Machine Vision: Three Generations of Commercial Systems

James L. Crowley

CMU-RI-TR-84-1

The Robotics Institute
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213

25 January 1984

Table of Contents

1 Introduction 1
1.1 Scope and Limitations of this Article 1
2 The First Generation: Binary Vision Systems 2
2.1 Basic Capabilities 2
2.2 System Architecture 3
2.3 Image Representation 5
2.3.1 Thresholds and Histogramming 5
2.3.2 Run Length Encoding 6
2.3.3 Connectivity Analysis 7
2.3.4 Feature Measurements 8
2.4 Object Classification 9
2.4.1 Training 9
3 The Second Generation: Edge-Based Systems 11
3.1 Basic Capabilities 11
3.2 System Architecture 12
3.3 Image and Pattern Representation 13
3.3.1 The Edge Detection Board 15
3.4 Smoothing and Edge Detection 15
3.4.1 Linear Filtering 15
3.4.2 Linear Spatial Smoothing 16
3.4.3 Edge Detection 17
3.4.4 Peak Detection and Labeling 18
3.4.5 Line Description and Vertex Detection 18
3.4.6 Line Fitting 19
3.5 Object Recognition 20
3.5.1 Model Representation 21
3.5.2 Training 22
3.6 Matching 23
3.7 Dynamic 2-D Scene Description 23
4 Dynamic 3-D Scene Analysis Systems 23
4.1 A Proposed System 24
4.2 The Initial Representation 24
4.3 Shape Experts 26
4.3.1 Simple Stereo 26
4.3.2 Generalized Stereo and Occlusion 27
4.3.3 Texture 27
4.3.4 Shape From Shading 27
4.3.5 Shape From Motion 28
4.3.6 The Composite Surface Model 28
4.3.7 3-D Object Models 30
4.4 Object Matching 30
5 Closing Remarks 30

List of Figures

Figure 1: The Architecture of a Binary Vision Module 3
Figure 2: A Typical Histogram of Pixel Intensities of an Image that Contains a Bright Object on a Dark Background. The Threshold is Typically Chosen as the Deepest Part of the Valley 5
Figure 3: Run Length Encoding on the Vision Module is given by Marking the Column Numbers where the Intensity Makes a Transition Across the Threshold. A Pair of Bytes is also Needed to Mark the Start and End of Each Row. 6
Figure 4: Connectivity Analysis Groups Overlapping Runs from Adjacent Rows into Blobs. 7
Figure 5: Selected Features Measured by a Binary Vision Module 8
Figure 6: Two Probability Distributions with Significant Overlap 9
Figure 7: Two Probability Distributions Which Do not Overlap 10
Figure 8: The Architecture of an Edge-Based Vision Module 13
Figure 9: A Block Diagram for a Processing Board for Smoothing, Edge Detection, and Peak Detection. 14
Figure 10: Framework for a Dynamic 3-D Scene Analysis System 25

Abstract:

Since 1980, machine vision systems for industrial applications have enjoyed a rapidly expanding market. The first generation machines are 2-D binary vision systems, patterned after the SRI Vision Module. These systems will soon be joined by a second generation, based on edge description techniques.

Both the first and second generation systems are pattern recognition machines. Research in machine vision is leading towards vision systems that will be able to dynamically model the 3-D surfaces in a scene. This research will lead to a third generation of vision systems which will provide a dramatic increase in capabilities over the first two generations.

This article describes these three generations of vision systems. The algorithms, data structures, and hardware architecture are presented for binary vision systems and edge-based vision systems. A framework is presented for the research problems which must be solved before a commercial vision system can be produced based on dynamic 3-D Scene analysis techniques.

1 Introduction

In 1980, two companies began selling a new type of automation equipment: The "Binary Vision Module". These systems enjoy an increasing market as factory managers and project engineers learn of their capabilities and integrate them into robotic manufacturing systems. This first generation of vision systems will soon be joined by a second generation: the "Edge Based Vision Module". Both of these systems are two dimensional pattern recognition machines. In the near future, current research will lead to a third generation of vision systems which perform dynamic 3-D scene analysis.

A fundamental issue in machine vision research is how to detect and represent visual information. The first two systems described below are based on two approaches to detecting and representing visual information that have competed with each other since the 1960's. These two approaches are often called "segmentation" and "edge detection". Since the late 1970's a new approach to representation has emerged: "multiple resolution representation". At the same time a new theme has emerged as a focus of research: "Dynamic 3-D Scene Analysis". Within the next 5 to 10 years, vision systems which perform dynamic 3-D Scene Analysis should reach the market as commercial vision modules.

1.1 Scope and Limitations of this Article

Machine Vision Systems detect, locate and recognize physical objects using images produced by a TV camera. The classes of objects that a system will recognized are learned by "training" the systems with examples of the object class.

There is not nearly enough room in this article to describe the large variety of systems which might be considered as examples of "machine vision". We have restricted the scope of this article, perhaps arbitrarily, to a class of systems which are useful in robotic manufacturing. We do not discuss such systems as blood cell counters, which are considered by some to have been the first successful commercial spin off of machine vision research.

In this article we focus on three generations of systems which are general purpose and are designed primarily for manufacturing applications. These systems are intended to be purchased by a relatively inexperienced user and applied to a large variety of situations where some arbitrary set of objects must be located and identified, usually to assist the operation of a robotic arm or other automation equipment.

2 The First Generation: Binary Vision Systems

During the mid 1970's researchers at SRI-International began work on a prototype "vision module". The goal of the project was to develop a system which would be an inexpensive, fast and general purpose pattern recognition machine for images. The result of this project became known as the SRI vision module. The SRI vision module was soon developed into a commercial product by two companies, and these became the first commercially available general purpose machine vision systems. These vision modules serve as an excellent example of how a commercial machine vision system can be designed so that it can be easily integrated into a factory-floor robotic system.

2.1 Basic Capabilities

The SRI Vision Module and its descendants are two-dimensional binary vision systems. These systems detect and classify connected binary regions in images, referred to as "blobs". Such systems provide two items of information about selected blobs which are particularly useful for robotic systems:

- a label, and
- the position and orientation.

Labels are assigned to blobs by a statistical pattern recognition algorithm. The most valuable aspect of labeling blobs is that it permits a system to reject or ignore blobs which are not of interest. It also makes it possible for a system to discriminate several different classes of objects.

The biggest drawbacks of these systems are

- they only make assertions about the two dimensional patterns projected by objects, and
- each blob is labeled individually

These systems are two dimensional; They locate and classify patterns in images, not objects. The two-dimensional nature of these system is a serious limitation; it means that they can only be used in a setting where the same stable repeatable pattern is imaged for each object. Because these systems operate on individual binary regions, the applications engineer must carefully design the workspace so that blobs correspond to individual objects. This is typically done by backlighting the object, or by placing dark objects on a light colored surface (or vice versa).

Processing speed for images is typically on the order of 0.5 to 2.0 seconds per image. Such rates bring these systems into a range where they can be useful in manufacturing environment, but they can also be a serious limitation in some situations. Most systems can operate with standard RS-170 Black and white camera signals and with CID solid state cameras (such as the GE TN-2500). Solid state cameras are prefered because of their ruggedness and because the imaging is much less subject to distortion and drift effects.

2.2 System Architecture

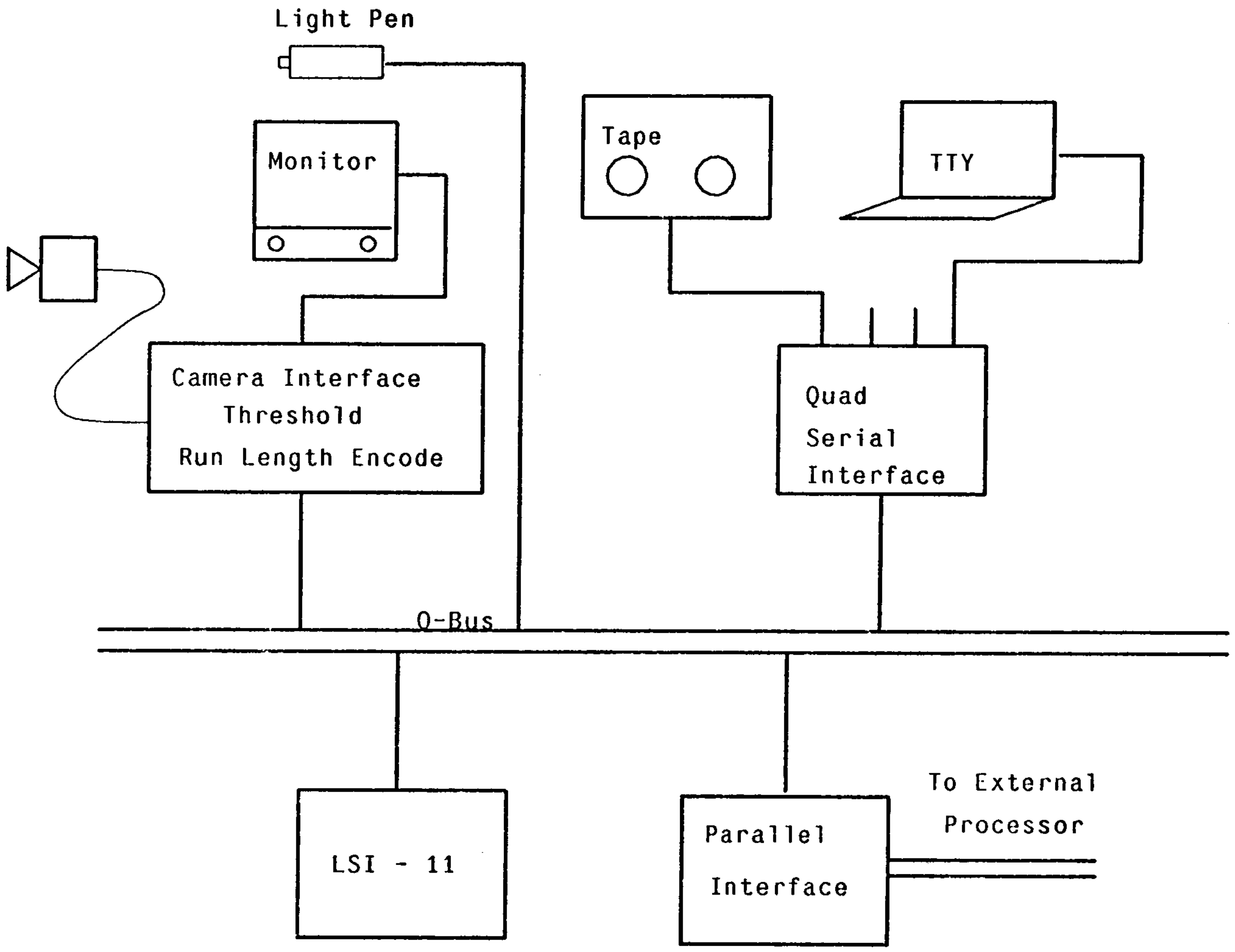

Figure 1: The Architecture of a Binary Vision Module

One of the first commercially available versions of the SRI vision module was the Machine Intelligence Corporation VS-100, which entered the marketplace in 1980 at a price of approximately $25,000. The functional components of the VS-100 are shown in figure 1. The system is implemented on a DEC LSI-11 micro-processor residing on a Q-Bus backplane and is programmed in Bliss-11.[1] In the early versions of the VS-100, the 32K word address space of the LSI-11 was a serious limitation. A newer version is based on a DEC 11/23 with memory management which brings the address space up to 128 K words. This permits the system to be implemented using the RSX-11 operating system, and to support DECNET communications.

[1]A very similar system was available the same year from Automatix. The Automatix system uses essentially the same program implemented on a Motorola 68000 and programmed in Pascal.

The primary intended mode of interacting with this system is via a light pen. A relatively easy to use menu system allows the user to set up and train the system using only the light pen and the monitor. Indeed, our MIC VS-100 did not come equipped with a terminal. The system monitor is used for viewing and monitoring system information, for receiving light-pen commands, for displaying both raw camera images and binary images, for displaying histograms and setting thresholds, and for observing the detection and classification of blobs.

The system contains a DEC DLV-11J quad serial interface board. Serial port 0 is typically connected to a cassette recorder, from which the system software is loaded. Definitions of patterns and other set-up information may also be saved or loaded with this cassette recorder.

Serial port 3, (the console port) supports a terminal interface which provides a variety of possibilities. A relatively simple and easy to use command interpreter is available on this port. By connecting a terminal, the user may request and set various switches and values, as well as direct the system to capture and process images. Even more importantly, this serial port provides the easiest method for interfacing the vision module to a larger system.

A parallel port (DEC DRV-11) is also available for interfacing to another processor. The parallel port provides considerably faster communication. The parallel communications protocol is both more powerful and somewhat harder to use. The usual method for interfacing to this parallel port is via a another DRV-11 board on second LSI-11 or 11/23.

The board which makes the binary vision module possible is the camera interface. This board contains a frame buffer which captures images from the selected camera.[2] This board contains hardware which thresholds the image and then encodes the regions above (or below) threshold in a run-length code. Run length encoding, described below, provides a great reduction in information from the original image thereby making both processing and communications faster. The runs, described below, are passed to the system processor over the system bus. There is also a strobe lamp control signal provided at the time each image is captured.

[2]Our VS-100 supports up to 4 software selectable cameras.

2.3 Image Representation

Binary vision modules of this class derive their speed from the image representation. A 256 by 256 image requires 65,536 bytes of storage. If the image is thresholded, then each byte is replaced by a single bit, giving a reduction of a factor of 8 to 8192 bytes. If a binary image contains only a small number of blobs, then run length encoding can provide a further reduction in the amount of information. For example, the number of bytes required to represent a simple convex blob without holes is twice the number of rows which the blob occupies.

2.3.1 Thresholds and Histogramming

The operation of thresholding replaces each pixel with a 1 or a 0, according to whether it is above or below a certain "threshold" level.

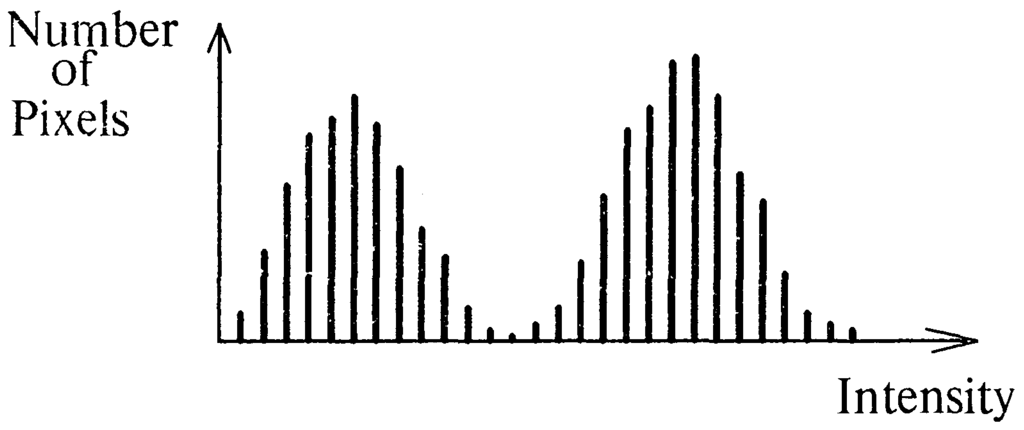

Figure 2: A Typical Histogram of Pixel Intensities of an Image that Contains a Bright Object on a Dark Background. The Threshold is Typically Chosen as the Deepest Part of the Valley

The first step in thresholding an image is to choose the threshold. A method which can work for very simple images is to base the threshold on a "histogram" of the pixel values in the image. If the image contains a bright object on a dark background, then the histogram will exhibit two distinct modes, as illustrated in figure 2. Choosing the deepest part of the valley between the modes will often yield a good threshold. Binary Vision Modules usually have a command which computes and displays the histogram of an image.

It is generally much more reliable to choose the threshold by observing the binary image that results as the threshold is changed. Thus, it is particularly desirable for a vision module to have a mode for interactively selecting the threshold. The VS-100 has a "threshold Selection Mode" in which the threshold may be varied up or down by touching one of two boxes with the light pen. The resulting binary image is automatically displayed on the monitor screen. This is usually the easiest and most reliable way to choose the threshold.

2.3.2 Run Length Encoding

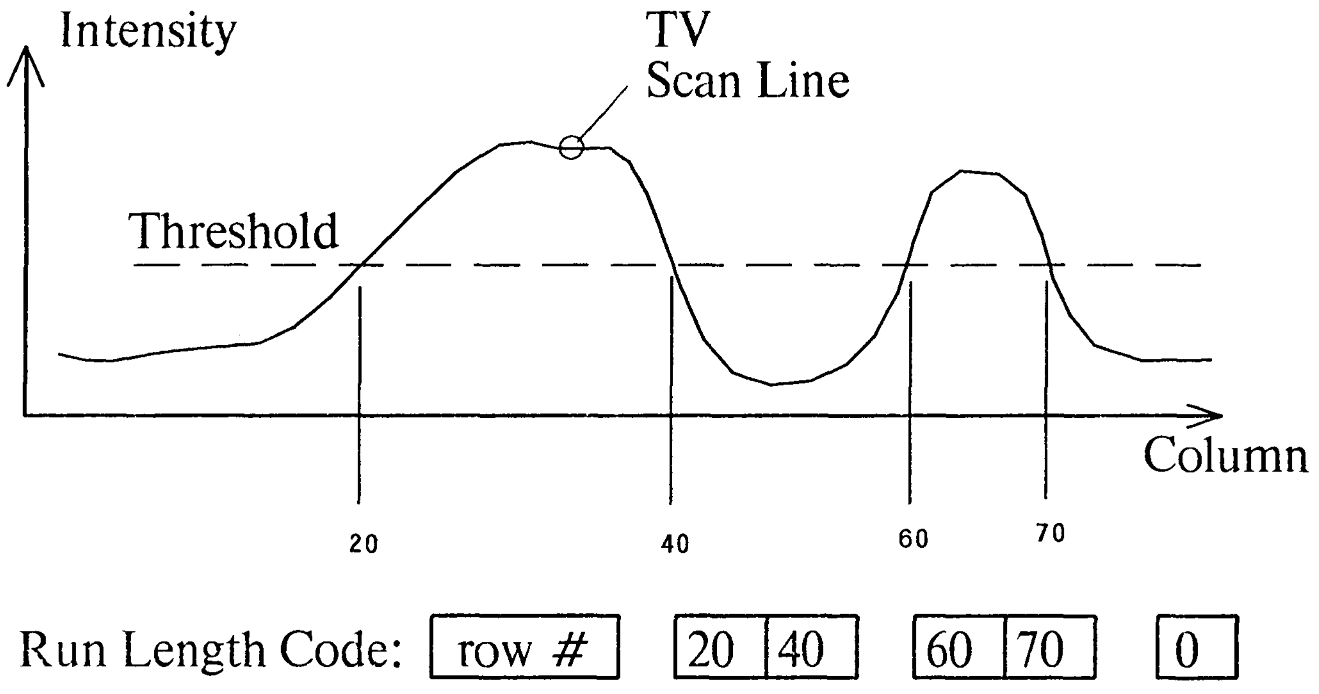

Figure 3: Run Length Encoding on the Vision Module is given by Marking the Column Numbers where the Intensity Makes a Transition Across the Threshold. A Pair of Bytes is also Needed to Mark the Start and End of Each Row.

The principles of run length encoding are illustrated in figure 3. For 256 by 256 (or smaller) images, all of the information may be represented with 8 bit bytes. For each row with any pixels above threshold, the row number is recorded. The pixel values along the row are then compared to the threshold. The column at which the values become greater than the threshold is saved as the start of a run. The column number at which the pixel intensity drops below the threshold marks the end of a run. Thus if a blob is convex, only 2 bytes are needed for each row that it crosses. A byte with the value zero is needed to mark the end of the row.

2.3.3 Connectivity Analysis

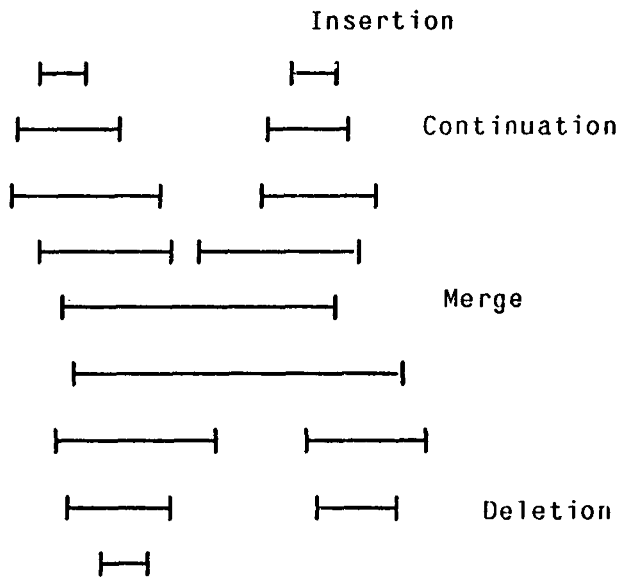

Figure 4: Connectivity Analysis Groups Overlapping Runs from Adjacent Rows into Blobs.

The run codes for each row in an image are passed to the processor through a parallel port. The process starts with the top row and operates sequentially through the rows. The communication is driven by a process called "Connectivity Analysis", which groups overlapping runs from adjacent rows into a data structure called a "blob descriptor". The connectivity analysis algorithm which is described here was developed by Agin for the SRI Vision Module [1]. This algorithm maintains a list of "active" blobs as it obtains the runs from each row. There are 4 cases, illustrated in figure 4, which the connectivity algorithm must handle as it processes each row.

Insertion	When a run occurs which does not overlap with an existing blob, a new blob descriptor must be created. This new blob descriptor is said to be "inserted" into the active blob list.
Continuation	When a run overlaps with the run on a previous row of an existing blob (using 4 neighbor connectivity), then the run must be added to the blob descriptor.
Deletion	If the process passes the columns for an existing blob without finding a run, then the blob descriptor is removed from the list of active blobs and stored.
Merge	When a run is found to overlap with two distinct blobs then these blobs must be merged into a single blob descriptor.

A deletion is always accompanied by a merge of the opposite color. The result of this process is a list of blob descriptors for the image. Most of the features described below are calculated during the connectivity process.

2.3.4 Feature Measurements

Color	Major	Peround	AngMod
NHoles	Minor	RMin	Width
Area	Orientation	RMax	Height
XCent	Perimeter	RMinAng	HoleArea
YCent	TotalArea	RMaxAng	CGDist(**2)

Figure 5: Selected Features Measured by a Binary Vision Module

The vision module labels blobs using a statistical pattern recognition algorithm described below. Statistical pattern recognition is based on the measurement of a number of "features" which describe each blob. Some of the features measured by the VS-100 are listed in figure 5. Most of these features can be calculated during the connectivity analysis process.

The measurement used for most of these features is obvious from the names. **Color** refers to white (above threshold) or black (below threshold). **NHoles** is the number of oppositely colored blobs completely contained inside a blob. **Area** is the area covered by blob, excluding holes, given in calibrated units. These calibrated units are established by calibrating the vision module to a standard size circular pattern. Calibration establishes the distance covered in the object plane by each row and each column in the image.

The features **XCent** and **YCent** are particularly important. These are the x and y coordinates of the center of gravity of the blob. These coordinates are frequently used as the location of the blob. **Major** and **Minor** refer to the second moments of the blob. These coincide with lengths of the major and minor axes of the best fit of an ellipse to the blob. **Orientation** is the angle, in degrees, of the major axis.

Perimeter describes the number of pixels along the perimeter of the blob. **TotalArea** is the total area including hole area covered by the blob in calibrated units. **Peround** is the ratio of the **perimeter** squared to the **area**.

RMin and **RMax** are the minimum and maximum radii from the center of the blob to its boundary.

RMinAng and **RMaxAng** are the directions, in degrees relative to the x axis, of Rmin and Rmax. **AngMod** is the angle between **RMaxAng** and **RMinAng**.

Width and **Height** refer to the dimensions of the bounding box of the blob. A bounding box ranges from the top most to the bottom most rows covered by the blob, and the left most to right most columns. **HoleArea** is simply **TotalArea** - **Area**. The feature **CGDist(**2)** is the square of the distance from the center of gravity of the blob to the calibrated origin of the image.

2.4 Object Classification

The SRI Vision Module and its descendants can be trained to recognize up to 9 classes of blobs. This pattern recognition ability permits the system to discriminate between various objects, as well as filter out noise patterns and objects in which there is no interest.

2.4.1 Training

The probability distribution for each feature in each object class is approximated by a Gaussian (or Normal) distribution. The purpose of training is to determine the average and standard deviation for the features for each pattern class. Training these systems is extremely easy and can be done by just about anyone. Training is accomplished by putting the system into Training Mode with the light pen. The user then places examples of the pattern class in front of the camera and touches the "Process Image" pad on the monitor menu with the light pen. The system will assume that the largest blob is the blob to be learned, but the user may select alternate blobs if desired. The system prompts the user for the name of the class to which the blob belongs. It is suggested that the system be trained with at least 5 examples from each pattern class.

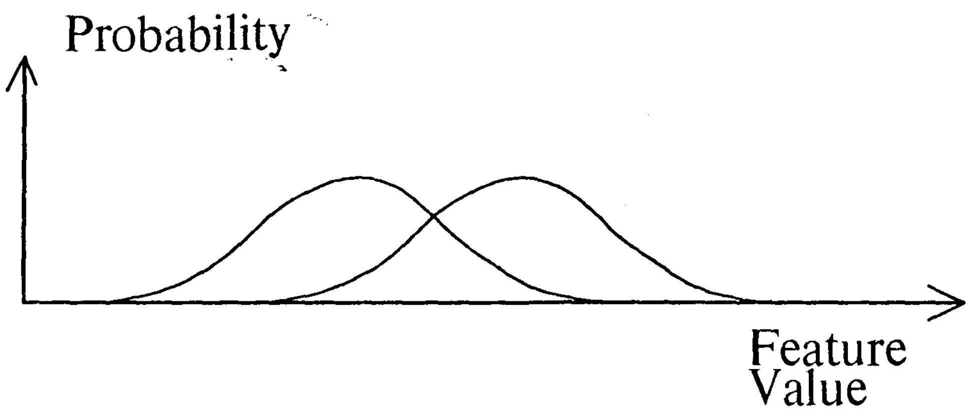

Figure 6: Two Probability Distributions with Significant Overlap

To understand how statistical pattern recognition works, let each feature define an independent dimension in a multi-dimensional space. The fact that many of the features are correlated is ignored. As the system is

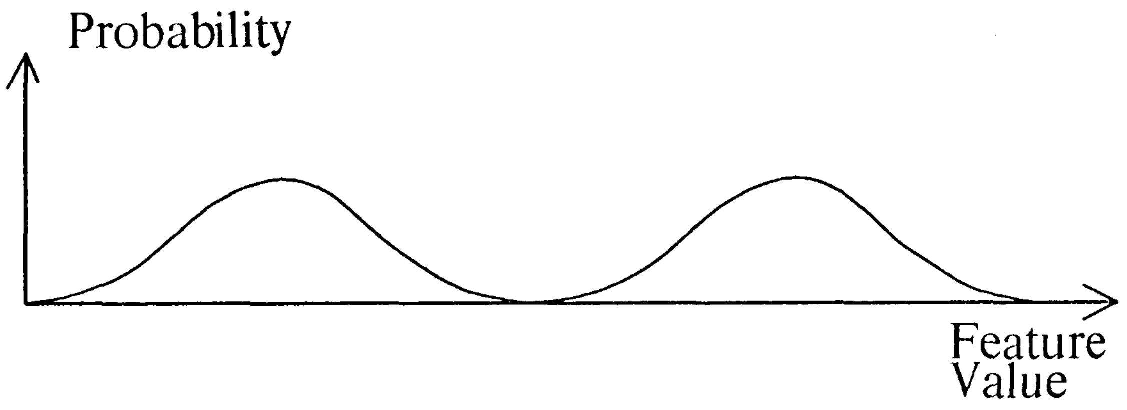

Figure 7: Two Probability Distributions Which Do not Overlap

trained on examples from a pattern class, the average and the standard deviation is computed for each feature. These define a multi-dimensional Gaussian (or Normal) probability distribution for each feature for that pattern class. For simplicity we can ignore the joint probabilities, and consider the distribution in each dimension to be independent.

Let us consider the distributions for a particular feature observed by training with two object classes. If the distributions have a significant overlap, as shown in figure 6, then this feature can not be used to reliably discriminate the two patterns. If the two distributions do not significantly overlap, as shown in figure 7, then this feature can be used to discriminate the objects. The probability that a particular object belongs to either class can be estimated by measuring this feature, and determining the height of each distribution at the feature value.

In the SRI Vision Module an effort was made to determine which of the features can be used to reliably discriminate the pattern classes on which it has been trained. The MIC Vision Module permits the user to select the features to be used. A default list, which includes most of the features, is set in the system software. This default list includes only features which are invariant under position and orientation. The user may alter this list with the light pen.

The selected features are trained to determine a multi-dimensional probability distribution which ignores cross terms. The probability that a blob belongs to each class is computed by measuring the height of this distribution at the multi-dimensional feature vector that is computed for the blob. The class label for which the probability value is highest is assigned to the blob. If none of the probability values are above a threshold, then the pattern is rejected.

Training determines the range of orientations, sizes and deformities over which each class will be recognized. For example, some of the features are size specific. If training examples are given with the blob at a variety of sizes, then these features will be found to have a large standard deviation. Any value of these features will then contribute equally to the likelihood that a pattern belongs to this class. Other features, which are invariant to size will be found to have small standard deviations. These features will automatically play an important role in determining the probability that a given object belongs to this class. Note that if a particular object is can exist at many sizes and all of the training examples are taken with the object at the same size, it becomes likely that the system will not recognize the same object if it occurs at a different size.

3 The Second Generation: Edge-Based Systems

Machine vision systems which operate on edge descriptions of objects have been developed for a number of defense applications. Commercial edge-based systems with pattern recognition capabilities should reach the market during the next year or two.

The goal of edge detection is to find the boundaries of objects by marking points of rapid change in intensity. There is a tendency among some people to refer to systems that operate on edge descriptions of images as "gray-level" vision systems. These systems are not sensitive to the individual intensities of patterns but to changes in pixel intensity. The assumption underlying an edge-based vision system is that the small edge elements detected in an object correspond to the object boundary. This is not always true; highlights, shadows, and surface texture also contribute to the edge elements detected in an image. Nevertheless, in proper lighting conditions, and with many objects, the edges in an image can be used to describe the shape of objects.

As with binary vision modules, the key to making a commercial edge based system is a board which converts the image into a more efficient representation. Boards which compute edge descriptions of images have been designed by a number of manufacturers in recent years. Machine vision systems that employ these boards should reach the market in the next few years.

The system described below is hypothetical. It is based on research performed at the Robotics Institute at C-MU to develop an edge based vision module [22].

3.1 Basic Capabilities

As with binary vision modules, systems based on edge detection are fundamentally two dimensional. They do not describe objects, but the boundaries of objects as detected by the sharp changes in intensity in the image. Of course, it is also quite possible to obtain an edge description from a binary image. Boundary description is somewhat more expensive, computationally, than classifying binary blobs, but it can provide a number of advantages.

The primary benefit of boundary description is that the object to be described does not have to be entirely above some threshold in intensity. This can greatly simplify, and thus reduce the cost of setting up the lighting in the workplace. A second advantage is that articulated objects may be recognized by such a system. Objects which are flexible, or that have pivoting points can still be recognized so long as the necessary structural "landmarks" can be recognized.

Finally, the techniques described below permit more than simple pattern recognition. These techniques permit a system to show where an observed part differs from a model. It is also possible to use such a system for dynamic 2-D scene analysis; that is, to dynamically describe the structure of objects as they change during some manufacturing operation. This can permit planning of actions or monitoring of processes. The ability to perform dynamic 2-D scene analysis stems more from the techniques used, than from whether the representation is binary or edge based. It is, however, easier to implement a dynamic scene analysis system with an edge based system.

3.2 System Architecture

The system architecture for a hypothetical edge-based vision module is given in figure 8. This system is similar in form to the binary vision module. It is suggested that a processor with a larger address space such as a Motorola 68000 or an Intel 8086 be used as the main processor.

The system contains a digitizer which converts analog picture information into a sampled and digitized image. The digitized image is stored in a frame buffer. The contents of the frame buffer are accessed through a special bus by the smoothing and edge detection board. The use of a set of "private" busses between boards solves one of the biggest problems in fast image analysis: communicating the large amount of information in an image.

The results of edge detection are passed to a dedicated micro-processor which does one task: convert the edge intensities into a collection of line segments. These line segments are then passed to the second processor which recognizes objects or maintains a model of the 2-D scene at which the camera is pointed.

As with the binary vision module, a monitor and light pen are provided as an easy to use user interface. Parallel and serial ports are available for integration of the module into a larger system.

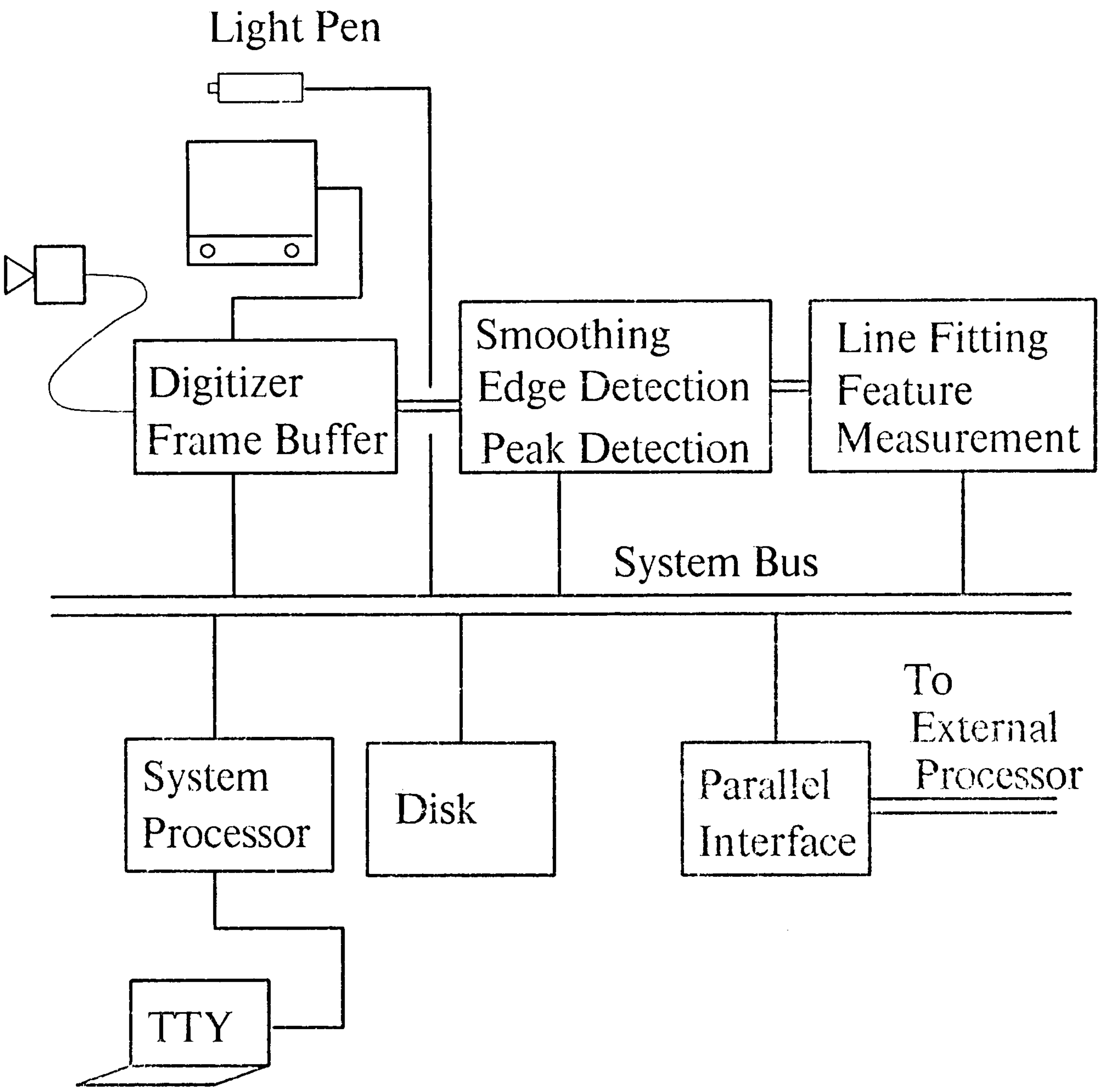

Figure 8: The Architecture of an Edge-Based Vision Module

3.3 Image and Pattern Representation

Edge based vision systems represent the patterns in an image as a network of line segments. These line segments represent linear sequences of peaks in an "edge image". The edge image is, in turn, constructed by applying a local edge detector to the input image.

We have found that a reliable method for detecting edge lines is to first smooth the image, apply the edge

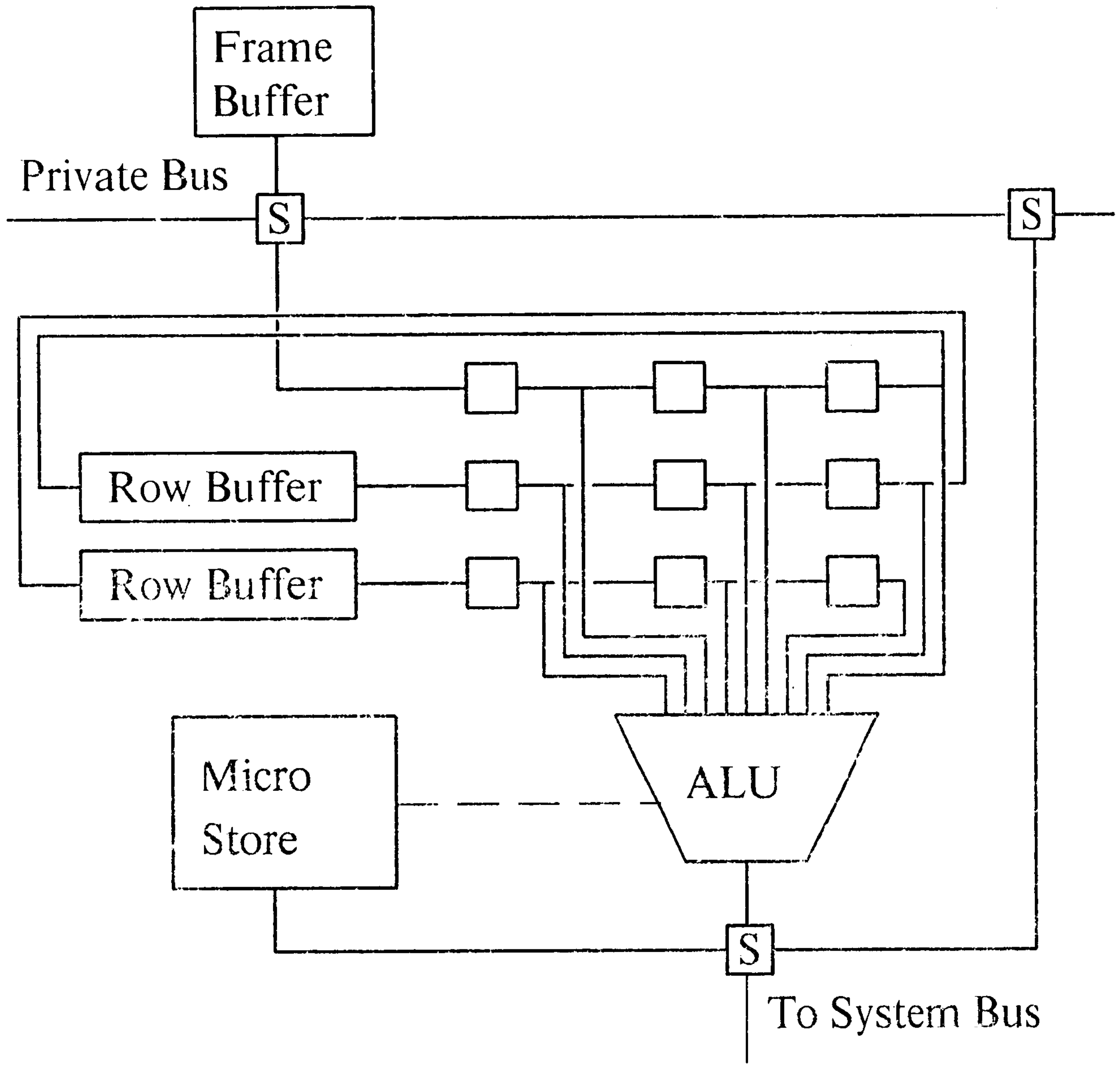

Figure 9: A Block Diagram for a Processing Board for Smoothing, Edge Detection, and Peak Detection.

detector, and then detect 1-D maxima in each of 4 directions. Edge image points which are 1-D maxima in at least 2 directions provide a connected sequence of points along the edge.

3.3.1 The Edge Detection Board

The operations of smoothing, edge detection, and peak detection can all be performed as operations on a 3 by 3 neighborhood. Thus, these three operations are all computed on the same "preprocessing" card in our hypothetical edge-based vision module. This preprocessing card is based on a micro-programmable ALU which can be implemented with bit-slice processors. On the card there is a frame buffer with enough memory to store a single image (65,536 pixels) at 8 bits per pixel. There are 2 "row buffers" which will hold the 2 rows of image data during processing, and shift the data from these rows into 3 sets of 3 neighborhood registers. These 9 neighborhood registers hold the contents of a neighborhood as it is processed.

Image data is read from the frame buffer into the first set of 3 neighborhood registers. As it is shifted out of these registers it goes into the first row buffer. During processing of the next image row it is shifted out of the first row buffer, through the second set of 3 neighborhood registers and then into the second row buffer. During processing of the following image row it is passed into the third set of 3 neighborhood registers and then discarded. Meanwhile, the results are stored back into the frame buffer. After peak detection, the column locations of the peaks on each row are passed to the line fitting processor.

3.4 Smoothing and Edge Detection

In order to detect the boundaries of regions, the image is convolved with an edge detector. If the resulting edge image is sufficiently smooth, a connected sequence of edge points can then be detected by detecting local maxima in the edge image. This smoothness in the edge image can be obtained by smoothing the raw image before convolution with the edge detector. Equivalently, the linear masks of the edge detector can be pre-convolved with the smoothing operator.

3.4.1 Linear Filtering

Both spatial smoothing and edge detection are based on an operation known as discrete convolution. The formula for the discrete convolution of an N by N linear operator s(x,y) with a discrete image p(x,y) is

$$s(x,y) * p(x,y) = \sum_{j=-N}^{N} \sum_{k=-N}^{N} s(j,k)\, p(x-j, y-k)$$

Linear operations, such as convolution, have the very desirable property that they have a set of "eigen-functions", which are the complex exponentials:

$$e^{-j(ux+vy)} = \cos(ux+vy) - j\sin(ux+vy)$$

The result of applying any linear operation, such as convolution, to an eigen-function, is the same eigen-function scaled in amplitude and shifted in phase. The amplitude scaling and phase shift is given by a complex function, called the "transfer function" $H(u,v)$, of the linear operator. That is

$$H(u,v)e^{-j(ux+vy)} = \sum_{i=-N}^{N} \sum_{k=-N}^{N} h(i,k)e^{-j(u(i+x)+v(k+y))}$$

The formula for the transfer function of a discrete filter may be found very simply by factoring and canceling the term

$$e^{-j(ux+vy)}$$

from both sides of the above equation [11]. This yields:

$$H(u,v) = \sum_{i=-N}^{N} \sum_{k=-N}^{N} h(i,k)e^{-j(ui+vk)}$$

The transfer function of a discrete coefficient operator is an infinite periodic function. Only the period $-\pi < u < \pi$ is of interest. The frequency $u = \pm\pi$ is called the Nyquist frequency and corresponds to a cosine of 1 cycle every 2 samples. Any frequency higher than the Nyquist frequency will be *alliased* by the sample rate to appear as a lower frequency.

For small masks with integer coefficients, the transfer function may be easily derived by paper and pencil. The result is a sum of sin and cosine functions in the two-dimensional (u,v) frequency domain.

The cost of implementing a convolution can often be held down by restricting the filter (or mask) to coefficients which are a power of 2. This makes it possible to implement the multiplication in the convolution formula as a shift. It is also possible to design masks which are separable into a convolution of 1-D masks in the row and column direction. This can make it possible to replace an operation which makes one pass through the image, but requires N^2 multiplications with an operation which makes 2 passes through the data, but requires a total of only 2N multiplications.

3.4.2 Linear Spatial Smoothing

Smoothing helps to assure that the edge image, produced by the edge detector, will contain clean connected edge functions along which there is a sharp maximal ridge. This permits the use of local maximum detection (also called "peak" detection) to mark the edge points. Without smoothing, the edge ridge would be more prone to breaking up, or to having parallel ridges and a much more expensive shrinking or relaxation process would be required to locate a connected sequence of edge points.

Historically, the most popular linear smoothing operator has been a uniform 3 by 3 mask. This corresponds to convolution with a 2-D filter whose weights are

$$\begin{vmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{vmatrix} / 9$$

This mask may be separated into the convolution of two 1-D masks of the form [1 1 1] in the row and column directions. Thus its transfer function is also separable into independent components in the u and v directions. The transfer function of the 1-D mask [1 1 1] is $1+2\cos(u)$, which has a zero crossing where $cos^{-1}(u)=0.5$.

We can easily construct a smoothing mask whose transfer function drops monotonically to 0 at the Nyquist rate (u = π) by using the mask [1 2 1] as our separable mask. This filter has a transfer function of $2+2\cos(u)$, which is identically 4 at DC (0 hz) and identically 0 at the Nyquist frequency. This mask also has the property that all of its coefficients are powers of 2. If a copy of this mask along the rows is convolved with a copy along the columns, the result is the 2-D smoothing mask:

$$\begin{vmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{vmatrix}$$

The result of convolution with this mask may be normalized so that the maximum gain is 1.0 by dividing by the sum of the coefficients. Because the sum of the coefficients is 16, this division can be implemented by simply shifting each convolution result to the right by 4 bits.

There are a number of techniques that can be used to compute a fast convolution with this mask. These include separability, the fact that all of the coefficients are powers of 2, and the fact that the 1-D kernel mask [1 2 1] can be implemented as a cascaded of 2 convolutions with the smaller kernel [1 1].

3.4.3 Edge Detection

During the late 1960's and early 1970's much research was focused on techniques for detecting edges. While there are a variety of edge detection operators described in the literature, one of the simplest and most reliable is the operator which has become known as the Sobel Operator [6]. This operator consists of 2 masks which are 90° rotations of each other, followed by a magnitude estimation step. The two masks of the sobel operator are composed of two separable kernels [1 0 -1] (a first difference) and the smoothing kernel [1 2 1]. The transfer function of the kernel [1 0 -1] is $-2j$ Sin(u), which approximates a first derivative for all but high spatial frequencies (small forms). The masks of the Sobel Operator are:

$$\begin{vmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{vmatrix} \qquad \begin{vmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{vmatrix}$$

If we refer to these masks as $m_1(x,y)$ and $m_2(x,y)$, then for picture $p(x,y)$ the edge function $e(x,y)$ is given by the following formula:

$$e(x,y)=\sqrt{(m_1(x,y)*p(x,y))^2+(m_2(x,y)*p(x,y))^2}$$

The square root of the sum of the squares can be approximated by a sum of the maximum of the absolute values causing a slight loss in sensitivity to diagonal edges.

3.4.4 Peak Detection and Labeling

A linear sequence of points where the edge detection operation produces a local maximum can be found by a simple local peak detection test. In this test, the edge values in each 3 by 3 neighborhood are compared to find local maxima in each of the 4 possible directions (horizontal, vertical, and the 2 diagonals). Edge points which are greater than or equal to both neighbors in 2 of the 4 directions are marked as an edge point. The edge value is also compared to a small threshold (say 5) to avoid responding to blank regions and small round-off errors.

After peak detection, peak points can be labeled based on the configuration of peaks in their immediate neighborhood. Labeling the peaks makes the process of fitting lines to the peak points much simpler. In the C-MU Popye edge-based vision system, one of 4 labels are assigned to each edge point based on the configuration of edge points in its nearest 8 neighbors. An edge point is defined as any point with 2 or more peak flags set. These labels are:

Isolated Point:	An edge point with no other edge points in its neighborhood.
End Point:	An edge point that terminates a line.
Line point:	An edge point interior to a line.
Node:	All other edge points. Typically junctions and small blobs.

The actual neighborhood configuration for each label were explicitly defined.

The results of peak marking and labeling are passed to the line fitting board. Unless the image is very cluttered, the most efficient coding for this communication is to pass the column numbers on which a peak has been marked for each row. The peak label can be appended as a second byte of information.

3.4.5 Line Description and Vertex Detection

Constructing a line segment description does not require an extremely large amount of memory or an unusual processor architecture. These processes described below can be implemented on a dedicated micro-processor with on-board memory, or on the general system processor.

A "simple to implement" line extraction algorithm was used in the C-MU Popye system. This system performs a raster scan of the image until an edge is detected. The edge is then tracked and stored as a linked list of edge-points. The edge point labels were used to control the tracking. Edge points are marked as they are tracked by deleting them from the image buffer. Afer tracking an edge, the raster scan resumes from the point at which tracking began.

A technique which operates more efficiently can be designed based on the connectivity analysis algorithm developed by Agin for the SRI binary vision module. This algorithm can be easily integrated into the communication between the edge detection board and the line fitting board. This algorithm involves a raster scan of the edge point frame buffer. As the edge points for each row are detected, they are transmitted to the line fitting board. Points which overlap from one row to the next are grouped into a data structure and held for line detection.

The line encoding algorithm is similar to Agin's binary connectivity analysis algorithm which is described above. However, the algorithm is complicated by the possibility of an edge running along a row. As with connectivity analysis, the process maintains a list of "active edges" as it scans each row.

Insertion	This is the case where an edge point has no active edge as a neighbor on the previous row, or to the left on the same row. A new edge is inserted in the active edge list. If there is an edge along the upper row a pointer is made to this edge.
Continuation	When an edge point has a neighbor on the upper row, or to the left on the same row, its direction is added to the end of the edge list.
Deletion	When the end of an edge has no neighbor at this row, its structure is removed from the list of active edges.
Fork	When an edge point has 2 lower neighbors, the edge is removed from the active list and 2 new active edges are inserted. Pointers are made between these edges.
Merge	When 2 edges come together at a single edge point, both edges are removed from the active edge list, and a new edge structure is inserted. Pointers are made between these three edge structures. A filter process will eliminate the new edge if it contains only one point.

3.4.6 Line Fitting

The result of edge description is processed by a recursive line fitting algorithm. As with the Sobel Edge Detector, this algorithm is described in the textbook by Duda and Hart [6].

Recursive line fitting begins by computing the equation of the line which connects the end points of the edge. The line equation has the form:

$$Ax + By + C = 0.$$

There is a very simple analytic formula for this line equation which may be found in many textbooks on analytic geometry [20].

The recursive line splitting algorithm travels along the connected edge points to find the point that is furthest from the line. The furthest point is found by evaluating the line equation $Ax + By + C$ at each edge point (x,y). The value of this expression will rise monotonically with distance from the line. The point where the absolute value is a maximum is selected as a break point.

The distance to the line at the break point may be computed from line equation. If this value is above a tolerance, then the procedure calls itself recursively on each side of the break point. When the furthest point is within the tolerance, then a line segment is recorded. The tolerance determines how accurately the lines must describe the data. The result is a linked list of line segments which describe the edge.

The connectivity between the line segment lists was established during the edge coding stage. This connectivity may be maintained for the line segment lists. Connections to nearby vertices may also be made by searching for end points within a distance tolerance.

3.5 Object Recognition

Objects may be recognized by comparing the line segments from an observed image to the line segments in models of objects. The line segment descriptions of objects are sometimes called *structural descriptions*, and the process of comparing them to models to find the most similar model is referred to as *Structural Pattern Recognition* [17]. Structural pattern recognition is based on finding a correspondence between lines in the observed image and lines in the object models. An evaluation function based on the correspondence determines which model best matches the pattern observed in the image.

If the comparison is done purely on the basis of the connectivity of the line elements, then the problem becomes a graph (or sub-graph) isomorphism problem which is known to be computationally expensive.[3] One popular approach has been to represent the models as a grammar and attempt to parse the observed pattern of lines with this grammar. If the observed pattern is a legal sentence in the grammar then it is recognized as an instance of that model. This approach is called *Syntactic Pattern Recognition* [8] and is known to be very sensitive to irregularities and noise in the pattern.

At C-MU, we have developed a matching technique for matching random graphs such as line segment descriptions [22]. This approach is based on the measurement of a set of attributes of the line segments, and the use of Probability Theory to determine the most likely correspondence between segments in the observed pattern and object models. With this approach, each model is composed of elements which have observed probabilities of occurrence. Each element has a set of "recognition features" which are described by

[3]General Subgraph isomorphism has been shown to be N-P Complete!!

probability distributions. Each element also has a set of attributes which are measured at each instance of the primitive and then used as features for higher level recognition. There are also connections to other elements which are described with a probability distribution. As with statistical pattern recognition, the elements and probabilities in the models are learned by a training process.

3.5.1 Model Representation

Object models are composed of a hierarchy of symbols. The attributes of the symbols at each level are defined in terms of the relative attributes of symbols at the previous level. The levels in this hierarchy are:

Level 1:	Line Segments
Level 2:	Vertices
Level 3:	Lists of Vertices.

The process of abstracting hierarchical primitives can be continued for more than 3 levels. However 3 levels seems to provide a rich enough vocabulary of unique primitives for recognizing most simple 2-D edge patterns.

At the lowest level an object model is composed of a set of line segments which represent edges and a set of interconnections between line segments. Each line segment has the attributes of length and orientation and a link to line segments to which it is connected. A search process links disconnected segment endpoints to other disconnected endpoints within a small distance tolerance. Each edge-line segment is defined by the following structure:

Edge-Line Segment:

L:	Length of the line
Θ:	Angle of line segment (0° to 180°)
(x_1,y_1):	Position of the first endpoint
(x_2,y_2):	Position of the second endpoint
Pointer List:	Pointers to connected edge-line segments.

Each instance where two or more line segments meet defines a second level primitive called a vertex. Vertices have both recognition features and attributes. The recognition features and the attributes both depend on the number of lines at a vertex. For example, a vertex with 2 line segments, called a pair, has the recognition features of the relative angle ($\Delta\Theta$) and the length of the two line segments. Each recognition feature is described by a Gaussian (or Normal) probability density function, given by a mean (μ) and a variance (σ^2). There is also a probability of occurrence for the vertex. Vertices have attributes of the number of line segments of which they are composed, and the absolute orientation and position of the entire structure.

Recognition features are used in determining the likelihood that an observed vertex is a particular instance of a vertex in an object model. Attributes are available for calculation of "higher level" features. There is also a list of other vertices to which the vertex is connected. Thus the data structure for representing a vertex of type "pair" is

```
Vertex Type Pair:
        Probability of Occurrence:  p(V)
        Recognition Features:
                Relative Angle:  (μ, σ²)
                Length of Line 1:  (μ, σ²)
                Length of Line 2:  (μ, σ²)
        Attributes:
                Position:  (x,y)
                Orientation:  α
        List of Connected Vertices.
```

Object models are given by a network of vertex primitives whose recognition features are established during training. Vertices are connected to other vertices with which they share a segment. The link between vertices may be described by probability density functions for the distance between the vertices and for the relative orientations of the vertices. In the case of an articulated object, the relative orientations for these links will have a large density function, and hence contribute little to the classification probability.

3.5.2 Training

The purpose of training is to derive the vertex types, probability of occurrence, and the density functions of the recognition features for the vertex primitives. Training also provides the probability density functions for the connections between vertices.

Training proceeds by showing the system instances of each object class. In order to determine whether a pair of line segments in two training samples correspond, the relative position and orientation of the training samples must be specified. This is done by explicitly specifying the same pair of line segment endpoints in each training sample. These endpoints are called handle vertices. The system aligns the training sample to the specified object model and alters the cumulative probability density functions for the vertex recognition features. During training, if no observed vertex is found at the position predicted by a model vertex, a search is made within a small distance (called an acceptance region) for the nearest vertex. This permits the system to tolerate minor distortions in the shape of the pattern. The network of connected vertices, with probability density functions for the position and orientation between connections is also updated with each training sample. In the case of articulated parts, it is necessary to specify a pair of handle vertices for each rigid component.

3.6 Matching

Matching an observed pattern to an object model is a search process. The system must determine the most likely correspondence between each observed vertex and a model vertex. Combinatorial explosion is averted by only evaluating the combinations for which the vertices have a high probability. The process can also be guided by the probabilities to try the most likely positions and orientations first.

During training, a dictionary of vertex primitives may be compiled. Vertex primitives which have a similar mean and standard deviation for their recognition features can be merged into a single entry in the dictionary. A list which gives the object models and the location of the primitive in the object model can be compiled for each vertex primitive. The dictionary can be ordered on the basis of the means of the recognition features.

Candidate models can be selected using this dictionary with a subset of the vertex primitives in a pattern. Each vertex primitive yields a list of model vertices from the dictionary that have a high probability of fitting the observed vertex. This list then provides a list of possible object models. Object models which occur in a large number of the lists generated by the set of primitives are likely candidates for matching. The probability may then be computed that each of these candidate models matches the observed network of vertex primitives. The model that yields the highest probability is the most likely match for the observed pattern.

3.7 Dynamic 2-D Scene Description

A system which can describe images as a network of line segments can be used for other tasks besides recognition. In many situations, the identity of a part is not as important as monitoring the position, orientation, or shape of the part as some system operates on it or near it. By matching between the line segments from consecutive images it is possible to build up a "composite" description of a 2-D scene. This description may be updated as each image is processed. This matching searches within an acceptance region for a line segment with a similar length and orientation. The end-points of the line are updated as each new image is matched. Tracking a rapidly moving object requires that an estimated velocity be attached to a rigid collection of line segments. A dynamic scene description can be used for planning actions by a robot arm, or to monitor a process for the occurrence of some event.

4 Dynamic 3-D Scene Analysis Systems

Vision in a 3-D world is fundamentally a 3-D process. Humans (and other species) employ a form of vision that could be described as dynamic 3-D scene analysis. The design of a machine vision system that dynamically monitors a scene, and interprets the forms in the scene as a collection of 3-D surfaces is rapidly becoming a popular paradigm for machine vision research.

There is no doubt that dynamic 3-D scene analysis will require large amounts of computing. However, computer power alone alone is not holding up development of such a system. Before dynamic 3-D scene analysis can even be demonstrated, much less made available on a commercial vision module, a number of difficult theoretical problems must be solved. Progress at this time is rapid; in recent years there has been a convergence toward agreement on what problems must be solved. Much work remains to be done.

4.1 A Proposed System

The following is a framework for a dynamic 3-D scene analysis system proposed by the author as part of his research in this area. The components of this system are described in figure 10. When vision systems which perform dynamic 3-D scene analysis become commercially available, we believe that they will have these components.

4.2 The Initial Representation

Images come into the system as a time sequence of stereo pairs. They are immediately converted into an initial representation which greatly reduces the bandwidth and is designed to facilitate the processing of a set of "shape" experts.

There is evidence that the human visual system uses an initial representation which independently describes the image information at a number of resolutions. Such a representation can be provided by a "Laplacian Pyramid", which provides a sequence of band-pass versions of each image. Matching information in two or more images is fundamental to the operation of most of the shape experts. A multiple resolution representation has been found by a variety of researchers to greatly simplify the problem of matching image information [15], [16]. The one shape expert which does not rely on matching is the "shape from shading" operation. However, it has recently been shown that surface curvature may also be determined using a Laplacian operation [18].

A particularly suitable "Laplacian Pyramid" is provided by the Difference of Low-Pass (DOLP) transform [4]. The DOLP transform expresses an N by N image as a set of $2 \log_2 N$ band-pass images such that the sum of the band-pass images, plus a low-pass residue, yields the original image. This proves that the DOLP transform is reversible and thus does not lose any information in the image. The band-pass filters in the DOLP transform are a set of copies of a prototype filter which have been exponentially scaled in size. All of these filters are defined as difference of 2 circularly symmetric low-pass filters which differ in size by a factor of square root of 2. A fast algorithm has been developed which computes a DOLP transform of an image composed of M samples in 3M additions and multiplications. The result of the transform occupies 3M storage locations [3].

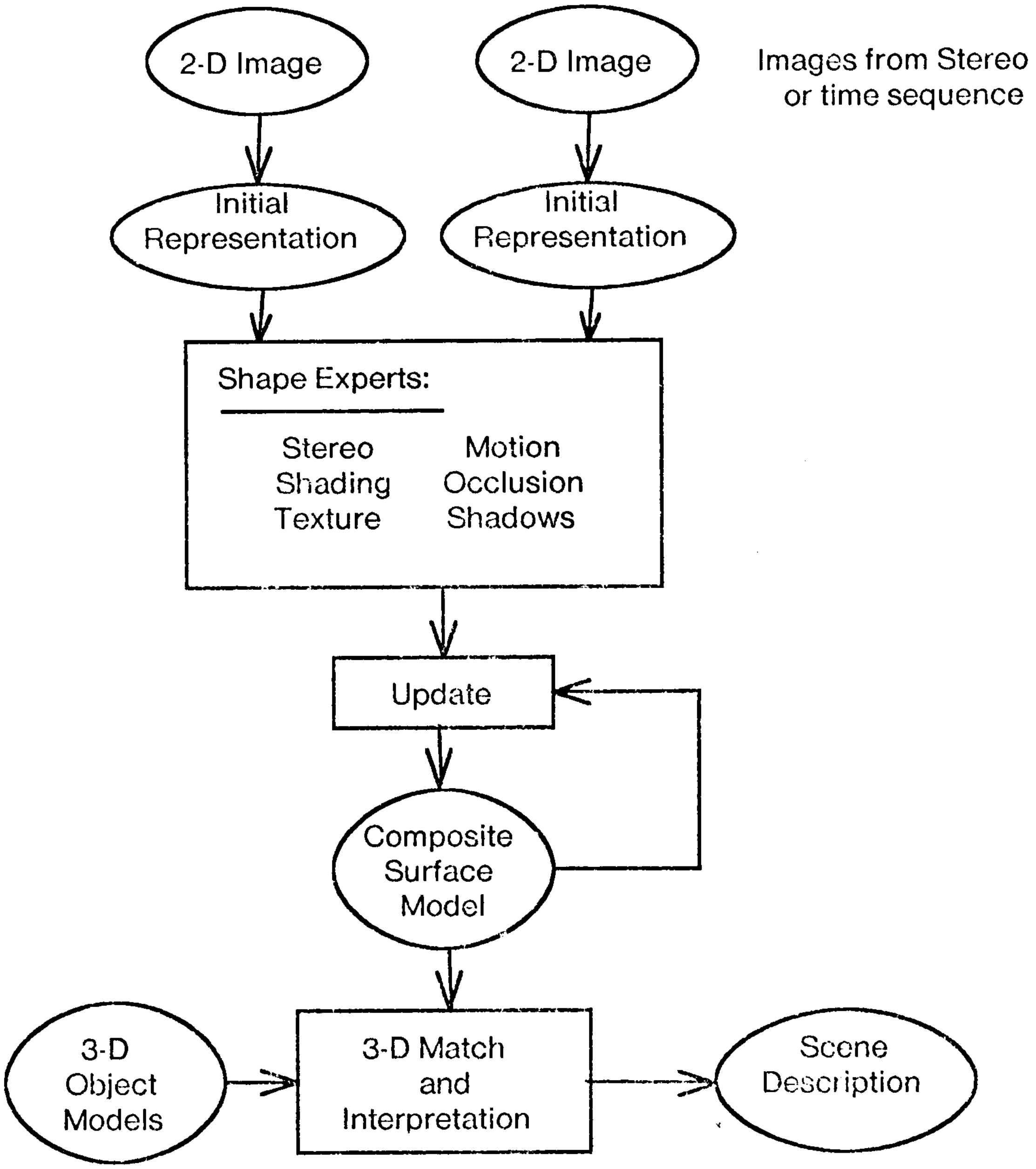

Figure 10: Framework for a Dynamic 3-D Scene Analysis System

Local positive maxima and negative minima (Peaks) in each band-pass image mark places where the band-pass filters are a close fit to forms in the image. This occurs for individual forms, ends of elongated forms, and at corners. Local positive and negative ridges describe edges and elongated shapes of forms. By detecting the peaks and ridges at each band-pass level, and then connecting them across levels, a tree may be constructed which describes the forms in an image at every resolution. At coarse resolutions there is very little information and matching images between levels is relatively easy. As the resolution becomes finer, more detail emerges, and more information is stored. A matching process can use the coarse resolution levels to detect forms and constrain possible matches. Continuing to higher resolutions provides increasing amounts of detail about the form, and and increasing precision in the matches between forms in different images.

The set of band-pass images in the Laplacian Pyramid forms a 3 dimensional space, whose dimensions are x, y, and resolution. Peaks in the band-pass images occur at adjacent locations at adjacent band-pass images. Connecting adjacent peaks in adjacent band-pass images yields a shape primitive called a "Peak-Path". Peak paths describe the gross structure of forms, ends of elongated forms, and corners. We have recently demonstrated probabilistic 2-D shape matching using Peak-paths as the shape primitives [5]. There is also a continuity across levels of ridge information. Detecting the largest ridge samples across levels yields a shape primitive called a "Ridge-Path". Ridge-Paths describe elongated forms and boundaries of forms. The graph of Peak-Paths and Ridge-Paths provides the initial representation for the shape primitives.

4.3 Shape Experts

The shape experts are a set of independent processes which provide information about the shapes of 3-D surfaces. Each shape expert operates on the initial representation and contributes information to the composite surface description. The information provided by the different shape experts complements and verifies consistency of the surface description. The shape information is represented by a data structure called the "Composite Surface Model". The Composite Surface Model is dynamically maintained and kept consistent by an updating procedure.

4.3.1 Simple Stereo

One of the best understood methods for recovering the shape of surface information is "simple stereo". Simple stereo provides shape information from the difference in position of gray-scale forms in two images taken at the same time from two cameras whose image planes are at a known position with respect to each other. Simple stereo is distinguished from generalized stereo by the fact that the camera geometry provides the constraint that patterns along a known line in one image must match to patterns along a known line in the second image. These lines, called epipolar lines, are defined by the intersection of a single plane with the two stereo images. A variety of techniques has been developed to perform simple stereo matching along epipolar lines, including a system developed at C-MU which uses the peaks from the DOLP transform.

4.3.2 Generalized Stereo and Occlusion

Animals learn much information about surface shape by moving their heads from side to side and forward and backward. Such movements cause corresponding movements in the patterns in the images. The problem of determining 3-D shape from the individual movements of these patterns is called "Generalized Stereo". Generalized stereo includes both detection of occlusion contours, and changes in size of patterns. Occlusion contours generally correspond to the edges of surfaces and are a very important clue about surface shape. The rate of change of the size of patterns as the camera moves forward or backward is an important clue about the distance to the pattern. While the matching problem is harder for generalized stereo, it has been recognized as a very important source of surface information for a 3-D scene analysis system [15], [10].

4.3.3 Texture

Surface texture can also be a very important clue about the shape of surfaces. There are several ways in which surface information can be obtained from changes in texture patterns [13]. The most obvious technique is based on assuming that the underlying texture patterns are all approximately the same size. Statistics about the rate of change of the size of the patterns provides an estimate of the surface orientation. Boundaries between texture regions can be an important clue about the edge of a surface. If the texture pattern is assumed to have a known shape, such as a cirlce, then an estimate of the orientation of the surface on which the pattern resides may be obtained by determining the affine transform that the element has undergone. Contours in a texture can also be an important clue about the shape of surfaces [23].

4.3.4 Shape From Shading

It is possible, in some cases, to estimate the rate of change of surface orientation from the rate of change of the shading on the surface. Shape from shading can provide surface information in just those cases where stereo and texture provide the least information. Several elegant techniques have recently been published to describe how surface orientation may be obtained from changes in shading [18], [24].

The orientation of a patch of a 3-D surface may be described by the two parameters of its gradient often referred to as (p,q) [2]. The (p,q) plane is known as the "gradient space". When a scene is illuminated by a single source, the observed intensity at a pixel is a function of:

- the illumination intensity,
- the angle between the illumination source and the surface patch normal (i), and
- the angle between the line from the camera, and the surface patch normal (e).

For a given configuration of camera, illumination source, and 3-D surface, there is a set of values of (p,q) that are closed and connected, and that correspond to a given intensity at a pixel. This set of values corresponds to a contour in the space of (p,q). If the scene is observed with illumination from a second light source, then a second contour in (p,q) results from the observed intensity. In general, there will be two values for (p,q) that lie on both contours. A third image with illumination from yet another 3-D point will disambiguate the situation and give a unique value of (p,q). This shape measurement technique is known as "photometric stereo". Alternatively, knowledge about the possible shapes of objects can be used to disambiguate the shape from two images.

Photometric stereo is not usually practical because of the difficulty of placing a point light source in three known places, and because most surfaces do not have a truly lambertian albedo. However, the contours of equal reflectance in (p,q) space can be used in another way to obtain local shape information. Given the values of (p,q) for a point, the gray levels along any line from that point map into a contour in (p,q) space, provided that

- no discontinuities in surface orientation are crossed,
- the values of (p,q) are assumed continuous and smoothly varying, and
- the surface reflectance ("albedo") is constant.

Thus, knowledge of (p,q) at a point allows knowledge of relative (p,q) at adjacent points.

4.3.5 Shape From Motion

A rigid object may be described by a set of landmark points on its surface, and a distance and orientation of the vector between these landmark points. When a rigid object moves in a scene, the 3-D length and relative 3-D orientations of these vectors remain constant, while the observed 2-D lengths and relative 2-D orientations change. A sequence of images of a moving rigid object in which a set of keypoints are detected can allow the 3-D lengths and orientations of the lines between keypoints to be determined.

As with stereo, there are two parts to the problem: determining the correspondence of landmark points and inferring 3-D shape from this correspondence information. The initial representation based on peaks and ridges in the Difference of Low-Pass Transform is useful for determining the correspondences of patterns in the images.

4.3.6 The Composite Surface Model

The composite surface model serves as a common data structure into which the ensemble of shape experts place their interpretation of surface shape in the scene. It also permits the shape experts to read the interpretation from the other shape experts so that they may modify or guide their interpretation in ambiguous situations. It is the data structure in which a description is built up from many views taken over time. It is also the data structure in which inconsistent information about surfaces from the different knowledge sources is resolved.

In addition to integrating a surface description from many knowledge sources, the composite surface model is also the data structure on which a variety of "higher level" processes operate. Such processes perform tasks such as:

- Recognize 3-D objects and construct a scene description,
- Plan actions with respect to the scene,
- Monitor the execution of actions,
- Learn the shape of objects, or collections of objects.

The composite surface model is "viewer centered"; it contains a description of surfaces seen from the viewer from a particular perspective, perhaps including surfaces which are temporarily occluded. It is referred to as composite because it is a composition of information obtained, over time, from different views, and from all of the shape experts.

The development of a representation for the composite surface model is currently an important research issue. Ideally, the the composite surface model should be composed of entities which represent surface patches. A number of competing techniques exist for representing these surface patches, each with its own short-comings.

The most obvious implementation for a composite surface model is as a "depth map", that is a 2-D array of the form $z = f(x,y)$. The most obvious problem with this representation is how to represent surfaces that are vertical with respect to the viewer. In this case there are multiple surface points at a given location (x,y). A second problem is an inability to represent surfaces which are temporarily occluded.

One possible implementation for the composite surface model is to represent surface regions as patches which are enclosed in closed contours where the surface shape is discontinuous. Each such patch could be approximated by a plane or a second order curve. A surface description of this form can be implemented as a graph of "surface patch elements", with each element linked to its adjacent neighbors. A planar representation composed of triangles has been developed by Fuchs [9]. A more general scheme which employs polyhedral approximations to 3-D objects has been developed by Faugeras et. Al. [7]. An alternative to representing each patch as a planar or second order element is to represent each patch as a network of surface normals. Such a set of normals can be represented by a spatial proximity graph [12].

4.3.7 3-D Object Models

Objects have 3-D shapes. In general, there is no way to know a-priori the 3-D angle or distance from which an object is likely to be seen. Thus a model is needed which describes the complete 3-D shape of an object. The system must be able to determine from this model what surfaces and surface features will be seen from a given viewing angle. Because such a model is represented independent of viewing angle it should be "object centered". That is surfaces in the model are represented relative to an internal coordinate system.

A powerful method for representing the 3-D shapes of objects is the technique known as "generalized cylinders" developed by Agin [1]. A generalized cylinder is described by three components:

1. a spine, or 3-D curve which is the center axis of the object,
2. a cross section, and
3. a sweeping rule which transforms the cross section as it is swept along the spine.

Objects which have more than one spine are described as a configuration of generalized cylinder models. Agin used parametric functions to represent cross sections. Thus, a generalized cylinder model of an object was a symplification which ignored many small shapes on a surface.

Marr has proposed a scheme for representing shapes as a hierarchy of generalized cylinders [14]. In this representation scheme, the description at each level is kept very simple. A pointer refers to a more detailed description of each component. Brooks has demonstrated a model driven visual interpretation system named ACRONYM which uses such a representation [2].

4.4 Object Matching

Construction of a scene description from a composite surface model involves matching object centered 3-D models to the viewer centered composite surface model. A dictionary of observed forms, similar to that described for matching edge-line descriptions, can be used to select candidate 3-D object models. As individual forms are described, their structure can be used as an index into a dictionary which yields a list of possible object models to which they can match. Models which are suggested by many of the forms can be compared to the observed patches in the Composite Surface Model. Land-marks can be used to determine the position and orientation at which matching is tried.

5 Closing Remarks

Machine vision is both a science and an engineering discipline. As a science it is still in its infancy; in a stage of development where exciting and surprising developments occur every year.

For a reader who is interested in learning more about either Pattern Recognition or Image Analysis we recommend the textbook written by Duda and Hart [6]. This text thoroughly describes both fields and has continued to surprise us by its relevance as new problems have emerged. The reader may find a rather different treatment, more oriented toward image processing, in the classic text by Azriel Rosenfeld [21] or in the newer text by Pratt [19]. Readers who are interested in the problems of dynamic 3-D scene analysis are urged to read David Marr's book [15]. Such readers may also enjoy the perspective given by Eric Grimson in his book on Stereo [10].

References

[1] G.J. Gleason and G.J. Agin.
A Modular Vision System for Sensor-Controlled Manipulation and Inspection.
In *9th Symposium on Industrial Robotics*, pages 57-70. Society of Manufacturing Engineers, Washington D.C., March, 1979.

[2] Brooks, R. A.
Symbolic Reasoning among 3-D Objects and 2-D Models.
AI Journal 16:285-348, 1981.

[3] Crowley, J. L. and R. M. Stern.
Fast Computation of the Difference of Low-Pass Transform.
IEEE Trans. on P.A.M.I., March, 1984.

[4] Crowley, J. L.
A Representation for Visual Information.
PhD thesis, C-MU Dept. of Electrical Engineering, Nov., 1981.

[5] Crowley, J. L. and Sanderson A. B.
Probablistic Matching of Multiple Resolution Image Descriptions.
To be submitted to IEEE Trans on PAMI, 1984.

[6] Duda, R. O. and P. E. Hart.
Pattern Classification and Scene Analysis.
Wiley, New York, 1973.

[7] Faugeras, O. D. Hebert, M. and Pauchon, E.
Segmentation fo Range Data into Planar and Quadratic Surfaces.
Proceedings of the Conf. on CVPR-83 20(10):8-13, June, 1983.

[8] Fu, K. S.
Syntactic Methods in Pattern Recognition.
Academic Press, New York, 1977.

[9] Fuchs, H., Kedem, Z. and Uselton, S. P.
Optimal Surface Reconstruction from Planar Contours.
CACM 20(10):693-702, October, 1977.

[10] Grimson, E.
From Images to Surfaces.
MIT Press, Cambridge, Mass., 1981.

[11] Hamming, R. W.
Digital Filters.
Prentice Hall, Englewood Cliffs, N. J., 1977.

[12] Henderson, T. C.
Efficient 3-D Object Representation for Industrial Vision Systems.
IEEE Trans. on PAMI PAMI-5(6):609-618, November, 1983.

[13] Kender, John R.
Shape from Texture.
PhD thesis, Dept. of Computer Science, Carnegie-Mellon University, November, 1980.

[14] Marr, D. and H. K. Nishihara.
Representation and Recognition of the Spatial Organization of Three Dimensional Structure.
Proc. of R. Soc. London (B) 200:269-294, 1978.

[15] Marr, David.
Vision.
W. H. Freeman and Co., San Francisco, 1982.

[16] Moravec, H. P.
Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover.
PhD thesis, Stanford University, September, 1980.

[17] Pavlidis, T.
Structured Pattern Recognition.
Springer Verlag, New York, 1977.

[18] Pentland , A. P.
Local Computation of Shape.
In *Procceedings AAAI-82,* pages 22-25. AAAI, 1982.

[19] Pratt, William K.
Wiley-Interscience: Digital Image Processing.
John Wiley & Sons, 1978.
page 322.

[20] Purcell, Edwin J.
Calculus with Analytic Geometry.
Meredith Corporation, 1965.

[21] Rosenfeld, A. and Kak A.C.
Digital Picture Processing.
Academic Press, New York, 1976.

[22] Crowley J. L. and A. B. Sanderson.
Probablistic Matching of Edge-Line Descriptions.
To be submitted to IEEE Trans on PAMI, 1984.

[23] Stevens, K. A.
Surface Perception from Local Analysis of Texture and Contour.
PhD thesis, MIT Dept of EE, 1979.

[24] Woodham, R. J.
Reflectance Map Technique for Analyzing Surface Defects in Metal Castings.
AI memo 457, AI Laboratory, MIT, 1978.