NARRATOR: This film is a summary of computer vision research, one of several research projects carried out in the MIT Laboratory of Artificial Intelligence. We believe that studying vision and manipulation is a good way to approach many important problems about the theory of general intelligence. During this film, you will see how experimental computer programs extract line drawings from pictures, and use knowledge about the three-dimensional world to understand these line drawings. And you will also see how new ideas about artificial intelligence are used in these processes.

All the programs depicted deal with scenes that are unfamiliar to the robot. The programs can work with a broad range of particular structures and can recover from many kinds of errors. We'll start with a description of some high-level processing, and then show the detailed program operations. To begin our discussion, Marvin Minsky explains how the robot can learn new concepts.

MARVIN MINSKY: I'm going to describe very briefly how a program written by Pat Winston works. This program is supposed to learn by example. And it learns in a rather different way from previous programs that have to do learning.

It needs a good teacher. The teacher comes up to the machine, which contains a vision system for describing things. And I give it an example-- for example, that, and say, "This is an arch."

Well, the machine describes it. And the description, the structure that the machine builds up inside its memory, is something like this-- there are three objects found by the vision system. They're all identified as blocks. And it notices certain relations between these objects-- for example, that this one supports that, and this one supports that.

These are the legs of the arch. And that's the top. And notice there's a lot of other things that I haven't got time to describe, such as that this block is standing up, and this one's lying down, and so forth.

And it records that as an arch. It's the only thing it knows about arches, so far. Then we give it another example.

Let's show it that. And we say, "That's not an arch." Well, the machine can only do one thing with the picture itself.

It has to describe it. And its description looks something like this. To you and me, this doesn't look very much like an arch at all.

But in some sense, the description is quite similar. It still has three objects. This one is lying down. These are standing up. They're all blocks.

But there's a very large difference as far as the machine's descriptive structure is concerned. The relation between the blocks-- this one supporting that and this one supporting that-- is missing. Well, since the machine has been told it's not an arch, what it has to do is change its description of arch so that it won't think that's an arch anymore.

And what can it do? It can say, well, it must be supported. So this is its first step in learning what the teacher's trying to get it. It now knows that it must be supported.

Let's give another example. We tell it that's not an arch. And again, the machine has to describe the scene, and its description now looks something like this.

Again, the description is very much like the description of all the others. There are three blocks. There is support here.

But there's one more relation that wasn't there before that has quite a high priority in Winston's program. And that's contact-- these two blocks are touching one another. And again, the teacher tells it the answer. In this case, that's not an arch.

The program says, oh, dear, I must change my description of arch so that that will be rejected. And since the most important difference is this contact relation, it puts in a new relation which now is, must not contact. And you see it's getting a pretty good idea of what an arch is.

Already, it's a structure of three things which must have the right support relations. And they must not touch, the two supports, which means that there will be a hole in it. The program doesn't have the idea of whole, but it has a pretty good practical equivalent.

Finally the teacher might give it a fourth example-- this, and say, "That is an arch." And the description of this structure agrees with the description that's been building up except for one small detail-- the top thing is no longer a block, it's a wedge. And the program has to say, I'll accept things that are wedges as well as blocks.

And that's pretty easily changed by saying this can be block or wedge. Or in the actual program, it generalizes and says that can be a prism. Well, the point of the program is that it doesn't learn so much a little bit at a time as in the traditional reinforcement theories of learning, which work very well for rats and very badly for people.

But for each example, the machine jumps to some sort of conclusion-- learns a new relation and it can learn very fast. It's learned a lot from four examples. On the other hand, it takes a good teacher. If you gave it misleading examples, where there are many differences between the things it's seen and the new things and it will be at sea. There will be a lot of differences that it could put in here. And it won't have any good way deciding which differences to represent in its final result.

NARRATOR: Once a learning program has grasped a general concept, recognition programs can find examples of this concept. The machine can find the arch, even when the scene is cluttered with other objects. It can also recognize many variations of the arches it was originally taught.

Other ideas can be taught by other learning sequences. The machine is able to learn that a pedestal is a board on top of a standing brick, a table is a board on a group of standing bricks, and a chair is an arch standing on one side of a table. But before the machine can do this high-level learning and identification, the raw data from the eye must be transformed into a usable scene description.

As the first step in this transformation, goal-oriented procedures guide low-level subroutines through the array of light intensity values. An electronic eye can supply the light intensity value for any point in the scene requested by the robot. Once procedures find the beginning of a line, other routines follow it, and search for all other lines emanating from its ends. Other routines hypothesize that regions may be parallelograms, and suggest where vertices might be found. The result is that these subroutines usually examine only a small fraction of the available points, because the machine has good ideas about where to look for lines once a few are found.

David Waltz has written a program which adds depth to the emerging description by generating labels for a line drawing. These labels specify the physical nature of each individual line. There are labels for shadows, obscuring edges, convex edges, and concave edges.

Physical constraints limit the number of ways that scene features can form vertices. Most combinations of line labels simply cannot occur at any junction. Furthermore, labellings for adjacent junctions must match, because the nature of a line cannot change along its length. This makes the selection of labellings like the assembly of a jigsaw puzzle, because previous selections strongly limit each new choice.

In our sample scene, the L junction starts with a complete set of 123 possible labellings. And this number is immediately reduced to 76, because the relevant region brightnesses preclude certain shadow labellings. Next, the mutual constraints between junctions have an effect. The arrow labellings are reduced from the initial 79 to the 52 compatible with the neighboring L. Similarly, only 32 of the options for the L are now compatible with the remaining arrow labellings.

Sometimes, the analysis of a particular junction can cause an effect to propagate for a considerable distance. Paths propagate through the drawing until activity dies at junctions where no new changes are forced.

By the time the last junction is considered, the overall arrangement typically forces a single interpretation for the entire line drawing. After determining basic facts about the lines, other specialist programs attempt to find the identity, location, and dimensions of each object. Once an object is found to be a brick, the dimension programs must find three un-obscured edges along each of the brick's axes.

This set of three lines is called the skeleton. But sometimes, a brick may be severely obscured. This arch presents a difficult problem, because such a small portion of the left post is in view. The identification specialist decides that the object is a brick. But the dimension specialist cannot find three complete orthogonal lines for a skeleton.

Programs by Timothy Finin are used in this situation. They determine the missing information by first using context to form a reasonable hypothesis, and then using a checklist to confirm the hypothesis. The robot's internal deliberation shows that the checklist is needed, because there may be a clear contradiction to the proposed hypothesis. Here the hypothesis is that the dimensions of the two posts are identical.

Looking behind the scene explains why the hypothesis failed. In scene analysis, the programs work towards abstract descriptions. In design, the process is reversed. Programs by Rich Boberg accept general facts from a user and envision a scene compatible with those facts.

Envisioning structures is a kind of automatic design. Copying structures with the hand is a kind of automatic assembly. Here, the robot prepares to copy a structure consisting of one cube and several assorted bricks.

Using programs described earlier, the robot finds the lines, regions, and skeletons of the objects, and then joins the regions marked R1, R2, and so forth into bodies like B1 and B2. Once the robot finds all the objects, it knows the dimensions and locations. Other robot programs know facts about building structures-- for example, that the bottom blocks must always be placed first. These programs generate a complete building plan for the structure to be copied. Once a plan is formed, the robot's arm assembles a copy using spare parts from a warehouse area.

If the arm misplaces an object, the error is detected by inspection programs, which compare the copied scene to the original model.

A small wrist motion moves the object to a better position. New arms under development are more skillful because of better design and improved force sensors. An arm built by David Silver can position objects with high accuracy. Force feedback allows the arm to turn a crank.

To build complicated structures, the robot must use sophisticated planning. A new program by Scott Fahlman knows how to build a balanced structure using a stable substructure, a scaffold, or a temporary counterweight. New studies are also underway to investigate the use of continuous visual feedback.

You've seen only a sampling of the work we have done. But we hope this film has given you some feeling for how these difficult problems can be approached.