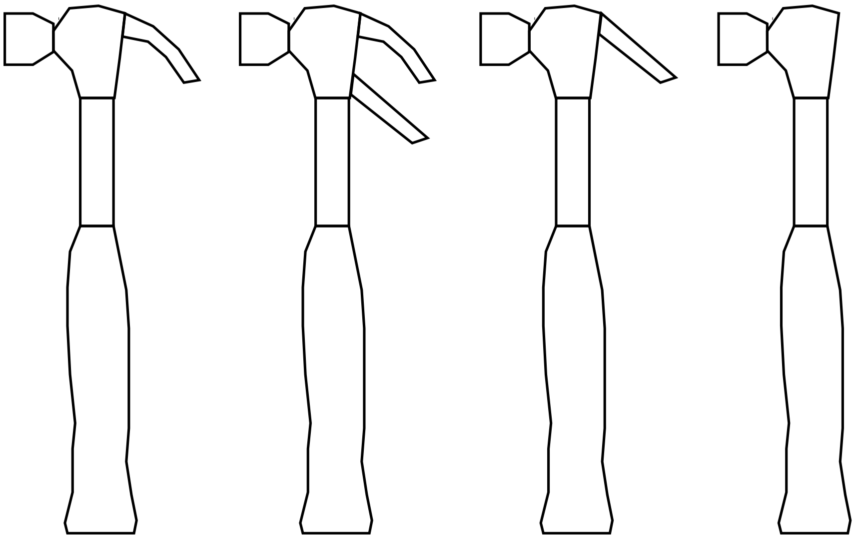


Figure 5: **Competition among object schemas.** (a) The sequence of positions of attention in processing the hammer input. (b) Activities of the object schemas. Initially, VISOR was unable to determine what the input object was. However, after looking at all the components, VISOR confidently concluded that it was a hammer.

For experiments 1–4 and 6, VISOR was trained only once with only one stereotypical example of each object and scene. For the fifth experiment, demonstrating that VISOR can learn to encode how important objects are in activating scene schemas, VISOR was trained twice with different sets of example scenes consisting of varying combinations of objects from Table 1. In all the training and experiments, the same set of system parameters was used (listed in Appendix C).

4.2 Experiment 1: Competition Among Object Schemas

This experiment illustrates how the schemas compete to match an input object. After VISOR had successfully learned the scene and object schemas of Table 1, it was shown the image of a hammer, and was set to initially attend to the handle of the hammer (Fig. 5). Since the shape of the handle looks similar to that of the knife and the spoon (Fig. 1), VISOR's recognition process began in an ambiguous state, and the schemas were activated by roughly equal amounts (recognition step 1). The second component of the hammer resembles the knife's blade more than the spoon's elliptical part. As a result, at step 2, both the activities of the hammer and the knife schemas increased more than that of the spoon schema. However, the hammer schema's activity was still lower than the knife's. Disambiguation occurred at step 3 when VISOR focused attention the component that exists only in the hammer. As VISOR focused successively at the remaining parts, the hammer schema's activity increased indicating that it matched the input better and better. In addition, the hammer schema suppressed the activations of the knife and the spoon schemas through inhibitory connections between the schemas' output units, causing the activities of the knife and the spoon schemas to decrease. After looking at all the components, VISOR clearly indicated that the image was most likely a hammer.



(a) (b) (c) (d)

Figure 6: **Variations of the hammer.** (a) A typical hammer presented to VISOR during the training. (b) A hammer with an extra component, (c) one with a different part, and (d) one missing a part.

schema's activity	figure			
	6a	6b	6c	6d
hammer	**0.96**	**0.96**	**0.93**	**0.83**
knife	0.39	0.39	0.39	0.40
spoon	0.25	0.25	0.25	0.25

Table 3: **Confidence of the object recognition process.** The table shows the schema activations after processing the hammer variations of Fig. 6. As the input object differed increasingly from the schema, VISOR's confidence that the object was a hammer decreased slightly. Nevertheless, the hammer schema's activities were much larger than those of its strongest competitors indicating that VISOR was able to recognize the inputs as variations of the hammer.

4.3 Experiment 2: Confidence of the Object Recognition Process

In real images, the inputs often differ slightly from prototypical examples due to normal variation, occlusion, damage to the input objects, noise, and so on. The second experiment shows that VISOR can recognize an object despite such variations and can also indicate the confidence of its analysis. VISOR had learned to encode the hammer (as well as the other objects listed in Table 1) based on the perfect image shown in Fig. 6a. In this experiment, variations of the hammer (Fig. 6) were presented to VISOR. The recognition results are summarized in Table 3.

When shown the standard hammer (Fig. 6a), the hammer schema attained an activity level of 0.96 whereas the activities of the other schemas were below 0.5, indicating that VISOR was very confident of its analysis. The hammer schema's activity remained at 0.96 in response to the image in Fig. 6b because the additional component did not match any component of the hammer schema or its competitor, and was simply ignored (similar results have been observed in psychological experiments of human object recognition; Biederman 1987a). In general, the additional part might match a competing schema, in which case the activity of the competitor would increase, and,

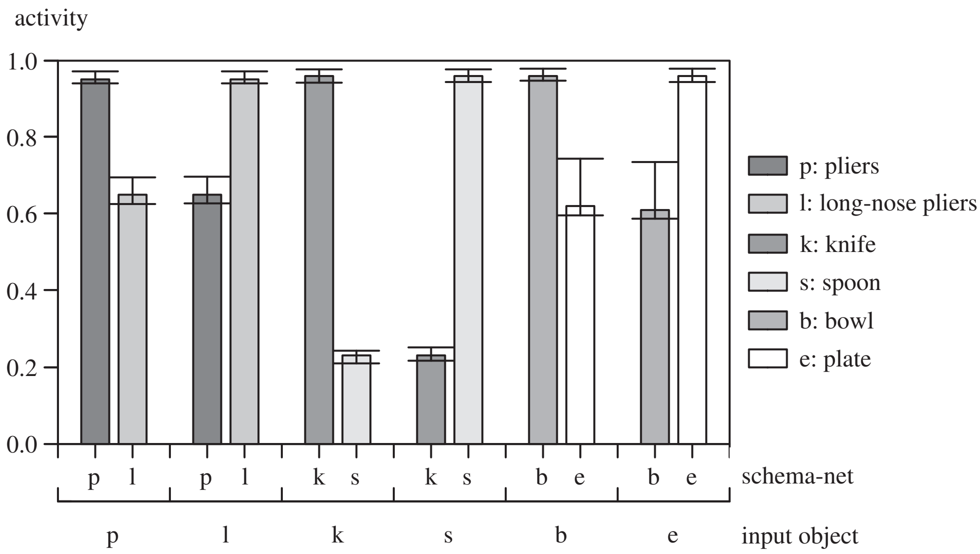


Figure 7: **Distinguishing between similar objects.** This graph summarizes the results of six input presentations to VISOR. The input object is named at the bottom of the figure, and the columns show for each input the resulting activation of the correct schema and its closest competitor. For example, given ordinary pliers as the input, the pliers schema becomes highly activated with its competitor, long-nose pliers, a distant second, although the two objects are very similar. The difference indicates how confident VISOR is of the distinction. The error bars show the highest and lowest activations under varying parameter settings (discussed in Experiment 10).

as a result, lower the hammer schema's activity. The claw of the hammer in Fig. 6c differed slightly from that of the standard hammer (Fig. 6a). The claw's shape, as encoded in the shape feature cells, only partially matches that of the standard claw represented in the hammer schema's connection weights. As a result, the hammer schema's activity decreased slightly, to 0.93. The more the input shape differs from the encoded one, the lower the hammer schema's activity. When the claw was completely missing, as in Fig. 6d, the hammer schema's activity dropped further to 0.83. Nevertheless, the hammer schema's activity was still significantly larger than those of its competitors. VISOR can therefore recognize the variations by determining how well they match the single prototype encoded in the schema.

4.4 Experiment 3: Distinguishing between Similar Objects

As was shown in Experiment 2, VISOR can robustly recognize variations of the same object, but it can also perform the complementary task: distinguishing between different objects that look alike, such as the ordinary pliers and the long-nose pliers, the knife and the spoon, and the plate and the bowl. In the third experiment, the above objects were presented to VISOR. Fig. 7 shows that the schema that correctly classifies the input object (e.g. the pliers schema in the first case) always attains a very high activity level, and its activity is always clearly larger than that of the competing schema (i.e. the long-nose pliers schema).

The ability to perform both the recognition and discrimination tasks robustly results from the interplay between schema matching and schema competition. Given variants of the hammer (Experiment 2), only the hammer schema matches the inputs well, and it receives little competition from other schemas. VISOR is thus able to conclude that such inputs must all be variants of the hammer. On the other hand, although an input object such as pliers looks like long-nose pliers, and activates the long-nose pliers schema to a degree, the pliers schema is activated even more. The competition among the schemas magnifies the difference, and the pliers schema emerges as a clear winner.

Certain input objects may be ambiguous and may match, say, both the schema for pliers and the one for long-nose pliers equally well. Both schemas would be equally strongly activated, but neither would be very highly active, indicating that the objects could belong to either category. In artificial vision applications of VISOR, such a classification of ambiguous inputs into several possible categories can be very useful. It constitutes a *characterization* of the object, and the final disambiguation may be performed based on additional information not available to VISOR.

4.5 Experiment 4: Cooperation and Competition Among Object and Scene Schemas

The fourth experiment illustrates how VISOR processes an entire dining table scene, and demonstrates how object and scene schemas cooperate and compete to interpret the input. In the experiment, an atypical dining table scene was presented to VISOR. In addition to the objects that typically appear on a dining table (Table 1), this scene also contained a hammer and a screwdriver that normally appear only in the workbench scene. VISOR was given a false lead: it was set to begin by focusing attention at the hammer. At step 5, the hammer was identified (Fig. 8a) and VISOR believed that the input scene was probably a workbench. After processing the hammer, VISOR shifted attention to the screwdriver. When the screwdriver was recognized at step 9, VISOR's initial belief was strengthened. After identifying the cup and the mug at steps 12 and 15, the dining table schema received strong activation from these schemas. Its activity started to increase and reduced the workbench schema's activity through competition. At step 15, VISOR was very confused: the dining table and the workbench schemas were equally active. However, after recognizing the plate (step 17), the knife (19), the spoon (21), and the bowl (23), the activity of the dining table schema increased drastically, and the belief in the workbench schema decreased even further. In the end, VISOR was very confident that the input scene was a dining table. VISOR thus demonstrated behavior similar to human recognition process: It tolerates variations in a scene, and recognizes the scene even if it accidentally contains some foreign objects. At the beginning of the recognition process, VISOR may be misled by the foreign object, however, after looking at other parts of the scene, it can change its initial belief and finally make the correct interpretation.

4.6 Experiment 5: Importance of an Object in Activating Scene Schemas

Some schema components are more important than others in activating a schema. For instance, a hammer appears more often than a knife in a workbench scene. Finding a hammer in the input scene therefore contributes stronger evidence that the scene is possibly a workbench. If a schema component i appears n_i times during the learning process, then the normalized number of appearances $n_i/\sum_j n_j$ indicates the importance of the component in activating the schema. VISOR learns to encode this information in the weights of the feedforward connections from the SAM units to the schema's output units (Fig. 4).

In the fifth experiment, VISOR had already learned about the desk, the dining table, and the kitchen-top, and it was trained to recognize the workbench under two different learning conditions, labeled L90 and L90/50:

1. **L90:** Each salient object (listed in Table 1) was present in 90% of the training scenes;
2. **L90/50:** Four of the salient objects were present 90% of the time, and the other two were

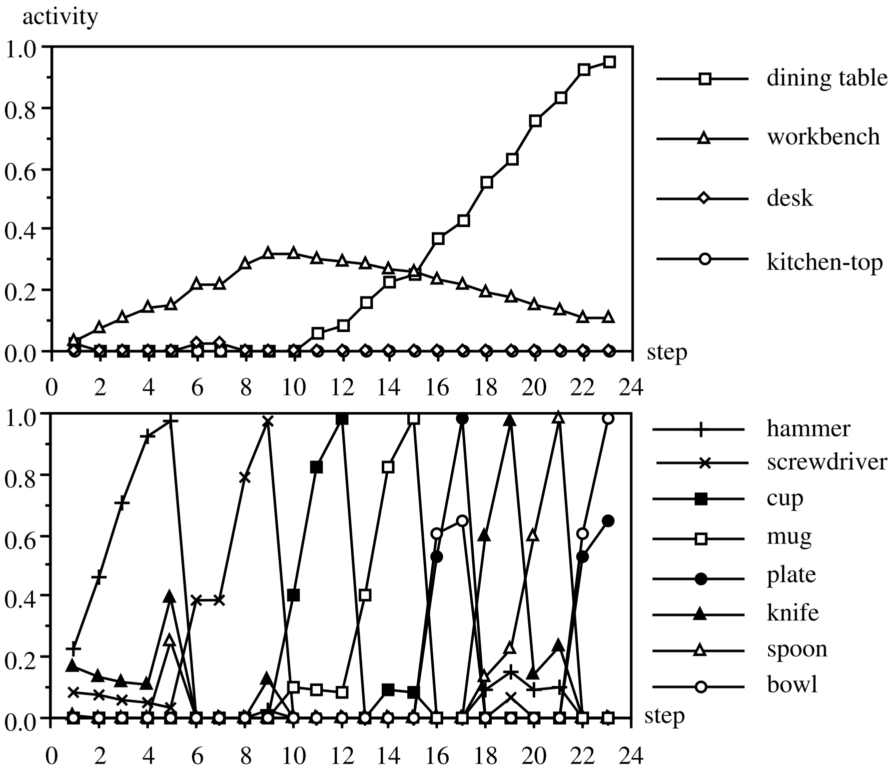


Figure 8: **Cooperation and competition among the object and scene schemas.** VISOR analyzed an atypical dining table scene that contained a hammer and a screwdriver. (a) The activities of the object schemas, and (b) those of the scene schemas. VISOR was set to begin by focusing attention at the hammer. After identifying the hammer (step 5) and the screwdriver (step 9), VISOR thought that the input scene was most likely a workbench. However, as other objects in the scene were recognized, VISOR concluded that the input scene was actually a dining table.

present 50% of the time.

The results show that the feedforward weights indeed learned to approximate the importance of schema components. Figures 9 show the feedforward weights learned under the condition L90/50. During training, four of the salient objects appeared 90% of the time, and the weights that represent these objects approximate the theoretical value of 0.196 ($= 0.9/4.6$, where the expected number of objects in the scene is $4.6 = 0.9 * 4 + 0.5 * 2$). The other two objects appeared only 50% of the time, and the corresponding weights approximate 0.109.

After learning under each condition, VISOR was given workbench scenes to recognize with varying number of salient and distractor objects. The distractors (mug, pen, and knife) had only appeared in scenes other than the workbench during training. As the number of salient objects decreased and the number of distractor objects increased, VISOR's confidence about its analysis decreased gradually (Table 4). The confidence decreased *slower* under condition L90/50 because the missing salient objects appeared only 50% of the time during training (weight ≈ 0.109), and, therefore, VISOR already expected to see some of them missing. In other words, VISOR used the importance encoded in the feedforward weights to determine how important the objects were in activating the scene schema and the confidence of its analysis.

In artificial vision applications, the ability to indicate confidence is important because the

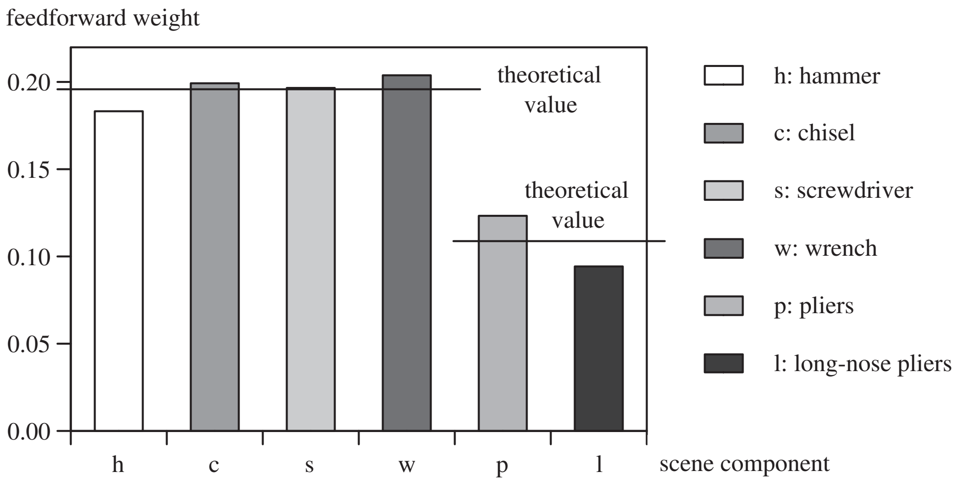


Figure 9: **Learning to encode object importance.** Under condition L90/50, four salient objects appeared 90% of the time, and two others appeared only 50% of the time. The feedforward weights representing the four objects approximate the value 0.196, and those of the other two objects approximate 0.109.

number of distractors	L90			L90/50		
	number of salient objects					
	6	5	4	6	5	4
0	**0.95**	**0.85**	**0.84**	**0.95**	**0.92**	**0.90**
1	**0.95**	**0.85**	0.64	**0.95**	**0.92**	0.81
2	**0.95**	**0.82**	0.64	**0.95**	**0.91**	0.81

Table 4: **Recognition results under different learning conditions.** As the number of salient objects decreases and the number of distractors increases, VISOR's confidence about the scene decreases gradually. Its confidence decreases slower under condition L90/50 because the missing salient objects appeared only 50% of the time during learning. That is, these missing objects are less important in identifying the scene.

confidence of the analysis is more useful to the user than a binary yes/no answer. With a confidence measure, the user can set different error tolerance levels to determine whether to accept or reject the analysis. In applications that require great accuracy, the tolerance level can be set very low, whereas in applications that can tolerate larger errors, an analysis with low confidence can be accepted. Even when the analysis is rejected, the information about the possible alternatives may still be useful to the user. This way, the user always has more info available than just the label of the best match.

4.7 Experiment 6: Contextual Disambiguation

The sixth experiment was conducted in two parts. First, VISOR was trained with only images of individual objects and regions listed in Table 2 (instead of complete scenes). After training, a house scene containing a road, house, car, tree foliage, and grass were presented to VISOR. The road, house, and the car are unique and could be identified individually without the context. However, when VISOR focused attention at the tree foliage (step 3) or the grass (step 4), both the foliage and the grass schemas attained the same high activity level (Fig. 10). The first part of

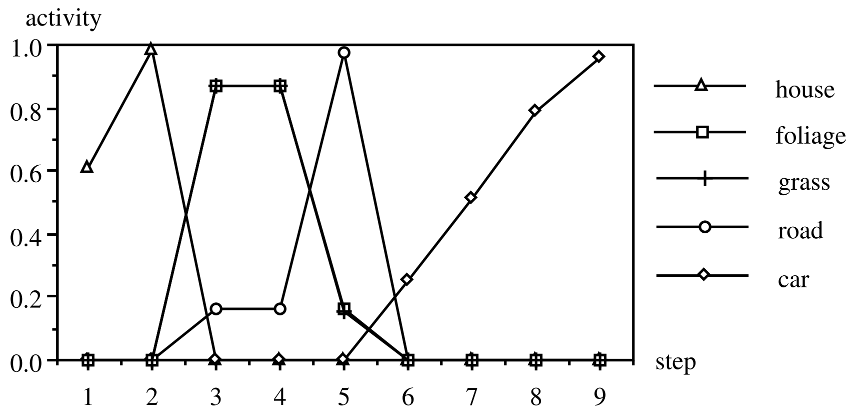


Figure 10: **Processing a road scene without contextual disambiguation.** The road, house, and car are unique objects and can be always identified even without the scene as context. However, when VISOR focuses attention at either the trees (step 3) or the grass (step 4), both the foliage and the grass schemas attain the same high activity level indicating that VISOR cannot discriminate between them.

the experiment thus shows that VISOR cannot discriminate between tree foliage and grass without using the spatial layout of the scene as context.

In the second part, the entire scenes were used as the training images (Table 2), and VISOR learned to encode the spatial layout of the objects and regions in the scenes. After training, a road scene consisting of a car, road, grass, and tree foliage was shown to VISOR (Figs. 2, 11). The schemas for the road scene and house scene were activated after recognizing the car (step 4). When VISOR focused attention at the ambiguous region at the bottom of the scene (step 5), the top-down inputs from these schemas increased the activity of the grass schema which, in turn, slightly decreased the foliage schema's activity through competition. Conversely, when VISOR focused attention at the area in the middle of the scene (step 7), the foliage schema attained a higher activity level than the grass schema. Therefore, VISOR was using the spatial information stored in the scene schemas to discriminate ambiguous region schemas.

5 Stability and Robustness

The experiments presented in the previous section demonstrated VISOR's properties as a schema processing system. To be useful in the real world, such a system also needs to have two other important properties: *stability* and *robustness*. Stability means that the activation of the system will eventually settle during recognition, and that the weights will settle during learning. Robustness means that a small change in the inputs or in the system parameters will result in only a small change in the system's behavior. These properties make VISOR more reliable, scalable, and easier to adapt to new tasks.

In this section, five experiments are presented that illustrate VISOR's stability and robustness:

7. Stability of activation;

8. Stability of learning;

9. Robustness against input variations;

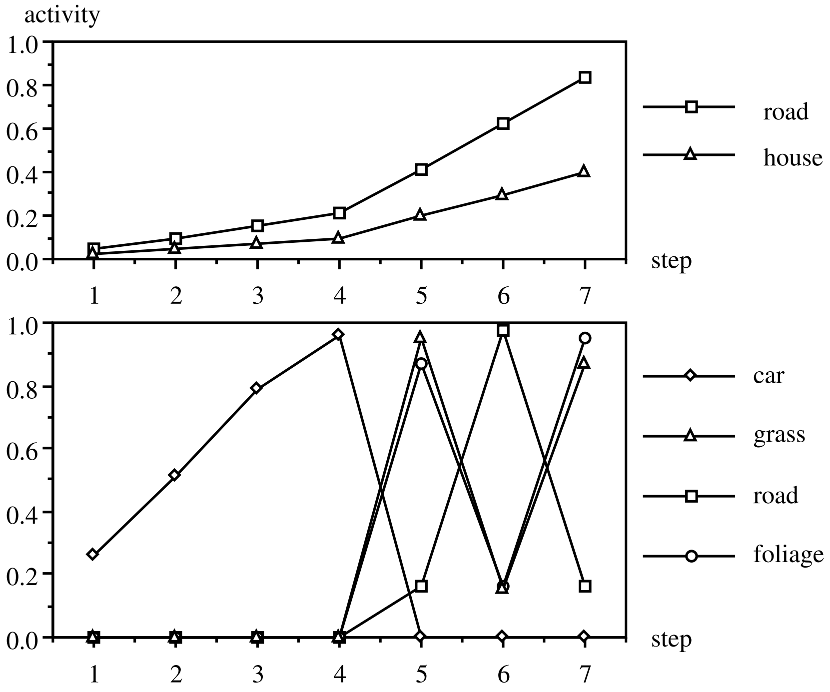


Figure 11: **Processing a road scene with contextual disambiguation.** The bottom graph plots the activities of the object and region schemas, and the top graph those of the scene schemas. At step 4, the car was recognized, and the road and house scene schemas were activated. When VISOR focused attention at the grass region (step 5), the top-down inputs from the scene schemas increased the grass schema's activity. Conversely, when VISOR focused attention at the tree foliage (step 7), the foliage schema attained a higher activity level than the grass schema.

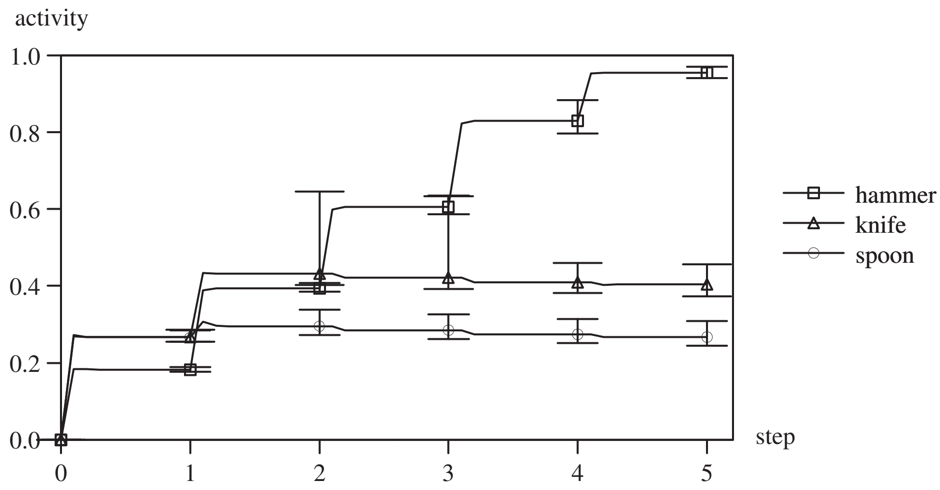
10. Robustness against parameter variations; and

11. Robustness against impulse noise.

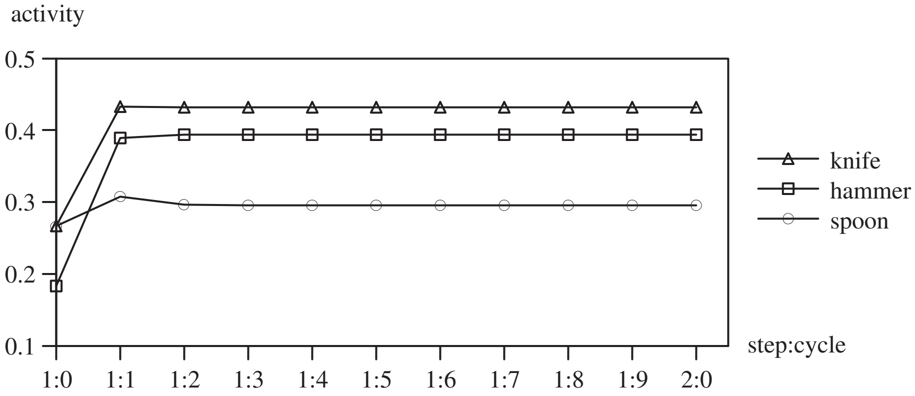
In Experiments 7 and 9–11, the same trained VISOR system was used as in Experiments 1–4 and 6. In Experiment 8, VISOR was trained with a small training set of 3 objects with the same system parameters as before.

5.1 Experiment 7: Stability of Short-Term Memory

Short-term memory (STM), refers to the activities of the units in a neural network. The STM is temporally stable if, given fixed inputs, fixed weights, and continuous updating of the units' activities, the STM will settle at some equilibrium state. A feedforward network is always stable because the units' activities never feed back, but in a recurrent network the units' activities can keep changing and never settle to a stable state. One way to ensure STM stability is to connect the network in such a way that it satisfies the *Cohen-Grossberg stability condition* (Cohen and Grossberg 1983). Since the recurrent connections in VISOR are highly structured, it is difficult to prove that the network satisfies this condition. However, due to strong mutual inhibition among the schema-nets' output units, VISOR has been found to be very stable in practice, settling to an equilibrium state within a small number of updating cycles.



(a)



(b)

Figure 12: **Stability of VISOR's activation.** Each recognition step consisted of 10 cycles during which the schema-nets' activities were updated. (a) In the beginning of each step, VISOR shifts attention to a different part of the hammer input. The activities change abruptly but quickly settle to new stable values. The error bars show the highest and lowest settled activations under different parameter settings (discussed in Experiment 10). (b) An expanded view of the second recognition step.

To demonstrate STM stability, a hammer was presented to VISOR, and changes in the schema-nets' activities were recorded in detail. At each recognition step, VISOR focused attention at a different part of the input. Each step consisted of 10 cycles during which the schema-nets' activities were updated. At the beginning of each step, the change in the input caused the schema-nets' activities to change abruptly. After a period of competition through mutual inhibition, their activities quickly settled to new stable levels. The settling was very strong as can be seen from Fig. 12: the activities stabilized in just a few cycles. This behavior is very robust and has occurred in all experiments with VISOR so far, even with significantly deflected parameter values (as will be discussed in Experiment 10).

5.2 Experiment 8: Stability of Long-Term Memory

Long-term memory (LTM), refers to the weights of the connections between neural units. Suppose that a network is learning to encode each category of input patterns by the weights of a different unit. As learning progresses and weights are modified, the category encoded by the unit may change. The network's LTM is said to be stable if each unit eventually settles to encode a fixed category. Not all learning algorithms are stable. For example, it has been shown that in competitive learning (Rumelhart and Zipser 1985), the categories may keep switching units indefinitely and the system may never settle to a fixed categorization (Grossberg 1976). Such instability is a result of the so-called *stability-plasticity dilemma* (Carpenter and Grossberg 1987): on one hand, the network should be plastic enough so that it can learn to encode new categories; on the other, what it has already learned should not be erased. One way to achieve LTM stability is to change only those weights that encode the category of the current input. This strategy is adopted in the ART network (Carpenter and Grossberg 1987) as well as in VISOR.

To show that VISOR's encoding of schemas is indeed stable, three objects, spoon, knife, and hammer, were presented repeatedly, in that order, to VISOR to learn. Fig. 13 shows the schema activations over the course of learning. In the beginning, no objects had been encoded, and when a spoon was shown to VISOR, an uncommitted schema-net was activated and it learned to encode the spoon. The weights were initially very small, and the newly-formed spoon schema did not have high activation. During the second presentation, a knife was shown to VISOR. Since a knife looks like a spoon, the spoon schema-net matched it better than the other still uncommitted schema-nets. VISOR produced the *spoon* label as the response, which was incorrect. The environment delivered a punishment signal, which suppressed the activation of the spoon schema-net. As a result (shown as 2b), an uncommitted schema-net became activated and learned to encode the knife.

In the third presentation, a hammer was given to VISOR. Since the hammer's handle resembles the handles of the spoon and the knife, both the spoon and the knife schema-nets were activated. The spoon schema-net had a slight advantage, and VISOR produced the *spoon* label as the response. This response was incorrect and a punishment signal was delivered, suppressing the spoon schema. The knife schema-net became most active (in 3b), which was again incorrect and was consequently suppressed. In 3c, an uncommitted schema-net finally became most active and was allowed to learn to encode the hammer. Similarly in subsequent presentations, only the correct schema-nets were allowed to encode the current inputs, and they eventually became very good encodings of the spoon, knife, and the hammer.

VISOR's learning strategy is similar to that of the ART network (Carpenter and Grossberg 1987). In ART, when an input activates a wrong encoding unit, the top-down pattern generated from the encoding unit will not match the bottom-up input pattern. The mismatch will generate a reset signal to suppress the currently most active encoding unit so that another unit can become the most active. In VISOR, the punishment signal generated by the environment serves the same role. This way, only the correct schema-net (or an uncommitted one) can modify its connection weights to encode the current input. Therefore, VISOR's learning is stable and learning new inputs does not erase what has already been learned.

5.3 Experiment 9: Robustness Against Input Variations

To test how well VISOR can tolerate deviations from the schemas, variations to the input objects listed in Table 1 were systematically created by modifying the shape of one of their components. For each shape attribute in turn, the pattern over the shape units was moved to the next unit, which in the current encoding meant changing the attribute values 50% to 80% (see appendix B).

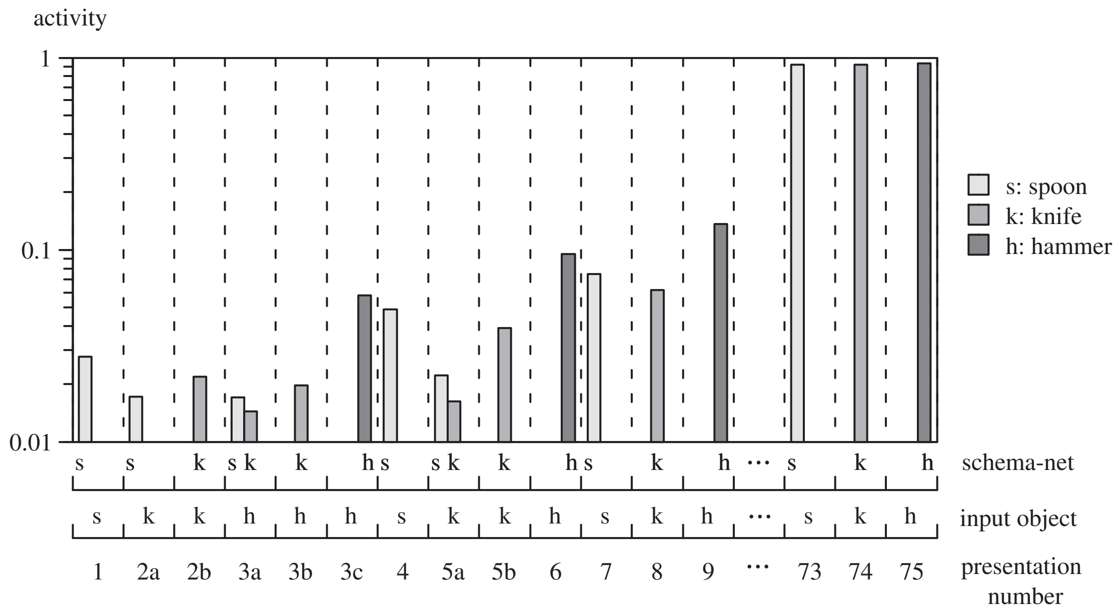


Figure 13: **Stability of VISOR's learning.** If an incorrect schema-net is activated (as in the 2nd, 3rd, and 5th presentations), VISOR will output an incorrect label and a punishment signal is delivered, suppressing the incorrect schema-net so that another net can become the most active. Only the correct schema-nets (or uncommitted ones) are allowed to encode the currect inputs, which makes learning stable even in prolonged training.

These variations were then presented to VISOR for recognition. VISOR correctly recognized all variations. Each column in Fig. 14 stands for the activity of the correct schema-net in response to the standard object. The error bars indicate the range of activities of this net in response to different variations of the same input. The standard object fits the schema the best, and therefore the variations all have lower activations. As can be seen from the figure, the schema-nets' activities do not vary much with variations in the input objects. The more components an object has, the smaller the variations are. Even when the object has only two parts, the matching schema-nets' activities do not usually differ more than 15%, which can be easily tolerated in the recognition process.

5.4 Experiment 10: Robustness Against Parameter Variations

To demonstrate that VISOR is robust against variations in parameter values, two properties must be shown:

1. Stability of schema activations is not affected; and
2. VISOR's ability to distinguish between different objects that look similar is not compromised.

There are three main independent parameters in VISOR (as described in Appendix A.1 and C):

1. The feedback coefficient β_B, which controls how strongly a schema-net's output unit feeds back to its SAM units;
2. The top-down coefficient β_D, which controls how strongly a high-level scene schema-net's SAM unit activates a low-level object schema-net's output unit; and

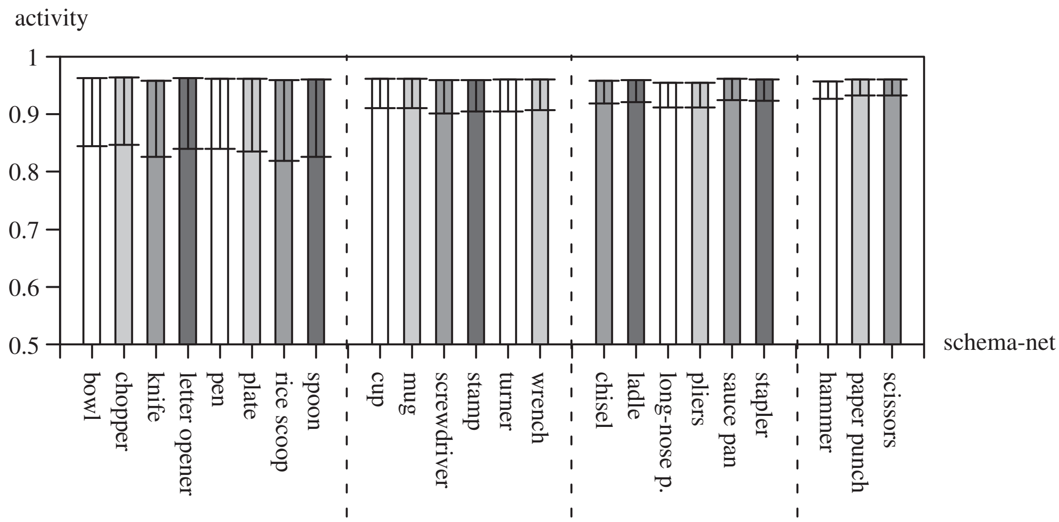


Figure 14: **Effect of input variations on the schema-nets' activations.** Each column indicates the activation of the matching schema-net in response to the standard version of each of the 23 objects, and the error bars indicate the range of activations in response to the variations. Activities are very robust against variations: Even when the input object has only two parts, the differences are usually less than 15%. The objects are grouped according to the number of parts, from 2 at left to 5 at right.

parameter	default	range of values				
feedback coefficient β_B	0.15	0.1	0.125	0.15	0.175	0.2
top-down coefficient β_D	0.15	0.1	0.125	0.15	0.175	0.2
inhibitory coefficient β_H	0.6	0.4	0.5	0.6	0.7	0.8

Table 5: **Variations of system parameters.** One of the parameters is varied over a range of $\pm 30\%$ while the other two are kept at the default values.

3. The inhibition coefficient β_H, which controls how strongly a schema-net inhibits other schema-nets at the same level in the schema hierarchy.

In this experiment, each one of these parameters was varied in turn over a range of $\pm 30\%$ while the others were kept constant at their default values (Table 5).

In the first test, a hammer was presented to VISOR and the schema activities were studied in detail. In each step of the recognition process, the activities settled in just a few steps just like with the default settings. Fig. 12 shows the variation in the final schema-net activities. The main plots show the activities with the default settings, and the error bars indicate the highest and lowest settled activations under the different parameter settings. In most cases, these values are very close to the default case. The error bars are higher at steps 2 and 3 because of one particular case: when the top-down feedback (i.e. $beta_D$) is very strong, the knife schema becomes a more likely candidate than the hammer. This is because knife has fewer components and therefore the feedback has a larger effect on its activation. Even in this case the activities are stable, and after looking at more components, VISOR easily arrived at the correct interpretation in the end. This experiment shows that, despite significant variations in parameter values, the stability of schema-nets' activations is not affected.

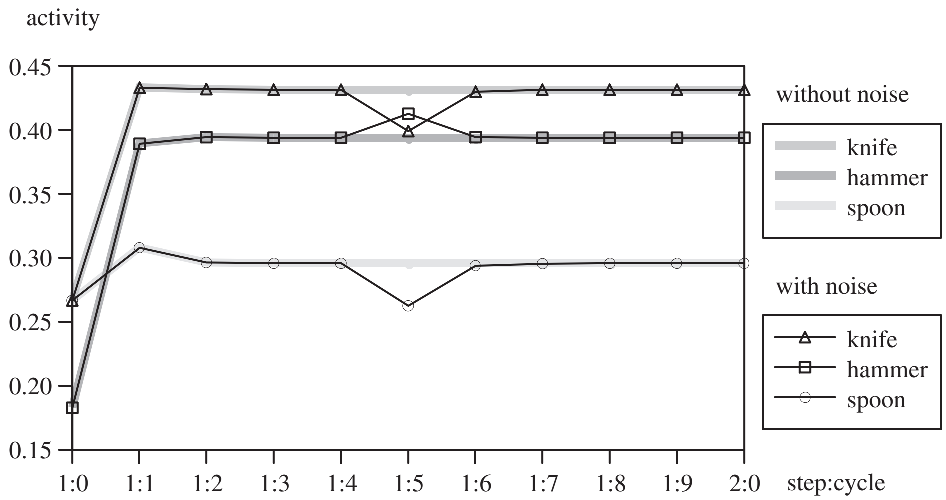


Figure 15: **Robustness against impulse noise.** Schema activations during the second recognition step are shown. The thick gray lines indicate how the schema-nets' activities settle to stable levels in the absence of noise (as in Fig. 12b). In this experiment, however, an impulse noise was introduced to the hammer schema-net's output unit during the fourth updating cycle, which changed the schema-nets' activities abruptly. After the impulse noise disappeared, the activities rapidly settled back to the same stable levels as before.

The effect of parameter variation on object discrimination is illustrated in Fig. 7, which shows what happens when different objects that look alike are presented to VISOR under the different parameter settings. The vertical columns show the final schema-net activities with default parameters, and the error bars indicate the highest and lowest activities under different parameter settings. As can be seen from the figure, the schema-nets' activities did not vary much and VISOR was able to distinguish between different objects that look alike.

5.5 Experiment 11: Robustness Against Impulse Noise

Again, a hammer was presented to VISOR, and the schema-nets were activated to match the input. Details of the activation during the second recognition step are shown in Fig. 15. The thick gray lines show how the schema-nets' activities settle to stable values in the absence of impulse noise (as in Fig. 12b). In this experiment, however, during the fourth cycle, an impulse noise was introduced to the output unit of the hammer schema-net. As a result, the schema-net's activity increased abruptly while the other schema-nets' activities decreased abruptly. After the impulse noise disappeared, the schema-nets' activities rapidly settled back to the same stable levels as before, showing that the system is robust against such impulses.

6 Cognitive Effects

In this section, VISOR's recognition and learning behavior is compared with well-known human perceptual phenomena. The purpose is to show that VISOR's processes are consistent with those of humans, suggesting that a schema system similar to VISOR's might be underlying human perceptual behavior. Three phenomena are considered (see Leow 1994; Leow and Miikkulainen 1994a for more details):

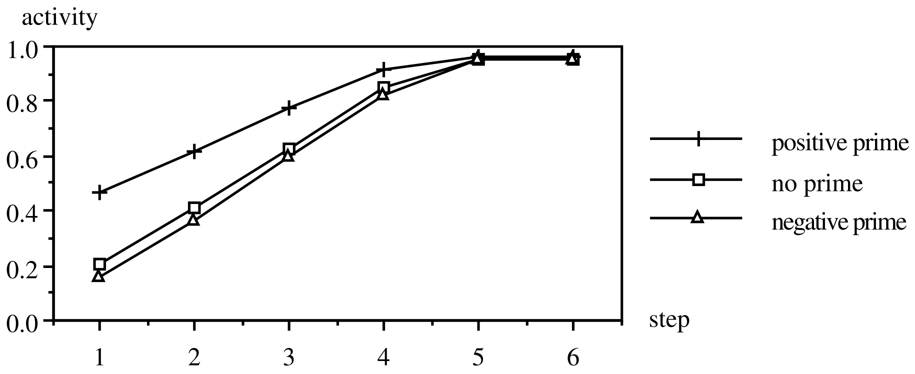


Figure 16: **Effects of priming on VISOR activation.** Positive priming reduced VISOR's response time, and negative priming increased it slightly.

12. Priming effects;

13. Perceptual reversal; and

14. Circular reaction in learning.

The same trained VISOR system was used in Experiment 12 as before (except with activity decay this time); Experiment 14 focused on one aspect of that training process. In Experiment 13, VISOR was trained on only 2 objects for simplicity, with the same system parameters as before, and with two extensions to the schema activation mechanism.

6.1 Experiment 12: Priming Effects

A person's object recognition performance can be facilitated if he is given advance information of the object, that is, by priming (Carr et al. 1982; Henderson et al. 1987). The subject recognizes an object (called the *target stimulus*) faster if he is shown a preceding object (called the *priming stimulus*) that is similar to or strongly correlates with the target stimulus. On the other hand, if the priming stimulus is very different from the target, the recognition takes slightly longer.

Priming effects can be tested in VISOR in the same experiment as with humans, with one refinement to the schema activation mechanism: without input, the activities of the schemas' output units and the SAM units are assumed to decay. In this experiment, a priming object (a hammer for positive priming, or a pair of pliers for negative priming) was first presented to VISOR. After it had recognized the object, it was removed from the input, and the schema activities started to decay. Before they reached zero, the target object (a hammer) was presented to VISOR. The residual activation in the network affected the time it took for VISOR to settle. As with humans, positive priming reduced VISOR's response time and negative priming increased it slightly.

6.2 Experiment 13: Perceptual Reversal

Fig. 17 is an example of an ambiguous object: It can be perceived either as a rabbit facing right or as a duck facing left. When a person views the figure continuously, her perception switches spontaneously from one to the other. With some practice, she can bias her perception towards either the duck or the rabbit. However, the reversal cannot be completely controlled or prevented from occurring. Two factors have been suggested to be causing perceptual reversal: (1) cognitive mechanisms such as attention and expectation, that allow the subject to bias her perception

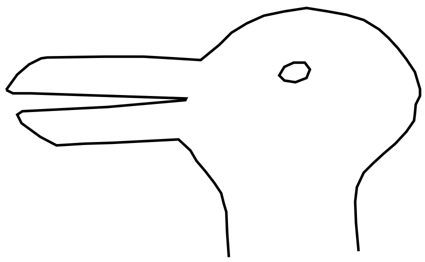


Figure 17: **Duck or rabbit?** In one half of the training presentations, VISOR was trained to recognize this figure as a rabbit, and in the other half, as a duck.

(Attneave 1971; Bugelski and Alampay 1961; Chastain and Burnham 1975; Lindauer 1989; Riani et al. 1984; Rock 1983; Rock and Mitchener 1992; Tsal and Kolbet 1985), and (2) neural satiation, which causes the representation of the currently dominant perception to gradually get weaker and eventually cause a switch (Attneave 1971; Babich and Standing 1981; Cornwell 1976; Long and Toppino 1981; Orbach et al. 1963; Spitz and Lipman 1962).

For the perceptual reversal experiment, two extensions to the basic VISOR architecture were made. Cognitive factors were modeled in VISOR by sending top-down input to the output unit of one of the schemas. To model the expectation that the figure is a duck, the top-down input is fed into the duck schema's output unit; when rabbit is to be expected, it is sent to the rabbit schema's output unit. Neural satiation and recovery from fatigue were modeled with probabilistic activation. A unit's activation was determined primarily by the input it receives, with a probabilistic component added to it that indicates how satiated the neuron is. If the neuron is activated higher than the *satiation threshold*, the probability of further high activation gets smaller. As soon as the neuron's activity falls below the threshold, it begins to recover and its probability for high activation gradually increases.

Ambiguous perception was set up by training VISOR to recognize Fig. 17 as a duck in one half of the learning presentations and as a rabbit in the other. After training, VISOR viewed the input figure continuously, focusing attention at different components. Fig. 18 shows the duck and the rabbit schema activations when a cognitive bias of 0.05 was fed into the rabbit schema unit. VISOR's perception spontaneously switched between duck and rabbit, but it was biased into perceiving a rabbit more often. However, when the top-down input was increased to 0.15, VISOR perceived the figure as a rabbit almost all the time. Even with extreme neural satiation, the top-down bias could sustain the rabbit percept for a long period of time, modeling the effect of strong cognitive bias in humans.

6.3 Experiment 14: Circular Reaction

Circular reaction is a concept developed by Piaget to describe an infant's learning behavior (Ginsburg and Opper 1969; Gruber and Vonèche 1977; Piaget 1952). When a behavior by chance produces interesting results, the infant will repeat it indefinitely. For instance, she moves her arm and by chance causes a toy attached to her cradle to rattle. The rattling interests her, and she desires to continue with it. Over a period of time, she learns the correct arm movement to rattle the toy whenever she likes. Circular reaction, therefore, is the idea that the repeated practice of actions discovered by chance induces learning of intentional actions. This is a very powerful learning principle and has been used, for example, to model the human visual saccade (Cohen et al. 1988; Grossberg 1978; Grossberg and Kuperstein 1989) and to build intelligent robots that learn

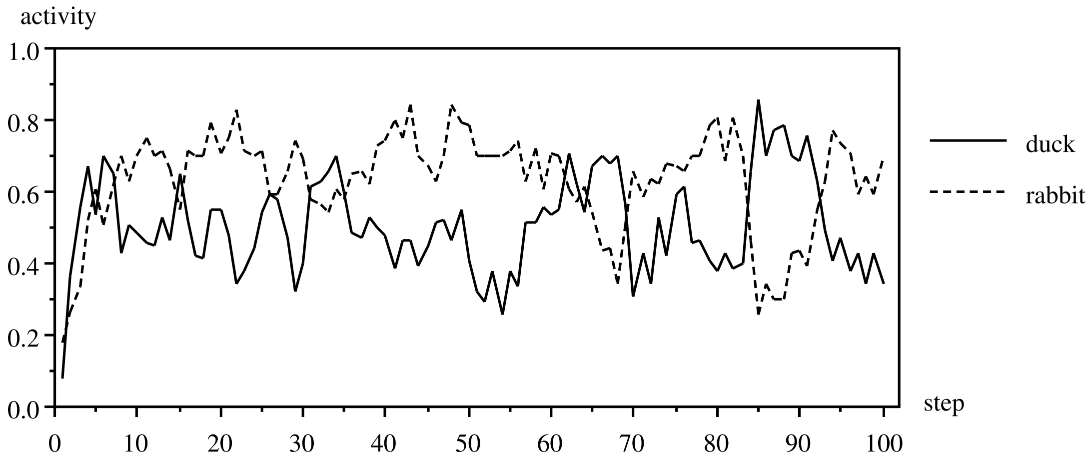


Figure 18: **Perceptual reversal mediated by top-down input and neural satiation.** A top-down activation of 0.05 was fed into the rabbit schema's output unit. As a result, VISOR perceived the ambiguous figure as a rabbit more often, but couldn't prevent the perception from occasionally switching to duck.

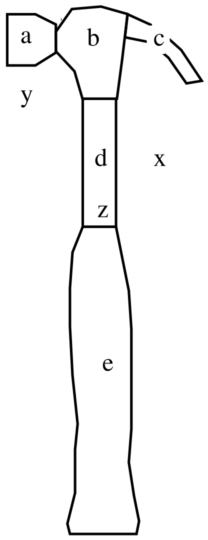
by exploring their environment (Fagg 1993; Lacaze and Meystel 1993).

Circular reaction is also central in VISOR's learning of new schemas. When VISOR first encounters a new object, it focuses attention only at positions where there are inputs in the scene. After VISOR has formed a schema for the object, it will shift attention to places where the object parts are expected. In other words, the shifting of attention evolves from a purely bottom-up, reactive process to a top-down, intentional behavior.

VISOR's attention shift is driven by two competing processes: (1) the Low-Level Visual Module (LLVM) always suggests a next position where there are inputs in the scene even if they do not match any schema component, and (2) based on the schema structure, the active schemas suggest positions where inputs are expected. VISOR makes a decision preferring small shifts suggested by highly-active schemas (see Leow 1994 for details). Consider, for example, how VISOR learned to encode the hammer as part of the training described in Section 4.1. From the first presentation of the hammer, an initially random schema started to encode the spatial structure of the hammer. Its shift suggestions were random because no information had been encoded at this point (Fig. 19b). The schema was also very weakly activated because it did not match the input well. Consequently, VISOR always adopted the LLVM's suggestions for the next position. At this stage, VISOR was only reacting to positions that happened to have input.

The weight changes made during learning are small, and it took several presentations for the schema to learn an accurate representation of the object's spatial structure. As the hammer schema gradually developed, it began suggesting more and more locations where inputs were expected. However, during the learning, when the schema was still not fully developed, it occasionally suggested inaccurate positions such as x (Fig. 19c).

After the schema network had learned the structure of the hammer, its recognition confidence was high, and, as a result, its output activity was large enough so that its suggestions were always adopted by VISOR. VISOR was no longer just reacting to the inputs in the scene; instead, it decided where to focus attention according to where inputs were expected—an act of intention. Even when a component was missing, VISOR shifted attention to the location where it was supposed to be. For example, when a hammer without the claw (component c) was presented to VISOR, the sequence



(a)

step	curr	schema	LLVM
1	c	x	b
2	b	y	a
3	a	y	d
4	d	z	e
5	e		

(b)

step	curr	schema	LLVM
1	d	a	b
2	b	c	a
3	a	c	c
4	c	x	e
5	e		

(c)

step	curr	schema	LLVM
1	d	b	b
2	b	c	a
3	c	a	a
4	a	e	e
5	e		

(d)

Figure 19: **Circular reaction in learning to recognize a hammer.** (a) The outline of the hammer object. Positions a–e denote the locations of the object parts, and positions x, y, and z are other possible positions in the input space. (b) At the beginning, VISOR focused attention at positions suggested by the LLVM, i.e., where there were inputs (curr: current position, schema: schema's suggestion, LLVM: LLVM's suggestion). (c) As the hammer schema gradually learned to represent the structure of a hammer, it began suggesting more and more locations where inputs were expected. However, since the schema was not yet fully developed, it still occasionally suggested random positions such as x. (d) After sufficient learning, a hammer without part c was presented to VISOR. The hammer schema always suggested positions where inputs were expected, including c, and its suggestions were always adopted by VISOR indicating it had learned intentional actions.

of positions shown in Fig. 19d resulted. At the second step, the LLVM suggested position a instead of c because there was no input at location c. However, VISOR still focused attention at c, as suggested by the hammer schema, because it was a location where a part was expected. These experiments demonstrate that the circular reaction principle can be used to describe learning of visual schemas, and perhaps other perceptual processes as well.

7 Discussion and Future Work

In previous sections, the properties of VISOR as a schema representation and learning system were demonstrated in detail. The experience with VISOR also helps to bring about insights and open new possibilities for future research in the general areas of neural network learning, schema theory, and object recognition.

7.1 Learning of Structure

Multi-layer feedforward networks trained with backpropagation (BP, Hertz et al. 1991; Rumelhart et al. 1986a) have been successful in many applications, and form a standard against which VISOR's methods of representation, recognition, and learning should be compared.

Backpropagation is a method for learning pattern transformations, and it does not apply well to domains with structured, variable input, such as scene recognition. The BP network would require an extremely large number of units (to cover the entire input scene), and it would have to be trained with all possible combinations of objects at all possible locations, an approach which is intractable in practice. In other words, a straightforward BP approach could not be used to implement an entire scene analysis system.

It might, however, be possible to train a BP network to recognize single objects, and it is interesting to compare its performance to VISOR in this limited task. To make the BP training time reasonable, the object maps were reduced to 3×7 spatial locations by removing three large objects (sauce pan, stapler, and ladle) from the list of objects in table 1. For the remaining objects, consisting of 7 different types of components, the BP network needed $3 \times 7 \times 7 = 147$ input units, 20 output units (one for each object), and 85 hidden units. The target output was 1 for the output unit that represented the input object, and 0 for the other units.

Backpropagation training turned out to be very sensitive to the learning rate, momentum, and initial connection weights (cf. Kolen and Pollack 1990). Many different combinations of parameters were tried before finding one set that produced satisfactory training results (with learning rate $= 0.5$, momentum $= 0.1$, and total sum-squared error < 0.1). It took slightly more than 100 presentations of each object to train the network.

In comparison, VISOR's learning is very insensitive to parameters. In general, a small learning rate, such as 0.1, is preferable so that the schema can learn to encode a smooth average of different training examples of the same object. If the training set has only one example for each object, as in this experiment, VISOR could learn to encode the object in a single presentation if large enough learning rate was used. In other words, VISOR's learning is much more efficient than BP in this task.

Another important characteristic of VISOR's learning scheme is that the training examples can be presented incrementally. The learning of new schemas does not affect existing schemas because only the weights of the new ones are modified. This property would be very useful in practice. For example, less common objects can be introduced into an application after VISOR has already learned about the more typical ones. VISOR can learn to encode the new objects without having to relearn the existing ones, which is very difficult for the BP algorithm (Grossberg 1987a; McCloskey and Cohen 1989; Ratcliff 1990). At every presentation, BP modifies all the connection weights, and learning of new objects quickly erases the encoding of the existing ones. Therefore, when the BP network learns about new objects, it also has to relearn the entire set of previous objects at the same time.

After learning, both networks were able to recognize variations of the object. However, there is an interesting difference in the recognition strategy. The BP network quickly learned to pay attention to those components that distinguish each object from the rest. For example, only the hammer has three components located at the top part of the map. If these components were part of the input, BP's confidence was high even if the other parts of the hammer (like the handle) were missing. In contrast, VISOR identifies the input by matching it with the entire schematic structure. The hammer schema is highly activated only if the shapes and the positions of the input parts match the corresponding schema components. Even if a unique part such as the claw is missing, VISOR can still recognize the hammer based on how well it fits the rest of the schema.

It is interesting to note that humans have yet another strategy for object recognition at their disposal, namely one that depends on the functionality of the object. For example, if an object that looks like a hammer has a sharp head instead of a flat one, it cannot be used like a hammer and would not be categorized as one. In other words, there are important components that must be present in an object for it to be a valid variation. Such information comes from general world knowledge, and would have to be supplied to VISOR by an external system.

In summary, VISOR's learning methods are more structured than BP, resulting in representations that encode the structure of the input as well as the recognition result. Because learning takes place at separate locations, it can be several orders of magnitudes faster. The scale-up properties are also good because new structures can be learned incrementally. These properties are exactly what is needed for learning schemas from examples, and constitute a promising general approach to learning structure with neural networks.

7.2 Schemas and Schema Instances

Schema theory makes an important distinction between *schemas* and *schema instances* (Arbib 1987, 1990; Rumelhart 1980). A schema is a package that contains knowledge about a class of objects or actions. A schema instance, on the other hand, contains information such as the size and orientation of a specific input object, or a specific action such as "pick up the white cup." Traditionally, schema instances are viewed as active agents that interact with each other to interpret input and perform motor actions, whereas schemas are just passive templates of knowledge structures.

The idea of passive schemas and active instances seems to have arisen from symbolic implementations of schemas in traditional AI systems. The symbolic approach makes a clear distinction between data and procedures. Data describe the objects that the system recognizes and the actions that can be performed on them. Procedures are programs executed by the system to perform the actions. Using the above example, the object recognition system dynamically creates a schema instance (i.e., a dynamic data structure) to represent each cup. After identifying the cup instance that the instruction "pick up the white cup" refers to, the system creates an instance of the pick-up schema for the cup, and forwards the location of the white cup, encoded in the cup schema instance, to it. The system then executes the procedure encoded in the action schema instance to carry out the action. When the instances are no longer needed, they are destroyed from the short-term (or working) memory of the system.

In neural networks, however, both the data and the mechanisms that perform recognition and actions are implemented by the same neural units and connections. It is impractical to create copies of the same neural hardware to represent indefinitely many instances at the same time. A more practical solution is to implement just one copy of the neural hardware (the general schema), and use it to recognize different input instances one at a time.

This approach leads to a reversal in the roles of schemas and instances (see also Arbib 1993; Goodale and Arbib in press): schemas are the active interacting agents whereas instances are passive copies. For example in VISOR, there is only one schema representation net that encodes the knife schema. When attention is focused on a knife in the scene, the knife schema is activated and its SAM units contain the confidence of finding the knife's components in the scene. When attention is shifted to another knife in the scene, the same schema now contains detailed information about the components of the new knife. Detailed information about the previous knife is lost; only a "pointer" indicating that the knife was found at this particular location is maintained in the SAM of the higher-level schemas.

Even without the detailed instance copies, VISOR can determine what the scene depicts because the activations of the scene schemas depend only on the confidence of finding the objects in the scene. In this particular schema-based task, therefore, only one active schema representation is needed for each class of object and scene. An interesting question can thus be raised: Is one active schema representation always enough? While further research is necessary to answer this question definitively, the experience with VISOR indicates that this may indeed be the case.

7.3 Building Schema-Based Vision Systems

Although the main emphasis in VISOR research is on representing and learning schemas, it also forms a framework in which some of the central issues in object recognition can be addressed. Let us see how these issues can be approached from the VISOR perspective.

One of the hardest problems in the area of scene analysis and object recognition is handling variations. An object may look different when it is observed from different viewpoints, and parts of the objects may be occluded. Even given a fixed, clear viewpoint, the retinal image can vary in size and orientation. A robust vision system should be able to recognize the object regardless of such variations. The traditional approach to recognizing an object located at any position and orientation is to transform the *entire* image of the object into a canonical representation that can be matched with typical objects stored in the memory. The most straightforward method is to resize, translate, and rotate the input image into an image with standardized size, position, and orientation (Anderson and van Essen 1987; Olshausen et al. 1993). A more sophisticated method is to transform the input image into a non-spatial representation. For example, the log-polar-Fourier transform of an image is represented in the frequency domain and is invariant to scaling, orientation, and translation (Cavanagh 1978, 1985; Pollen et al. 1971; Seibert and Waxman 1989). Since these techniques are typically applied to the entire image of an object, they are not well-suited to handling variations that result from the changes in the shapes, sizes, positions, and orientations of individual components.

One way to deal with many such variations is to identify each component one at a time, as in VISOR. The positions of the components in VISOR are represented in RPMs, relative to the objects center of gravity, and their recognition is therefore invariant with respect to the object's location. Their orientations are measure relative to the object's main axis and therefore they can be recognized regardless of the object's orientation. Small variations in the components' expected locations are acceptable as long as they fall in the area represented by the correct SAM unit. As was seen in Experiment 9, small variations in the shapes of the components can also be tolerated, and the confidence in recognition is automatically indicated in the process.

VISOR could be extended to handle other variations as well, such as size. Instead of representing the shape of an input component in terms of its actual size, VISOR can encode the shape in terms of its size relative to the size of the object. This way, the shape attributes would be invariant with respect to the object's size. A possible drawback of this method is that the relative sizes may change drastically when a large component is missing from the input object. For example, an input hammer with a missing handle is much shorter than a standard hammer. As a result, the relative sizes of the components will become much larger than those of the standard hammer. However, an object that is missing a large component should perhaps not be considered an acceptable variant anyway, because large components, such as the hammer's handle, typically constitute major parts of an object.

VISOR could also be extended to recognize articulated objects. The spatial structure of such objects could be encoded based on the joints connecting the components instead of the positions of the parts. After identifying an object part and the joint that connects it to another component, VISOR can shift attention to the second component and determine whether it matches the one encoded in the schema. It can also determine whether the orientation of the component falls within the range of permitted angles. The main difficulty in this approach is that an object part can be connected to several other parts, and a joint can connect together many components (as in a Swiss-army knife). VISOR may have to represent many-to-many mappings between components and the joints.

To help focus the research effort, the mechanisms of segmentation and visual attention were left largely unspecified in the Low-Level Visual Module. Although not essential to the goals of VISOR, they are important issues in machine vision on their own right. Several promising approaches have been developed by other researchers, such as Grossberg's (1987b), Mozer et al.'s (1992), and Finkel and Sajda's (1992) segmentation models, and Mozer's (1988) and Olshausen et al. (1992) attentional mechanisms. Especially Grossberg's model seems compatible with VISOR and could possibly be adapted to function in the LLVM (Leow 1994). Such a combined model would make it possible to study how feedback from the schema hierarchy affects segmentation and attention. Also, VISOR does not currently include mechanisms for dealing with occluded objects. However, VISOR can recognize objects with missing parts, which is a good starting point. Additional mechanisms are necessary for recognizing occlusion, and modifying VISOR's propagation processes so that foreign components are not interpreted as part of the object.

At present, VISOR does not encode objects as three-dimensional (3-D) structures. There are two possible alternatives how this could be done: with an object-centered or a viewer-centered frame. With the object-centered approach, the Subschema Activity Maps in the current VISOR implementation would be extended to encode depth, and thus represent the 3-D spatial structures of the objects such as enclosure and overlay. VISOR would also need an additional mechanism for determining the viewpoint from the input image, and for reconstructing the 3-D representation from it. How exactly to do that remains an open question at this point.

With the viewer-centered approach, VISOR could represent different views of an object in separate schemas. The Subschema Activity Maps still need to be extended to represent 3-D spatial relationships such as enclosure and overlay. However, this approach does not require a mechanism for determining the correct viewpoint. Given an input image, VISOR will just activate the schemas that represent different views to determine which one best matches the input. The viewer-centered approach can thus be more readily implemented in VISOR, and to the extent that VISOR's representation method is consistent with that of the human visual system, suggests that the viewer-centered approach might be in use in the human visual system as well.

In summary, it seems that shifting of attention and recognizing the components one at a time could lead to translation, rotation, and scale invariant recognition of an object. Such a strategy might also make it possible to tolerate slight variations in the components' shapes, sizes, positions, and orientations, and possibly occlusion. In future research with VISOR, such extensions towards a robust vision system based on schema hierarchy will be studied, and the hypothesis that recognition of 3-D objects is based on a viewer-centered frame will be tested.

8 Conclusion

The main contribution of VISOR is to demonstrate how knowledge about structure such as visual schemas can be processed in neural networks using simple architectures such as maps and connections among them. Cooperative, competitive, and parallel bottom-up and top-down processes emerge naturally from neural networks, and they give rise to robust, stable, and cognitively plausible recognition behavior. From the point of view of schema theory, VISOR demonstrates how schemas can be learned by a combination of unsupervised adaptation and reinforcement. It also leads to a new, neurally inspired view of the schema system, where the schemas are the active agents and the instances are passive copies. VISOR's methods of representing and learning schemas should help in building robust vision systems, and they may be applicable in other domains such as motor control and natural language processing as well. Such systems may form a long-sought bridge between neural processes and high-level cognitive behavior.

Acknowledgements

This research was supported in part by NSF grant #IRI-9309273 and in part by Texas Higher Education Coordinating Board Grant #ARP-444.

Appendices

A Details of the Recognition and Learning Processes

Let us use the symbols A_i to denote the output unit of a general schema-net i, and symbols $a_{ix\mu}$ the units of the 3-D SAM, where x denotes the 2-D spatial position and μ the location within a SAM column (for object schemas, the subscript μ is redundant since their SAMs are two-dimensional spatial maps). In addition to the SAM and the output unit, a schema-net has three other important components:

1. The Current Position Map (CPM) c_{ix}, which represents the current position of attention by activating a single unit on the map.
2. The Next Position Map (NPM) n_{ix}, which encodes the next position of attention according to where a subschema is expected.
3. The Current Subschema Activity Map (CSAM) $b_{ix\mu}$, which temporarily holds the SAM activations.

The CSAM is needed for the following reason: When VISOR focuses attention at an object in the scene, the position of the object is encoded by a single active unit in the CPM. This unit gates the column of units in the corresponding location of the SAM, and allows them to update their values and learn to expect the object in that location. Learning consists of strengthening the connections between the currently active object schema and the corresponding unit in the SAM column. However, several units in the column can be active at the same time, because several objects may already have been recognized at that part of the scene. In such a case, it is impossible to tell which one of the SAM units should modify its weights.

The solution is to split SAM into two parts: the SAM proper records which subschemas have been recognized so far, and the CSAM (isomorphic to SAM) indicates which unit in the column is currently active. The bottom-up connections from the subschemas go to CSAM, and they can be learned unambiguously. CSAM is connected to SAM with one-to-one connections, and over time, the activations of CSAM units are recorded in the SAM.

A.1 Recognizing Objects and Scenes

Let us now see how the schema-nets cooperate and compete to match the input scene. Given an input scene, the LLVM focuses attention at one component of the scene, say the elliptical part of the spoon, and generates a feature representation of its shape in the shape feature cells L_j. The single active unit r_x in the Relative Position Map (RPM) encodes the position of the elliptical part relative to the entire spoon. The RPM unit r_x activates all object schemas' CPM units c_{ix}, indicating the current position of attention. The active CPM units, in turn, enable all object schemas' CSAM units $b_{ix\mu}$ at position x so that only those CSAM units, and in turn, the SAM units, that match the current position of attention x (in this case the left middle units) can update their activities. The SAM units' activities indicate how strongly the schemas' components are believed to be present in the scene. In this example, the left middle SAM unit of the spoon schema is most strongly activated.

Let us next see how the CSAM and SAM units are updated. For an object schema, the total bottom-up input $U_{ix\mu}$ to CSAM unit $b_{ix\mu}$ is a weighted sum of the feature cells' activities L_j:

$$U_{ix\mu} = \sum_j W_{ix\mu j} L_j \tag{1}$$

where $W_{ix\mu j}$ denotes the weight of the connection from feature cell L_j to CSAM unit $b_{ix\mu}$.

For higher-level (scene) schemas, weights of the bottom-up connections are divided into two components, $W_{ix\mu j}$ and $X_{ix\mu j}$ (Section A.2). The factor $X_{ix\mu j}$ is the contribution of the source unit to the total connection strength (corresponding, for example, to the amount of neural transmitters in the presynaptic neuron), and the weight $W_{ix\mu j}$ is the contribution of the unit receiving the connection (corresponding, for instance, to the number of receiving sites in the postsynaptic neuron). The effective weight of the connection is then given by the product $W_{ix\mu j}X_{ix\mu j}$, which is large only for the connection from an object schema to its unique CSAM unit in the scene schema.

For higher-level schemas, the bottom-up input $U_{ix\mu}$ is a weighted sum of the lower-level schemas' output activities A_j:

$$U_{ix\mu} = \sum_j W_{ix\mu j} X_{ix\mu j} f_U(A_j) \tag{2}$$

The activation function $f_U(A_j)$ is a function that picks the winner among the lower-level schemas.

In the schemas, the current position unit c_{ix} enables the CSAM column at position x, and a unique CSAM unit $b_{ix\mu}$ in the column is activated by the winning object schema A_j through the bottom-up input $U_{ix\mu}$:

$$b_{ix\mu} = \begin{cases} c_{ix}U_{ix\mu} & \text{if } c_{ix}U_{ix\mu} = \max\limits_{y,\nu} c_{iy}U_{iy\nu} \ , \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

The unique CSAM unit then propagates its activation to the SAM unit at the same map location:

$$a_{ix\mu} = \max\left(b_{ix\mu} + \beta_B m_{ix\mu} A_i \ , \ 0\right) \tag{4}$$

while other SAM units' activities remain unchanged. The symbol $m_{ix\mu}$ denotes the weight of the feedback connection from the schema's output unit A_i to its SAM unit $a_{ix\mu}$. The positive weighting parameter β_B should be much smaller than 1, such as 0.1, since bottom-up inputs are more important in activating the SAM units than top-down expectation.

The schemas' output units A_i sum up their respective SAM activities $a_{ix\mu}$, top-down inputs from higher level schemas, and mutual inhibition from other schemas at the same level:

$$A_i = \sum_{x,\mu} w_{ix\mu} a_{ix\mu} + \beta_D D_i - \beta_H H_i \tag{5}$$

where $w_{ix\mu}$ denotes the weight of the connection from SAM unit $a_{ix\mu}$ to output unit A_i, D_i and H_i the total top-down inputs and total inhibition, and β_D and β_H the positive weighting coefficients.

The total top-down input D_i represents the expectations from higher-level schemas:

$$D_i = \min\left(1, \ \sum_{j,y,\nu} c_{jy} M_{jy\nu i} a_{jy\nu}\right) \tag{6}$$

where $M_{jy\nu i}$ denotes the weight of the connection from SAM unit $a_{jy\nu}$ of superschema j to schema i's output unit A_i. For example, a knife is expected in a dining table. When the dining table schema is active, its activity is propagated down, slightly activating its subschemas. Again, such top-down inputs should be small compared to bottom-up inputs so as not to overwhelm the bottom-up ones (i.e., $\beta_D \ll 1$, such as 0.1).

The total inhibition H_i received by schema i is the weighted-sum of the activities of the other schemas at the same schema level:

$$H_i = \frac{\sum_k n_k A_k}{\sum_k n_k} \tag{7}$$

where A_k is output of the schema-net k at the level of schema-net i, and n_k is the number of components encoded by schema-net k. The more components a schema-net encodes, the stronger is its inhibition of other schema-nets. Similar inhibitory interaction between neural units has also been adopted in (Cohen and Grossberg 1987; Grossberg 1978).

The object schemas' output activities are propagated to the scene schemas' SAM units, and at the same time, the scene schemas also propagate their activities down to the object schemas as top-down expectations. The schemas' activations are updated repeatedly until they stabilize. With appropriately large inhibitory coefficient β_H such as 0.6, the activities typically stabilize within 3–5 update cycles.

After processing the inputs at the current position, the schemas each suggest a next position of attention where another component is expected. The suggested next positions are encoded by the single active units in their respective Next Position Maps. Based on the schemas' and the LLVM's suggestions, the Schema Module selects one position to be adopted by the whole system. The selected position is sent to the schemas and the LLVM, which then shift attention to the new location, that is,

$$c_{ix} = 0 \quad \text{and} \quad c_{iy} = 1 \tag{8}$$

where x is the current position and y the new location. The above recognition process then repeats until VISOR has attended to all the input components in the scene. At this time, the most active object schema represents the last attended object, and the most active scene schema identifies the scene, in this example, a dining table.

A.2 Learning to Encode Schemas

As discussed in section 3.3, only the most active schema-net can modify its connection weights to encode the spatial structure of the input. The weights are adapted through variations of the Hebbian principle (Hebb 1949; Anderson 1985; Sejnowski and Tesauro 1989).

1. Bottom-up Weights W_{ixj} (object schemas)

The object schema-nets receive bottom-up connections $W_{ix\mu j}$ from the LLVM feature cells L_j. Initially, all weights are limited between 0 and 1. During learning, they are modified to encode the shape (or texture) of the object's component, as represented in the feature cells L_j:

$$\frac{\mathrm{d}}{\mathrm{d}t} W_{ix\mu j} = c_{ix}\left[-W_{ix\mu j} + L_j\right] \ . \tag{9}$$

Equation 9 modifies W_{ixj} towards L_j, thereby encoding the feature pattern in the weights W_{ixj}.

2. Bottom-up Weights $W_{ix\mu j} X_{ix\mu j}$ (higher-level schemas)

Whereas the object schemas' bottom-up connection weights encode the shape of the objects' components, those of the scene schemas map the object schemas to unique CSAM units. The scene schemas' bottom-up weights therefore require a different learning process. The weight modification is most easily expressed in terms of two weight values on a single connection: $W_{ix\mu j}$ at the unit $b_{ix\mu}$ receiving the connections, and $X_{ix\mu j}$ at the source unit A_j. Both weights are adapted through a process that normalizes the total weights (Leow 1994):

$$\frac{\mathrm{d}}{\mathrm{d}t} W_{ix\mu j} \quad = \quad c_{ix}\left[-W_{ix\mu j}\sum_i \Psi_W W_{ix\mu i} b_{ix\mu} A_i + \Psi_W W_{ix\mu j} b_{ix\mu} A_j\right] \ , \tag{10}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}X_{ix\mu j} \quad = \quad c_{ix}\left[-X_{ix\mu j}\sum_{\nu}\Psi_X X_{ix\nu j}b_{ix\nu}A_j + \Psi_X X_{ix\mu j}b_{ix\mu}A_j\right] \tag{11}$$

where Ψ_W and Ψ_X should be larger than 1, such as 10. Initially, the weights are assumed to be bounded between 0 and 1. The learning process rapidly strengthens the initial connection between the most active object schema and the most active CSAM unit, while weakening the other connections. As a result, a unique connection is formed mapping the object schema to the unique CSAM unit.

3. Feedforward Weights $w_{ix\mu}$

The schema's output unit A_i receives connections $w_{ix\mu}$ from its SAM units $a_{ix\mu}$. The weights $w_{ix\mu}$ encode how important the component at location (x, μ) is in activating the schema. Initially, the weights have small random positive values. During the learning process, they are modified by a process that normalizes the total weights:

$$\frac{\mathrm{d}}{\mathrm{d}t}w_{ix\mu} = -w_{ix\mu}\sum_{y,\nu}A_i a_{iy\nu} + A_i a_{ix\mu}\ . \tag{12}$$

Equation 12 normalizes the total weight $\sum_{x,\mu} w_{ix\mu}$ to 1 (Leow 1994). If all the n components of the schema appear with the same probabilities, then the weights $w_{ix\mu}$ representing these components will have approximately the value $1/n$, while other weights have 0 values. On the other hand, if a component at, say, location (x, μ), appears more often than one at (y, μ), then the weight $w_{ix\mu}$ will attain a larger value than $w_{iy\mu}$ during learning, indicating that the component at (x, μ) is more important in activating the schema.

4. Feedback Weights $m_{ix\mu}$

The schema's output unit A_i also sends feedback connections to its SAM units $a_{ix\mu}$. The weights $m_{ix\mu}$ of these connections are modified by a process that polarizes the weights towards 0 or 1 (Leow 1994):

$$\frac{\mathrm{d}}{\mathrm{d}t}m_{ix\mu} = -\Lambda_m m_{ix\mu} + (1 - m_{ix\mu})A_i a_{ix\mu} \tag{13}$$

where Λ_m is the decay rate, typically much smaller than 1, such as 0.1. Initially, the weights have small positive random values. During the learning process, the weights $m_{ix\mu}$ at positions where there are input components ($a_{ix\mu} \gg 0$) will be increased towards 1, indicating that schema components are expected at those locations. On the other hand, the weights $m_{ix\mu}$ at positions where there are no inputs ($a_{ix\mu} = 0$) will decay towards 0, indicating that no schema components are expected.

5. Top-down Weights $M_{ix\mu j}$

The top-down expectation from schema i is first propagated from its output unit A_i to its SAM units $a_{ix\mu}$. It is then propagated through the top-down connections $M_{ix\mu j}$ to specific subschemas j (Fig. 4). The weights $M_{ix\mu j}$ are modified through a process similar to that for the bottom-up weights so that the scene schemas are connected to only the objects schemas that they contain:

$$\frac{\mathrm{d}}{\mathrm{d}t}M_{ix\mu j} = -M_{ix\mu j}\sum_{k}\Psi_M M_{ix\mu k}a_{ix\mu}A_k + \Psi_M M_{ix\mu j}a_{ix\mu}A_j \tag{14}$$

where Ψ_M is a large positive parameter such as 10.

B Input Representation

In the current implementation of VISOR, the Low-Level Visual Module is simulated using procedural programs, so that the research can concentrate on the main issues of representing, applying, and learning visual schemas. Currently, the input to VISOR consists of symbolic representations of scenes such as those depicted in Figs 1 and 2. Each component is described by seven shape features: length, breadth, closure, vertical tilt, horizontal tilt, expansion, and curvature. In addition, ten textural features, corresponding to Gabor filters with different frequencies and orientations, are used in describing the textured regions in outdoor scenes.

As an example, a simple workbench scene that contains a hammer and a knife would be given to the LLVM as follows:

```
scene: workbench
objects:
  x   y  ori   l   b  name
 17   5  100  32  12  hammer
  4   9   45  26   2  knife
components:
obj   x   y  ori   l  b    c    e     k     t
 0   16   2   10   3  3    1    0.25  0     -1
 0   17   2  100   5  3    1    0.1   0     -1
 0   18   2   55   6  1.6  0.7  0     0.15  -1
 0   17   4  100   7  2    1    0     0     -1
 0   17   7  100  18  3.5  1    0.1   0     -1
 1    5   8   45  14  1.6  1    0.06  0     -1
 1    3  10   45  12  2    1    0     0     -1
```

The names of the scene and the objects represent the target labels given to the system during learning. The hammer (object number 0) has 5 components listed in the rows with `obj` = 0, and the knife (object number 1) has 2 components. The columns labeled `x` and `y` indicate the actual locations of the objects and their components in the scene. The `ori` columns indicate their orientations measured in degrees from the x-axis. Given the orientation values, the LLVM computes the vertical and horizontal tilt of each component.

The columns marked `l` and `b` indicate length and breadth of the objects and their components measured in an arbitrary unit length. Those labeled `c`, `e`, and `k` indicate the closure, expansion, and curvature measured in radian. The texture column `t` has the value -1 in all the rows indicating that texture is not used as a feature to describe the shape of the components. In the representation of an outdoor scene, the `t` attributes of the grass, tree, and road regions would have the values that identify their texture.

Based on this symbolic representation of the input image, the program that simulates the LLVM activates the feature cells to encode the shape and texture of the currently attended component. The value of each attribute is represented by a bell-shaped activity pattern over four feature cells. The four length and breadth cells stand for ranges 1–1.5, 1.5–4, 4–8, and over 8; the closure, tilt, expansion, and curvature cells represent ranges 0–0.05, 0.05–0.2, 0.2–0.5, and over 0.5. The LLVM program also computes the component's relative position based on its actual location in the scene and the overall size of the object, and activates the RPM unit at that relative position. Such featural and positional information are sent to the Schema Module and are used by the schemas to determine how well they match the inputs.

name	notation	value
feedback coefficient	β_B	0.15
top-down input coefficient	β_D	0.15
inhibitory coefficient	β_H	0.6
decay rate	Λ_m	0.01
growth rates	Ψ_M, Ψ_W, Ψ_X	10

Table 6: **Values of the system parameters.** Values of the system parameters used in the experiments. Other values may be used as long as they satisfy the following conditions: The feedback coefficient β_B and top-down input coefficient β_D should be small because bottom-up input (with default coefficient of 1) is more important in activating a schema. The inhibitory coefficient β_H should be large enough so that the schemas' activities can stabilize rapidly. The growth rates Ψ_M, Ψ_W, and Ψ_X should be large enough to provide strong competition among the weights.

C Simulation Parameters

The values of the system parameters used in the experiments are summarized in Table C. The parameters consist of balance coefficients β and decay and growth rates Λ and Ψ. The balance coefficients are needed because a schema's output unit receives bottom-up inputs from its SAM units, top-down inputs from higher-level schemas, and inhibition from other schemas, and the contributions of these inputs on the schema activation need to be properly weighted. The decay and growth rates are denoted by Λ and Ψ, and subscripted by the symbols for the weights. They are necessary for proper tuning of the learning process.

References

Anderson, C. H., and van Essen, D. C. (1987). Shifter circuits: A computational strategy for dynamic aspects of visual processing. *Proceedings of National Academy of Sciences USA*, 84:6297–6301.

Anderson, J. A. (1985). What Hebb synapses build? In Levy, W. B., Anderson, J. A., and Lehmkuhle, S., editors, *Synaptic Modification, Neuron Selectivity, and Nervous System Organization*, 153–173. Hillsdale, New Jersey: Lawrence Erlbaum.

Arbib, M. A. (1975). Artificial intelligence and brain theory: Unities and diversities. *Annals of Biomedical Engineering*, 3:238–274.

Arbib, M. A. (1987). Levels of modeling of mechanisms of visually guided behavior. *Behavioral and Brain Sciences*, 10:407–465.

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

Arbib, M. A. (1990). Schemas for high-level vision: The problem of instantiation. In Schwartz, E. L., editor, *Computational Neuroscience.* Cambridge, Massachusetts: MIT Press.

Arbib, M. A. (1993). "what", "where", and the architecture of action-oriented perception. In Arbib, M. A., editor, *Proceedings of Workshop on Neural Architectures and Distributed AI: From Schema Assemblages to Neural Networks.*

Attneave, F. (1971). Multistability in perception. *Scientific American*, 225:63–71.

Babich, S., and Standing, L. (1981). Satiation effects with reversible figures. *Perceptual and Motor Skills*, 52:203–210.

Bartlett, F. C. (1932). *Remembering.* Cambridge University Press.

Biederman, I. (1987a). Matching image edges to object memory. In *Proceedings of 1st International Conference on Computer Vision*, 384–392.

Biederman, I. (1987b). Recognition-by-components: A theory of human image understanding. *Psychological Review*, 94(2):115–147.

Binford, T. O. (1971). Visual perception by computer. In *Proceedings of the IEEE Conference on Systems and Control (Miami, FL).*

Bovik, A. C., Clark, M., and Geisler, W. S. (1987). Computational texture analysis using localized spatial filtering. In *Proceedings of the Workshop on Computer Vision*, 201–206.

Bugelski, B. R., and Alampay, D. A. (1961). The role of frequency in developing perceptual sets. *Canadian Journal of Psychology*, 15:205–211.

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.

Carr, T. H., McCauley, C., Sperber, R. D., and Parmelee, C. M. (1982). Words, pictures, and priming: On semantic activation, conscious identification, and the automaticity of information processing. *Experimental Psychology: Human Perception and Performance*, 8(6):757–777.

Cavanagh, P. (1978). Size and position invariance in the visual system. *Perception*, 7:167–177.

Cavanagh, P. (1985). Local log polar frequency analysis in the striate cortex as a basis for size and orientation invariance. In Rose, D., and Dobson, V. G., editors, *Models of the Visual Cortex*, 85–95. New York: Wiley.

Chastain, G., and Burnham, C. A. (1975). The first glimpse determines the perception of an ambiguous figure. *Perception & Psychophysics*, 17(3):221–224.

Cohen, M. A., and Grossberg, S. (1983). Absolute stability of global pattern formation and parallel memory storage by competitive neural networks. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-13(5):815–826.

Cohen, M. A., and Grossberg, S. (1987). Masking fields: A massively parallel neural architecture for learning, recognizing, and predicting multiple groupings of patterned data. *Applied Optics*, 26(10):1866–1891.

Cohen, M. A., Grossberg, S., and Stork, D. G. (1988). Speech perception and production by a self-organizing neural network. In Lee, Y. C., editor, *Evolution, Learning, Cognition, and Advanced Architectures*. Singapore: World Scientific.

Cornwell, H. G. (1976). Necker cube reversal: Sensory or psychological satiation? *Perceptual and Motor Skills*, 43:3–10.

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.

Fagg, A. H. (1993). Reinforcement learning for robotic reaching and grasping. In Bennett, K. M. B., and Castiello, U., editors, *New Perspectives in the Control of the Reach to Grasp Movement*. Amsterdam: The Netherlands: Elsevier.

Feldman, J. A. (1985). Four frames suffice: A provisional model of vision and space. *Behavioral and Brain Sciences*, 8:265–313.

Finkel, L. H., and Sajda, P. (1992). Object discrimination based on depth-from-occlusion. *Neural Computation*, 4:901–921.

Ginsburg, H., and Opper, S. (1969). *Piaget's Theory of Intellectual Development*. Englewood Cliffs, New Jersey: Prentice-Hall.

Goodale, M. A., and Arbib, M. A. (in press). The Cognitive Architecture of Vision and Action. In Pylyshyn, Z. W. (Ed.) *Options for Cognitive Theory: Issues and Methods for a Science of Cognition*. Norwood, NJ: Ablex.

Grossberg, S. (1976). Adaptive pattern classification and universal recoding. I: Parallel development and coding of neural feature detectors. *Biological Cybernetics*, 23:121–134.

Grossberg, S. (1978). A theory of human memory: Self-organization and performance of sensory-motor codes, maps, and plans. In Rosen, R., and Snell, F., editors, *Progress in Theoretical Biology*, vol. 5. New York: Academic Press.

Grossberg, S. (1987a). Competitive learning: From interactive activation to adaptive resonance. *Cognitive Science*, 11:23–63.

Grossberg, S. (1987b). Cortical dynamics of three-dimensional form, color, and brightness perception, I: Monocular theory. *Perception & Psychophysics*, 41(2):87–116.

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

Gruber, H. E., and Vonèche, J. J. (1977). *The Essential Piaget.* New York: Basic Books.

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.

Head, H., and Holmes, G. (1911). Sensory disturbances from cerebral lesions. *Brain*, 34:102–254.

Hebb, D. O. (1949). *The Organization of Behavior.* New York: Wiley.

Henderson, J. M., Pollatsek, A., and Rayner, K. (1987). Effects of foveal priming and extrafoveal preview on object identification. *Experimental Psychology: Human Perception and Performance*, 13(3):449–463.

Hertz, J., Krogh, A., and Palmer, R. G. (1991). *Introduction to the Theory of Neural Computation.* Redwood City, California: Addison-Wesley.

Hinton, G. E. (1990). Mapping part-whole hierarchies into connectionist networks. *Artificial Intelligence*, 46:47–75.

Hochberg, J. E. (1978). *Perception, 2nd Ed.* Englewood Cliffs, New Jersey: Prentice-Hall.

Hummel, J. E., and Biederman, I. (1992). Dynamic binding in a neural network for shape recognition. *Psychological Review*, 99:480–517.

Kolen, J. F., and Pollack, J. B. (1990). Scenes from exclusive-or: Back propagation is sensitive to initial conditions. In *Proceedings of 12th Annual Conference of Cognitive Science Society*, 868–875.

Lacaze, A., and Meystel, M. (1993). Baby sub: Using schemata for conceptual learning. In Arbib, M. A., editor, *Proceedings of Workshop on Neural Architectures and Distributed AI: From Schema Assemblages to Neural Networks.*

Leow, W. K. (1994). *VISOR: Learning Visual Schemas in Neural Networks for Object Recognition and Scene Analysis.* PhD thesis, Dept. of Computer Sciences, Univ. of Texas at Austin, Austin, Texas.

Leow, W. K., and Miikkulainen, R. (1993). Representing visual schemas in neural networks for object recognition. In *Proceedings of International Conference on Neural Networks*, vol. III, 1612–1617.

Leow, W. K., and Miikkulainen, R. (1994a). Priming, perceptual reversal, and circular reaction in a neural network model of schema-based vision. In *Proceedings of 16th Annual Conference of Cognitive Science Society.*

Leow, W. K., and Miikkulainen, R. (1994b). VISOR: Schema-based scene analysis with structured neural networks. *Neural Processing Letters*, 1(2):18–23.

Lindauer, M. S. (1989). Expectation and satiation accounts of ambiguous figure-ground perception. *Bulletin of the Psychonomic Society*, 27(3):227–230.

Long, G. M., and Toppino, T. C. (1981). Multiple representations of the same reversible figures: Implications for cognitive decisional interpretations. *Perception*, 10(2):231–234.

Marr, D. (1982). *Vision*. San Francisco, California: W. H. Freeman.

McCloskey, M., and Cohen, N. J. (1989). Catastrophic interference in connectionist networks: The sequential learning problem. *The Psychology of Learning and Motivation*, 24:104–169.

Mozer, M. C. (1988). A connectionist model of selective attention in visual perception. In *Proceedings of 10th Annual Conference of Cognitive Science Society*.

Mozer, M. C., Zemel, R. S., Behrmann, M., and Williams, C. K. I. (1992). Object discrimination based on depth-from-occlusion. *Neural Computation*, 4:650–665.

Olshausen, B., Anderson, C., and Van Essen, D. (1992). A neural model of visual attention and invariant pattern recognition. Technical Report CNS Memo 18, Computation and Neural Systems Program, Division of Biology, California Institute of Technology, Pasadena, California.

Olshausen, B. A., Anderson, C. H., and Van Essen, D. C. (1993). A neurobiological model of visual attention and invariant pattern recognition based on dynamic routing of information. *Neuroscience*, 13(11):4700–4719.

Orbach, J., Ehrlich, D., and Heath, H. A. (1963). Reversibility of the Necker cube: I. An examination of the concept of "satiation of orientation". *Perceptual and Motor Skills*, 17:439–458.

Piaget, J. (1952). *The Origins of Intelligence in Children*. International University Press.

Piaget, J. (1971). *Biology and Knowledge*. Edinburgh University Press.

Pollen, D. A., Lee, J. R., and Taylor, J. H. (1971). How does the striate cortex begin the reconstruction of the visual world? *Science*, 173:74–77.

Ratcliff, R. (1990). Connectionist models of recognition memory: Constraints imposed by learning and forgetting functions. *Psychological Review*, 97:285–308.

Riani, M., Oliva, G. A., Selis, G., Ciurlo, G., and Rossi, P. (1984). Effect of luminance on perceptual alternation of ambiguous patterns. *Perceptual and Motor Skills*, 58:267–274.

Rock, I. (1983). *The Logic of Perception*. Cambridge, Massachusetts: MIT Press.

Rock, I., and Mitchener, K. (1992). Further evidence of failure of reversal of ambiguous figures. *Perception*, 21(1):39–45.

Rumelhart, D. E. (1975). Notes on a schema for stories. In Bobrow, D. G., and Collins, A. M., editors, *Representation and Understanding: Studies in Cognitive Science*. New York: Academic Press.

Rumelhart, D. E. (1980). Schemata: The building blocks of cognition. In Spiro, R. J., Bruce, B. C., and Brewer, W. F., editors, *Theoretical Issues in Reading Comprehension*. Hillsdale, New Jersey: Lawrence Erlbaum.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986a). Learning internal representations by error propagation. In Rumelhart, D. E., and McClelland, J. L., editors, *Parallel Distributed Processings*. Cambridge, Massachusetts: MIT Press.

Rumelhart, D. E., and Ortony, A. (1977). The representation of knowledge in memory. In Anderson, R. C., Spiro, R. J., and Montague, W. E., editors, *Schooling and Acquisition of Knowledge*. Hillsdale, New Jersey: Lawrence Erlbaum.

Rumelhart, D. E., Smolensky, P., McClelland, J. L., and Hinton, G. E. (1986b). Schemata and sequential thought processings in PDP models. In McClelland, J. L., and Rumelhart, D. E., editors, *Parallel Distributed Processings*. Cambridge, Massachusetts: MIT Press.

Rumelhart, D. E., and Zipser, D. (1985). Feature discovery by competitive learning. *Cognitive Science*, 9:75–112.

Schmidt, R. A. (1975). A schema theory of discrete motor skill learning. *Psychological Review*, 82:225–260.

Seibert, M., and Waxman, A. M. (1989). Spreading activation layers, visual saccades, and invariant representations for neural pattern recognition systems. *Neural Networks*, 2:9–27.

Sejnowski, T. J., and Tesauro, G. (1989). The Hebb rule for synaptic plasticity: Algorithms and implementations. In Byrne, J. H., and Berry, W. O., editors, *Neural Models of Plasticity*, 94–103. New York: Academic Press.

Spitz, H. H., and Lipman, R. S. (1962). Some factors affecting Necker cube reversal rate. *Perceptual and Motor Skills*, 15:611–625.

Tsal, Y., and Kolbet, L. (1985). Disambiguating ambiguous figures by selective attention. *Quarterly Journal of Experimental Psychology: Human Experimental Psychology*, 37(1):25–37.

Zemel, R. S., Mozer, M. C., and Hinton, G. E. (1990). TRAFFIC: Recognizing objects using hierarchical reference frame transformations. In Touretzky, D. S., editor, *Advances in Neural Information Processing Systems 2*, 266–273. San Mateo, CA: Morgan Kaufmann.