[1] One approach to determining the three-dimensional shape of an object is the use of light stripes. In this approach, applicable especially in industrial robotics, the pattern of stripes of light (top) is interpreted by the computer to determine the orientation of the surface. Here the orientations determined are color-coded (middle). Once the surface orientations are known, the computer—in the example illustrated here from work by David Smith and Takeo Kanade at Carnegie-Mellon University in Pittsburgh, Pa.—isolates each object by edges and selects certain simple geometric shapes, such as cylinders and cones, to fit to the surfaces observed (bottom). When such surfaces are correctly fitted, the computer can "recognize" the object as a cup, pan, and so on, by their descriptions in memory as combinations of simple shapes.

Computer vision: the challenge of imperfect inputs

Humans can see and understand extremely complex scenes at a glance. By contrast, the best computer vision systems now available commercially have very primitive capabilities. They can rapidly recognize and locate objects on a conveyor belt. but only under strictly constrained conditions and lighting—in essence, circumstances that reduce the objects to be identified to isolated, two-dimensional black and white silhouettes. The development of a general-purpose computer vision system that can approach the abilities of the human eye and brain is remote at present, despite recent progress in understanding the nature of vision.

Vision is difficult for a computer for a number of reasons. For one thing, the images received by a sensing device—for example, a vidicon tube-do not contain sufficient information to construct an unambiguous description of the scene observed. Most importantly, depth information is lost, and there are many parts of an image where objects overlap.

Also. many different factors are confounded in the image—a surface may *appear* dark, for example, because of its low reflectance, shallow angle of illumination, insufficient illumination. or unfavorable viewing angle. The interpretation of what is Seen requires a large body of knowledge. not only about what various objects look like—such as cars, houses, roads, and trees—but also about how these objects can be expected to fit together.

Finally, vision involves a large amount of memory and many computations. For an image of 1000 by 1000 pixels (image elements). some of the simplest procedures require 10^9 operations. The human retina, with 10^8 cells operating at roughly 100 hertz, performs at least 10 billion operations a second, and the visual cortex of the brain has undoubtedly higher capacities.

Three steps to vision

Most computer Vision systems arc based on a three-step process that works upward from the raw image to some sort of scene desaiption. Low-level processing extracts features from the image, detecting edges and then connecting *edges* into lines or curves or segmenting the image into regions that have more or less uniform properties [see chart on p. 90].

Intermediate-level processing derives from the image of the features of the scene—that is, the real objects that produced the image. At this point three-dimensional information, such as surface orientation and distance from the viewer, are derived from the two-dimensional image.

The highest level of processing produces a description of the scene—what is actually seen. Here the computer attempts to match models of known objects, such as cars, buildings, or trees. to the sane description and thus determine what is there.

Although this three-level process applies to many vision systems, there arc some types that attempt to go from raw image to at least partial scene description in one step. In addition, most commercial Vision systems omit one or more steps because they operate in a simple environment that does not require such elaborate processing.

In typical commercial *systems,* a gray-level image is converted into a binary black and white representation and the outlines of the objects are obtained. These outlines are then measured and

Takeo Kanade and Raj **Reddy**
Carnegie-Mellon University

compared with models stored in memory. When the observed outline and the model outline match, the object—say, a part on a conveyor belt—can be identified and its position and orientation deduced. In this instance, no 3-D interpretation is needed.

The General Motors Corp.'s Consight system, used in connection with assembly robots. is an example of a computer-vision machine. Similar vision systems are now performing such tasks as inspection, parts sorting, and the guidance function in arc welding and are assisting robots in a variety of industries.

More sophisticated systems are required to go beyond the limited factory applications to complex tasks—for example, photointerpretation or automatic navigation—where the vision system would operate in an unconstrained environment in which lighting and other variables ace not tightly controlled. These systems are now only in the early stages of development. Effort has been divided mainly between the intermediate-level task of obtaining 3-D information and the higher-level task of description.

Determining 3-D shapes from images

The basic approach to determining 3-D shapes from images is to use shading, textures, shadow edges, and other image features as constraints on the shapes that may be present. Even though each constraint may allow many different shapes, a unique shape can be obtained by combining a number of constraints. Until relatively recently, most vision programs used such constraints in specialized ways. But in the past few years a powerful new approach has been formulated, and it has allowed a set of computational modules to be developed.

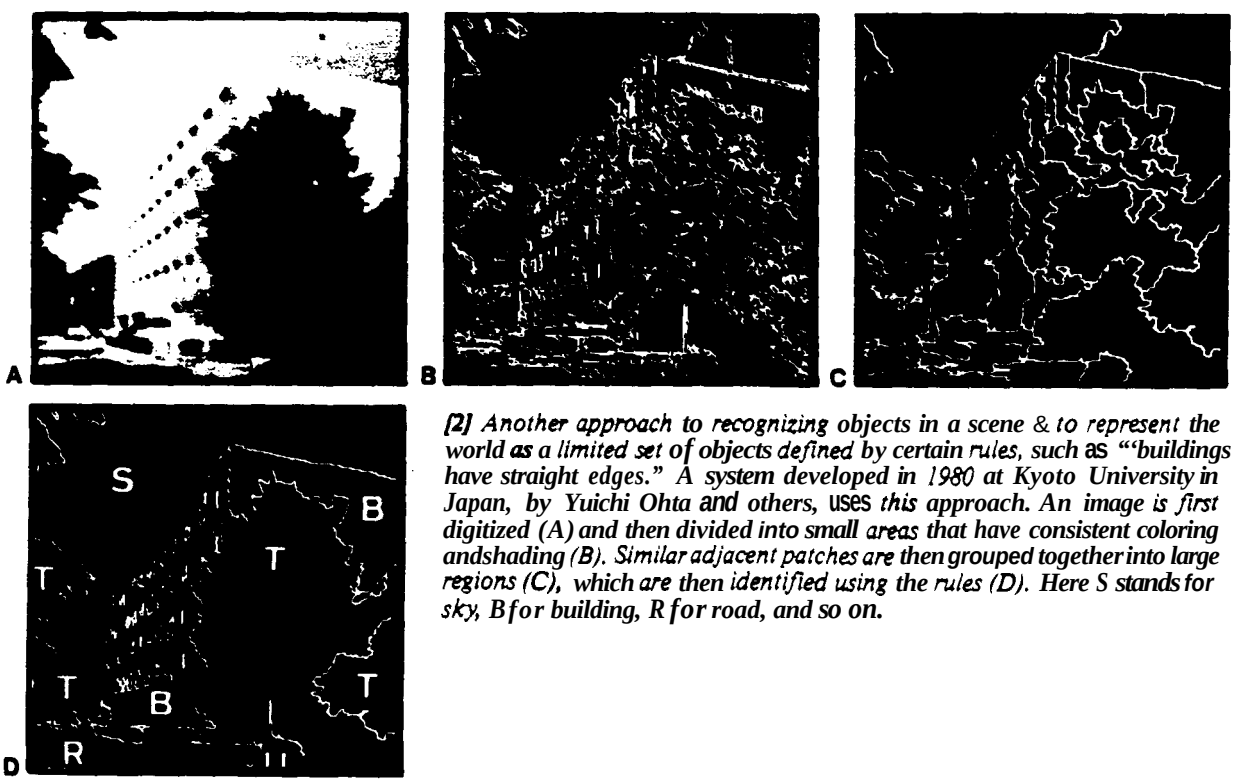
The new approach involves three steps. First, an image quantity, or primitive, is paired with a scene feature that it constrains. For example, the intensity or shading of an image may be matched with the surface orientation of an object in the scene. The physical and geometrical rules that govern the process of generating the image primitive from the Scene primitive is then represented mathematically. For instance, the laws relating surface orientation, angle of illumination, and surface reflectance will determine the intensity at a point on the image. By using these laws, the set of constraints on the scene primitive (in this example, surface orientation) can be derived from a set of measurements of the image (in this case, intensity).

Modules have been developed that use shading, texture, contours, shadows, stereooptical disparity, and other image qualities to produce constraints on the scene features. Though successful programs for determining 3-D shapes from arbitrary images have not yet been developed. sufficient progress has been made so that systems now can analyze images made from restricted types of scenes. For example, the shape-from-shading method developed by Berthold K.P. Horn of the Massachusetts Institute .of Technology in Cambridge, interprets images of objects that are smooth and illuminated by diffuse lighting.

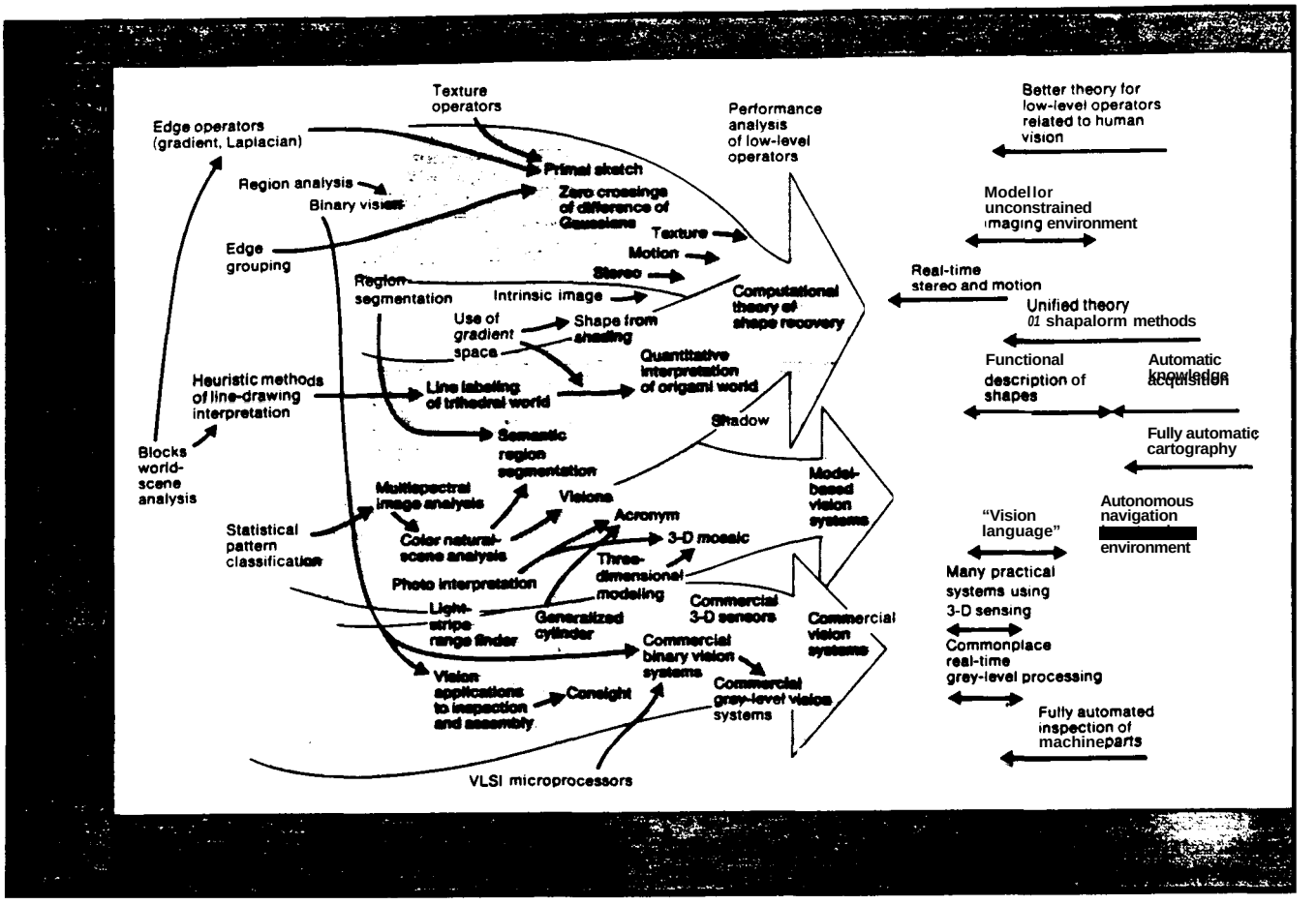
One obstacle to general systems of this sort is the number of computations required to determine the constraints, especially when more than one is applied. Highly parallel, special-purpose computers are required to perform such computations in real time. Once processors an developed with capabilities of 10^{10} to 10^{13} instructions per second—comparable to those estimated for the human retina—it seems reasonable to believe that extensions of current techniques will make it possible to deduce 3-D features from arbitrary scenes in real time.

Such capabilities are already being approached in indoor applications where active lighting techniques can be used. In active lighting, the source of illumination is so structured as to give immediate clues to the 3-D shape of the objects imaged. One of the most common typa of active lighting is stripe lighting, in which the source projects parallel stripes of light across the scene. The bumps and jogs in the light stripe accurately reflect the shapes of the objects independent of their other qualities, such as reflectance Fig. 1]. This technique is already being applied in a number of products.

Thus the first problem of computer vision—perceiving the shape of a three-dimensional world from two-dimensional images—is on its way to systematic solution. More difficult is the higher-level problem of recognizing the shapes deduced as ob-



[2] Another approach to recognizing objects in a scene & to represent the world as a limited set of objects defined by certain rules, such as "'buildings have straight edges." A system developed in 1980 at Kyoto University in Japan, by Yuichi Ohta and others, uses this approach. An image is first digitized (A) and then divided into small areas that have consistent coloring andshading (B). Similar adjacent patches are then grouped together into large regions (C), which are then identified using the rules (D). Here S stands for sky, B for building, R for road, and so on.



Computer vision efforts have advanced over the past 20 years along three fronts: low-level vision, the extraction of basic features, such as edges from an image; intermediate-level vision, the deduction of the three-dimensional shape of objects from the images; and high-level vision, the recognition of objects and their relationship. Some representative research projects include the Hand-Eye robotic vision project initiated at the Massachusetts Institute of Technology in Cambridge and at Stanford University in Palo Alto, Calif.; the pattern-information processing system (PIPS) project in Japan, one of the earliest focused research programs sponsored by the Ministry of International Trade and Industry; the U.S. Defense Advanced Research Projects Agency's image-undemanding system (IUS) project; and the current Darpa next-generation project.

jects and identifying them. Here progress has been more limited.

The main approach to the problem of object recognition and scene description is called model-based vision. In this approach a simplified model of the world is prepared, depending on the task at hand. For example, an urban-scene analysis program would model the world as consisting of certain types of objects—buildings, cars, trees, roads, sky, and so on. The models of objects would be described in memory as having certain characteristics, and the program would attempt to match these against various parts of the image.

This task is similar to that of speech recognition, in which the program tries to match models (words) against the incoming speech signals. The difficulty in scene analysis is that a given object looks different from various angles, so the program must be capable of manipulating the model in three dimensions to match with images in many different ways.

Several approaches have been taken to match models and images. One involves performing correlations between a stored image of an object and the image itself. At the point where the object appears on the image, a correlation peak is formed. Such correlations are frequently carried out by optical means, although they can also be performed digitally.

Though correlators have been developed that recognize such objects as tanks from a variety of different viewpoints in a complex background, the flexibility of this approach appears to be limited. The objects searched for must be identical with the model rather than an example of a certain type of object.

Another approach exemplified by the system, Argos, developed by Steven Rubin at Carnegie-Mellon University, Pittsburgh, Pa., used entire images of a given scene as the basic models. Leaving the program with the limited task of deciding which view of the scene—say, downtown Pittsburgh—it was looking at. The approach was in part inspired by speech-understanding techniques that limited the speech input to a finite number of sentences, in one case several thousand sentences, from which the program had to decide the one that was spoken.

Argos stored each view as a network of relationships—in a form like "large brown object to left of small brown object." By searching through the network and matching the relationships against the image presented to it, Argos could identify the correct viewpoint 80 percent of the time. However, such a system has severely limited applications—such as hdping a robot navigate around a small room.

A more versatile approach is to represent the world as a limited set of objects defined by a set of rules that either describe properties of objects ("buildings have straight edges") or procedures that the program should carry out under given circumstances ("if an object has straight edges, look for repeating rows of windows

to confirm it is a building"). An outdoor scene system using this approach was developed at Kyoto University in Japan in 1980by Yuichi Ohta and others. The system first portions the image into many small patches having the same color and shading. It then groups the adjacent patches into large areas, which it identifies through the rules as trees, roads, cars, buildings, sky, or a half dozen other categories [Fig. 21.

These systems. although they could label some of the objects viewed, were two-dimensional and could not give a 3-D description of the relationships among the objects. Rodney Brooks and Thomas Binford of Stanford University, Palo Alto, Calif., developed the Acronym system in 1981, which models objects three-dimensionally as generalized cylinders. Images are analyzed and divided into combinations of ribbons (cylinders) and ellipses (ends of cylinders). Then the system matches objects made up of cylinders—such as aircraft—to the images by estimating the dimensions of the cylinders and their orientations. Acronym has been applied to airport scenes, and the Hughes Aircraft Co., in Los Angeles, Calif., is using this approach to develop a photo interpretation system for port monitoring.

Urban scenes can be used to produce 3-D descriptions in the Incremental 3-D Mosaic system, under development at Carnegie-Mellon University by Marty Merman and others. This system employs block models of buildings. Several images from different angles are used, and the system builds up its model of the scene as new portions come into view. The system has successfully constructed a three-dimensional description of part of Washington, D.C.

Much research remains to be done

Clearly the general problem of recognizing objects in a scene and describing their relations in three dimensions is far from solved. Existing systems can deal only with restricted types of scenes and they operate slowly. To dcvdop generic systems, much more knowledge of the world has to be incorporated into the program. There must be a mechanism to store large-scale spatial information about an area, from which relevant data can be extracted and into which newly acquired information can be fed.

Finally, there must be a dramatic increase in the speed of vision processors. The computing requirements can be estimated if one considers that a mobile cart developed by Hans Moravec at Stanford University navigated at a speed of 3 to 5 meters per hour while analyzing one image of its surroundings for each 1-meter lurch with a 1-million-instruction-per-second computer. To guide a similar cart at a walking speed of a meter pa second will therefore require from 10^9 to 10^{10} instructions pa second.

Once such high-speed processors are available, highly computationally intensive methods may be attempted that have not been tried so far, leading to more versatile systems. ●

SOFTWARE ENGINEERING

The power of computer hardware has improved by a factor of many thousands over the last 20 years, but programmer productivity has at best only doubled. Furthermore, as software systems have grown larger and more complex, the absolute number of bugs in them has increased, as have the potential effects of program mors. In some cases, systems have been abandoned after the expenditure of tens or even hundreds of millions of dollars.

Raymond Yeh University of Maryland

Yet it appears that computer systems relying on increasingly complex software systems will play a growing part in people's lives in the future.

How will the software industry increase both the quality of its products and productivity? The apparent solution is automation—using the same kinds of tools that software developers have made available to others—to aid the process of creating software.

Use of software tools lags

Although a fair number of software tools—compilers, formal design methodologies, testing programs; program-design languages, and others—arc available in the United States and elsewhere, they are sddom used in the software industry (see table]. Among the problems inhibiting tool use is that corporate management generally has little if any software background and therefore is not sympathetic to the need for such tools. Furthermore, there is usually no entity within a corporation to provide tools on a centralized basis; tools must generally be paid for out of the funds for a particular project and so there is little incentive for introducing them.

Some of these organizational problems hindering the introduction of software tools are less prevalent in Japan than in the United States because Japanese companies may consider the benefits versus costs of software tools over a wider base than a single project. In addition, the relative lack of mobility of Japanese software engineers leads to an increased incentive for introducing tools; programmas are more likely to reap the long-term benefits of tool use if they stay at the same company for the bulk of their careers. Ya another factor tending to favor tool use in Japan is the willingness to do postmortems on finished projects to determine what might have been &ne differently and what lessons might be learned from them, rather than immediately proceeding to the next project.

The progress of software-engineering techniques is also slowed because many programmers must spend a large portion of their time maintaining a huge inventory of existing. ill-structured software—estimated at over $200 billion. Software tools designed to work with well-structured programs written in high-level languages are of little use in such work, and software organizations that spend most of their time maintaining and upgrading old code have little time to apply formal methodologies to new programs. Some software organizations report spending as much as 70 percent of their time correcting errors in and making changes or enhancements to existing programs.

New alternatives are needed

All of these problems, however, may simply be masking underlying inadequacies in current programdevelopment techniques. It is estimated that full use of all existing programming tools. starting with the *requirements* and specifications stages and running through unit and system testing, would reduce the cost of software development by only 40 percent. Further gains may be made because of the reduced costs of maintaining well-structured programs with good documentation, but a factor of two is about the best that can be expected in software productivity by applying existing tools.

Inadequacies in techniques come from two areas. First, it is impossible to prescribe fully the requirements of a program; the user always looks at the completed system and decides that certain operations should be carried out slightly differently or that functions must be added. Second, there an no facilities for reusing previous designs or sections of program code. To achieve the gains in software productivity that will be required for next-