NASA CR-165,012

JPL PUBLICATION 81-92

# Computer Vision

Donald Gennery
Robert Cunningham
Eric Saund
John High
Carl Ruoff



November 1, 1981

National Aeronautics and
Space Administration

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

JPL PUBLICATION 81-92

# Computer Vision

Donald Gennery
Robert Cunningham
Eric Saund
John High
Carl Ruoff

November 1, 1981

National Aeronautics and
Space Administration

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

The research described in this publication was carried out by the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration.

ABSTRACT

The field of computer vision is surveyed and assessed, key research issues are identified, and possibilities for a future JPL vision system are discussed.

CONTENTS

# 1. INTRODUCTION

## 1.1 Scope of Document

The purpose of this document is to review the state of the art of computer vision, to identify some key research issues, and to describe the capabilities and architecture that a future JPL vision system might have.

The document can be used as a brief introduction to the field of computer vision. For more detailed information the reader can consult the cited references. In particular, Duda and Hart [1973] provide a good text on some of the basic principles of computer vision; Pavlidis [1977] discusses in detail some of the algorithms of computer vision; Winston [1975a] presents a few significant pieces of work; and Aggarwal *et al.* [1977], Hanson and Riseman [1978a], and Barrow and Tenenbaum [1981] provide surveys of some of the more important work, portions of which are cited elsewhere herein. Also, Rosenfeld provides an annual bibliography of image processing and computer vision. (For example, Rosenfeld [1981] covers the year 1980 and contains 897 references.)

The term "computer vision" is considered here to be synonymous with "machine vision" and "robot vision". The terms "scene analysis," "image understanding," and "pictorial pattern recognition" often are also considered to be synonymous to these, although some authors use the latter three terms in more restricted senses. The general field of pattern recognition includes the recognition of abstract patterns in arbitrary non-pictorial data, and is not covered here. Also, techniques for analyzing highly specialized two-dimensional scenes (as in character recognition) are not covered here.

We have attempted to provide a survey of the more significant developments in the field of computer vision, but very likely some important work has been omitted. For this we apologize.

## 1.2 Overview

There are many ways in which a description of computer vision could be organized. For example, the field could be divided according to the systems developed by different individuals or groups, by the nature of the scenes being processed, by the nature of the information desired, by the kind of techniques that are used, or by the progression from low-level (close to the image) to high-level (close to the desired final results) processing. This document uses primarily the latter approach in Sections 2, 3, and 4, but elements of some of the other organizations appear in other sections. In addition to these considerations, it is difficult to produce a coherent organization because of the wide variety of approaches that are used and the overlap among them. This is caused by two facts. First, the vision task is very difficult and requires complicated methods. Second, vision research is still in a very primitive state. There is no consensus on the best techniques at any level of processing.

The terms "representation," "description," and "modelling" are used with various meanings and sometimes are used interchangeably. However, here definitions are assigned somewhat arbitrarily, as follows. "Representation" denotes the choice of low-level features derived from the picture which capture most of the important information in the picture but do not explicitly describe the global nature of the scene. Section 2 describes the representation techniques usually used, arranged roughly from the

lowest level towards higher levels. Section 3 covers "description," defined here as the ways of describing a scene or object more globally, perhaps in terms of the basic representation components and the relations among them. "Modelling" is considered to be essentially the same as description, except that it is applied to abstract models of objects which are searched for in actual pictures. One important vision task is recognition, which is considered to be the matching of a description derived from a picture to one or more abstract models, perhaps out of a large number of possible object models. Ways of doing this are described in Section 4. The situation often is not as simple as this straightforward description-model-matching scenario implies, however, for this same type of process can repeat at several levels in the analysis of a scene.

Another important vision task is verification, in which it is known what object should be present and approximately where it is, and it is desired to verify its presence and correct the estimate of its location. In such a case the a priori information can be used to guide the finding of important features used in the representation, and comparing the position of these to their predicted positions enables the model to be updated. A similar task is the tracking of moving objects. Here the predicted information comes from the results at previous times. Verification and tracking are discussed in Section 5.

Three-dimensional information can be measured directly by some devices, as mentioned in Section 2.1. However, usually it is obtained indirectly from two-dimensional pictures. If only a single two-dimensional picture is available, the depth information must be inferred by means of heuristics, some of which are inherent in the recognition techniques described in Section 4. However, if more than one picture is available, often

one of several techniques here called "stereo vision" can be used to obtain the depth information, as described in Section 6.

Section 7 discusses methods for getting the necessary information concerning object models into the computer vision system. Section 8 discusses methods for controlling and calibrating the cameras. Section 9 discusses issues of system architecture, both in terms of computational structures and hardware. Section 10 summarizes our conclusions concerning the state of vision, the key research issues of vision, and the possible nature of a future JPL vision system.

## 1.3 Typical Vision Systems

Before discussing particular aspects of the vision problem in the rest of the document, a few representative computer vision systems will be briefly described in this section.

First some operational industrial vision systems will be described. Many computer vision systems have been developed to provide visual feedback to a robot. Typically, these systems identify objects in the workspace of the robot, estimate the position and orientation of objects, and in some cases estimate the velocity of moving objects. In some cases, object identification may include inspection to detect defective products. Some systems are designed around a single object and use ad hoc techniques which may not apply directly to any other application. Other systems are designed for generic classes of objects. These systems have a programmable data base which can be loaded with models of specific objects for any given application. The vision system is programmed to extract a standard set of features to generate descriptions of objects in a scene. This is followed by a matching procedure which compares object descriptions obtained from the image to prototypes in an

object model data base. For a given application, the system provides some means of loading this data base with descriptions of specific prototypes.

At the National Bureau of Standards (NBS), VanderBrug et al. [1979] have developed a vision system using structured light and a camera mounted on the wrist of a robot arm. The camera is used to locate an object resting on a flat surface and to estimate its position and orientation so that it can be grasped by the arm. A description of the object is built up through multiple views obtained by moving the arm and camera. The structured light source, also on the arm, is a stroboscopic flash behind a cylindrical lens which produces a sheet of light. The camera line-of-sight is oblique to the plane of light. Image analysis consists of detecting the stripe of light on the surface of an object. The oblique viewing geometry causes the stripe to take on different appearances depending on the geometry of the object and its orientation. For example a rectangular object viewed head on produces a straight line, whereas a V-shaped line occurs when the object is viewed obliquely. Cylinders produce curved stripes when viewed parallel to the circular cross-section. Prismatic objects with grooves or ridges produce broken stripes. Image analysis consists of interpreting stripe features to identify objects. Knowing the geometry of the camera and light source makes it possible to extract 3-D measurements of illuminated points on the object, and thus ultimately the position and orientation of the object in robot coordinates.

CONSIGHT is a hand-eye system developed at General Motors (Ward et al. [1979]) for the purpose of picking up parts off a moving conveyor belt. A linear-diode-array camera is mounted directly above the conveyor belt. The image of an object is built up through a sequence of one-line images taken as the object passes through the field of view of the camera. When the

entire object has passed by the camera, a statistical description derived from its 2-D silhouette is used to identify it and to determine the position and orientation (in a plane) of a predetermined grasp point. The vision system can identify multiple two-dimensional objects which are in the field of view simultaneously. The only requirement is that parts do not touch or overlap. The vision system is programmed to recognize a part using a teach-by-showing method. In this mode, the part description derived from an image of the part is stored along with its name, which is entered by the operator. CONSIGHT also uses structured light consisting of two focused line light sources, one on either side of the camera along the direction of the conveyor belt movement. The light sources are aimed obliquely to the conveyor belt so that they both illuminate a thin line on the surface of the conveyor belt perpendicular to the direction of its motion and visible to the linear-diode-array camera. When the belt is empty, the camera sees a continuous white stripe. When an object is present, the oblique illumination of the object causes the thin line of light to move along the object towards the light source and out of view of the linear-diode-array camera. The amount of line of light movement is proportional to the height of the object. Thus objects appear as dark blobs on a bright background. The main advantage of this structured light approach is that parts can be detected independently of their contrast with the belt.

The SRI Vision Module (Nitzan et al. [1979]) is very similar to CONSIGHT, and in fact served as an inspiration for the latter system. The SRI system uses a more conventional 2-D array camera, and is thus suited to other applications as well as looking at parts on conveyor belts. Objects are detected as blobs in a binary image obtained by thresholding. Contrast is enhanced by careful lighting, including backlighting, so that objects are significantly brighter or darker than the background.

The system is programmed to recognize parts using the teach-by-showing method. During the teaching phase, the part is viewed several times in different positions and orientations in a plane to obtain a statistical distribution of the features (mean and standard deviation). The statistical distribution of feature values can optionally be used by the program to automatically generate an optimal binary decision tree for blob classification. Otherwise, classification is done by "nearest neighbor" matching in feature space. SRI has used the vision system in several experiments including picking parts from a moving conveyor belt, packing and unpacking boxes, inspection, and object tracking.

Although the above systems can perform useful vision tasks in real time, their performance is very limited. Fenerbaum et al. [1979] point out the following limitations of current industrial vision systems: high contrast, no shadows, no occlusion, two-dimentional models, rigid objects, and standard viewpoint. Next we discuss the ACRONYM system developed at Stanford University which overcomes all of these limitations to a certain extent. (For a more complete description of ACRONYM see Brooks et al. [1979], Brooks and Binford [1980], and Binford et al. [1980].) ACRONYM is not an operational system. It is a research vehicle still under development, which runs on large time-shared computer, and uses pre-stored images. However, it appears that it will be one of the most advanced vision programs yet produced, and it has a large degree of generality in the domain of identifying man-made objects.

ACRONYM models scenes and objects as specified by the user in terms of generalized cones (described in Section 3). An object consists of a hierarchical structure (an object graph) in which the volume primitives are generalized cones. In a generic model the number of each type of part and the dimensions, relative position, and relative orientation of the parts can vary

over specified ranges. A predictor and planner module converts the models into prediction graphs, which predict the appearance of objects within the scene, and provides a plan for lower-level descriptive processes and a matcher to find instances of the objects in the image. The edge mapper module detects edges using the method of Nevatia and Babu (described in Section 2.4) and forms these into ribbons, which are the two-dimensional analogue of generalized cones. The result is an observation graph. The matcher then matches the observation graph (produced from the image) and the prediction graph (produced from the model) to produce the interpretation graph, from which the interpretation of the scene is derived. In this process the predictor and planner can be invoked again to extend the graphs when a submatch is successful. It is planned to add stereoscopic vision to ACRONYM in the future. Certain aspects of ACRONYM are discussed further in Sections 3, 4, and 7.

## 2. REPRESENTATION

### 2.1 Pixels

The lowest-level representation of a digitized picture is the pixel. Each pixel represents the value of one or more quantities at some point in the two-dimensional picture. Usually the pixels form a uniform rectangular array over the picture, but sometimes other arrangements are used, such as a hexagonal array as advocated by Golay [1969].

Usually the pixels represent the brightness and perhaps color in a projection from a three-dimensional scene. (The fundamentals of image formation and color are discussed by Pratt [1978], and the way in which surface properties determine image intensities is discussed by Horn [1977].) In a monochromatic picture each pixel is represented by a single numerical value. In a color picture, each pixel is represented by two or more (three, if human vision is simulated) values representing brightness in different wavelength bands. (These values can be converted to other values such as hue, saturation, and brightness.) In general, however, the pixel values do not have to represent light intensities. Other media, such as sound or tactile pressure, could be used.

In fact, the pixels do not have to represent intensities at all. They can represent distances to the corresponding points in the three-dimensional scene, in which case the pixel array is referred to as a "range image." Such data is produced by a scanning laser rangefinder as discussed by Lewis and Johnston [1977] or could be produced by an appropriate sonar device. (Also see the next paragraph.) Similar data can be obtained somewhat less directly by a triangulation method using a laser and an ordinary camera, as described by Agin and Binford [1973].

If distance is not measured directly, it must be inferred indirectly from the two-dimensional pictures if three-dimensional scenes are being considered. A low-level method of estimating relative distance and surface orientation from a single picture is described by Barrow and Tenenbaum [1978]. They use heuristics based on the rate of change of brightness across a picture. Horn [1975] and Woodham [1977] provide methods based on the assumption of a reflectivity function that is constant over an object and some assumptions about the illumination. Methods based on more than one picture and high-level methods are discussed in later sections.

Nitzan et al. [1977] obtain registered range and intensity data by scanning the scene with an amplitude-modulated laser transmitter. A receiver outputs the amplitude and modulation phase shift of the reflected laser light, which are proportional to the intensity and (within one phase period) the range, respectively, of the reflecting surface. Thus both a range picture and a conventional brightness picture are produced simultaneously by the same device. The main drawback to such an approach in computer vision for robotics is that such a device is currently much slower than a standard TV camera.

If the pixels represent brightness, these brightness values usually suffice for later processing, since the important quantity is often relative brightness rather than absolute brightness. However, if it is desired to identify objects by their absolute color, the reflectance (lightness) of their surface (in several wavelength bands if color is used) must be determined. Thus the effects of illumination must be separated from the effects of surface lightness, which combine to produce the measured brightness. Human beings are quite good at this, even though the general problem is insoluble. Several investigators have proposed heuristics by which reasonable

results can be obtained for typical scenes. Land [1971] proposed a method based on edges (spatially sudden changes in brightness), assuming that these are due to changes in lightness whereas spatially gradual changes are due to illumination effects. Horn [1974] extended Land's work, using the inverse of the Laplacian operator as a means of integrating the information across two dimensions in order to obtain lightness at each point. Gilchrist [1979] showed the importance of three-dimensional position information in performing the separation of illumination and lightness, in addition to the two-dimensional information used by Land and Horn, but did not produce an algorithm for computer implementation. Ali *et al.* [1979] use normalized color values to try to minimize the effects of shadows, as described in Section 2.3.

## 2.2 Texture

Texture is a local variation in pixel values (whether brightness, color, or any other information mentioned in Section 2.1) that repeats in a regular or random way across a portion of an image or object. Texture can be characterized in various ways that result in descriptions including numerical or symbolic data, or both. Such a description can vary as a function of position within an image. However, the texture information is usually obtained at a lower resolution than the original image data from which it is derived, because several pixels are required to determine each texture element.

Once the texture has been measured, it can be used for several purposes. The nature of the texture may aid in identifying a particular object or material, the scale of the texture in the image may be used to determine distance if the scale of the texture on an object is known, and nonisotropy of texture may aid in determining surface orientation. Also, the

variations in texture across an image can be used in the same way as variations in untextured pixel values. Segmentation using the edge detection techniques and region finding techniques to be described in later sections can be done based on the texture information instead of on brightness, color, or other original pixel information. It is even possible to apply the definition of texture recursively in order to obtain a texture of textures.

Texture information is likely to be useful on natural outdoor scenes, since these tend to be highly textured. However, man-made objects usually have fairly uniform surfaces. Thus for the near-term NASA applications, which involve assembly work, texture is unlikely to be important. Therefore, it is not discussed in further detail here. The many approaches to texture analysis that have been used are surveyed by Haralick [1978].

## 2.3 Regions

A region is a set of connected pixels that share a common property such as average gray level, color, or texture in an image. The assumption in forming regions is that pixels sharing the above properties will also share the property of being images of points on the same object (or part of an object, or a collection of objects - in other words, an entity of interest to the vision system). Regions are typically described by statistical features such as perimeter, area, first and second moments, average gray level (or color), etc. In some cases, explicit shape information such as the location of corners or tabs is included, or possibly the entire boundary is represented for more general shape analysis. Lists of region records containing some portion of the above set of features for each region in the image are often structured as a tree or graph to indicate nesting and adjacency relationships between regions. Zucker [1976a] and Riseman and Arbib [1977] survey several region

growing techniques, some of which are discussed below. Also, Kanada [1978] discusses how the region segmentation problem relates to the rest of the vision task.

The simplest approach to region growing is to look for clusters of 0's and 1's in a binary image obtained by thresholding a gray-level image. The approach used by SRI (Nitzan et al. [1979] and CONSIGHT (Ward et al. [1979]) is connectivity analysis performed in a one-pass raster scan of the image. The result is a tree-like list of region records where links down the tree indicate nesting; i.e., if a region record is not a leaf node of the tree, then the children of the region are completely surrounded by it.

Both of the above systems use a global threshold to obtain the binary image. Each pixel of the image is thresholded at the same level. This approach works best in man-made environments such as a manufacturing area where scene parameters such as illumination and background composition can be controlled to insure high contrast images suitable for global thresholding. In a survey of threshold selection techniques, Weszka [1978] identifies three generic approaches to thresholding. One is global thresholding as defined above. Global thresholds are often selected by a user on the basis of experimentation to achieve the best results. Automatic global threshold selection usually involves analysis of the gray level histogram of the image to locate a well defined local minimum between two peaks, and setting the threshold at this gray level. Attempts to improve the results of histogram analysis include weighting the histogram on the basis of the response to a local operator such as gradient or Laplacian operators. The second approach is called local thresholding. In this case, the threshold is allowed to vary from pixel to pixel depending on some function of gray levels in a neighborhood of the pixel. The third approach is dynamic

threshold selection. Thresholds are chosen for a subset of image points using local threshold selection techniques. Thresholds at the remaining points are obtained by interpolation. Thus the threshold is a function of a pixel's location in the image. Otsu [1978] devised an improved threshold selection method that does not require the detection of a minimum in the histogram. This method is equivalent to fitting to the original picture by means of least-squares the two-valued (or multivalued for multiple thresholds) picture obtained by thresholding.

The above techniques can be extended to color images. Ohlander, et al. [1978] compute histograms for nine color parameters: red, green, blue; intensity, hue, saturation; and Y, I, Q. (These parameters are defined in Pratt [1978].) Regions are extracted by thresholding the parameter that exhibits the "best" histogram; i.e., a strong peak with well defined local minima on either side. This process is called region splitting. The algorithm is applied recursively to each region extracted in a previous iteration until no more regions can be split.

Ali et al. [1979] used chromaticity coordinates and normalized intensity to segment color photographs of airplane runway scenes. The chromaticity coordinates are obtained by dividing the intensity in each of the red, green and blue bands by the sum of all three intensities. The sum of the red, green and blue intensities is divided by the maximum possible total intensity (3 x 255 for 8-bit digitization in each band) to obtain normalized intensity. The interesting result of this work was the ability to segment runways as a single region in spite of shadowing since the normalized color coordinates are roughly independent of the shadows. Attempts to locate camouflaged airplanes in the same scenes were encouraging but not quite as successful.

For most scenes a simple threshold is inadequate. The region-splitting technique of Ohlander mentioned above performs better in some cases. Brice and Fennema [1970] use a region-merging technique. Starting with small, fairly uniform regions, two heuristics are used that merge regions so that the regions formed tend to be of simple shape and weak boundaries tend to be eliminated. Horowitz and Pavlidis [1974] use a split-and-merge procedure. It starts from an initial approximation to the desired segmentation, and proceeds both to split regions and to merge regions until the process stabilizes. In this way a better and faster segmentation can be achieved in some cases than with just splitting or merging.

It is possible to incorporate semantic information about the nature of the scene into the region segmentation process in order to produce a better segmentation than can be produced using picture information only. Yakimovsky and Feldman [1973] use a decision-theoretic region-merging approach. They attempt to maximize a probability based on the properties of the regions, such as color, and the properties of the boundaries between regions, such as their crude shape and orientation, and how these properties relate to their semantic interpretation. Heuristics are used in order to avoid an exhaustive search. Their technique has been used on natural outdoor scenes.

Ohta *et al.* [1978] produced a system using semantic knowledge about objects with substructures that is able to analyze outdoor scenes containing buildings. It produces a preliminary segmentation by means of a recursive thresholding technique similar to Ohlander's with local thresholds, and a data structure describing the relationship of the low-level features found. A plan is generated from this segmentation based on the larger lowest-level regions. The interpretation process uses the plan and a set of production rules which contain the semantic

knowledge. In this process some of the lower-level regions are remerged and an interpretation in terms of objects is assigned to each region.

The segmentation system developed by Tenenbaum and Barrow [1976] uses relaxation techniques (described in section 4.1) to iteratively refine the partitioning of an image into regions. At each step, beginning with a very elementary partitioning of single pixel regions or regions composed of a few pixels with identical attributes, the system performs the most complete interpretation of regions in the current partition. Based on this interpretation, a pair of regions is merged. The choice of this pair is based on minimizing the risk of merging regions that are not part of the same object. The risk is calculated in terms of the current interpretations of regions and relational constraints in a model of the scene. The new partitioning thus obtained is re-interpreted, and so on, until there are no safe merges available.

If the pixels contain range data instead of gray level or color information, regions in which the gradient is approximately constant are meaningful, since these correspond to planar surfaces. Milgram and Bjorklund [1980] determine such regions by first fitting local planes to small areas around each pixel by means of least squares. Then regions are grown from these according to how well adjacent planes agree and how small the residuals of the fit are.

## 2.4 Edge and Line Elements

An edge is a step in pixel values between two regions of relatively uniform values. The detection of edges is often an important step in the segmenting of scenes. (An alternative approach, the detection of regions, was discussed in the previous

section. Nevatia and Price [1978] compare these two methods.) For the present purposes, a line is defined as a thin region (perhaps only one pixel wide) of roughly uniform pixel value between two regions of roughly equal pixel values. A line is thus a double edge. An edge element or line element is a short (several pixels) length of an edge or line that can be assumed to be straight, even though the complete edge or line may be curved. One approach to the detection of edges and lines is the use of a local detector to find these short elements, which can then be linked together by higher-level methods. The term "edge detector" is used here to denote the local detectors, which are the subject of this subsection.

Even though an edge detector is a very low-level operator and the concept of an edge element or line element is fairly simple, many edge detectors have been proposed that differ in various ways that make a simple comparison difficult, and no one stands out as the best. Surveys of edge detectors have been provided by Davis [1975], Fram and Deutsch [1976], and Shaw [1979]. A few of the more popular or significant detectors are described below. Unless otherwise specified, these are designed to operate on monochromatic images and to detect edge elements only.

One of the most popular edge detectors was designed by Hueckel [1971]. It was later generalized to detect an edge-line combination, and other improvements were made (Hueckel [1973]). The Hueckel detector operates on a circular field several (typically nine) pixels in diameter. It attempts to fit to the data an ideal edge function consisting of constant brightness on each side of a perfectly sharp edge which can be at any position and orientation within the field. Thus four parameters are solved for (six in the generalized version which fits an edge-line combination). For speed, the operator only approximates a

least-squares fit by means of a set of orthogonal functions. Since the edge does not have to pass through the center of the field, the operator can be applied on a grid with sufficiently small spacing so that there is some overlap of the fields, instead of applying it centered on every pixel. This results in fairly good overall speed. However, the detection of off-center edges is somewhat degraded. Nevatia [1977] generalized the Hueckel operator to use color information.

Several edge detectors are based on the use of very small (two-by-two or three-by-three) weighting functions which are convolved with the input data to approximate the two components of the gradient, from which the magnitude and direction of the gradient can be computed. A sufficiently large magnitude is considered to represent an edge, but these points usually must be thinned if a one-pixel thick edge is desired. A popular detector of this type is the Sobel operator (described by Duda and Hart [1973]). In the elementary form of these operators as stated above, they are fast but are quite susceptible to noise. To improve their noise rejection (at the cost of less speed and resolution) their weighting functions are sometimes spread out by applying each weight to the average of several pixels in a square area, as described by Shaw [1979].

Frei and Chen [1977] use a variation on the method of the previous paragraph. They first find the magnitude of the gradient by means of a three-by-three operator. Next they determine how well this fits an ideal line by using an orthogonal set of three-by-three functions, and then threshold the result according to the goodness of fit instead of by the magnitude.

Nevatia and Babu [1979] designed an edge detector that convolves the image with a set of ideal edge masks several pixels wide, each of which has a different edge direction. (In

practice, five-by-five masks with orientation every 30° have been used.) The mask orientation that produces the highest output at each pixel is considered to be the edge orientation for that pixel. However, an edge is reported at a pixel only if its edge magnitude is a maximum along the normal to the edge direction, its edge direction agrees approximately with its neighbors, and its magnitude exceeds a threshold. This edge detector has been used on distance data by Inokuchi and Nevatia [1980].

Marr and Hildreth [1979] note that different edges are found depending upon the size of the edge mask. To capitalize on this, they use information from several spatial frequency channels by convolving the original image with Gaussian smoothing filters of various sizes. Edges are located by finding the zero crossing of the second spatial derivative of the smoothed image. Computationally, this amounts to finding the zero crossings in the convolution of the original image with the Laplacian, $\nabla^2$, of the Gaussian smoothing filter for each spatial frequency channel used. This transformation contains nearly all of the information present in the original image. Edges are said to occur where the zero crossings from several spatial frequency channels concur.

All of the above edge detectors perform satisfactorily on high-quality images. Of course, perfect results cannot be expected because of the imperfections in real images, so the higher-level processing must be able to handle occasional errors. However, on poor-quality images the performance of these detectors degrades in different ways depending on the amount of image noise, blurring of edges, faintness of edges, and smooth variations in pixel values superimposed on the edges. It appears that none of them is the last word in edge detectors.

## 2.5 Curves

Long edges or lines can be found either by using an edge detector as discussed in the previous section and linking these into a long smooth curve, filling in gaps and ignoring stray elements, or by a procedure which accomplishes a similar result by operating directly on the image data, bypassing the need for an edge detector. In either case, the algorithm can operate sequentially by proceeding along the curve as it links edge elements or pixels, in which case it often is called a line follower (or tracker), edge follower, or curve follower, or it can operate on an effectively parallel or gestalt basis.

Several investigators have used sequential techniques that link edge or line elements. For example, Shirai [1975] used a pair of parameters that vary according to how continuously and smoothly the elements are being found. These parameters determine thresholds for deciding when to accept a new element according to how close it lies to the linear continuation of the current tracking and when to stop the tracking. Roberts [1965] used an elaborate line-finding method that contained elements of this kind of technique.

Martelli [1976] used a global heuristic search instead of a local search. His method operates directly on the brightness values instead of using a separate edge detector. It attempts to optimize a cost function that depends on the curvature and the degree to which the curve separates regions of different brightness. Yachida *et al.* [1979] used a similar method based on the output of a local edge detector.

Kelly [1971] devised a method in which edges found in a low-resolution version of the picture and selected according to global knowledge of the shape being sought are used to form a

plan, which is then used for linking the edge elements in the full-resolution picture.

Eberlein [1976] proposed a relaxation method (see Section 4) for linking edges found by a local detector. This is an effectively parallel method. On each iteration the strength of each local edge element is changed according to how well it agrees with its neighbors, until the process converges to a thin continuous line.

When curves are derived from edge data, a useful preprocessing step is thinning. Thinning algorithms reduce contours to a single-pixel width by discarding redundant edges while maintaining the global connectivity of all contours. Some methods, such as Eberlein's just mentioned, include thinning as an inherent part of the operation. Stefanelli and Rosenfeld [1971] describe a thinning algorithm for binary images which decides the fate of each edge based on the states of its eight nearest neighors in a 3 by 3 window. Nevatia and Babu [1979] perform thinning by accepting the edge which has a maximal gradient value compared to adjacent pixels with similar gradient orientation, as mentioned in Section 2.4.

Hough [1962] proposed a global parallel method for finding straight lines, which was improved and extended to other curves by Duda and Hart [1972]. In this method the desired curve is represented by a few parameters. (For a straight line, two parameters are needed, for which the angle of its direction and its normal distance from the origin are recommended.) Each point in the image that is a candidate for being on the curve (by producing significant magnitude from an edge detector, for example) is transformed into a curve in the parameter space. The parameter space is quantized, and each cell accumulates the total number (or total edge magnitude or other weight measure) of

transformed points that pass through it. A large peak in the resulting histogram in the parameter space then represents a curve in the original image space. This is a fast method for finding curves that require only a few parameters, if the required accuracy is not too high. However, if it is necessary to quantize the parameters very finely or if the number of parameters is more than about three, the number of cells becomes very large, resulting in the need for much computing and a large storage space. The computing problem can be alleviated somewhat if the directional information from the edge detector that produced the points is used to restrict the number of cells incremented for each point. However, since this local directional information is seldom accurate, it is still usually necessary to increment cells corresponding to a band of directions. These matters are discussed by Wechsler and Sklansky [1977], among others.

Even though the relatively global knowledge about the shape of the curve (whether its precise shape or just its smoothness) that is used in the above methods tends to reduce the errors made on the basis of only local evidence, perfection cannot be expected with real images. Each higher level of processing must be able to tolerate the errors from the lower levels and hopefully to filter out some of them, in order to reduce the burden on the yet higher levels.

## 2.6 Chain Code

Chain code is a compact representation of region boundaries or, more generally, of any line structure in an image. A chain-coded boundary record consists of a header containing the image coordinates of the starting point and the length of the boundary, followed by a list of chain links, or vectors, which represent the boundary as a sequence of moves from boundary point to

boundary point. Several chain-coding schemes are described in Freeman [1974]. In its most typical form, there are eight possible vectors corresponding to links between a point and each of its eight nearest neighbors in a three-by-three window. The vector to each neighbor position is assigned a unique number from zero to seven. Thus, in its most compact form, each chain element requires only three bits of storage.

Chain code is of limited usefulness in modelling objects for pattern recognition since it is difficult to compute general rotations or scale changes required for matching. It can be useful, however, as an intermediate-level description of the image from which useful features are extracted. Wilf and Cunningham [1979] describe a boundary traversal algorithm for computing region moments from a chain-coded boundary. The moments can be used to derive the area, centroid, and orientation (axis of minimum moment of inertia) of a region, among other things. Freeman and Davis [1977] describe an algorithm for locating corners in line drawings represented by chain code. (See Section 2.7) Chain code can also serve as a starting point for a higher-level boundary description consisting, for example, of arbitrary length line segments and circular arcs. Freeman [1974] describes other chain-code manipulations such as smoothing, rotation, and correlation matching.

## 2.7 Corners

As an alternative to more or less straight lines (including edges) or in addition to these, corners may be useful as features to be used by higher-level processes. A corner in a two-dimensional image can be defined as a point (within the resolution of the image) from which two or more lines emanate at various angles. If there are only two, the corner is merely an abrupt change in the direction of the curve.

In the latter case, one possibility for detecting the corner is as an integral part of the curve-fitting process. Martelli [1976] included such an ability in his method mentioned in Section 2.5. Duda and Hart [1973] discuss an iterative end-point fit method for doing this when straight lines are being fit by means of least squares.

Once a curve has been followed with only very local (if any) constraints, it is possible to detect corners on the curve. (After this has been done, the segments of the curve can be smoothed if desired.) Cederberg [1978] detects points of maximum curvature in order to locate the corners, by using a recursive smoothing filter technique designed to be implemented on a cellular processor. Kruse and Rao [1978] detect corners by means of a matched filter operating on the second derivative of the length of a chord connecting two points that are a constant distance apart along the curve.

Freeman and Davis [1977] detect corners in chain-coded curves by calculating the slopes of chords connecting edge elements i and i + n for some arbitrarily chosen constant n. By looking at the slope of these chords as a function of i, the position on the curve, corners are detected on the basis of an abrupt change in the slope.

A similar approach is used by Rosenfeld and Weszka [1975]. In their algorithm several chords are defined for each edge element i by connecting edges i + k and i - k for some range of values of k. By analyzing the slopes of the set of chords thus obtained, the edge is classified as a corner or non-corner.

Perkins and Binford [1973] devised a method for finding a corner of two or three lines when it is known approximately where

the corner should appear in the image. The expected area is searched for straight lines which approximately match the expected lines from the model. Then a corner is found as the intersection of these lines. Several refinements in the method are included to improve its reliability.

## 2.8 Pyramids and Quadtrees

A pyramid data structure represents an image at several levels of resolution simultaneously. The base of the pyramid is the original full resolution image, usually assumed to be a $2^n$ by $2^n$ square array. The next level of the pyramid is formed by partitioning the original image into nonoverlapping 2 by 2 cells and mapping the four pixels in each cell to a single pixel in the next level. Various mappings are possible. Examples are the average gray level of a cell, the minimum or maximum value of a cell, or the output of an edge detector applied to the cell. The complete pyramid is formed by repeating the process at each level until the image has been compressed to a single pixel at the highest level. The result is a set of images of sizes $2^n$ by $2^n$, $2^{n-1}$ by $2^{n-1}$, ... , 2 by 2, 1 by 1, each representing the same scene at different resolutions. The pyramid representation can be generalized by defining an arbitrary n by m partitioning at each level.

The usefulness of pyramids lies in being able to extract features at an appropriate level of resolution. This simplifies recognition by reducing the search space (image array size) and suppressing unnecessary details. Generally speaking, gross features can be extracted from high levels of the pyramid. Finer detail can be extracted where necessary from lower levels. As an example, the results of edge detection or region growing in a high level of the pyramid can be used to constrain the search in lower levels as object descriptions are refined in higher

resolution. Uhr [1972] proposed a "recognition cone" model for image analysis in which successive operations produce abstracted or simplified versions of the original image at increasingly lower resolutions. Tanimoto and Pavlidis [1975] used a pyramid obtained by averaging 2 by 2 blocks. Pyramids are also central to the scene analysis work of Hanson and Riseman [1978b] and Levine [1978].

The quadtree representation of a $2^n$ by $2^n$ image is obtained in a top-down manner by recursively splitting the image into quadrants, the quadrants into subquadrants, and so on. The process continues until the quadrants are one pixel in size, or all pixels in a quadrant are uniform with respect to some feature. The result is a tree where each non-terminal node has four children, and terminal nodes represent square uniform regions of the image. In a binary image, this means that each terminal node is a block of all white or all black pixels. Whereas a pyramid represents the image at multiple resolutions, a quadtree is a variable resolution image, representing each area of the image by the largest square region possible. Samet and Rosenfeld [1980] have adapted several standard binary image processing algorithms to operate on quadtrees, such as boundary tracking, connectivity analysis, genus (number of holes) computation, and extraction of features such as area, moments, and perimeter. They have also developed efficient tree traversal algorithms which do quadtree/raster and quadtree/chain code conversions.

For a recent survey of image analysis techniques using pyramids and quadtrees see Rosenfeld [1980b].

# 3. DESCRIPTION

## 3.1 General Information

Ways of describing a scene or object and of deriving the description from an image will now be discussed. We are concerned here with fairly high-level descriptions, as opposed to the lower-level concerns of Section 2. At these higher levels, it becomes more difficult to judge what the best approaches are. As a result a wide variety of techniques has been used. We can provide here only a cursory view of some of the more important work. For more information the reader can consult the surveys by Pavlidis [1978], Shirai [1978a], Bajcsy [1980], and Barrow and Tenenbaum [1981].

## 3.2 The Two-Dimensional World

This section discusses descriptions that are basically two-dimensional. These may be used for planar surfaces contained within a three-dimensional scene, or they may be used for projections of flat three-dimensional objects of known orientation relative to the camera axis.

When an image has been segmented by the techniques described in Section 2.3, a description of each region, or blob, can be generated consisting of a list of statistical features. These features typically include area, perimeter, first and second order moments, color, etc. The individual blob descriptors are linked to form a tree data structure which represents nesting relationships. The parent of any blob in the tree is the adjacent blob which completely surrounds it. The SRI Vision Module (Nitzan *et al.* [1979]) and CONSIGHT (Ward *et al.* [1979]) use this approach. Milgram [1979] presents the details of a one-

pass algorithm which constructs a blob tree description of a binary image.

Perkins [1978] describes objects as a set of "concurves." A concurve is an ordered set of straight line segments and circular arcs which approximate the boundary of an object. An object such as a connecting rod for an automobile is modelled as three concurves, one for the outer boundary and one each for the crankshaft and piston pin holes, respectively. Concurves associated with objects in an image are derived from edges. If the resulting concurves form closed boundaries, statistical features similar to those described above are computed for the enclosed region. Also, associated with each concurve is a number describing its rotational symmetry. The main advantage of the concurve representation is that objects may be recognized on the basis of partial views by mate ing a subset of the lines and arcs in a model concurve with the image data.

Shapiro [1979] surveys data structures used for description and pattern recognition. The paper concludes with a discussion of a recursive data structure for representing line drawings. A PICTURE is constructed as a result of evaluating a picture expression. A picture expression consists of primitives (LINE, ARC, CIRCLE, SQUARE, etc.) and possibly other pictures, with provisions for specifying the relative or absolute position and orientation of various components of the image.

Blum [1967] proposed a method of representing planar regions, known as the medial-axis transformation or prairie-fire transformation. In this method a region is described by a skeleton which is the locus of points equidistant from the boundaries of the region on each side of the axis, and by the value of this distance for each point on the skeleton.

Nevatia and Price [1978] describe two-dimensional scenes by means of a graph structure in which regions and lines are the nodes and the positional relationships between them are the arcs.

Rosenberg et al. [1978] produce a relative depth map for regions in a two-dimensional view by using heuristics that indicate the occlusion relationships among these regions. A scene is first segmented into regions by some technique as discussed in Section 2.3. Heuristics are used to indicate which regions may be occluding other regions. Then a probabilistic relaxation process (see Section 4) is used to resolve the contradictions that the heuristics have produced.

## 3.3 The Blocks World

In this section three-dimensional objects are considered, but they are restricted to simple polyhedra with uniform surface reflectance and usually diffuse illumination.

Roberts [1965] produced perhaps the first program for analyzing three-dimensional scenes. This method first extracts a line drawing from a picture as described in Section 2.5. This line drawing must be a topologically perfect projection of objects made of three simple geometrical models (a cube, a wedge, and a hexagonal prism) that can be stretched in each dimension, rotated, and translated. The line drawing is then matched to the models one at a time by comparing the polygons intersecting at a point in the picture to the faces of the models, until the entire scene is described in terms of the models. The distance and hence the size of each object is obtained by assuming that each object is supported by another object seen or by the assumed ground plane.

Guzman [1968] produced a program that analyzes a perfect

line drawing that represents a projection of arbitrary polyhedra. By means of heuristics it groups the polygons in the line drawing into objects, without the use of any object models other than the knowledge that the objects are polyhedra. The program usually does quite well on complicated scenes including occlusion, but the fact that it requires perfect line drawings is a serious limitation. Brice and Fennema [1970] used a technique similar to Guzman's together with some semantic knowledge to identify the floor and walls in a room scene. Then some heuristics were used to insert missing lines. Falk [1972] extended Guzman's heuristics and used a set of nine fixed-size three-dimensional models in terms of which the scene is interpreted, so that perfect line drawings are not necessary. Grape [1973] used two-dimensional models of edge structures for convex objects to identify missing lines.

Huffman [1971] and Clowes [1971] eliminated the need for heuristics in interpreting perfect line drawings. They recognized that each line in the picture represented either a convex edge, a concave edge, or an occluding edge in the three-dimensional scene, and they constructed a catalog of possible vertices with allowable line labellings. A scene can then be analyzed by starting at one vertex and proceeding through the line drawing performing a tree search, limiting the number of possible line labellings at each step according to the catalog, until a consistent labelling for the entire scene is obtained. Waltz [1975] extended this work by including shadows and cracks. He produced a catalog of a few thousand possible vertex types, and used a relaxation-type procedure (see Section 4) to decide on the correct labelling for each line according to the possibilities in the catalog. This procedure converges rapidly (usually to a unique interpretation) regardless of the complexity of the scene. Waltz also included a limited ability to handle imperfect line drawings by including in the catalog some of the

most common cases of missing edges. Freuder [1980] showed how simpler catalogs of vertices could be used together with occasional examination of the scene to obtain more information.

Shirai [1975] analyzed scenes of polyhedra by first finding the lines separating the objects from the background by using the assumption that these are high-contrast edges. Then the generally fainter edges separating objects or faces of objects are hypothesized by means of heuristics and searched for in the image. A heterarchical structure is used in the program, rather than the usual hierarchical structure. That is, it is organized as a community of experts that communicate with one another.

Winston [1975b] produced a program that, given a line drawing representing polyhedra, produces a description of the scene in terms of a network of objects and their relations such as "supports," "above," "left of," "in front of," and so forth. A method similar to Guzman's is used to segment the scene into objects. Then some rules and heuristics are used to derive the relations. In this process groups of objects are formed by a process of conjecture, criticism, and revision. The conjectures find objects linked by relation chains or bearing the same relation to some common object. Then the criticism and revision delete from a group objects whose membership is weak compared to the average for the group. The resulting description networks are used by his learning system described in Section 7.

### 3.4 The More General World

In this section more complicated objects and scenes, often with curved surfaces, are considered. Although some of these techniques have a degree of generality and have been used for recognition as described in Section 4, they still fall short of what is needed for a general, powerful vision system. For

example, two tasks that are beyond the capability of any existing computer vision system are the recognition of parts in a jumble in a bin and the operation of a robot vehicle in a complicated outdoor environment.

Marr [1978] described three levels of representation. From lowest to highest, these are the primal sketch, the 2-1/2-D sketch, and the 3-D model. The primal sketch is in iconic (image) form, but it makes information about the location of lines and edges explicit. The 2-1/2-D sketch is also iconic, but it represents depth information and surface orientation relative to the viewer, and it makes depth discontinuities explicit. The 3-D model is an object-centered representation that describes the object in a convenient way, perhaps in terms of generalized cones (described below).

Ohta *et al.* [1978] use a semantic network for describing outdoor scenes containing buildings. The network consists of a hierarchical structure describing part-whole relationships, two-dimensional positional relationships, and properties such as color. The method of segmenting the scene to produce this structure was described in Section 2.3.

Barrow and Tenenbaum [1980] propose a method for interpreting curved line drawings as three-dimensional surfaces. To interpret a two-dimensional curve they compute a three-dimensional curve projecting to it that minimizes a combination of variation in curvature and departure from planarity. For example, an ellipse would be interpreted as a circle, since a circle has constant curvature and is planar. To interpolate surfaces between boundaries, they attempt to make the two observable components of the surface normal vary as linearly as possible relative to the image coordinates.

Binford introduced the concept of generalized cones (also known as generalized cylinders) as a means of representing three-dimensional objects. (See Agin and Binford [1973].) A generalized cone is defined by a space curve, called the spine or axis, and planar cross sections normal to the spine. The function which describes how the cross section changes along the axis is called the sweeping rule or cross-section function. Generalized cones are useful for describing three-dimensional solids whose cross sections change smoothly along an axis, especially elongated solids. Complicated objects often can be broken down into parts of this nature.

Agin and Binford [1973] fit generalized cones to portions of objects by using three-dimensional data. The spine of a generalized cone was represented merely by a list of points. The cross sections were circles, whose radii were a linear function of the position along the axis.

Nevatia and Binford [1977] derive descriptions of complicated articulated curved objects in terms of generalized cones. They use three-dimensional data, but only the boundaries of the object as seen from the camera (the depth discontinuities) are used. Initial approximations for the axes are formed by using the midpoints of the intersections of the boundaries with evenly spaced lines with about eight different orientations. Then an iterative process finds cross sections normal to a straight line fit to axis points at the centers of these cross sections. The axes are extended in both directions, and a new straight-line fit is started when a new axis point deviates from the old fit by more than a threshold. A large jump in cross section denotes the end of the generalized cone. The possibility of multiple representations of the same piece of object is eliminated by using the cone with the longest axis. Summary descriptions for each piece of the object represented by a

generalized cone are produced by computing the length of axis, the average cross-section width, and the cone angle corresponding to a linear fit to the cross-section function. The joints where two or more pieces of a object connect are determined. The object description then consists of the connectivity relations of the pieces and joints, which is equivalent to a graph structure; the summary descriptions of each piece; and some summary information about the object, including the number of pieces, number of elongated pieces, number of joints, and information about pieces distinguished by their large width or elongation. This description is used for matching to an object model as described in Section 4.

In ACRONYM (Brooks et al. [1979], Brooks and Binford [1980], and Binford et al. [1980]) two-dimensional scenes are described in terms of ribbons, which are pairs of roughly parallel edges, and the spatial relationships among the ribbons. The ribbons are produced by a rule-based system which links edge elements by means of a best-first heuristic search. Three-dimensional objects are modelled in terms of structures composed of generalized cones and their spatial relationships. There exist multiple levels of representation of objects, from coarse to fine. The object model is a graph, subgraphs represent parts and subparts of the object, and so on to the individual generalized cones. The particular generalized cones in ACRONYM use a cross section whose boundary can be decomposed into straight-line segments and circular arcs, a sweeping rule which is continuous and piecewise linear, and a spine which is continuous and composed of straight line segments or, under some restrictions on the cross section and sweeping rule, circular arcs. This is a more general class of generalized cones than is usually used in other systems.

Woodham [1979b] showed that the shapes of some surfaces,

including a subset of generalized cones, can be determined from the shading (brightness) information in a single view, if the reflectance properties of the surface are constant.

Baker [1977] describes irregular three-dimensional objects by approximating their surfaces with circular-arc wire-frame models. The vertices of the model correspond to points where the surface curvature changes significantly. Burr and Chien [1977] use piecewise-linear wire-frame models in which the wire frame corresponds to edges in brightness, usually caused by illumination effects at intersections of planar surfaces of the object. They obtain depth information by means of stereo vision and match the perceived edges to a pre-stored model.

The geometric modelling system developed at IBM by Wesley et al. [1980] enables the user to describe complicated three-dimensional objects such as mechanical parts. The volume primitives stored internally are polyhedra, which can be combined as needed by the operations of union, intersection, and difference. Curved objects are approximated by high-order polyhedra. Objects and assemblies are represented in a graph structure that indicates part-whole relationships, attachment, constraint, and assembly. Physical properties of objects and positional relationships between objects are also included. The relevance to computer vision is that the system can determine the appearance of an object for an arbitrary view. This information could be used by a recognition system to guide the search for features to match an image to the model.

Shapiro et al. [1980] describe objects in terms of three primitive types of shape and the relationship between these primitive parts in the object. The three primitives are sticks, plates, and blobs, which are meant to approximate roughly the parts of the object with significant extent in one, two, or three

dimensions, respectively. The relations show how the parts connect, indicate their spatial relationships, and limit the sizes of the parts. Some global information about the object is summarized in a numeric vector. This summary information can be used to find likely candidates for matching with the full relational model when recognition is attempted.

Gennery [1980] produced a method of describing three-dimensional natural outdoor scenes. The ground surface is approximated by one or more planes or paraboloids, and objects lying on the ground are approximated by ellipsoids. The method derives this description from three-dimensional data in the form of points densely spaced over the scene (such as might be produced by the stereo techniques described in Section 6). The ground surface is found first by a process which finds a set of points that form a well-defined surface such that there are few points well below it, since these lower points most likely would represent errors. However, many points above the surface can be tolerated, since these may lie on objects. Then the points sufficiently above the computed ground are clustered into tentative objects. Ellipsoids are fit to these clusters in such a way as to use the information that points of any kind should not be hidden from the camera by an object and to tolerate occasional incorrect points. In this process clusters of points are split and merged as needed to produce the most reasonable fits. This form of description was devised for describing the surface of Mars for a roving vehicle; thus the objects would be rocks. For man-made objects, the ellipsoidal representation seldom would be suitable. However, the technique for finding the ground might be suitable for finding the planar surfaces of man-made objects in some cases.

Minsky [1975] proposed the concept of "frames" as a way of representing knowledge. A frame is a data structure for

representing a stereotyped situation. The frame includes information about how to use the frame, what can be expected to happen next, and what to do if these expectations are not met. The frame contains slots, which represent variables to be instantiated when the frame is used. The entities to fill these slots often are other frames. An important feature is the fact that each slot has a default assignment, which is replaced only when specific information overrides it. Related frames are linked into frame systems. The frame concept is meant to apply to the whole field of artificial intelligence. In vision, frames would be useful in recognition; in other areas, they would be used in reasoning and in understanding discourse. Some non-visual systems using frames have been implemented, for example KRL by Bobrow and Winograd [1977].

# 4. RECOGNITION

## 4.1 General Recognition Methods

The process of recognition consists of matching a description derived from an image to a description of a stored model, perhaps chosen out of a large number of possible models according to the best match. This process can occur at many levels in the analysis of a scene. At the highest levels, the descriptions to be matched are those of entire objects or scenes, as described in Section 3.

The simplest form of matching is correlation, in which the cross-correlation coefficient or a similar mathematical function is maximized. For binary data this reduces to template matching. Such a method usually is suitable only for the lowest levels of a vision task. For example, some of the edge detectors described in Section 2.4 and some of the stereo techniques described in Section 6 use this form of matching.

A slightly more elaborate form of recognition is statistical pattern classification. In this method, numerical values for a set of features are measured. The scene is classified according to where the vector of values lies in a multidimensional feature space. This method is suitable only for very specialized vision tasks or for minor parts of elaborate vision systems. Such techniques are described by Duda and Hart [1973].

Relaxation is a method of selecting appropriate labels for a set of interrelated units. In a recognition task, the units would be features extracted from an image, and the labels would be the corresponding features in an object model. Two basic forms of relaxation exist: discrete and probabilistic (or

continuous). In discrete relaxation, a set of possible labels is initially associated with each unit. On each iteration, labels are discarded for a unit if they are inconsistent with all of the remaining labels on related units. In probabilistic relaxation, each label associated with each unit is assigned an initial correctness probability estimate. On each iteration, the probabilities are adjusted as a function of the label probabilities on related units, according to given compatibility functions. In either case, on each iteration all of the adjustments are assumed to be done simultaneously, using the old values for the related units. Thus the method is well suited for parallel computation. In many cases the process converges after a few iterations. Relaxation has been used in some low-level and intermediate-level vision tasks and may be useful in high-level tasks also. Zucker [1976b] describes the basic principles of relaxation, Rosenfeld [1978b] surveys some of the work on relaxation, Hummel and Zucker [1980] discuss its theoretical foundations, Yamamoto [1979] discusses the derivation of the compatibility functions, Faugeras [1980] describes some improvements to the basic method, and Ullman [1979] shows how to perform constrained optimization by means of relaxation.

One approach to recognition is syntactic analysis. In this approach a formal language is devised corresponding to the model of the scene or object. The syntax of this language defines the hierarchical structure of the model. Fu and Swain [1969], Pavlidis [1977], and Rosenfeld [1979b] discuss such methods. The advantage of this approach is that a parser, which does the matching, is independent of the knowledge in the models. This makes it easy to change the models or to make other modifications. However, applying such methods to more than one dimension and allowing for uncertainty have proven difficult.

At the highest levels, most recognition methods that have

been used do not fall strictly into any of the above categories, although elements of some of them often appear. Usually some kind of heuristic search is performed, in which features are matched one at a time to produce a tree structure which must be searched. Heuristic search is a common task in artificial intelligence. General search methods are discussed by Nilsson [1980]. Very often the recognition process involves the matching of graph structures. The relevant properties of graphs are discussed by Pavlidis [1977].

For more information see Barrow and Tenenbaum [1981].

## 4.2 Particular Recognition Efforts

This section discusses some recognition programs and proposals that operate at a fairly high level.

A pattern classifier capable of limited two-dimensional position, rotation, and distortion invariant recognition is the recognition (Fukushima [1975] and Fukushima and Miyake [1980]), in which feature detectors are self-organized in a network through unsupervised learning. (See Section 7.) This method is computationally extremely expensive.

Barrow *et al.* [1972] discuss ways of recognizing objects by matching relational structures, which are graphs showing the relationships between features in the scene or object model. They propose the hierarchical synthesis method, in which the object model is broken into substructures which are searched for in the scene. Then combinations of the substructures are found, and so on through as many levels as necessary.

Fischler and Elschlager [1973] describe a way of performing "rubber-sheet" matching of two-dimensional structures. The

structure consists of features connected by "springs," and the matching process attempts to find a matching of the features that minimizes a function of the stretching of the springs. They describe a dynamic programming method that finds the optimum match and a "linear embedding algorithm" that is much faster but is not guaranteed to be optimum.

Price [1976] produced a method of matching two symbolic scene descriptions. Elements of the scene are matched one at a time. For each element in one scene the best match is selected according to a minimization of the weighted sum of absolute values of differences of feature values.

Gennery [1980] produced a method of matching scene descriptions consisting of sets of feature vectors with estimated uncertainties. It is assumed that the transformation between the scenes is known except for a few parameters (such as translation and rotation, for example). The method performs a search by sequentially matching the features of one scene to those of the other, solving for the transformation parameters, computing the probabilities of these matches by means of Bayes' theorem, and using these probabilities to prune the search. The method was devised to match the scene descriptions consisting of ellipsoidal objects suitable for the Martian surface, as described in Section 3. In that case each feature vector describes an ellipsoid. However, with different feature vectors (perhaps corners) the method may be suitable for matching known man-made objects.

The world model of Ballard et al. [1978] includes information about the visual characteristics of objects. This intermediate stage between the image and the world model is a "sketch map," in which instantiations of elements of the world model are explicitly correlated with features of the current image, with accompanying location descriptors. Recognition is

said to occur when, based on correlation of features in the image with stored world knowledge, a model of the scene is constructed in the sketch map from elements of the total world model.

Hanson and Riseman [1978c] use several intermediate stages of image processing as they progress to a high-level, symbolic representation. They use the concept of "schema" (similar to Minsky's Frames described in Section 3) in their highest level, by which they exploit the fact that certain objects and features are often found together.

Bolles [1980] uses the local feature focus method to locate occluded two-dimensional objects. In this method a reliable (focus) feature of the object is located first, and the secondary features are located relative to it to identify the focus feature and to determine the position and orientation of the object. Bolles determines the focus features and their secondary features automatically in a training-time computation in which the program analyzes a model of the object to determine those features that can most reliably and cheaply determine the location of the object.

Neumann [1978] performs recognition of two-dimensional objects with occlusion. The outlines of the objects are used, represented by straight-line segments. A search for best match is done among tentative matches based on corners. The match to a given object model can vary in position, orientation, and scale factor.

Perkins [1978] relies on explicit shape matching to recognize industrial parts. This is done by transforming a model concurve to bring it into registration with an image concurve (described in Section 3.2). Candidate concurves are selected on the basis of gross features such as area, perimeter and other

statistical values (if the image concurve is closed, allowing similar quantities to be computed from the image), the number and type of components that make up the concurves, and symmetry. The analysis is 2-dimensional -- objects are constrained to lie in a plane at a fixed distance from the camera -- so the transformation of the model is expressed as (x, y, Θ), representing a 2-D translation and rotation. Matching based on partial views is done in a similar manner by trying various subsets of model concurve components. After the transformation of the model is determined, a global measure of the goodness of the match is obtained by computing the distance between the model and image concurves at selected points along the boundary. This system is able to correctly identify several overlapping parts, and it tolerates fairly high levels of noise.

The SRI Vision Module (Nitzan et al. [1979]) uses two matching techniques to identify blobs. Both methods require that position uncertainty is limited to 2-D translation and rotation in a plane parallel to the image plane and that objects are entirely in the field of view without touching any other objects. One is the nearest neighbor method. A set of n features, each of which can be expressed as a single number, is chosen to classify objects. Each object in the model is represented as an ordered n-tuple of feature values which can be thought of as the coordinates of a point in an n-dimensional space. A blob in the image is compared to each object in the model by computing the distance between the n-tuple extracted from the blob and each model point. The blob is classified as an instance of the object corresponding to the nearest model point, if the distance is within some tolerance set by the expected variation of feature values. Otherwise, the object is rejected as unknown. This is also the method used in CONSIGHT (Ward et al. [1979]).

The second method used by SRI (Agin and Duda [1975]) is the

binary decision tree method. Non-terminal nodes of the binary tree specify a feature to be tested and a threshold to determine branching. Single features are tested sequentially, with each test reducing the number of possible classifications until a terminal node is reached which represents the desired classification. The method is optimal given certain assumptions about the distributions of the feature values being used. The only drawback to this method (if the assumptions are met) is that it cannot reject unrecognizable objects without resorting to computing the distance between the blob n-tuple of features and the n-tuple corresponding to the object represented by the terminal node.

Vamos et al. [1979] produced a system which can recognize simple industrial parts with arbitrary three-dimensional orientation. Their system detects edges in a two-dimensional view and assembles these into line segments and arcs. Probabilities are assigned to these according to the strength of evidence. The resulting description is matched to wire frame models (including hidden-line elimination) in the data base by means of a heuristic search. It is intended that the system be able to recognize an object out of ten or twenty possible object models.

Shirai [1978b] uses an iterative approach to recognize objects. The basic processing sequence consists of edge detection, curve fitting (straight lines and ellipses), and recognition. The first iteration recognizes as many objects as possible using a conservative edge detection threshold. Subsequent iterations obtain more edges by lowering the threshold, and use previous recognition results in combination with relational constraints in the model to recognize new objects or additional parts of partially recognized objects.

Nevatia and Binford [1977] produced recognition of complicated articulated curved objects, using the description in terms of generalized cones described in Section 3. First, in order to avoid a lengthy, detailed comparison with all models in a large data base, a description code summarizing some important features of the description is used to index into the data base to find a few models with similar description codes. In practice, description codes based on the distinguished pieces are used. The descriptors used are the connectivity of the distinguished piece, whether it is distinguished because of its length or because of its width, and whether its cone angle exceeds a threshold. Then a detailed match against each model retrieved by indexing is performed, so that the best matching model can be chosen. In this process, similar distinguished pieces are tentatively matched first. Then the match is grown to include other pieces, according to the connectivity relations and allowing for missing pieces. In this way a tree search is performed. The best match is chosen based on how well the connectivity relations are preserved and how well the summary descriptions of the matched pieces agree. In test cases where the data base of models consisted of a doll, a toy horse, a toy snake, a glove, and a ring the system usually recognized the objects correctly even when multiple objects were present in the scene, the limbs were variously articulated, and moderate amounts of occlusion were present. The computer time required for description and recognition of a typical scene was from five to ten minutes on a DEC KA-10 processor.

In ACRONYM (Brooks et al. [1979], Brooks and Binford [1980], and Binford et al. [1980]) an image-derived description based on ribbons is matched to a model based on generalized cones. (See Sections 1.3 and 3.4) A predictor and planner produces a two-dimensional prediction graph from the model, and an edge mapper produces an observation graph from the image. The matcher

matches the two graphs by finding globally consistent subsets of local matches, invoking the predictor and planner again where necessary. The matching process is mapped back to three-dimensional models to ensure global consistency. ACRONYM is able to detect multiple objects in a scene, where each object is an instantiation of a generic object model that has been given to the system as described in Section 7. In early tests of the incomplete system it has located aircraft in an aerial photograph of an airport.

# 5. TRACKING AND VERIFICATION

## 5.1 Tracking

This section is devoted primarily to object tracking. A discussion of verification vision is included because the approach to object recognition in both cases is similar. The goal of object tracking is to process sequences of images in real time to describe the motion of one or more objects in a scene. Often real time implies processing every image from a TV camera operating at 30 Hz. In other words, an image is digitized, features are extracted from the image, the object or objects are located in the image, and position and velocity estimates are updated 30 times a second, although in practice slightly slower rates are sometimes used. At the present time, the approaches which achieve real-time operation rely on simplifying assumptions about the nature of the scene, track very few objects in a given scene, and incorporate varying levels of special-purpose hardware designed for the particular tracking algorithm.

The field of object tracking has been surveyed by Nagel [1978] and Martin and Aggarwal [1978]. Real-time tracking programs have been developed for a variety of applications. Griffin *et al.* [1978] use an object-tracking program to provide feedback for closed-loop guidance of a breadboard Mars-rover vehicle. Gilbert *et al.* [1980] developed a system for real time identification and tracking of missiles and aircraft. Pinkney [1978] describes a one-camera system which tracks four artificial markers on an object to control a manipulator visually as it approaches an object to be grasped. The intended application is to the manipulator on the Space Shuttle. A similar approach using stereo cameras is proposed in Brooks [1980] for supervisory control of a teleoperator manipulator. Chien and Jones [1975] reported on the use of real-time tracking to aid in stacking

blocks with a manipulator and inserting a peg in a hole. SRI (Nitzan et al. [1979]) is investigating using their vision module to track objects, such as a part which is suspended from an overhead conveyor, for feedback to an industrial manipulator. Tsugawa et al. [1979] describe a real-time stereo vision system for detecting certain road conditions to operate an automobile autonomously.

The fundamental problem in object tracking is to devise a robust matching algorithm which is able to repeatedly recognize the same object or objects in successive images, and is computationally feasible; i.e., the algorithm must execute in approximately 1/30 second. In general, matching algorithms conduct a search in a window believed to contain the object, looking for the best registration between image features extracted from the window and features associated with an internal model of the object. The very nature of object tracking simplifies this problem to a considerable extent. However, it is still by no means trivial. Since successive images are only 1/30 second apart in time, the appearance of the object will change very little from image to image. The object can be modelled adaptively as it was last seen by the tracker, with the expectation that a good match between the object model and the features in the current image is available. Furthermore, the location of the object in the image can be predicted very accurately by using the latest available position and velocity estimates coupled with the short elapsed time between images. As a result, the search window need only be large enough to contain the object up to a few pixels uncertainty. This limits the required computation to a manageable level and, more importantly, greatly reduces the probability of a false match occurring.

Real-time implementations typically rely on features which can be computed directly from the image without resorting to

actual 3-D measurments of object features. The object tracker used by Griffin et al. [1978] uses gray-level correlation implemented in software to match images. The signature of the object is a small spatial sample of gray levels taken from an arbitrary part of the object. This sample is correlated over a small window in the current image, with the match determined by the maximum value attained by the correlation function. Although correlation is notorious for obtaining false matches, it works reliably in this application due to the small search window. While rotating objects can be tracked, the rotation cannot be measured on the basis of the correlation value, so only translational velocity is measured.

Hirzinger and Snyder [1980] use a contour-based approach. The analog video signal is processed by special purpose hardware to detect significnt contrast areas in the image. This is done by recording transitions as the video level rises or falls past a threshold, with the processing taking place inside a programmable tracking window. The coordinates of each "contour" point are recorded, and tracking is based on four values -- the extrema in the horizontal and vertical directions. The position of the object is taken to be the centroid of the rectangle defined by these four values. This is a very simple approach which would seem to be easily fooled in scenes of moderate complexity. Tsugawa et al. [1979] use a similar video processing approach. In their case, the analog video signals from two cameras, mounted one above the other and oriented so that the scan lines are vertical, are differentiated and compared to obtain a stereo match of contrast edges. The matched edges are used to estimate the position of road features such as traffic cones, curbs and guard rails. This information is used to guide an automobile to follow a road and avoid obstacles. Pinkney [1978] also extracts features from the analog video signal. In this case, it is necessary to locate four high-contrast markers on an object.

Special-purpose hardware thresholds the video in four separate windows, one per marker, and returns the centroid location of each marker. This is a highly specialized approach which assumes very high contrast between the markers and the background and which assumes that there are no other pixels of similar brightness in the window.

The system in Gilbert et al. [1980] is distributed over four processors, each processor consisting of a microprocessor and special purpose hardware. One processor classifies pixels as "target" or "non-target" based on a histogram analysis of the image in a tracking window. A second processor computes two orthogonal projections of the target pixels by summing the pixel values (1 = target, 0 = non-target) along horizontal and vertical lines. Assuming that the target has bilateral symmetry (the target is a missile or airplane), these projections uniquely determine the identity of the target and can be used to extract the position and orientation (in the image) of the target. The features computed from the projections are normalized to obtain scale invariance. Although the features are not rotationally invariant, they are nearly so for small rotations which occur between two images. The system is able to correct for rotations by electronically rotating the video scan pattern so that the object orientation is essentially constant. Image rotation is handled by a third processor which also controls camera pointing and zoom. The fourth processor evaluates the goodness of the match at each tracking iteration and outputs parameters for camera control and the size and location of the tracking window to the other processors.

At a higher level, a robust tracking program must deal with adverse conditions which occur in dynamic scenes. One of these is occlusion, where the object becomes only partially visible or cannot be seen at all as it passes behind another object. In the

case of partial occlusion, a tracking program has to be able to generate a match on the basis of an incomplete set of image features. Also, the occluding object may generate features which must be recognized as not belonging to the object of interest. Even if an object is in full view at all times, feature extraction algorithms such as thresholding and edge detection may produce variable results due to changes in the background or due to changes in illumination (i.e., the angle of incident illumination) which arise as the object moves both translationally and rotationally with respect to the light source. Of the examples discussed so far, only Hirzinger and Snyder [1980] attempt to deal with occlusion. Their approach involves detecting radical changes in the relative values of the four features derived from the contour extrema. They state that a much more robust solution is required to handle occlusion reliably. Correlation tracking (Griffin et al. [1978]) is immune to background changes if the signature mask is contained almost entirely in the object. Using normalized correlation makes the tracker insensitive to uniform changes in illumination intensity, but does not help for partial changes such as occur if the object is moving into a shadow. Gilbert et al. [1980] assign a confidence weight to each match based on how well the features agree with the expected values predicted by the internal model. If a low weight is assigned to the current image, then the tracker "coasts" through this image, basing tracking control decisions more heavily on previous higher-confidence images. This is intended to overcome changing background conditions such as the target moving past a cloud. Saund et al. [1981] track objects by means of features used in a least-squares adjustment of the internal model of the tracked object. The program rejects extraneous features on the basis of proximity of a particular feature to its expected location based on the internal model.

Due to the severe constraints on computing time, it is

virtually impossible to use all of these techniques in their full generality in a real-time implementation on existing computers. An example of a more general, slower approach is the program of Roach and Aggarwal [1979]. Their program operates in the blocks world. Image sequences are generated by storing images of static scenes containing variously shaped blocks which are moved by hand between images. Images are then retrieved in the same sequence and processed as long as necessary to extract all of the necessary information and to perform matching. An internal model contains a description of each block which has appeared in any scene. In matching two scenes, blocks in the current image are either recognized as blocks already in the model or are labelled as new and inserted in the model. Any block which was seen in the previous image but is not present in the current image is labelled accordingly. There are three levels of matching. At the highest level, the program attempts to match each block in the model with features in the image which are present at a location predicted by the model based on previous position and velocity information. The features used for matching are the number of visible edges and visible surface area (in 2-D image coordinates). If this fails, the second level is invoked which attempts to match objects on the basis of the relative positions of two or more objects using relations such as left, right, above, and below. The third level attempts to match individual faces of the blocks on the basis of relative positions. This is done primarily to disambiguate the original segmentation. Occlusion is inferred by the presence of "T-nodes"; it is assumed that the top of the "T" belongs to the occluding object. Matching is performed by trying various correspondences between the visible edges of the occluded object and the model of the object which is expected to be present at that location. This program is fairly general in the sense that occlusion is handled fairly well and there is no fundamental limit on the number of objects in a scene or on the number of objects which enter or

leave the scene in any given image. However, the blocks world is an idealized visual domain, so this approach is certainly not the final answer.

Fennema and Thompson [1979] developed a technique called the Gradient Intensity Transform Method which differs from any of the other examples discussed so far. Time variations in intensity and the spatial gradient (as measured by the output of the Sobel operator mentioned in Section 2.4) are recorded for each pixel in the image. The intensity and gradient variations place constraints on the possible directed velocity of an object imaged at any pixel. A Hough-transform method (see Section 2.5) is used to cluster points by parameterizing velocity in terms of changes in gray level intensity and orientation of the gradient. Clustering techniques applied to the parameter space of the Hough transform are used to find regions of pixels with similar velocities. In this way, objects can be segmented from the scene as well as assigned a velocity. To make this procedure work well, the image is first smoothed with an averaging filter. Thus a certain amount of detail will be lost and the accuracy of position measurements may suffer.

## 5.2 Verification

In verification vision the system knows what objects should be present in the scene and their approximate position and orientation. The goal is to verify the presence of these objects and to refine the estimates of their position and orientation. Bolles [1976] developed a verification vision system that uses a set of opertors to find features in the scene approximately at the positions where these features are expected from the a priori information. The system uses teaching and learning phases described in Section 7 in which operators are selected and statistics about them are gathered. Then in what Bolles calls

"planning time" the system ranks the operators according to their expected contribution to the solution, determines the expected number of operators to be needed, and predicts the cost of obtaining the solution. Finally, in "execution time" the system applies the operators one at a time and combines their results into estimates of confidence in the verification and precision of the refinement. For this purpose the probability distributions of the results of applying each operator that were gathered in the learning phase are used in Bayes' theorem. The refined position and orientation estimate is obtained by a least-squares adjustment, which includes an automatic editing feature for removing those features that do not seem to agree with the others. When the desired confidence and precision or the cost limit has been reached, no more operators are applied.

# 6. STEREO

In stereoscopic vision, triangulation between two or more views from different positions is used to determine distance. This avoids the high degree of ambiguity inherent in trying to determine depth by other clues in monocular views. However, there is still some ambiguity present in the process of matching points in the different views so that the triangulation can be done, since a small portion of one image may be similar to several portions of another image. This is especially true when the noise level is high, since small differences may be obscured by noise. Although using stereo makes the problem of extracting three-dimensional information easier than it is with monocular vision, the hardest parts of the vision problem, description and recognition, still remain.

It makes no essential difference for stationary scenes whether the multiple views are obtained from separate cameras simultaneously or from one moving camera, except that the calibration problem might be different. (Camera calibration is discussed in Section 8.)

Usually only two camera positions are used. However, if several positions are used, the problem of resolving the ambiguities becomes easier. In the close-together views, things have not shifted much between views and thus are easier to match. These results then can be used to resolve the ambiguities in the further-apart views, whose results produce greater accuracy. Nevatia [1976] and Moravec [1980] have explored ways of accomplishing this.

Stereo techniques differ in the way in which matching is done between the pictures, especially in the kind of entities

that are matched. One common method is area correlation, in which small areas (windows) a few pixels on a side are matched by maximizing the correlation coefficient, minimizing the mean squared difference, or using some variant of these processes. This method usually works well for highly textured scenes, such as natural outdoor scenes.

Hannah [1974] explored some of the basic properties of area correlation and developed a region-growing method using it. Gennery [1980] produced a refined correlation measure for area correlation and a search procedure that uses some local context to reduce the ambiguity in matching. Levine et al. [1973] use a correlation window that varies in size, so that it is small for high resolution where there are large brightness variations but large elsewhere for good noise rejection. They also first match sparsely spaced "tie points" with high information content, and then use these points to constrain the search for matching the nearby points. Yakimovsky and Cunningham [1978] also use a correlation window that varies in size, according to the magnitude of the local autocorrelation value. They use a sparse window for speed, then use a full window at the five points with the highest resulting correlation, to produce an accurate correlation. Moravec [1973] corrects for distortion within the window by means of a prediction-correction technique. On each iteration of this process, the depth map produced by the previous iteration is used to correct the perspective distortion within each window so that a better match can be made.

Instead of using ordinary area correlation, Marr and Poggio [1976] proposed a relaxation method that assigns a depth to each pixel. The method assumes that the depth is continuous except at occasional boundaries. Grimson and Marr [1979] produced a method in which the images are band-pass filtered and the zero crossings of the results are matched between the images. Various amounts

of filtering are used. The coarse zero crossings from the heavily smoothed images are used to resolve the ambiguities in the high-resolution zero crossings from the lightly smoothed images.

Scenes of man-made objects often are not highly textured but contain sharp brightness edges at boundaries of objects and at intersections of planar faces. For such scenes area correlation does not work very well. Instead, it is usually better to detect features in each image and to match these features.

Arnold [1978] produced a method that uses edge elements as the features to be matched. He first finds edge elements by means of the Hueckel operator. Then these are matched by a relaxation process that uses local context to resolve ambiguities. Baker [1980] matches edges by means of dynamic programming, using a coarse-to-fine strategy.

Ganapathy [1975] detected straight edges that correspond to the edges of polyhedra, then matched these in order to reconstruct the polyhedra in three dimensions.

Roth [1978] used a region-matching technique to extract three-dimensional surfaces from a stereo pair of images. Each image is converted to a gradient array based on local changes in intensity. The gradient arrays are partitioned into regions of uniform intensity (zero gradient) and uniform change (similar gradient orientation). Initial region matching is based on similarity of shape, size, average intensity and average gradient. Further match evaluation imposes disparity constraints and uses occlusion clues. The matching algorithm makes or breaks matches based on a confidence measure until a globally consistent high-confidence match is obtained.

Saund et al. [1981] combine the measured positions of features found in two camera views into a single least-squares adjustment for the three-dimensional position and orientation of a known object. Thus the stereo information is used implicitly in the adjustment, rather than being explicitly computed for each feature. Indeed, it is not necessary for any feature to be seen by both cameras; if enough features are seen in each image, the constraints of the object model allow depth to be obta' ed.

Woodham [1979a] proposed a novel approach in which, instead of moving the camera to different known locations, the light source illuminating the scene is moved to different known locations in order to obtain different images. Under some reasonable assumptions about the reflective nature of the surfaces in the scene, this allows the orientation of the surfaces in the scene to be determined unambiguously. Although, strictly speaking, this is not stereo vision, it is sufficiently similar so that the term "photometric stereo" has been applied to it.

# 7. TEACHING AND LEARNING

Computer vision systems recognize objects in a scene by matching image features with internal models. The models represent the vision system's knowledge about the world. The concepts of teaching and learning relate to the ways in which general-purpose systems incorporate knowledge into the data base of models. The teaching/learning process implies a dialogue between the computer and a human operator. If we show the vision system a new object (i.e., one that is not currently represented in the data base) and give it a name, and the system is able to recognize the object when it is seen again, then we can say that the system has learned to recognize the object. We can also say that the system was taught to recognize the object, especially if the human operator has given assistance in learning how to recognize the object, such as by pointing out important features, for example. Even if the operator comletely specifies the object model and does not show the system a training example at all, this process can still be called "teaching," as long as it involves a high-level interactive transfer of knowledge. Otherwise, it might better be called "programming."

One approach to learning in computer vision has been to simulate biological functions. The neocognitron (Fukushima [1975] and Fukushima and Miyake [1980]) is a self-organizing classifier for two-dimensional patterns constructed as a set of layered two-dimensional "cell" arrays. The cells are connected between and within layers by "synapses," some of whose strengths evolve with visual experience. Synapse modification is through unsupervised learning; the machine learns to recognize patterns it sees most often. The cells in layers near the "photoreceptive layer" become feature detectors, while the information represented with increasing depth becomes more abstract. The

neccognitron is able to tolerate shifts in position, rotations, and distortion in the shape of patterns. However, this method is computationally extremely expensive, requiring thousands of cells and tens of thousands of synapses, so will probably not be of practical use until computations can be performed in parallel for each cell.

At the simplest level, many computer vision systems can learn to recognize specific objects. In the case of the SRI Vision Module (Nitzan *et al.* [1979]) and CONSIGHT (Ward *et al.* [1979]), blob analysis is performed in a training mode and a record containing the features of the new object is stored in a list of possible objects. Several views of the object may be used to obtain a statistical distribution of the various feature values. Viewing distance and perspective are constant, so the object always appears essentially the same up to translations and rotations in the image plane. These systems do not have to infer what the object will look like for arbitrary translations and rotations. Perkins [1978] used a similar teach-by-showing method to generate a concurve representation which includes a list of statistical features extracted from the region enclosed by the outer boundary and a measure of the rotational symmetry of the object. His program also assumes that scale and perspective are constant.

Underwood and Coates [1975] developed a method of learning to recognize 3-D objects (convex polyhedra) from arbitrary views. Their method automatically generates a 3-D description of an object based on a sequence of images taken as the object is rotated in space. An object is described in terms of its surfaces and how they are interconnected, by matching successive views to determine what has been seen before and what is new.

In ACRONYM (Brooks *et al.* [1979] and Brooks and Binford

[1980]), the user specifies models of objects, generic object classes, and possible relationships among objects. A high-level modelling language is used with an interactive editor so that the user can conveniently specify the generalized cones to be used and how they connect to form objects. (See Section 1.3.) The user is aided by a library of useful prototypes and a graphics module that provides visual feedback.

The next level of learning, which represents a considerable leap from that discussed above, involves being able to model generic classes of objects by being shown examples, with no help from the human teacher other than selecting an appropriate sequence of training examples. As an example of this type of capability, a program would be able to recognize any chair after seeing a few examples of the various types of chairs. This requires the ability to determine the relevant components of a chair and their relationships to each other, leading to a description such as "a chair has four legs and a back attached to opposite sides of a seat." An outstanding example of this type of capability is described next.

Winston [1975b] produced a program that learns concepts by being shown positive and negative examples. The type of concepts used by Winston involves structures composed of simple objects. When it is shown a scene, the program constructs a description of the scene consisting of a network indicating the relationships among the objects, as described in Section 3.3. When the scene is designated as a positive example of a certain concept, the program uses the resulting description as its initial model of the concept if it had no previous model of it; otherwise it compares the description to its model, notes the difference, and generalizes its model accordingly. When the scene is designated as a negative example of a certain concept, the program compares the description of the scene to its current model of the concept

and notes the difference. If a single difference in the structural description is noted, the relationship missing in the negative example is marked in the model as being mandatory, or the relationship missing in the model is entered into the model as being prohibited. If there are multiple differences, a tree of models is produced, which can be disambiguated by later examples. Thus the negative examples that are near misses provide the most information.

Another aspect to learning is the selection of a recognition strategy. The binary decision tree used in the SRI Vision Module (discussed in Section 4.2) is generated automatically after all parts to be modelled have been presented to the system. Using the observed statistical distributions of features, the tree is constructed by selecting a feature and a threshold for values of that feature which most reliably partition the set of objects into two disjoint subsets. This is done recursively for each subset containing more than one object until all subsets contain exactly one object.

The verification vision system of Bolles [1976] described in Section 5 incorporates some ability to learn and to be taught. In what Bolles calls "programming time" the user interacts with the system to specify confidence, precision, and cost constraints for the task and to help in selecting operators for detecting features in the pictures. The system finds features and displays them and their properties, and the user can accept, reject or modify each operator and can specify additional operators. In "training time" the system applies the chosen operators to sample pictures and gathers statistical information about their effectiveness.

# 8. CAMERA CONTROL AND CALIBRATION

This section deals with the control of camera parameters which allow a computer vision system to adapt to changes in viewing conditions and camera calibration techniques which allow accurate measurement of 3-D object positions.

In order to deal with moving objects or to look at various locations in the environment, a pan-tilt head is necessary to point the cameras in the proper direction. This is essentially an engineering problem to build a suitable device, so we consider here some of the important design requirements. First, it should be capable of very rapid movement so that the robot can "glance around" to quickly check out situations. Second, it should be equipped with precise encoders so that information extracted from the pictures can be referenced to a fixed coordinate system. Third, it should be capable of precise servo control to keep a moving object centered in the field of view. Fourth, additional degrees of freedom of camera movement are desirable. For example, the ability to rotate each camera of a stereo pair so that their principal axes intersect at any desired range maximizes the common field of view at that range.

Generally speaking, image feature extraction algorithms perform best when iris and focus settings are adjusted to obtain the highest-quality image. If the vision system operates in a dynamically changing environment, automatic control of iris and focus is necessary to adapt to variations in illumination and object distance.

The best iris setting maximizes the dynamic range of pixel intensities and is nominally obtained by keeping the brightest pixel in the image just below saturation. This approach can be

refined by applying it to a window of the image containing the object of interest and allowing brighter regions (which are currently being ignored) to saturate. This is the approach used in IMFEX, a special-purpose image feature extraction device built at JPL (Eskenazi and Wilf [1979]). The maximum video level in a programmable rectangular window is available at 30 Hz to a microcomputer which servos the iris motor.

Focus control is generally based on maximizing the high-frequency content of the image. In a 3-D scene containing objects at varying distances, the best focus depends on which object is being analyzed. Thus the high-frequency content should be maximized in a window of the image which contains the object of interest. One way to measure high-frequency content is to look at the magnitude of a gradient edge detection operator (Eskenazi and Wilf [1979]). For best results, some quantity can be integrated over a window. The system described by Johnson and Goforth [1974] focuses by integrating either thresholded brightness data or the results of using high-pass filter on the image. For all of these methods, a hill-climbing strategy is used to drive the focus motor of the lens to the position that maximizes the parameter being measured. In some cases local maxima exist that do not correspond to the correct focus, but if the initial focus is sufficiently close to being correct, the hill climbing technique will find the global maximum.

The final camera parameter to be considered is focal length. Variations in object distance or varying field of view requirements for different tasks can be best handled if it is possible to change the focal length of the lens. One approach is to mount several fixed-length lenses on a turret and to rotate the appropriate one into place as viewing conditions require. Pingle and Tenenbaum [1971] used three lenses. A more general approach is to use a computer-controlled zoom lens. The

difficulty with varying the focal length is that camera calibration (to be discussed below) changes. In the case of multiple fixed-length lenses, the calibration is at least fixed for a given lens. For a zoom lens however, the calibration will vary continuously. Gibert et al. [1980] use a zoom lens in their tracking system but do not discuss the impact on camera calibration. It is worth noting here that changes in focus have a similar effect on calibration, although on a smaller scale, to changes caused by a zoom lens.

Camera calibration consists of determining a set of parameters which specifies the relationship between 3-D points in a scene and their projections onto the 2-D image plane. While there are various ways of formulating this relationship, in the case of a central projection it is generally equivalent to knowing the location of the lens center, the orientation of the principal axis of the lens, and the distance from the lens center to the image plane. Cameras are typically calibrated by determining the image coordinates of a set of reference points and solving for the calibration parameters. We are concerned here with calibration methods which allow general 3-D position measurements and stereo matching, as opposed to methods used in systems such as CONSIGHT (Ward et al. [1979]) which assume constant viewing distance and perspective.

In hand-eye systems, a target on the manipulator can be used as a calibration point. By moving the manipulator to several different positions and locating the target in the image each time, a set of 3-D points (obtained through manipulator position feedback) and their images are obtained which can be used to solve for the calibration parameters. This is the method used by the JPL hand-eye system (Yakimovsky and Cunningham [1978]). The target is a small light bulb on the hand which can be located easily and reliably. One of the advantages of this approach is

that systematic errors in the camera calibration relative to the manipulator calibration are eliminated.

In order to obtain accurate position information from an image, the transformation from three-dimensional space to the two-dimensional image plane must be known. This is usually assumed to be a central projection. However, cameras often have distortion caused by the lens or by the scanning mechanism that causes the true projection to depart from this ideal. Therefore, a distortion calibration may be necessary. One way of performing such a calibration is to take a picture of an array of dots whose positions are accurately known. A program then can find the dots, compare their positions in the image to the ideal positions, and fit a distortion correction function (perhaps a two-dimensional polynomial) to the discrepancies. Moravec [1980] describes a way of finding the dots in the image.

In using the stereoscopic vision techniques described in Section 6, it is highly desirable to know accurately the relative position and orientation of the cameras which produced the multiple views, because this knowledge constrains the search for matching points in the images, and because it enables absolute distances to be computed from the matches. Of course, if each camera's position and orientation have been precalibrated relative to some common coordinate system, the relative position and orientation are easily obtained. However, sometimes this individual calibration is insufficiently accurate or is absent. In such cases it is possible to obtain the desired relative calibration by using unknown points in the actual images, so that no special calibration data is needed, except that the distance between the cameras cannot be so obtained. If at least five points in general position are matched in two images, they can be used to compute the five parameters that define the position and orientation of one camera relative to the other, except for

distance. Gennery [1980] provides a way of performing such a stereo camera calibration, which obtains the matched points by using a method of Moravec [1980], performs a least-squares adjustment so that more than five points can be used effectively, individually weights each point based on its estimated accuracy, and automatically edits out points that have been mismatched. The distance information is usually available with sufficient accuracy from other sources, and, even if it is not, three-dimensional information (except for a scale factor) can still be computed from a stereo pair of pictures.

# 9. SYSTEM ARCHITECTURE

## 9.1 Computational Structures

In the simplest computer vision systems, a scene description is obtained in a sequential process. A TV image is input to a feature extractor whose output goes to a recognizer or classifier which in turn outputs a scene description. This characterization applies to fairly simple (and practical) vision systems such as the SRI Vision Module and CONSIGHT described in Section 1.3. The basic assumption in this type of approach is that there is a clean distinction between the processes of feature extraction and recognition, and that feature extraction can operate reliably in the absence of knowledge stored in the model that is used for recognition. The strict sequential approach makes it fairly straightforward to partition the processing into logically distinct units and to implement these computational units in special-purpose hardware where speed is critical.

The approach described above has proven to be useful in highly organized environments. However, it is woefully inadequate when applied to natural outdoor scenes or even general 3-D scenes of man-made (i.e., industrial) objects.

In developing possible architectures for more general computer vision systems, we must consider what types of computational tasks will be performed and what structures are best suited to perform them. No one knows yet how powerful high-level understanding and visual analysis will work, so we have very few hints as to how to design a system to do high-level vision. But there are some insights into what is required of low-level vision. Low-level vision must extract an economical description of a scene from a raw intensity image, without

necessarily recognizing objects or understanding much about the scene.

Much of the research that has been done in low-level vision (for example Barrow and Tennenbaum [1978], Hanson and Riseman [1978c], and Brady and Wielinga [1978]) indicates that a number of images of a scene in various stages of processing should be maintained concurrently, because these explicitly represented images interact with each other and with higher and lower levels as processing proceeds. The actual computations of low-level vision are usually local to one portion of an image, both within and between levels. Often the actual computing is by way of relaxation processing, whereby local constraints within and between images are used to arrive at a globally consistent result (as, for example, with Zucker [1978]). Thus, low-level vision might be well served by an architecture consisting of a large number of registered image buffers accessible by processing elements working in parallel.

An example of this kind of architecture for low-level vision processing is the stack organization proposed by Barrow and Tenenbaum [1978] (also described by Tenenbaum *et al.* [1979]). In this organization each level of the stack holds an iconic (in the form of an image) representation of various characteristics of the scene, called intrinsic images. For example, these can be brightness, illumination, reflectance, orientation, and distance. There is communication between nearby pixels in each level of the stack to enforce assumptions about the continuity of each characteristic, and there is communication between corresponding pixels at different levels of the stack to enforce the assumed relationships among the various characteristics. By an effectively parallel iterative computation based on these assumptions, the intrinsic images are recovered. (Special hardware could be built for implementing this scheme, as

described in Section 9.2, but it can be implemented on any general-purpose computer with sufficient memory and speed.)

A possible architecture for performing some low-level vision tasks is the cellular automaton, in which simple operations are performed at each step on each pixel as a function of the neighboring pixels at the previous step. (It thus is similar to one level of the Barrow and Tenenbaum method described above.) Such methods are discussed by Rosenfeld [1979b]. They can be implemented efficiently by the single-instruction-stream multiple-data-stream hardware described in Section 9.2.

Another computational structure that has received much attention is the "recognition cone" (Uhr [1972]) and its variants. This is a hierarchical approach with several layers of processing organized similarly to the pyramid data structure discussed in Section 2.8. Uhr proposes a "parallel-serial" computer architecture (Uhr [1978]). Each layer is viewed as a parallel processor which transforms (and shrinks) the data at one level to the next higher level. There may be several transforms (operators) at each level which operate in parallel. The various layers are processed serially in both a bottom-up and top-down fashion. This implies feedback to feature extractors based on partial recognition results, something absent in the simple vision system architecture described at the beginning of this section.

Hanson and Riseman [1978b] propose a hierarchical processing cone computational model for low-level vision processing such as extracting line and region data. In their model, there may be several planes of data at any given level representing processed outputs from the level below. The processing at each level is carried out by an array of local processes. In addition to top-down and bottom-up processing, there would be communication

between processes at the same level, adding lateral control decisions to the computational structure. All of this processing could take place in parallel, leading to a host of unknown computational methods awaiting much further research.

## 9.2 Hardware

The basic hardware device necessary for digital image processing is an analog-to-digital (A/D) converter and a computer interface for access to the digital image. Advances in semiconductor technology have led to fast A/D converters and fast random-access memories which allow continuous eight-bit digitization of 512-by-512 (or larger) images with full frame buffering at the standard video frame rate of 30 Hz. For the most flexibility, the computer should have random access to any pixel in the image without disrupting the digitization and buffering. One of the first devices to offer this capability was RAPID (Yakimovsky *et al.* [1976]). RAPID digitizes (8 bits/pixel) and buffers 192-by-240 images while providing concurrent computer access to any pixel in the frame buffer in 4 microseconds. This device enabled the implementation of the real-time correlation tracker discussed in Section 5 (Griffin *et al.* [1978]).

In order to achieve reasonable competence, vision requires enormous amounts of computational power. It is possible that no existing sequential computer comes within six orders of magnitude of being powerful enough to see as well as a human being. Even the modest performance of some of the existing systems requires several minutes of computing in order to analyze a single scene. Although the speed of processors will increase, it is apparent that a different architecture than the single general-purpose processor will be required in order to produce the large gains needed. Two principal possibilities are special-purpose hardware devices dedicated to computing certain operations needed in

vision much faster than a general-purpose computer can, and parallel computation in general purpose computers. Special-purpose hardware can produce large gains in speed but it is limited to low-level operations at present. At the higher levels, the greater complexity needed may cause it to remain noncompetitive with general-purpose hardware. Although parallel computation can be used in a special-purpose device, it can also be used in a general-purpose computer, so that large gains in speed can be achieved without loss of flexibility. (Some of the possible hardware architectures for computer vision have been surveyed by Reddy and Hon [1979].)

One of the simplest image preprocessing steps is thresholding, which can be done either digitally or in analog. In the analog case, the thresholder is essentially a one-bit A/D converter. The SRI Vision Module (Nitzan *et al.* [1979]) operates in either mode, obtaining binary images at frame rates. A 16 K-bit frame buffer allows storage of two binary images taken from one or more 128-by-128 cameras. Additional special hardware can access this memory to convert the raster image to run-length code and to compute the area and first-order moments (sum x, sum y) of blobs. All of this processing is confined to a programmable rectangular window so that analysis can be restricted to a single blob. The run-length code is processed by a general-purpose microcomputer (DEC LSI-11/02) to extract additional blob features.

Many low-level image processing algorithms convolve the image with a square or rectangular window. The windows typically range in size from 3 by 3 to 7 by 7, with larger windows being used occasionally. Most of these algorithms could be implemented in a parallel array processor consisting of M times N identical computational units to process an M-by-N image. However, the output of conventional TV cameras is serial, which means that the

processors would be idle for most of the 1/30 second frame time. An alternative approach is to process the image serially in real time by effectively scanning the image with a single computational unit. For an n-by-n operator, this is accomplished by buffering the last n-1 lines and accessing an n-by-n window from the current line and the buffered lines in parallel, using a pipeline architecture to perform the necessary computations, with new windows being accessed at the pixel rate. While an entire frame time is required to process an image, the net result approaches the efficiency of parallel array processing, since the processing is going on concurrently with the acquisition of the image by the camera. The effective processing time is thus the difference between the time when the last pixel is scanned by the camera and the time when the processed value corresponding to the last pixel is output. Clocking data through a computational unit at pixel rates results in a difference, or pipeline delay of (n-1)/2 line times (standard video line time is approximately 63 microseconds) plus possibly a few pixel times for an n-by-n operator.

An example of this type of pipeline processor is a device called IMFEX built at JPL (Eskenazi and Wilf [1979]). A video input signal is digitized and processed by four computational units. The first is a 3-by-3 gradient operator which enhances contrast edges, outputting the magnitude (8 bits) and orientation (quantized to 45-degree intervals) of the gradient. The second is a thinning algorithm (3-by-3) which filters the gradient output by passing only those pixels whose magnitude is greater than either of its two nearest orthogonal neighbors in a 3-by-3 neighborhood (e.g., the top and bottom neighbors of a horizontal edge). The thinned gradient image is thresholded to obtain a binary edge map. A second thinning algorithm deletes edges from the binary edge map which are not necessary to maintain global connectivity, by examining the eight nearest neighbors of each

edge in a 3-by-3 window. The thinned binary edge map is transferred to a block of memory in a DEC LSI-11/03, which performs further processing.

Nudd *et al.* [1979] have experimented with hardware implementation of several low-level algorithms using charge-coupled device (CCD) technology. One of the primary goals of their work is to integrate the image sensor and processor on a single CCD chip. Functionally, the approach is the same as that for IMFEX in that the image is processed serially as it is scanned by the camera. Most of the operators they have implemented (Sobel operator, Laplacian, spatial filter, and unsharp masking, for example) are 3 by 3. They have also implemented 5-by-5 and 7-by-7 programmable masks and a 26-by-26 convolution operator.

One possible type of parallelism that may be especially suited to low-level vision is an array processor using a single-instruction-stream mutiple-data-stream (SIMD) architecture. Such a system uses an array of simple processors (usually one per pixel) that all perform the same functions simultaneously under control of a master processor. Each cell in the array usually can communicate directly only with its neighbors in the array. The master processor is similar to an ordinary computer. It decodes the instructions in its program, and causes the array of processors to execute them. Since each cell is much simpler than a central processing unit of an ordinary comuter, a high degree of parallelism can be achieved at low cost. However, the kinds of algorithms which can use this sort of parallelism are limited.

Several SIMD devices have been built. They differ greatly in the complexity of the processors in the array and in the amount of data stored at each cell in the array. Golay [1969] designed a device that performed simple operations on binary

images based on the values of the six neighboring cells in a hexagonal array. A few bits per cell were stored. The ILLIAC-III computer (McCormick [1963]) performed very simple operations at each cell of either a rectangular array or a hexagonal array, whereas the ILLIAC-IV computer (Barnes *et al.* [1968]) can perform arithmetical calculations on data that can be accessed in a fairly flexible way; but it has only 64 processors. (The ILLIAC-IV was not intended for vision.) The CLIP4 system (Duff [1978]) uses a 96-by-96 rectangular array of processors, each of which can communicate with its eight nearest neighbors and can perform boolean operations, from which arithmetic operations can be built with software. Each cell can store 35 bits. Probably the most ambitious project of this sort so far is the Massively Parallel Processor (Schaefer [1980]) being developed by NASA. It will contain a 128-by-128 rectangular array of cells. Each cell stores 1024 bits of data, performs logical and arithmetic operations (both fixed-point and floating-point), and communicates with its four nearest neighbors. Bit-serial arithmetic is used. Thus some parallelism is sacrificed in order to keep the cost down enough so that the large amount of parallelism in the array is economically practical.

Another type of parallelism is a multiprocessor using a multiple-instruction-stream multiple-data-stream (MIMD) architecture. Such a system uses a number of central processing units that independently execute different instructions. It would be possible to have these connected only to their neighbors in an array as in the SIMD devices, but this would waste the generality of the processors. Some more general type of communication is needed. Preferably, it is desired to have all of the processors able to communicate directly with each other (without the delays that using a common bus would involve).

The above direct communication can be achieved by means of a

crossbar switch. To interconnect n items in this manner requires $n^2$ switching circuits. An example of such a system is the C.mmp system built at Carnegie-Mellon University (Wulf and Bell [1972]). It connects sixteen PDP-11/40E computers to sixteen 256 K memory modules by means of a sixteen-by-sixteen crossbar. Since the amount of circuitry in each switching circuit is considerably less than that in each processor, it probably is practical to connect a few hundred processors in this manner using current semiconductor technology.

As semiconductor technology improves, it will become practical to use a much greater number (perhaps millions) of processors in a computer. To connect such a large number by means of a crossbar probably will be impractical, since the number of components is proportional to the square of the number of units to be connected. However, Moravec [1979] has proposed a method based on the Batcher sorting network in which the number of components increases much less rapidly. Full interconnection is retained, but there is a slight loss of speed, since a message sent through the network must go through a number of stages of circuitry proportional to the logarithm of the number of units to be connected. (Because of pipelining the bandwidth is high, but the latency is fairly long.) Thus with this method it would be most appropriate for each processor to have its own memory, which it would access most of the time, with less frequent messages being sent to and from other processors or memory modules. Although elaborate systems software may be needed, once it is available the complexity of the system can be largely transparent to the applications programmer.

In cases where the number of processors is too great for the use of a crossbar and it is desired to avoid the complexity of the sorting network, a more limited interconnection scheme tailored for a particular type of task might be used. For

example, several processors could be connected by a crossbar to form an image processing unit that would operate on the contents of one image buffer, and several of these combinations could be connected by serial image transfer between image buffers through a crossbar, so that processing on different iconic representations could occur simultaneoulsy, corresponding to different stages of processing. As another example, the stack organization of Barrow and Tenenbaum for recovering intrinsic images (described in Section 9.1) could be implemented by having processors in each level of the stack that could communicate with their neighbors in that level and with the processors at the same position in all other levels. An existing system with limited interconnections is the Cm* system built at Carnegie-Mellon University (Swan _et al._ [1977]). It contains 48 LSI-11 computers connected in clusters. It must be emphasized, however, that where the complete interconnection of processors is practical it is better to use such a general system and to put it into the configuration of these examples under software control, rather than to build hardware for these specialized interconnections. A good rule to follow is not to build a special-purpose device if a general-purpose device can be built almost as cheaply and can perform almost as fast.

# 10. CONCLUSIONS

## 10.1 State of Computer Vision

The statement "Vision is hard" is found often in the computer vision literature. There are several reasons for the difficulty. In the first place, an image contains an enormous amount of information, much of it irrelevant to the task at hand, and it is an imperfect projection of the real world, containing noise and distortion. From this the relevant information must be extracted. In the second place, the transformation from the image to the real world is highly ambiguous. Thus world knowledge must be relied on to resolve the ambiguities. (This is especially true in monocular vision of three-dimensional scenes, but it is also true to a lesser extent in stereo vision.) In the third place, an object seen may only vaguely resemble others of its generic type or even itself at other times or under other conditions. In the fourth place, in a powerful vision system an object must be recognized out of a large number of possible objects or generic types.

These facts appear to manifest themselves in two ways in practice. First, vision requires an enormous amount of computing. Second, it seems that the computational methods needed are very complicated, and it is unknown today what the right methods will be.

Even though the above two aspects of the problem are both important, there is a trade-off between them. For example, recognition could be done in principle by comparing the image to all possible views of all possible objects. This is a simple technique, but it is completely prohibitive in computational cost. More complex, smarter methods can reduce the computation

enormously. At the other extreme, one might hope that an extremely clever method might be invented that would make the amount of computing quite small. This doesn't seem likely, though, because of the large amount of information in an image, the large number of possibilities in a viewed scene, and the fact that biological evolution has not been able to come up with such a method. (The human brain devotes billions of neurons to the task of vision.) Thus, to match the capability of human vision will probably require several orders of magnitude times the computing power of today's most powerful computers. (If the current progress in electronics technology continues, sufficiently powerful parallel computers eventually will become available, as discussed in Section 9.)

One might hope also that some powerful simplifying principles might be discovered that would eliminate the need for much of the complexity, but there is no evidence that such principles exist. Study of the human brain has not been of much help in this regard or in regard to finding less powerful but practical principles, since neurophysiologists and psychologists have barely scratched the surface in understanding how it works. (Some of the current knowledge is summarized by Graham [1965], Julesz [1971], and Carterette and Friedman [1975].) Since the use of the techniques that ultimately will be successful probably will require much computing, these techniques cannot be developed until sufficiently powerful computers are available with which to experiment. Thus, much research using these powerful computers may be required before we learn how to use them effectively.

In spite of the above problems, some progress has been made. Some highly specialized systems have actually performed useful tasks in restricted domains. Some laboratory systems have a degree of generality in the domain of recognition of two-dimensional objects under well-controlled lighting, because of

the lesser amount of ambiguity and complexity in this domain. Some experimental systems hold promise for recognition of generic three-dimensional objects, although they require a large amount of computing time on existing computers. Some special-purpose hardware is becoming available, which enables some very low-level computations to be performed rapidly. Even in these cases, however, a variety of techniques are in use, with no consensus about which are the best. This becomes even truer as we move to the higher-level, more general, or more advanced areas. Furthermore, many of the approaches that have been used are ad hoc, with little promise of generality.

## 10.2 Key Research Issues

Some of the issues that seem important in computer vision research will be summarized.

In recognition, it is possible to proceed either in a bottom-up manner, detecting low-level features first, and organizing these into ever higher-level structures until the scene is completely analyzed, or in a top-down manner, starting with a hypothesized object and trying to find its features in the scene. A combination of both of these approaches is needed in most vision tasks. An important issue is the proper balance between these two approaches and how it varies with the nature of the task.

At the lowest level in vision the scene representation is iconic (in the form of an image), whereas at the highest level the representation is symbolic. The proper level for the dividing line between the two types of representation and how much they should overlap is an issue. (Funt [1977] touches on this question in the domain of problem solving. It is also discussed by Barrow and Tenenbaum [1981].)

A separate but related issue is the specific representations that should be used. That is, what sort of features should be extracted from the scene (edges, corners, regions, surface orientation, and so forth), and how should objects be modelled (wire-frame models, generalized cylinders, and so forth)? At the highest level this issue is part of the general knowledge representation problem in artificial intelligence.

If there is a very large number of models in the data base, the problem of how to index efficiently into it is important.

Another issue is whether parallel methods such as relaxation are merely a programming style as claimed by Marr [1978], or whether they lead to inherently different algorithms than sequential methods, and if so, which are more appropriate to which types of tasks.

There are often several types of information available in portions of a vision task. For example, depth information can be obtained stereoscopically and by means of various monocular clues. Also, information obtained from a sense of touch or from other information in an intelligent robot may be available, in addition to vision. Means of combining such different types of information need to be explored. At the lower levels, relaxation processes such as advocated by Barrow and Tenenbaum [1981] may be appropriate. At the higher levels, one possibility is the "blackboard" (a central communication medium for the representation of hypotheses, partial solutions, and pending activities) approach used in the Hearsay speech-understanding and knowledge-based expert systems (Reddy et al. [1973], Erman et al. [1980], and Balzer et al. [1980].)

Once a strategy is chosen for the above general issues, the

question remains as to what particular methods should be used in each case. This is a problem at all levels, but it is particularly wide open at the high levels. Indeed, at the highest levels vision merges into the rest of the field of artificial intelligence. Thus the question of how high-level processing should work cannot be limited to vision only but is part of the problem of how any high-level processing might be done no matter what sensory mechanisms an intelligent machine possesses.

Another issue concerns means for the vision system to learn object models by being shown the objects and to be taught object models by means of a convenient user interface. An even more difficult problem is the learning and teaching of generic types (for example, learning the concept of a chair by being shown examples of chairs). A related issue is how to make the system versatile by having it programmable at a very high level.

It is possible that research in vision would be greatly helped by the availability of a very-high-level programming language especially designed for vision. Very little has been done along this line.

Finally, the type of hardware to be used is impor' A specific question in this regard is how much parallelism and what kind of parallelism should be used. (Some of the options were discussed in Section 9.)

## 10.3 Future JPL System

A JPL vision system that can be developed in the next few years must operate in a restricted domain, because of the limited advancement of the state of art of computer vision that can be expected in that time. It is expected that the system will be

capable of recognizing and tracking, perhaps in real time, known objects that can be modelled by the composition of a few simple geometrical shapes. The objects can be selected from a reasonably large set of possible objects and can have arbitrary three-dimensional position and orientation. It is desirable, but not necessarily achievable until later, that multiple objects could be present, some partially obscuring others. It may be necessary in some cases at first to have objects identified by means of special colors or markings. The object models can be taught by the user, and perhaps can be learned by means of vision. The vision system will produce data suitable for grasping and manipulating objects.

In order to make the above capability achievable, certain hardware will be required. The cameras should have at least 240 non-interlaced lines of vertical resolution and roughly equivalent horizontal resolution or better. They should produce eight-bit monochromatic pictures, and some sort of color capability should be available. Since stereoscopic vision will be used, at least two cameras are required. The cameras should be mounted on a smoothly-operating pan-tilt head equipped with precise position encoders.

Special hardware for performing some low-level vision operations at high speed should be available. This would be similar to the present IMFEX but probably much more powerful and versatile.

In order to perform the remaining computation at or near real time, either an extremely powerful processor or many processors will be needed. One possibility is a multiprocessor mainframe computer. Another possibility is a few hundred microprocessors operating in parallel (perhaps connected by a crossbar), and a single-processor mainframe computer. It remains

to be seen what combination will produce the most computing for the money. In any case sufficient memory will be needed to store large programs and many images at various stages of processing at one time.

A large on-line disk storage system will be needed for convenient storage of programs, images, and other data. Good programming practice requires that programs be tested on stored images, so that reproducible results can be obtained.

Interactive graphics display terminals will be needed for the user interface, so that information concerning object models can easily be entered and intermediate results of computations can be displayed.

A system such as described above will allow significant contributions to the state of the art of computer vision to be made at JPL, and will allow the development of techniques that NASA will find useful in the future use of robots in space.

## REFERENCES

The following abbreviations for a few journals and conference proceedings are used:

| | |
|---|---|
| AI | Artificial Intelligence |
| CGIP | Computer Graphics and Image Processing |
| CACM | Communications of the Association for Computing Machinery |
| JACM | Journal of the Association for Computing Machinery |
| T-C | IEEE Transactions on Computers |
| T-PAMI | IEEE Transactions on Pattern Analysis and Machine Intelligence |
| 1AAAI | First Annual National Conference on Artificial Intelligence, The American Association for Artificial Intelligence, Stanford University, August 1980 |
| 2ICPR | Second International Joint Conference on Pattern Recognition, Copenhagen, August 1974 |
| 3ICPR | Third International Joint Conference on Pattern Recognition, San Diego, November 1976 |
| 4ICPR | Fourth International Joint Conference on Pattern Recognition, Tokyo, November 1978 |
| 5ICPR | Fifth International Conference on Pattern Recognition, Miami Beach, December 1980 |
| 2IJCAI | Second International Joint Conference on Artificial Intelligence, London, September 1971 |
| 3IJCAI | Third International Joint Conference on Artificial Intelligence, Stanford University, August 1973 |
| 4IJCAI | Fourth International Joint Conference on Artificial Intelligence, Tbilisi USSR, September 1975 |
| 5IJCAI | Fifth International Joint Conference on Artificial Intelligence, Cambridge, Mass., August 1977 |
| 6IJCAI | Sixth International Joint Conference on Artificial Intelligence, Tokyo, August 1979 |

Aggarwal, J.K., Duda, R.O., and Rosenfeld, A. (eds.) [1977]. Computer Methods in Image Analysis, IEEE Press & Wiley.

Agin, G.J. and Binford, T.O. [1973]. "Computer Descriptions of Curved Objects," 3IJCAI, pp. 629-640.

Agin, G.J. and Duda, R.O. [1975]. "SRI Vision Research for Advanced Automation," Second USA-Japan Computer Conference, Tokyo, August 1975, pp. 113-117.

Ali, M., Martin, W.N., and Aggarwal, J.K. [1979]. "Color-Based Computer Analysis of Aerial Photographs," CGIP 9, pp. 282-293.

Arnold, R.D. [1978]. "Local Context in Matching Edges for Stereo Vision," in Proceedings: Image Understanding Workshop, Defense Advanced Research Projects Agency, Cambridge, Mass., May 1978 (Science Applications, Inc., Report No. SAI-79-749-WA), pp. 65-72.

Bajcsy, R. [1980]. "Three-Dimensional Scene Analysis," 5ICPR, pp. 1064-1074.

Baker, H.H. [1977]. "Three-Dimensional Modelling," 5IJCAI, pp. 649-655.

Baker, H.H. [1980]. "Edge Based Stereo Correlation," in Proceedings: Image Understanding Workshop, Defense Advanced Research Projects Agency, Baltimore, April 1980, pp. 168-175.

Ballard, D.H., Brown, C.M., and Feldman, J.A. [1978]. "An Approach to Knowledge-Directed Image Analysis," in Hanson and Riseman [1978a], pp. 271-281.

Balzer, R., Erman, L.D., London, P., and Williams, C. [1980]. "Hearsay-III: A Domain-Independent Framework for Expert Systems," 1AAAI, pp. 108-110.

Barnes, G.H., Brown, R.M., Kato, M., Kuck, D.J., Slotnick, D.L., and Stokes, R.A. [1968]. "The ILLIAC IV Computer," T-C 17, pp. 746-757.

Barrow, H.G., Ambler, A.P., and Burstall, R.M. [1972]. "Some Techniques for Recognizing Structures in Pictures," in Frontiers of Pattern Recognition, S. Watanabe (ed.), Academic Press, pp. 1-29. (Also in Aggarwal et al. [1977], pp. 397-425.)

Barrow, H.G. and Tenenbaum, J.M. [1978]. "Recovering Intrinsic Scene Characteristics from Images," in Hanson and Riseman [1978a], pp. 3-26.

Barrow, H.G. and Tenenbaum, J.M. [1980]. "Interpreting Line Drawings as Three-Dimensional Surfaces," 1AAAI, pp. 11-14.

Barrow, H.G. and Tenenbaum, J.M. [1981]. "Computational Vision," Proceedings of the IEEE 69, pp. 572-595.

Binford, T.O., Brooks, R.A., and Lowe, D.G. [1980]. "Image Understanding Via Geometric Models," 5ICPR, pp. 364-369.

Blum, H. [1967]. "A Transformation for Extracting New Descriptors of Shape," in Models for the Perception of Speech and Visual Form, W. Wathen-Dunn (ed.), MIT Press. (Also in Aggarwal et al. [1977], pp. 153-171.)

Bobrow, D.G. and Winograd, T. [1977]. "An Overview of KRL, a Knowledge Representation Language," Cognitive Science 1, pp. 3-46.

Bolles, R.C. [1976]. "Verification Vision Within a Programmable Assembly System," AIM-295, STAN-CS-77-591, Computer Science Dept., Stanford University.

Bolles, R.C. [1980]. "Locating Partially Visible Objects: The Local Feature Focus Method," 1AAAI, pp. 41-43.

Brady, J.M. and Wielinga, B.J. [1978]. "Reading the Writing on the Wall," in Hanson and Riseman [1978a], pp. 283-301.

Brice, C.R. and Fennema, C.L. [1970]. "Scene Analysis Using Regions," AI 1, pp. 205-226. (Also in Aggarwal et al. [1977], pp. 79-100.)

Brooks, R.A. and Binford, T.O. [1980]. "Interpretive Vision and Restriction Graphs." 1AAAI, pp. 21-27.

Brooks, R.A., Greiner, R., and Binford, T.O. [1979]. "The ACRONYM Model-Based Vision System," 6IJCAI, pp. 105-113.

Brooks, T.L. [1980]. "Supervisory Manipulation Based on the Concepts of Absolute vs. Relative and Fixed vs. Moving Tasks," Proceedings of the International Computer Technology Conference, sponsored by ASME, San Francisco, August 1980, pp. 185-196.

Burr, D.J. and Chien, R.T. [1977]. "A System for Stereo Computer Vision with Geometric Models," 5IJCAI, p. 583.

Carterette, E.C. and Friedman, M.P. (eds.) [1975]. Handbook of Perception, Vol. V Seeing, Academic Press.

Cederberg, R.L.T. [1978]. "An Iterative Algorithm for Angle Detection on Digital Curves," 4ICPR, pp. 576-578.

Chien, R.T. and Jones, V.C. [1975]. "Acquisition of Moving Objects and Hand-eye Coordination," 4IJCAI, pp. 737-741.

Clowes, M.B. [1971]. "On Seeing Things," AI 2, pp. 79-116.

Davis, L. [1975]. "A Survey of Edge Detection Techniques," CGIP 4, pp. 248-270.

Duda, R.O. and Hart, P.E. [1972]. "Use of the Hough Transform to Detect Lines and Curves in Pictures," CACM 15, pp. 11-15. (Also in Aggarwal et al. [1977], pp. 204-208.)

Duda, R.O. and Hart, P.E. [1973]. Pattern Recognition and Scene Analysis, Wiley.

Duff, M.J.B. [1978]. "A User's Look at Parallel Processing," 4ICPR, pp. 1072-1075.

Eberlein, R.B. [1976]. "An Iterative Gradient Edge Detection Algorithm," CGIP 5, pp. 245-253.

Erman, L.D., Hayes-Roth, F., Lesser, V.R., and Reddy, D.R. [1980]. "The Hearsay-II Speech-Understanding System: Integrating Knowledge to Resolve Uncertainty," Computing Surveys 12, pp. 213-253.

Eskenazi, R. and Wilf, J.M. [1979]. "Low-Level Processing for Real-time Image Analysis," JPL Report 79-79.

Falk, G. [1972]. "Interpretation of Imperfect Line Data as a Three-Dimensional Scene," AI 3, pp. 101-144.

Faugeras, O.D. [1980]. "An Optimization Approach for Using Contextual Information in Computer Vision," 1AAAI, pp. 56-60.

Fennema, C.L. and Thompson, W.B. [1979]. "Velocity Determination in Scenes Containing Several Moving Objects," CGIP 9, pp. 301-315.

Fischler, M.A. and Elschlager, R.A. [1973]. "The Representation and Matching of Pictorial Structures," T-C 22, pp. 67-92. (Also in Aggarwal et al. [1977], pp. 31-56).

Fram, J.R. and Deutsch, E.S. [1976]. "On the Evaluation of Edge Detection Schemes and Their Comparison with Human Performance," T-C 24, pp. 616-628.

Freeman, H. [1974]. "Computer Processing of Line-Drawing Images," Computing Surveys 6, pp. 57-97.

Freeman, H. and Davis, L. [1977]. "A Corner-Finding Algorithm for Chain-Coded Curves," T-C 26, pp. 297-303.

Frei, W. and Chen, C.-C. [1977]. "Fast Boundary Detection: A Generalization and a New Algorithm," T-C 26, pp. 988-998.

Freuder, E.C. [1980]. "Information Needed to Label a Scene," IAAAI, pp. 18-20.

Fu, K.S. and Swain, P.H. [1969]. "On Syntactic Pattern Recognition," in Software Engineering, J.T. Tou (ed.), Academic Press, pp. 155-182. (Also in Aggarwal et al. [1977], pp. 369-396.)

Fukushima, K. [1975]. "Cognitron: A Self-Organizing Multilayered Neural Network," Biological Cybernetics 20, pp.121-126.

Fukushima, K. and Miyake, S. [1980]. "Neocognitron: Self-Organizing Network Capable of Position-Invariant Recognition of Patterns," 5ICPR, pp. 459-463.

Funt, B.V. [1977]. "Whisper: A Problem-Solving System Utilizing Diagrams and a Parallel Processing Retina," 5IJCAI, pp. 459-464.

Ganapathy, S. [1975]. "Reconstruction of Scenes Containing Polyhedra from Stereo Pairs of Views," AIM-272, Computer Science Dept., Stanford University.

Gennery, D.B. [1980]. "Modelling the Environment of an Exploring Vehicle by Means of Stereo Vision," AIM-339, STAN-CS-80-805, Computer Science Dept., Stanford University.

Gilbert, A.L., Giles, M.K., Flachs, G.M., Rogers, R.B., and U, Y.H. [1980]. "A Real-Time Video Tracking System," T-PAMI 2, pp. 47-56.

Gilchrist, A.L. [1979]. "The Perception of Surface Blacks and Whites," Scientific American 240, No. 3, pp. 112-124, (March 1979).

Golay, M.J.E. [1969]. "Hexagonal Parallel Pattern Transformations," T-C 18, pp. 733-740. (Also in Aggarwal et al. [1977], pp. 172-179.)

Graham, C.H. (ed.) [1965]. Vision and Visual Perception, Wiley.

Grape, G.R., [1973]. "Model Based (Intermediate Level) Computer Vision," AIM-201, Computer Science Dept., Stanford University.

Griffin, M.D., Cunningham, R.T., and Eskenazi, R. [1978]. "Vision-Based Guidance of an Automated Roving Vehicle," paper 78-1294, Proceedings AIAA Guidance and Control Conference, Palo Alto, August 1978.

Grimson, W.E.L. and Marr, D. [1979]. "A Computer Implementation of a Theory of Human Stereo Vision," in Proceedings: Image Understanding Workshop, Defense Advanced Research Projects Agency, Palo Alto, April 1979, pp. 41-47.

Guzman, A. [1968]. "Decomposition of a Visual Scene into Three-Dimensional Bodies," in Proceedings of AFIPS Fall Joint Computer Conference 33, Dec. 1968, pp. 291-304. (Also in Aggarwal et al. [1977], pp. 324-337.)

Hannah, M.J. [1974]. "Computer Matching of Areas in Stereo Images," AIM-239, Computer Science Dept., Stanford University.

Hanson, A.R. and Riseman, E.M. (eds.) [1978a]. Computer Vision Systems, Academic Press.

Hanson, A.R. and Riseman, E.M. [1978b]. "Segmentation of Natural Scenes," in Hanson and Riseman [1978a], pp. 129-163.

Hanson, A.R. and Riseman, E.M. [1978c]. "VISIONS: A Computer System for Interpreting Scenes," in Hanson and Riseman [1978a], pp. 303-333.

Haralick, R.M. [1978]. "Statistical and Structural Approaches to Texture," 4IJCPR, pp. 45-69.

Hirzinger, G. and Snyder, W. [1980]. "Analysis of Time-Varying Imagery for Tracking Moving Objects," Proceedings of the International Computer Technology Conference, sponsored by ASME, San Francisco, August 1980, pp. 26-33.

Horn, B.K.P. [1974]. "Determining Lightness from An Image," CGIP 3, pp. 277-299.

Horn, B.K.P. [1975]. "Obtaining Shape from Shading Information," in Winston [1975a], pp. 115-155.

Horn, B.K.P. [1977]. "Understanding Image Intensities," AI 8, pp. 201-231.

Horowitz, S.L. and Pavlidis, T. [1974]. "Picture Segmentation by a Directed Split-and-Merge Procedure," 2IJCPR, pp. 424-433. (Also in Aggarwal et al. [1977], pp. 101-110.)

Hough, P.V.C. [1962]. "Method and Means for Recognizing Complex Patterns," U.S. Patent 3,069,654.

Hueckel, M.H. [1971], "An Operator Which Locates Edges in Digitized Pictures," JACM 18, pp. 113-125. (Also in Aggarwal et al. [1977], pp. 191-203.)

Hueckel, M.H. [1973]. "A Local Visual Operator Which Recognizes Edges and Lines," JACM 20, pp. 634-647.

Huffman, D.A. [1971]. "Impossible Objects as Nonsense Sentences," in Machine Intelligence 6, B. Meltzer and D. Michie (eds.), Halsted Press, pp. 295-323. (Also in Aggarwal et al. [1977], pp. 338-366.)

Hummel, R.A. and Zucker, S.W. [1980]. "On the Foundations of Relaxation Labelling Processes," 5ICPR, pp. 50-53.

Inokuchi, S. and Nevatia, R. [1980]. "Boundary Detection in Range Pictures," 5ICPR, pp. 1301-1303.

Johnson, E.T. and Goforth, L.J. [1974]. "Metaphase Spread Detection and Focus Using Closed Circuit Television," The Journal of Histochemistry and Cytochemistry 22, pp. 536-545.

Julesz, B. [1971]. Foundations of Cyclopean Perception, University of Chicago Press.

Kanada, T. [1978]. "Region Segmentation: Signal vs. Semantics," 4ICPR, pp. 95-105.

Kelly, M.D. [1971]. "Edge Detection in Pictures by Computer Using Planning," in Machine Intelligence 6, B. Meltzer and D.Michie (eds.), Halsted Press, pp. 397-409. (Also in Aggarwal et al. [1977], pp. 228-244.)

Kruse, B. and Rao, C.V.K. [1978]. "A Matched Filtering Technique for Corner Detection," 4ICPR, pp. 642-644.

Land, E.H. [1971]. "Lightness and Retinex Theory," Journal of the Optical Society of America 61, pp. 1-11.

Levine, M.D. [1978]. "A Knowledge-Based Computer Vision System," in Hanson and Riseman [1978a], pp. 335-352.

Levine, M.D., O'Handley, D.A., and Yagi, G.M. [1973]. "Computer Determination of Depth Maps." CGIP 2, pp. 131- 150.

Lewis, R.A. and Johnston, A.R. [1977]. "A Scanning Laser Rangefinder for a Robotic Vehicle," 5IJCAI, pp. 762-768.

Marr, D. [1978]. "Representing Visual Information," in Hanson and Riseman [1978a], pp. 61-80.

Marr, D. and Hildreth, E. [1979]. "Theory of Edge Detection," AI Memo No. 518, Artificial Intelligence Laboratory, Mass. Institute of Technology, April 1979.

Marr, D. and Poggio, T. [1976]. "Cooperative Computation of Stereo Disparity," Science 194, pp. 283-287.

Martelli, A. [1976]. "An Application of Heuristic Search Methods to Edge and Contour Detection," CACM 19, pp. 73-83. (Also in Aggarwal et al. [1977], pp. 217-227.)

Martin, W.N. and Aggarwal, J.K. [1978]. "Dynamic Scene Analysis," CGIP 7, pp. 356-374.

McCormick, B.H. [1963]. "The Illinois Pattern Recognition Computer ILLIAC III," T-C 12, pp. 791-813.

Milgram, D.L. [1979]. "Constructing Trees for Region Description," CGIP 11, 88-99.

Milgram, D.L. and Bjorklund, C.M. [1980]. "Range Image Processing: Planar Surface Extraction," 5ICPR, pp. 912-919.

Minsky, M. [1975]. "A Framework for Representing Knowledge," in Winston [1975a], pp. 211-277.

Moravec, H.P. [1979]. "Fully Interconnecting Multiple Computers with Pipelined Sorting Nets," T-C 28, pp. 795-798.

Moravec, H.P. [1980]. "Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover," AIM-340, STAN-CS-80-813, Computer Science Dept., Stanford University.

Mori, K., Kidode, M., and Asada, H. [1973]. "An Iterative Prediction and Correction Method for Automatic Stereocomparison," CGIP 2, pp. 393-401.

Nagel, H.-H. [1978]. "Analysis Techniques for Image Sequences," 4IJCPR, pp. 186-211.

Neumann, B. [1978]. "Interpretation of Imperfect Object Contours for Identification and Tracking," 4IJCPR, pp. 691-693.

Nevatia, R. [1976]. "Depth Measurement by Motion Stereo," CGIP 5, pp. 203-214.

Nevatia, R. [1977]. "A Color Edge Detector and Its Use in Scene Segmentation," IEEE Transactions on Systems, Man, and Cybernetics 7, pp. 820-826.

Nevatia, R. and Babu, K.R. [1979]. "Linear Feature Extraction and Description," 6IJCAI, pp. 639-641.

Nevatia, R. and Binford, T.O. [1977]. "Description and Recognition of Curved Objects," AI 8, pp. 77-98.

Nevatia, R. and Price, K. [1978]. "Locating Structures in Aerial Images," 4IJCPR, pp. 686-690.

Nilsson, N.J. [1980]. Principles of Artificial Intelligence, Tioga Publishing Co.

Nitzan, D., Brain, A.E., and Duda, R.O. [1977]. "The Measurement and Use of Registered Reflectance and Range Data in Scene Analysis," Proceedings of the IEEE 65, pp. 206-220.

Nitzan, D., Rosen, C., Agin, G., Bolles, R., Gleason, G., Hill, J., McGhie, D., Prajoux, R., Park, W., and Sword, A. [1979]. "Machine Intelligence Research Applied to Industrial Automation" 9th Report, SRI International, August 1979.

Nudd, G.R., Fouse, S.D., Nussmeier, T.A., and Nygaard, P.A. [1979]. "Development of Custom-Designed Integrated Circuits for Image Understanding," in Nevatia, R. and Sawchuck, A.A., "Semiannual Technical Report," USCIPI Report 910, Image Processing Institute, University of Southern California, Sept. 1979, pp. 120-145.

Ohlander, R., Price, K., and Reddy, D.R. [1978]. "Picture Segmentation Using a Recursive Region Splitting Method," CGIP 8, pp. 313-333.

Ohta, Y., Kanada, T., and Sakai, T. [1978]. "An Analysis System for Scenes Containing Objects with Substructures," 4IJCPR, pp. 752-754.

Otsu, N. [1978]. "Discriminant and Least Squares Threshold Selection," 4IJCPR, pp. 592-596.

Pavlidis, T. [1977]. Structural Pattern Recognition, Springer-Verlag.

Pavlidis, T. [1978]. "A Review of Algorithms for Shape Analysis," CGIP 7, pp. 243-258.

Perkins, W.A. [1978]. "A Model-Based Vision System for Industrial Parts," T-C 27, pp. 126-143.

Perkins, W.A. and Binford, T.O. [1973]. "A Corner Finder for Visual Feedback," AIM-214, STAN-CS-73-386, Computer Science Dept., Stanford University.

Pingle, K.K. and Tenenbaum, J.M., [1971]. "An Accommodating Edge Follower," 2IJCAI, pp. 1-7.

Pinkney, H.F.L. [1978]. "Theory and Development of an On-Line 30 Hz Video Photogrammetry System for Real-Time 3-Dimensional Control," Proceedings of the ISP Symposium on Photogrammetry for Industry, Stockholm, August 1978.

Pratt, W.K. [1978]. Digital Image Processing, Wiley.

Price, K. [1976]. "Change Detection and Analysis in Multi-Spectral Images," Ph.D. Thesis, Carnegie-Mellon University.

Reddy, D.R., Erman, L.D., Fennell, R.D., and Neeley, R.B. [1973]. "The Hearsay Speech Understanding System: An Example of the Recognition Process," 3IJCAI, pp. 185-193.

Reddy, D.R. and Hon, R.W. [1979]. "Computer Architectures for Vision," in Computer Vision and Sensor-Based Robots, G.G. Dodd and L. Rossol (eds.), Plenum Press, pp. 169-186.

Riseman, E.M. and Arbib, M.A. [1977]. "Computational Techniques in the Visual Segmentation of Static Scenes," CGIP 6, pp. 221-276.

Roach, J.W. and Aggarwal, J.K. [1979]. "Computer Tracking of Objects Moving in Space," T-PAMI 1, pp. 127-135.

Roberts, L.G. [1965]. "Machine Perception of Three-Dimensional Solids," in Optical and Electro-Optical Information Processing, J.T. Tippet et al. (eds.), MIT Press, pp. 159-197. (Also in Aggarwal et al. [1977], pp. 285-323.)

Rosenberg, T., Levine, M.D., and Zucker, S.W. [1978]. "Computing Relative Depth Relationships from Occlusion Cues," 4IJCPR, pp. 765-769.

Rosenfeld, A. [1969]. "Picture Processing by Computer," Computing Surveys 1, pp. 147-176.

Rosenfeld, A. [1972]. "Picture Processing: 1972," CGIP 1, pp. 394-416.

Rosenfeld, A. [1973]. "Progress in Picture Processing, 1969-71," Computing Surveys 5, pp. 81-108.

Rosenfeld, A. [1974]. "Picture Processing: 1973," CGIP 3, pp. 178-194.

Rosenfeld, A. [1975]. "Picture Processing: 1974," CGIP 4, pp. 133-155.

Rosenfeld, A. [1976]. "Picture Processing: 1975," CGIP 5, pp. 215-237.

Rosenfeld, A. [1977]. "Picture Processing: 1976," CGIP 6, pp. 157-183.

Rosenfeld, A. [1978a]. "Picture Processing: 1977," CGIP 7, pp. 211-242.

Rosenfeld, A. [1978b]. "Relaxation Methods in Image Processing and Analysis," 4ICPR, pp. 181-185.

Rosenfeld, A. [1979a]. "Picture Processing: 1978," CGIP 9, pp. 354-393.

Rosenfeld, A. [1979b]. Picture Languages, Academic Press.

Rosenfeld, A. [1980a]. "Picture Processing: 1979," CGIP 13, pp. 46-79.

Rosenfeld, A. [1980b]. "Quadtrees and Pyramids for Pattern Recognition and Image Processing," 5ICPR, pp. 802-811.

Rosenfeld, A. [1981]. "Picture Processing: 1980," CGIP 16, pp. 52-89.

Rosenfeld, A. and Weszka, J.S. [1975]. "An Improved Method of Angle Detection in Digital Curves," T-C 24, pp. 940-941.

Roth, S.D. [1978]. "Stereo 3-D Perception for a Robot," Ph.D. Thesis, California Institute of Technology.

Samet, H. and Rosenfeld, A. [1980]. "Quadtree Representation of Binary Images," 5ICPR, pp. 815-818.

Saund, E., Gennery, D.B., and Cunningham, R.T. [1981]. "Visual Tracking in Stereo," Joint Automatic Control Conference, sponsored by ASME, University of Virginia, June 1981.

Schaefer, D.H. [1980]. "NASA End-to-End Data System Massively Parallel Processor," internal memo, Computer Development Section, Goddard Space Flight Center, Greenbelt, Md., May 30, 1980.

Shapiro, L.G. [1979]. "Data Structures for Picture Processing: A Survey," CGIP 11, pp. 162-184.

Shapiro, L.G., Moriarty, J.D., Mulgeankar, P.G., and Haralick, R.M., [1980]. "Sticks, Plates, and Blobs: A Three-Dimensional Object Representation for Scene Analysis," 1AAAI, pp. 28-30.

Shaw, G.B. [1979]. "Local and Regional Edge Detectors: Some Comparisons," CGIP 9, pp. 135-149.

Shirai, Y. [1975]. "Analyzing Intensity Arrays Using Knowledge About Scenes," in Winston [1975a], pp. 93-114.

Shirai, Y. [1978a]. "Recent Advances in 3-D Scene Analysis," 4ICPR, pp. 86-94.

Shirai, Y. [1978b]. "Recognition of Real-World Objects Using Edge Cue," in Hanson and Riseman [1978a], pp. 353-362.

Stefanelli, R. and Rosenfeld, A. [1971]. "Some Parallel Thinning Algorithms for Digital Pictures," JACM 18, pp. 255-264.

Swan, R., Fuller, S., and Siewiorek, D. [1977]. "Cm* -- A Modular, Multi-Microprocessor," National Computer Conference 46, pp. 637-644.

Tanimoto, S.L. and Pavlidis, T. [1975]. "A Hierarchical Data Structure for Picture Processing," CGIP 4, pp. 104-119.

Tenenbaum, J.M. and Barrow, H.G. [1976]. "IGS: A Paradigm for Integrating Image Segmentation and Interpretation," 3ICPR, pp. 504-513. (Also in Aggarwal et al. [1977], pp. 435-444.)

Tenenbaum, J.M., Barrow, H.G., and Bolles, R.C. [1979]. "Prospects for Industrial Vision," in Computer Vision and Sensor-Based Robots, G.G. Dodd and L. Rossol (eds.), Plenum Press, pp. 239-259.

Tsugawa, S., Yatabe, T., Hirose, T., and Matsumoto, S. [1979]. "An Automobile with Artificial Intelligence," 5IJCAI, pp. 893-895.

Uhr, L. [1972]. "Layered 'Recognition Cone' Networks that Preprocess, Classify, and Describe," T-C 21, pp. 758-768.

Uhr, L. [1978]. "'Recognition Cones,' and Some Test Results; the Imminent Arrival of Well-Structured Parallel-Serial Computers; Positions, and Positions on Positions," in Hanson and Riseman [1978a], pp. 363-377.

Ullman, S. [1979]. "Relaxation and Constrained Optimization by Local Processes," CGIP 10, pp. 115-125.

Underwood, S.A. and Coates, C.L. [1975]. "Visual Learning from Multiple Views," T-C 24, pp. 651-661.

Vamos, T., Bathor, M., and Mero, L. [1979]. "A Knowledge-Based Interactive Robot-Vision System," 6IJCAI, pp. 920-922.

VanderBrug, G.J., Albus, J.S., and Barkmeyer, E. [1979]. "A Vision System for Real Time Control of Robots," 9th International Symposium on Industrial Robots, Washington, March 1979, pp. 213-232.

Waltz, D. [1975]. "Understanding Line Drawings of Scenes with Shadows," in Winston [1975a], pp. 19-91.

Ward, M.R., Rossol, L., Holland, S.W., and Dewar, R. [1979]. "CONSIGHT: A Practical Vision-Based Robot Guidance System," 9th International Symposium on Industrial Robots, Washington, March 1979, pp. 195-212.

Wechsler, H. and Sklansky, J. [1977]. "Finding the Rib Cage in Chest Radiographs," Pattern Recognition 9, pp. 21-30.

Wesley, M.A., Lozano-Perez, T., Lieberman, L.I., Lavin, M.A., and Grossman, D.D. [1980], "A Geometric Modeling System for Automated Mechanical Assembly," IBM Journal of Research and Development 24, pp. 64-74.

Weszka, J.S. [1978]. "A Survey of Threshold Selection Techniques," CGIP 7, pp. 259-265.

Wilf, J. and Cunningham, R. [1979]. "Computing Region Moments from Boundary Representations," JPL Publication 79-49.

Winston, P.H. (ed.) [1975a]. The Psychology of Computer Vision, McGraw-Hill.

Winston, P.H. [1975b]. "Learning Structural Descriptions from Examples," in Winston [1975a], pp. 157-209.

Woodham, R.J. [1977]. "A Cooperative Algorithm for Determining Surface Orientation from a Single View," 5IJCAI, pp. 635-641.

Woodham, R.J. [1979a]. "Analyzing Curved Surfaces Using Reflectance Map Techniques," in Artificial Intelligence: An MIT Perspective, P.H. Winston and R.H. Brown (eds.), MIT Press, pp. 161-182.

Woodham, R.J. [1979b]. "Relating Properties of Surface Curvature to Image Intensity," 6IJCAI, pp. 971-977.

Wulf, W. and Bell, C.G. [1972]. "C.mmp -- a Multi-Mini-Processor," in Proceedings of the Fall Joint Computer Conference 41 II, pp. 756-777.

Yachida, M., Ikedo, M., and Tsuji, S. [1979]. "A Knowledge Directed Line Finder for Analysis of Complex Scenes," 6IJCAI, pp. 984-991.

Yakimovsky, Y. and Cunningham, R. [1978]. "A System for Extracting Three-Dimensional Measurements from a Stereo Pair of TV Cameras," CGIP 7, pp. 195-210.

Yakimovsky, Y. and Feldman, J.A. [1973]. "A Semantics-Based Decision Theory Region Analyzer," 3IJCAI, pp. 580-588. (Also in Aggarwal et al. [1977], pp. 426-434.)

Yakimovsky, Y., Rayfield, M., and Eskenazi, R. [1976]. "RAPID - A Random Access Picture Digitizer, Display, and Memory System," JPL TM 33-772.

Yamamoto, H. [1979]. "A Method of Deriving Compatibility Coefficients for Relaxation Operators," CGIP 10, pp. 256-271.

Zucker, S.W. [1976a]. "Region Growing: Childhood and Adolescence," CGIP 5, pp. 382-399.

Zucker, S.W. [1976b]. "Relaxation Labelling and the Reduction of Local Ambiguities," 3ICPR, pp. 852-861. (Also in Aggarwal et al. [1977], pp. 445-454.)

Zucker, S.W. [1978]. "Vertical and Horizontal Processes in Low Level Vision," in Hanson and Riseman [1978a], pp. 187-195.

**End of Document**