

Rochester Institute of Technology

RIT Scholar Works

Theses

1989

A Simulated shape recognition system using feature extraction

Wendy Pan

Follow this and additional works at: <https://scholarworks.rit.edu/theses>

Recommended Citation

Pan, Wendy, "A Simulated shape recognition system using feature extraction" (1989). Thesis. Rochester Institute of Technology. Accessed from

This Thesis is brought to you for free and open access by RIT Scholar Works. It has been accepted for inclusion in Theses by an authorized administrator of RIT Scholar Works. For more information, please contact ritscholarworks@rit.edu.

Rochester Institute of Technology
School of Computer Science and Technology

A Simulated Shape Recognition System
Using Feature Extraction

by

Wendy Pan

A thesis, submitted to
The Faculty of the School of Computer Science and Technology
in partial fulfillment of the requirements for the degree of
Master of Science in Computer Sciences

Approved by: _____
John A. Biles

Edward R. Salem

Peter G. Anderson

May 16, 1989

Permission to Reproduce

Title of Thesis: A Simulated Shape Recognition System
Using Feature Extraction

I, Wendy Pan, prefer to be contacted each time a request for reproduction is made. I can be reached at the following address:

19 Locke Drive
Pittsford, NY 14534

Date: May 16, 1989

ACKNOWLEDGEMENTS

I would especially like to thank Prof. John Biles for his encouragement, guidance, inspiration, and many helpful comments. He also helped me greatly on improving my thesis writing, I am sincerely grateful. I would also like to thank Prof. Edward Salem who reviewed the manuscript and kept me on the right track.

ABSTRACT

A simulated shape recognition system using feature extraction was built as an aid for designing robot vision systems. The simulation allows the user to study the effects of image resolution and feature selection on the performance of a vision system that tries to identify unknown 2-D objects. Performance issues that can be studied include identification accuracy and recognition speed as functions of resolution and the size and makeup of the feature set. Two approaches to feature selection were studied as was a nearest neighbor classification algorithm based on Mahalanobis distances. Using a pool of ten objects and twelve features, the system was tested by performing studies of hypothetical visual recognition tasks.

Key Words: robot vision, feature extraction, simulation,
nearest neighbor algorithm, Mahalanobis distance.

TABLE OF CONTENTS

1. INTRODUCTION	1
2. BACKGROUND	5
2.1. Overview	6
2.2. Data Acquisition	6
2.2.1. Visual Input Devices	6
2.2.2. Other Hardware	7
2.3. Data Processing	7
2.3.1. Segmentation	7
2.3.2. Edge Extraction	8
2.4. Classification	9
2.4.1. Template Matching	9
2.4.2. Feature Extraction	10
2.4.3. Selection of a Feature Subset	15
2.4.4. Nearest Neighbor Classification Algorithm	16
2.5. Simulation by Shape Generation	19
2.6. Summary	19

3. IMPLEMENTATION	23
3.1. System Overview	23
3.2. Simulated Digitization	25
3.2.1. Shape Generation	25
3.2.2. Effect of Grid Size	26
3.2.3. Translation, Rotation and Scaling	29
3.2.4. Controlled Distortion	29
3.3. Classification by Feature Extraction	30
3.3.1. Feature Pool	31
3.3.2. Feature Selection	35
3.3.3. Classification	43
3.4. Flow Charts of Implementation	44
3.4.1. System Data Flow Diagram	44
3.4.2. Hierarchical Process Diagram	44
3.4.3. Create and Store Objects	47
3.4.4. Select Reference Objects	47
3.4.5. Select a Feature Subset	47
3.4.6. Identify Unknown Objects	50
3.5. System Hardware and Software	52
3.6. Summary	53

4. RESULTS AND DISCUSSION	54
4.1. Graphic Representation of Edge-Extracted Objects	55
4.1.1. Objects in the Object Pool	55
4.1.2. Digitized Images without Distortion	61
4.1.3. Types of Distortion	61
4.1.4. Digitized Images with Distortion	62
4.2. Properties of the Objects in the Object Pool ...	63
4.2.1. Average Feature Values and Their Standard Deviations	63
4.2.2. Mahalanobis Distances between Object Classes with All Features	66
4.3. Selection of Reference Objects from the Object Pool	68
4.4. Effect of Resolution	69
4.5. Selection of a Feature Subset	75
4.5.1. Elimination of Redundant Features	75
4.5.2. Selection of Features by Discrimination Ability	79
4.5.3. Mahalanobis Distances between Object Classes with a Feature Subset	82

4.6. Classification of Unknown Objects	84
4.6.1. Generation of an Unknown Object at a Randomly Distorted State	84
4.6.2. Confidence Level on Identification	85
5. CONCLUSION, FUTURE WORK AND OTHER APPLICATIONS	93
5.1. Future Work	93
5.2. Other Applications	95
BIBLIOGRAPHY	96
APPENDIX. Ten Objects in the Object Pool	99

CHAPTER 1

INTRODUCTION

The patterns we encounter fall into two categories: abstract and concrete. Examples of abstract items include ideas and arguments, and the recognition of such patterns is beyond the scope of this study. Examples of concrete items include characters, symbols, pictures, biomedical images, three-dimensional physical objects, speech waveforms and electrocardiograms [BOW84]. In the last couple of decades, extensive interest has focused on two types of concrete pattern recognition problems: optical character recognition and robot vision [BERT86]. This thesis focuses on the robot vision area.

Many robot vision (digital image processing) systems also have been developed. Overviews of these systems are presented in numerous papers [SUET86][ZIMM83][VANG86]. Robot vision systems consist of a computer, an image memory bank that is different from the computer RAM and connected to the computer bus, an interface that connects the camera with the image memory bank, and a sensor controller, which provides the camera with the necessary synchronization signals.

In recent years, the power of digital image processing also has been brought to the IBM PC and compatible computers. Imaging Technology's PCVISION Frame Grabber [PCVI86] is one of these commercially available frame grabbers. A software package is supplied with the PCVISION

Frame Grabber that enables the user to perform low level digital image processing functions. Among them are image averaging, image subtraction, convolution and edge enhancing algorithms.

For a robot vision system to be capable of complex automated assembly and inspection operations, it must be able to perform in real time. The system's aim is to sort randomly oriented objects on production lines at speeds of about 10 objects/sec. As a result, some of the functions, such as edge detection and feature calculations, are performed by dedicated hardware [CAGN86].

Although robot vision systems are being applied increasingly to manufacturing tasks, the installation of a complete vision system is still expensive. Companies are often reluctant to make such a major investment unless they are sure that the system can do the job.

This thesis describes a simulated shape recognition system that can be used to do feasibility studies of the suitability of robot vision systems. There is no need to go through all the steps in building a vision system just for a feasibility study because a simulated system can provide useful data for determining whether to invest in an actual vision system.

For instance, if one wants to differentiate a bolt, a nut and a washer on an assembly line, the first step is to image these objects electronically. This step and some of

the following steps can be simulated by directly feeding the digitized and edge extracted objects into a computer.

Using a real robot vision system, one often needs to image objects at random orientations and locations for a teaching set. This can become tedious and time consuming, but on a simulated system, the "picture" in computer memory can be easily rotated, translated and scaled in a matter of seconds.

By doing simulation, the following questions can be answered before one seriously considers the installation of a real vision system:

(a) What is the minimum required camera resolution, 64x64, 128x128 or ... ? It is easy to digitize an object mathematically at various grid sizes. The results will provide data to help in selecting the right kind of electronic camera.

(b) What is the minimum number of features that need to be examined to recognize a set of objects and what are those features?

(c) What features take a long time to calculate? Can these features be implemented in hardware?

Our objective was to provide a simple but complete simulated shape recognition system to help decide on appropriate resolution requirements and useful feature sets for a robot vision system in a given domain. We have tested our system by examining several 2-D geometrical shapes.

This system has a simulated shape digitizer, procedures to select and calculate feature values, and algorithms to perform classification. Reference objects will undergo controlled distortions to provide a range of feature values. The average value and its standard deviation for each feature of each type of object are then stored for future comparison.

Chapter 2 of this thesis gives an overview of the concepts and various components of a robot vision system. Prior work also is reviewed.

Chapter 3 discusses the ways we have implemented our simulated system. Some of our own ideas are presented here and several flow charts are included to show the details of our system.

Chapter 4 discusses the results of our work. Digitized images with and without distortions are plotted. Feature values and their standard deviations for various objects are tabulated. Mahalanobis distances between object pairs are shown. The algorithms for selecting an effective feature subset are also presented. Finally, identification and confidence levels are examined on several unknown objects.

Chapter 5 concludes our work and also suggests how to improve our system.

CHAPTER 2

BACKGROUND

2.1. Overview

In robot vision, one can divide an entire task into three phases: data acquisition, data processing and decision classification, as shown in Figure 2.1 [BOW84]. In the data acquisition phase, light intensity is measured by a sensor and converted to a digital format suitable for computer processing. The measured data then are used as the input to the data processing phase and are grouped into a set of characteristic features as output. The classification phase is implemented in the form of a set of decision functions. In the following sections, we will discuss each phase in great detail.

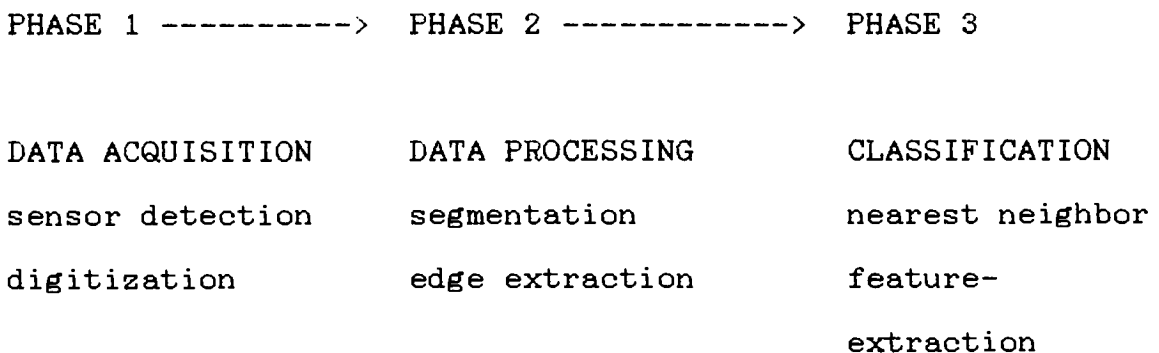


Figure 2.1. Three phases in a robot vision system.

2.2. Data Acquisition

An automatic visual inspection or robot vision system consists of the following subsystems: the part handling system, the optics and the sensor, the illumination, and the computer system. Each of them is briefly described below.

2.2.1. Visual Input Devices

A variety of devices have been used for visual input to robot vision systems. The most popular ones are solid-state array cameras, linear arrays and laser scanners [RAPA88][MUND83][AGIN80].

Solid-state array cameras include CCD (charge-coupled device) and CID (charge-injection device) cameras which contain area arrays of photosensitive elements. Uniformity of response between elements of the array was a problem in earlier devices, but high quality cameras available today have much improved uniformity.

Linear arrays are used where the scene to be scanned is in continuous linear motion, as for example on a constantly moving conveyor. The camera scans a line across the conveyor, and the motion of the part produces the orthogonal direction of the scan. Cost reduction can be achieved by using a linear array instead of an area array.

Laser scanners generally use an arrangement of rotating mirrors, which moves the laser beam across the material perpendicular to the direction of motion. Strategically placed photodetectors measure reflected, scattered, and

transmitted light from various angles. These systems are capable of extremely fast operation, but there are quite expensive.

2.2.2. Other Hardware

In addition to visual input devices, there is other required hardware [SUET86], depending on the task. For instance, when dealing with parts inspection, there should be a part handling system, which consists of a feeding system for the transportation of the parts to the sensor and a separation system for sorting the inspected objects.

Proper illumination is also important. The goal is to provide high contrast images to allow the objects of interest to be isolated from the background by simple thresholding.

Also, a computer is needed for data acquisition and further processing.

2.3. Data Processing

When data acquisition is completed, some processing of data is needed. First of all, the object of interest should be isolated. This step is called segmentation. Secondly, the edges of the object are detected to simplify data processing. This step is called edge extraction.

2.3.1. Segmentation

After a digitized image is captured, the objects of interest are separated from their surroundings. Several

methods exist for segmenting an image [SUET86]. The easiest way is called global thresholding, in which the objects are separated from the background by means of a fixed gray level or threshold value. This thesis is restricted to examine only binary images. If the gray level exceeds a threshold value, then that pixel is "on", otherwise it is "off".

If a frame contains more than one objects, connectivity analysis [ROSE66] [AGIN80] is needed to break an image into its connected components. For instance, if a wrench and a nut are in an image, this analysis will indicate that there are two separate objects so that each can be analyzed. This analysis also detects any holes in an object. For this thesis, segmentation was not necessary because the images that were processed were already segmented.

2.3.2. Edge Detection

Edge extraction [SHIO86][BOIE87] [CAEL87][PAVI75] is probably the most important step in image pattern recognition. The purpose is to reduce the amount of data points for the classification phase.

Edges are regions in which abrupt changes of brightness occur. The edges in an image, then, can be extracted by detecting these changes. A gradient operator [CHEN86], which yields high values in the regions with rapidly changing brightness, usually is used to detect edges. Dedicated hardware for edge extraction is becoming widely available in commercial robot vision systems [SUET86]. For this thesis,

the edge extraction is by-passed, since our mathematically generated shapes constitute only the edges.

2.4. Classification

Two major pattern recognition approaches, template matching and feature extraction, are discussed in this section, although only the latter was implemented in our system. Also two techniques for selecting a proper feature subset, and algorithms for doing nearest neighbor pattern classification [AGIN80][CASH86][GLEN87] are discussed.

2.4.1. Template Matching

Direct image-matching, frequently called "mask matching" or "template matching" [AGIN80], is a pixel-by-pixel comparison of one image (usually a "live" image) with another (a stored image to be used as a reference or model).

A fundamental limitation of template-matching systems is that the two images to be compared must be in perfect alignment for the comparison to be meaningful. If the position of the object to be inspected is not precisely known beforehand, some adjustments must be made. A brute-force technique to accomplish this is to compare images many times, shifting one image with respect to the other between comparisons, to find the amount of shift that maximizes the correlation (minimizes the difference) between the images. This technique is widely used in recognizing printed characters. As a result, commercial optical character

recognition (OCR) machines [TECH86] often require that "skew" is within ± 2 degrees, where skew is the angular deviation from the proper orientation of a character.

Faster methods for template matching exist [HUTT87]. One method is to search only a single row or column of the image before shifting and scaling, selecting the translation that yields the smallest difference in the single row or column, and then calculating the difference between the two entire images.

If one image is rotated with respect to the other, it is possible to perform a rotation in software before matching. However, rotation is much more time consuming than shifting.

To increase speed, one also can match an image with multiple templates [LI86], provided that each template has its own processor and the matching can be done in parallel.

2.4.2. Feature Extraction

Along with template matching, there is another major pattern recognition approach, called feature extraction [BOW84][SHET86]. The objective of feature extraction is to reduce the dimensionality of the measurement space (pixels in a raw image) to a feature space suitable for the application of pattern classification algorithms. During the process of feature extraction, only the features necessary for the recognition process are retained, so that

classification can be implemented on a vastly reduced feature space.

In an N-feature space, an object is represented as an N-dimensional vector, the vector components being the different feature values. Examples of features commonly used in current commercial vision systems are area, perimeter, centroid (the center of gravity), length of the minimum and maximum radius (Rmin, Rmax) from the centroid to the perimeter, the angles of the minimum and maximum radius, first and second invariant moments, and the length and width of the bounding box, which is the smallest rectangle that completely encloses the object.

Two dimensional moments have been used with success for a number of image processing tasks. In the robotics field, moments are used for motion tracking and for object orientation calculations [GOSH83], scene matching [WONG78] and character recognition [CASH87][HU62].

For a 2-D pattern, the moment of order (p+q) is defined as

$$M_{pq} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x^p y^q f(x,y) dx dy \quad (2.1)$$

where p, q = 0, 1, 2,

For a discrete image, the moments can be approximated by

$$M_{pq} = \sum_{m=0}^M \sum_{n=0}^N x^p y^q f(x,y) \quad (2.2)$$

where m and n are the horizontal and vertical dimensions, respectively, of the image, and $f(x,y)$ is the intensity (gray level) at a point (x,y) in the image. $f(x,y)$ can be either 1 or 0 for a binary image. $f(x,y)$ also may have 0 to 2^n gray levels, where n is the number of bits per pixel.

These "raw" moments are information preserving; the original image can be reconstructed acceptably using a finite, but sufficiently large, set of moments computed from the image [TEAG80].

The central moments of an image can be computed using

$$\mu_{pq} = \sum_{x=0}^m \sum_{y=0}^n (x-\bar{x})^p (y-\bar{y})^q f(x,y) \quad (2.3)$$

where

$$\bar{x} = \frac{m_{10}}{m_{00}}, \quad \bar{y} = \frac{m_{01}}{m_{00}}$$

are the coordinates of the centroid of the image. The central moments are invariant with respect to translation of an image.

Another set of moments may be derived from the central moments, which are also invariant with respect to the scale of an image. Denoted by η_{pq} , these normalized central moments are given by

$$\eta_{pq} = \frac{\mu_{pq}}{(R_{av}/R_o)^{p+q}} \cdot \sum_{x=0}^m \sum_{y=0}^n f(x,y) \quad (2.4)$$

where R_{av} is the average distance from the centroid to all "on" pixels. R_0 is a constant used to scale the magnitude of η_{pq} 's to a suitable level. R_0 was set to 2 in the current study. Note that η_{pq} is also invariant with respect to the value of $f(x,y)$.

Hu [HU62] went one step further and developed a set of seven moments that were invariant to translation, scale change and rotation. These moments are usually called "moment invariants" [ZAKA87][CAGN86].

Moment invariants were chosen as part of the features for this thesis, since the calculations are straightforward. In addition, dedicated hardware has already been developed for computing moment invariants [WU86]. Table 2.1 lists the formulas for the seven lowest order invariant moments.

Index	Formula
1	$(\eta_{20} + \eta_{02})$
2	$(\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2$
3	$(\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2$
4	$(\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2$
5	$(\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2]$ $+ (3\eta_{12} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]$
6	$(\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03})$
7	$(3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2]$ $+ (3\eta_{12} - \eta_{30})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]$

Table 2.1. Formulas for the 7 Lowest Order
Invariant Moments

2.4.3. Selection of a Feature Subset

The only guaranteed technique for choosing the best subset of N features from a set of M features is to try all possible combinations. This is computationally impractical for large numbers of features, so heuristic techniques are required. Mucciardi and Gose [MUCC71] compared several techniques on feature selection. Two of those techniques, which are easy to implement, are described here.

The first technique is to arbitrarily choose the first feature and to determine which two classes are most often confused in a multi-class problem. The feature that (when used alone) is the best discriminator between these two classes is the next addition to the set. The procedure is iterative.

The second technique is based on the idea that a feature that is very similar to another already in use adds very little additional discriminatory information. According to this technique, the second feature selected is the one least correlated with the first, which is arbitrarily selected. Subsequent features are those that have the minimum average correlation coefficients with those already chosen.

The above techniques, with slight modifications, were implemented in our system.

2.4.4. Nearest Neighbor Classification Algorithm

Given several reference object classes and an unknown object, the problem is to determine to which object class the unknown belongs. Objects may be thought of as points in an N-dimensional feature space, where N is the number of features. The nearest neighbor technique computes the distance from the unknown point to each of the reference points and chooses the object class closest to the unknown.

Figure 2.2 illustrates a hypothetical case where the number of features available for recognition has been reduced to two. The open shapes indicate feature measurements made on each of several reference objects. The solid shapes mark the centroids computed for each object class. The unknown, designated by the "X" in Figure 2.2, is identified by choosing the class whose centroid is the closest.

Intuitively, the feature that has a smaller variance (standard deviation) should contribute more to the decision process. Therefore, a Mahalanobis distance is suggested, in which each feature is weighted and the weight is the reciprocal of standard deviation.

The Mahalanobis distance [CASH86] between an unknown object and the i-th reference object class is defined as

$$D_m = \sqrt{\sum_{k=1}^n \left(\frac{U(k) - R_i(k)}{\sigma_i(k)} \right)^2} \quad (2.5)$$

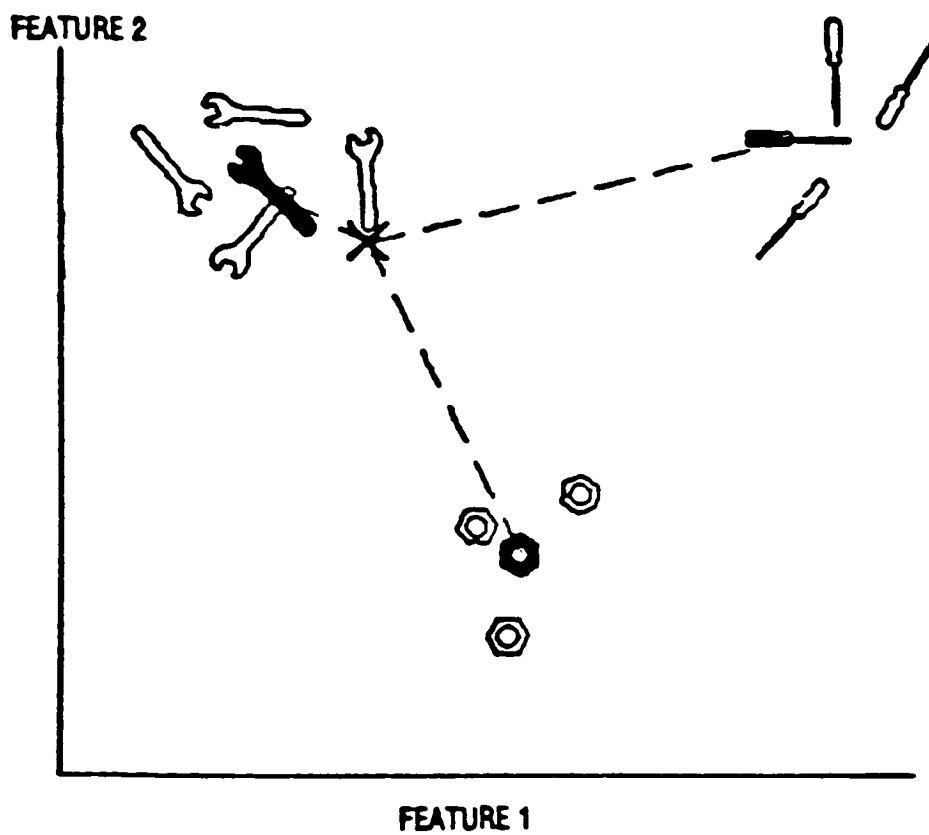


Figure 2.2. A nearest neighbor classifier.

where N is the number of features, $U(k)$ is the k -th feature value of the unknown object, $R_i(k)$ is the average k -th feature value for the i -th reference object class and $\sigma_i(k)$ is the standard deviation of that class. Note that the $U(k)$ and $R_i(k)$ values are normalized by the variance. In the present work, $\sigma_i(k)$ is set to 0.01 if it is less than 0.01.

We further extend the definition of Mahalanobis distance to include the distance between two object classes i and j ,

$$D_m = \sqrt{\sum_{k=1}^n \left(\frac{R_i(k) - R_j(k)}{\sigma_i(k) + \sigma_j(k)} \right)^2} \quad (2.6)$$

Note that during the calculation of Mahalanobis distance, only the average feature values and their variances are needed. This leads to less stored data and a much simplified computation.

Cash and Hatamian [CASH86] studied various classification techniques. They recommended the use of Mahalanobis distance for its adequate recognition rate and easy implementation. This study will use Mahalanobis distance along with the nearest neighbor algorithm for classification.

Other approaches, such as the binary decision tree [AGIN80], has been used for classification. However, we use only the nearest neighbor algorithm in this study because of its robust performance and easy implementation.

2.5. Simulation by Shape Generation

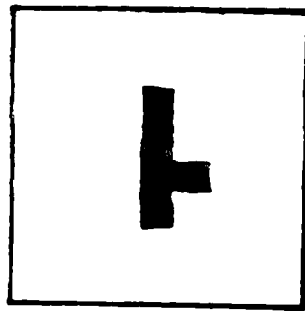
This study tried to bypass image acquisition, segmentation and edge detection. Instead, the digitized edges of a shape were generated mathematically and then processed directly.

Hooper and Klinger proposed a similar idea [HOOP86] when they advocated that artificial pattern generation was a useful technique for providing large banks of data that could be used as test data for pattern recognition experiments. The generated patterns were distorted under control in this thesis to yield a wide variety of samples that were different from, but similar to, the original pattern.

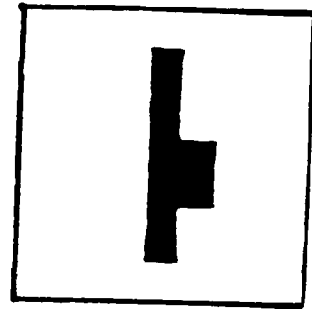
The distortions included linear stretching in either a horizontal or vertical direction, rotation and relocation, blurring and random noise. Figures 2.3 and 2.4 present some of the examples. Figure 2.3 gives an original image and its stretched, relocated and noisy-added images. Figure 2.4 shows a curve before and after blurring.

2.6. Summary

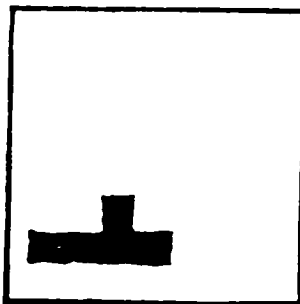
In this chapter, we have reviewed the hardware for the actual robot vision systems. We also have reviewed how to achieve edge extraction. However, edge extraction was simulated in our system by mathematically generating "edge-extracted" objects.



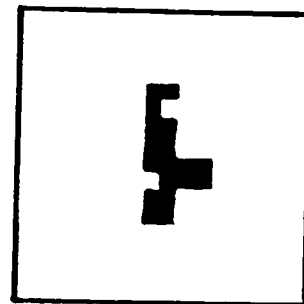
(a)



(b)



(c)



(d)

Figure 2.3. Examples of various distortion schemes.
(a) original
(b) vertical stretching
(c) rotation and relocation
(d) random noise within pattern

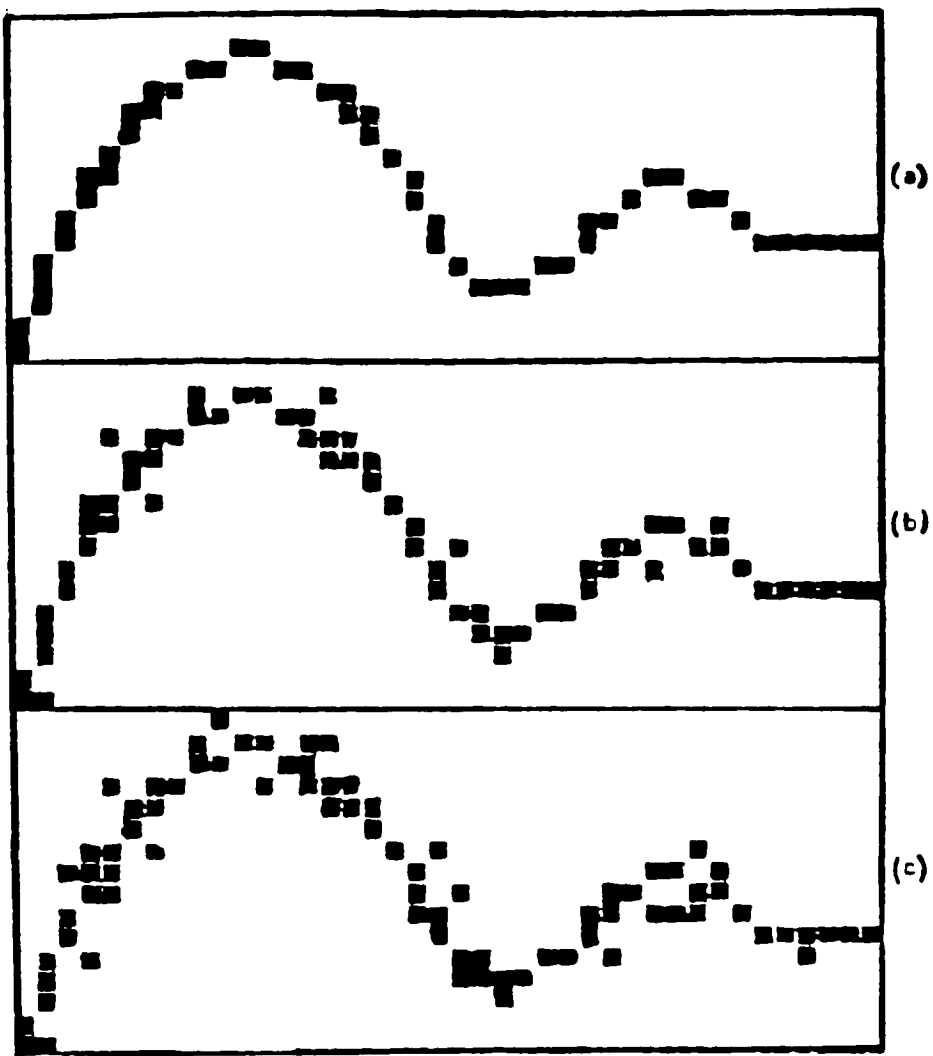


Figure 2.4. (a) the original digitized curve,

(b) the same curve after blurring by adding 10% of the cells adjacent to the curve,

(c) the result of blurring by adding 20% of the adjacent cells.

Two major pattern recognition approaches, template matching and feature extraction were discussed, but only the latter was considered suitable for our purpose. Invariant moments are useful and were chosen as part of the features in the pool.

Reduction the number of features can improve recognition speed. We thus have studied two techniques to automatically select useful features. One technique is to eliminate redundant features and the other technique is to pick a feature, for a pair of object classes, with high discrimination ability.

Identification was achieved by using the nearest neighbor classification algorithm. A Mahalanobis distance was introduced and discussed. This distance can be calculated between two object classes or between an unknown and an object class.

CHAPTER 3

IMPLEMENTATION

3.1. System Overview

In this chapter, we describe the implementation of image simulation and digitization, feature extraction, and unknown identification in our system.

Figure 3.1 shows an overall system data flow diagram. A new object and many variations can be created any time and stored in the object file. The digitized image of an unknown is generated mathematically. The stored feature values and the standard deviations of objects of interest can be retrieved when needed.

The task of recognition is restricted to associating an unknown with one of the selected reference objects. The unknown is classified by comparing feature values of the unknown to those of references. Classification is achieved by the nearest neighbor algorithm.

There are 12 features currently stored in our feature pool. We have derived two rules for selecting an effective feature subset. The system automatically does the selection.

Many other flow charts are provided to show the details of our implementation. Our system hardware and software are also described.

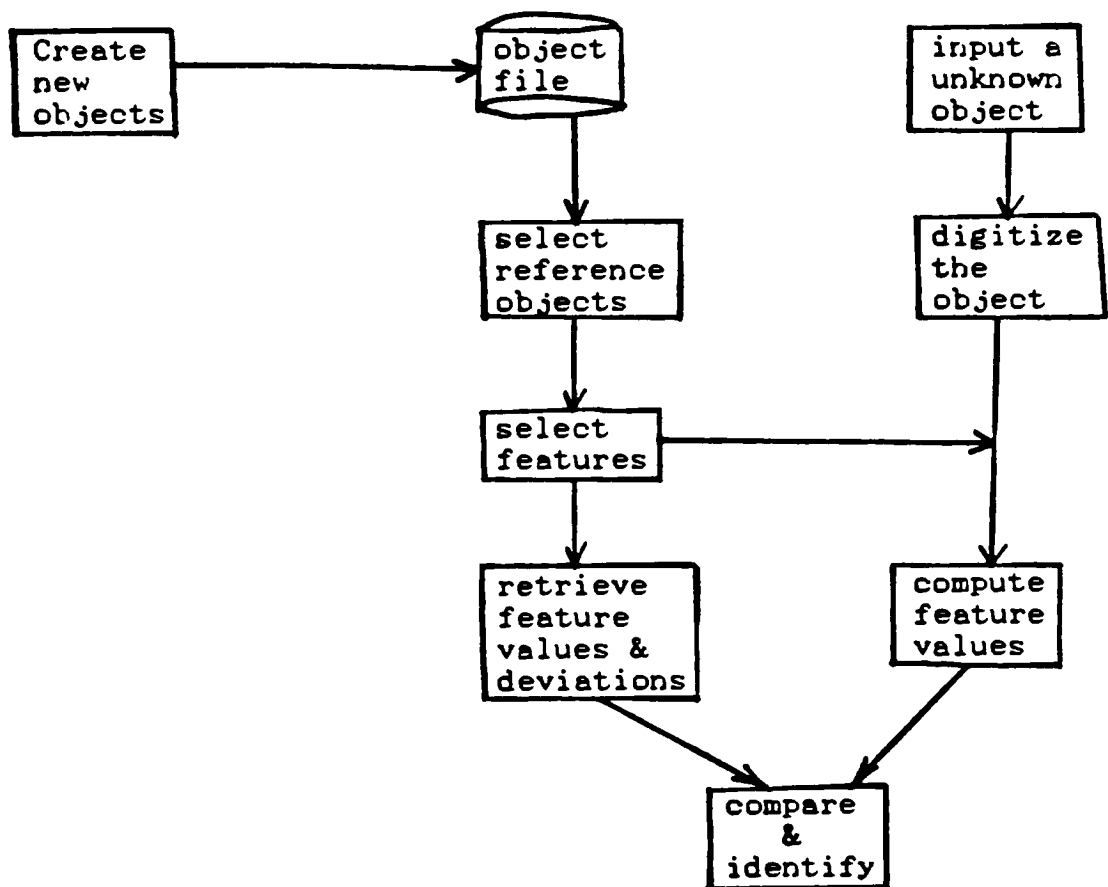


Figure 3.1. System data flow chart.

3.2. Simulated Digitization

Digitization and edge extraction are two important processes in robot vision. This thesis describes a system that simulates the results from these processes and mathematically generates the digitized edges of a shape. The shape undergoes a series of translations, rotations, scaling, and controlled distortions to form a class of similar images.

In this study, we have limited the system to binary images.

3.2.1. Shape generation

Here, we discuss how to generate an "edge extracted" image.

The user first needs to decide the size of an object and then plot the object on graph paper. The coordinates of key points of that object then can be obtained. We believe that any digitized curve can be represented by a series of line segments, although some line segments may be short.

For instance, a triangle is represented by the coordinates of its three vertices. A line segment is defined by two points. Then a linear equation $y=ax+b$ can be determined, and the projections of this line segment on a given grid can be calculated. The projections often will not fall precisely on individual pixels, but the nearest pixels to the projections will be turned "on". The mathematics here is straightforward, and we wrote an effective procedure to





perform this digitization task. An example is shown in Figure 3.2.

As shown in Figure 3.3, for a triangle with three vertices $(2,0)$, $(3,1)$ and $(-1,5)$, four points need to be specified, $(2,0)$, $(3,1)$, $(-1,5)$ and $(2,0)$, to indicate that it is a closed curve. Note that the first and the last points are the same.

Our system was designed in such a way that it automatically digitizes three line segments, $(2,0)$ to $(3,1)$, $(3,1)$ to $(-1,5)$ and $(-1,5)$ to $(2,0)$.

We also wrote a procedure for generating a digitized circle, which is defined by a center coordinate and a radius, based on Bresenham's circle algorithm [BRES77].

The following shapes will be examined by our system:


in which,  simulates a washer,  and  represent nuts, and others are simple geometrical shapes.

3.2.2. Effect of Grid Size

One typical goal for a user of our system would be to study the effect of camera resolution. This can be simulated by changing the grid size. This system was designed such that digitization can be performed at three grid sizes, i.e., 16×16 , 32×32 , and 64×64 . The user needs to input only the key points of a shape in any of the above sizes, and the



Figure 3.2. Digitization of a line segment.

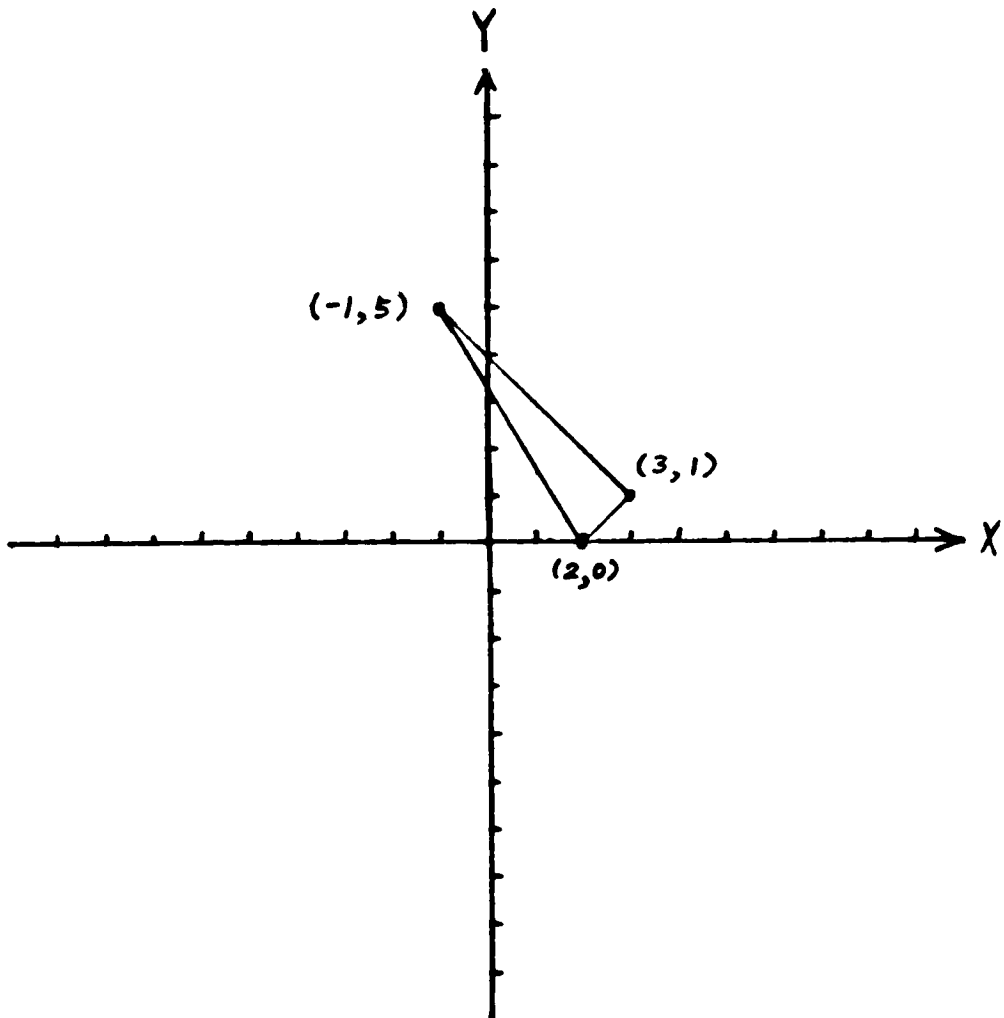


Figure 3.3. A triangle represented by three vertices.

system will scale up (or down) the coordinates of those key points accordingly and then perform digitization.

Objects are often easy to classify at a higher resolution. However, this benefit may diminish rapidly beyond a certain limit. This simulation can help the user to determine a proper cost-effective resolution by allowing him to experiment with different resolutions and object sizes.

3.2.3. Translation, Rotation and Scaling

When the pictures of an object are taken at various locations, orientations and camera-to-object distances, the results of digitization are all different. Therefore, this system provides translation, rotation and scaling operations to perform these simulation tasks. By performing various transformations, a series of similar feature values will be generated, which, along with those from original and other controlled distortions, are considered to belong to the same class. An average value and standard deviation for that class then can be determined.

3.2.4. Controlled Distortion

Distortion operations are used to simulate device inaccuracy. For instance, a circle may be distorted slightly to become an ellipse, and the edges may be noisy. In this system, an object can be shortened along the x direction before it is digitized.

Blurring is simulated by using a random number generator. During the calculation of intercepted points

between a shape and a grid, a small (e.g., -2 to 2 for a grid size of 64x64) random number is added to the calculated coordinates so that an "on" pixel may be shifted a few positions in the x and/or y directions.

The extent of x-axis compression (or elongation) and pixel relocation in our system are controlled by two parameters that are temporarily set at 5% and $\pm 2/64$, respectively, in our controlled distortion. Comparisons between the feature values of undistorted and distorted shapes also give us some idea of the effect of distortion on various feature values. For instance, if feature values change a lot with single axis compression, we may have to select an acquisition device that has a smaller compression/elongation distortion.

This system is aimed at simulating a real robot vision system. For a real vision system, the extent of distortions can be determined experimentally. The parameters used by our system for controlling distortions then can be re-set easily.

3.3. Classification by Feature Extraction

We intend to identify randomly oriented but well separated objects on a conveyor belt. Based on the arguments given in Section 2.4.1 and 2.4.2, the feature extraction method is more suited than the template matching method for recognizing randomly oriented objects. As a result, the

method of feature extraction is used in this study for classification.

There are twelve features in our current feature pool. In this section, we show how these feature values are computed. Two rules to select an effective subset of features are also discussed.

3.3.1. Feature Pool

Our system is limited to simulating robotic visual inspection and classification tasks. It is assumed that the objects to be inspected are isolated and randomly oriented. In addition, the object may not be at an exact position under the camera. As a result, the selected features should be independent of position, orientation, and size.

As an aside, we understand that size-dependent features can be useful. By using these features, for example, printed upper case characters can be differentiated easily from lower case ones. Our intent in using size-independent features is that someday we hope this system can be used to examine hand written characters of which the sizes often vary. As a result, the size independent features are especially useful for hand written characters.

Orientation dependent features are also useful in the assembly of machine parts, since parts often need to be properly aligned. The reason not to use orientation dependent features is to avoid time consuming rotation related computations.

This system can be modified easily to include size and/or orientation dependent features by simply adding such features to the feature pool. The following are the features used in our system. This list could be expanded easily if desired.

Feature 1 -- $\langle R_{\max}/R_{\text{av}} \rangle$,

where R_{\max} is the longest distance ("radius") from centroid to any "on" pixel and R_{av} is the average distance from centroid to all "on" pixels. Note that R_{\max} may be off by a lot if there is random noise. This problem can be lessened by taking the average distance for the 5% most remote "on" pixels, instead of a single most remote "on" pixel, as R_{\max} .

Feature 2 -- $\langle R_{\min}/R_{\text{av}} \rangle$,

where R_{\min} is the shortest distance from centroid to any "on" pixel. To lessen the effects of noise, one may also take R_{\min} as the average distance for the nearest 5% "on" pixels from centroid.

Feature 3 -- fraction of "on" pixels in circular shell 1 as shown in Figure 3.4. The radius for the outermost circle is R_{\max} . A "shell" is defined as the area between two neighboring circles.

Feature 4 -- fraction of "on" pixels in shell 2.

Feature 5 -- fraction of "on" pixels in shell 3.

As an example, Figure 3.4 shows the digitized edges of two rectangles with four circles drawn using the centroid as the center and R_{\max} , $0.75 \times R_{\max}$, $0.5 \times R_{\max}$, and $0.25 \times R_{\max}$ as

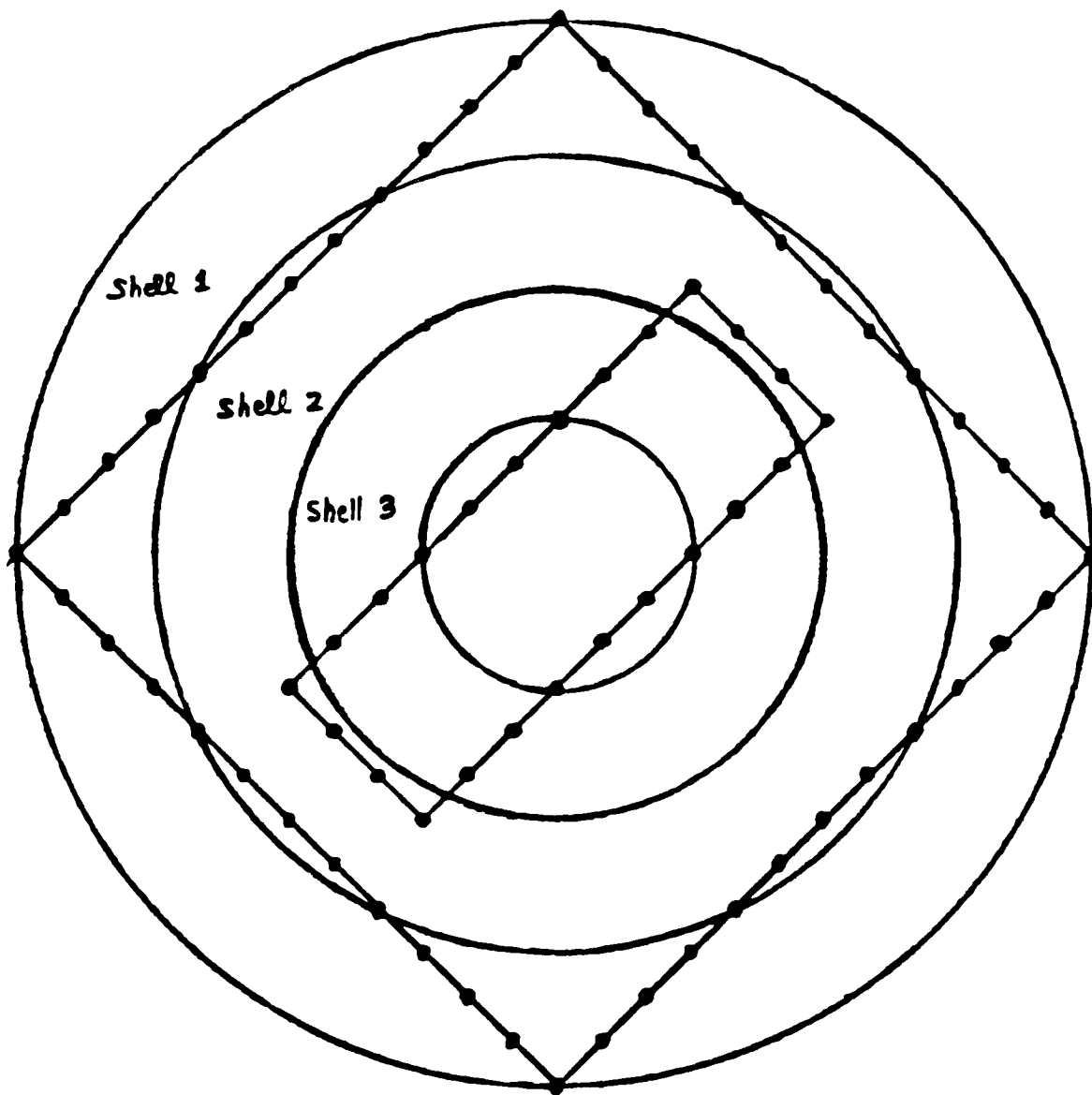


Figure 3.4. Example for the computation of features
3, 4 and 5.

the radii. The total number of "on" pixels for both rectangles is 72. The fraction of "on" pixels in shell 1 is 0.39 (28/72), which is the value of feature 3. Similarly, the values of feature 4 and 5 are 0.39 (28/72) and 0.11 (8/72), respectively. Note that the fraction of "on" pixels in the innermost shell is not included, since it is redundant to the other 3 shells.

Features 3 to 5 are proposed because it is obvious that their values are independent of orientation, size and position. They can be considered as discrete density functions, where the digitized image is divided here into four disjoint circular shells. It is reasonable to assume that the number of shells can be increased if a grid size greater than 64x64 is used.

Features 6 to 12 -- the 7 invariant moments derived by Hu [HU62] are the choices and the formulas for computing invariant moments are shown in Table 2.1.

Currently, there are a total of twelve features in our feature pool. Table 3.1 gives a summary of the features.

Feature number	Formula for computing feature value

1	R_{\max} / R_{av}
2	R_{\min} / R_{av}
3	Fraction of "on" pixels in shell 1 (Fig.3.4)
4	Fraction of "on" pixels in shell 2 (Fig.3.4)
5	Fraction of "on" pixels in shell 3 (Fig.3.4)
6	1st invariant moment (Table 2.1)
7	2nd invariant moment (Table 2.1)
8	3rd invariant moment (Table 2.1)
9	4th invariant moment (Table 2.1)
10	5th invariant moment (Table 2.1)
11	6th invariant moment (Table 2.1)
12	7th invariant moment (Table 2.1)

Table 3.1. Features in our feature pool.

3.3.2. Feature Selection

For the recognition of some given objects, one may not need all the features in the feature pool. The problem is to develop a systematic way to select an optimum subset of features that can achieve both adequate discrimination and high speed recognition. We have derived two selection rules from the algorithms described in section 2.4.3, and both were used in the system.

Rule 1 -- Choose the feature with good discrimination ability when the feature is used alone.

As shown in Figure 3.5a, the discrimination ability of feature k for object classes i and j is defined as

$$DA_{ij}(k) = \frac{(F_i(k) - F_j(k))}{\sigma_i(k) + \sigma_j(k)} \quad (3.1)$$

where $F_i(k)$ is the k -th feature value for object class i , and $\sigma_i(k)$ is the corresponding standard deviation. In the present study, $\sigma_i(k)$ or $\sigma_j(k)$ is set to 0.01 if it is less than 0.01.

Comparing with Equation (2.6), the above definition is, in fact, a one-feature Mahalanobis distance. For a given feature, when the ranges of feature values of two object classes touch each other as shown in Figure 3.5b, the discrimination ability for that feature will be unity. Conceptually, one may say that a feature is statistically "likely" to differentiate two objects if its corresponding discrimination ability is greater than unity.

In order to be more specific, let us assume that the distribution of feature values of an object in various states is a bell shaped normal function [DUDA73]. One then can calculate the probability of occurrences of a feature value falling within a specific range [FREU84]. The results are shown in Table 3.2.

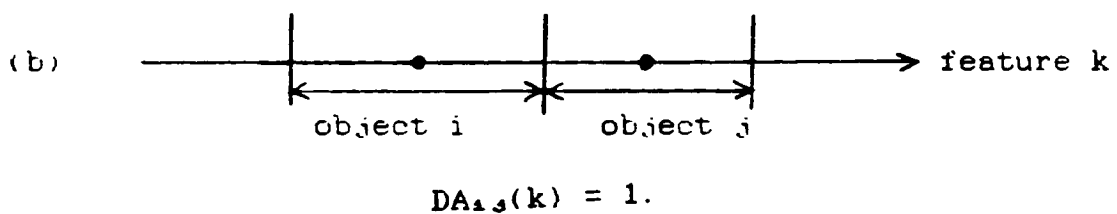
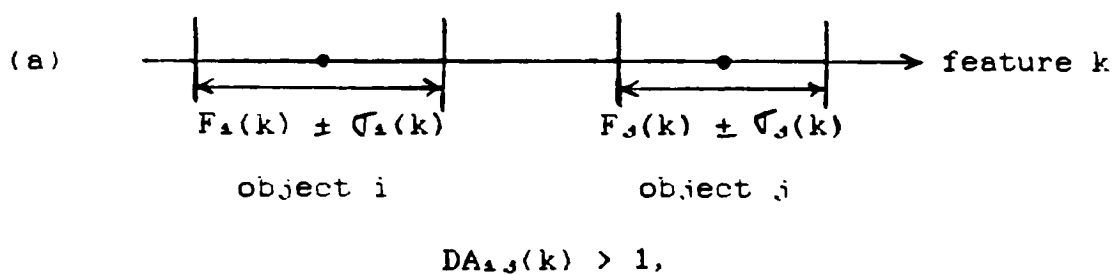


Figure 3.5. Discrimination abilities of feature k for object classes i and j .

Range	Probability
mean feature value $\pm 1 \sigma$	68.27%
mean feature value $\pm 2 \sigma$	95.45%
mean feature value $\pm 3 \sigma$	99.73%

Table 3.2. Probability of occurrence for a feature value to be within a range.

Armed with the information in Table 3.2, one can further calculate the probability of successful discrimination by feature k for object classes i and j at given $D_{i,j}(k)$ values. The results are shown in the table below.

$D_{i,j}(k)$ value	Successful discrimination
1	70.79%*
2	95.50%
3	99.73%

* $70.79\% = (0.6827 + 0.5 \cdot (1 - 0.6827))^2$, where probability of object i occurs inside one standard deviation from its mean = 0.6827, and probability of object i occurs outside one standard deviation from its mean but away from object j = $0.5 \cdot (1 - 0.6827)$.

Table 3.3. Successful discrimination at some $D_{i,j}(k)$ values.

In order to better explain our method, a hypothetical example is given in Table 3.4 which includes 3 objects and 3 features.

Feature 1		O ₁	O ₂	O ₃
	O ₁	0	1.5	0.3
	O ₂	1.5	0	1.1
	O ₃	0.3	1.1	0

Feature 2		O ₁	O ₂	O ₃
	O ₁	0	2.5	0.8
	O ₂	2.5	0	1.3
	O ₃	0.8	1.3	0

Feature 3		O ₁	O ₂	O ₃
	O ₁	0	0.7	3.0
	O ₂	0.7	0	1.5
	O ₃	3.0	1.5	0

Table 3.4. Discrimination abilities by each feature among 3 object classes.

Based on Table 3.4, for differentiating O₁ from O₂, feature 2 is the best choice; for O₁ and O₃, feature 3 is the best; and for O₂ and O₃, feature 3 is the best. As a

result, features 2 and 3 should be chosen as a subset for classification. Note that the user can always include additional features to the selected feature sub-set for improved accuracy, but with a compromise in speed.

In our current study, if the best feature provides a discrimination ability less than 2, then the next best one also will be selected. For the object pair 2 and 3 in Table 3.4, both features 3 and 2 are selected since neither of the features has a discrimination ability greater than 2.

What if, for a pair of objects, no feature with discrimination ability greater than unity can be found? In such a case, it is always difficult to discriminate between the object pair. It is thus advised to look for a better feature outside the current feature pool.

Rule 2 -- If two features are redundant for given reference objects, one of them is deleted.

An example is illustrated in Figure 3.6, where 5 objects are plotted in a 2-dimensional feature space and all objects fall onto a straight line. This means that $F_1(k) - F_3(k)$ is proportional to $F_1(1) - F_3(1)$ for any object pair, i, j . This also means that if an object pair can be discriminated by feature k , it also can be discriminated by feature 1. As a result, one of these features can be eliminated because of the redundancy.

Then, which feature should be deleted? To make that decision, we chose to compute the sum of discrimination

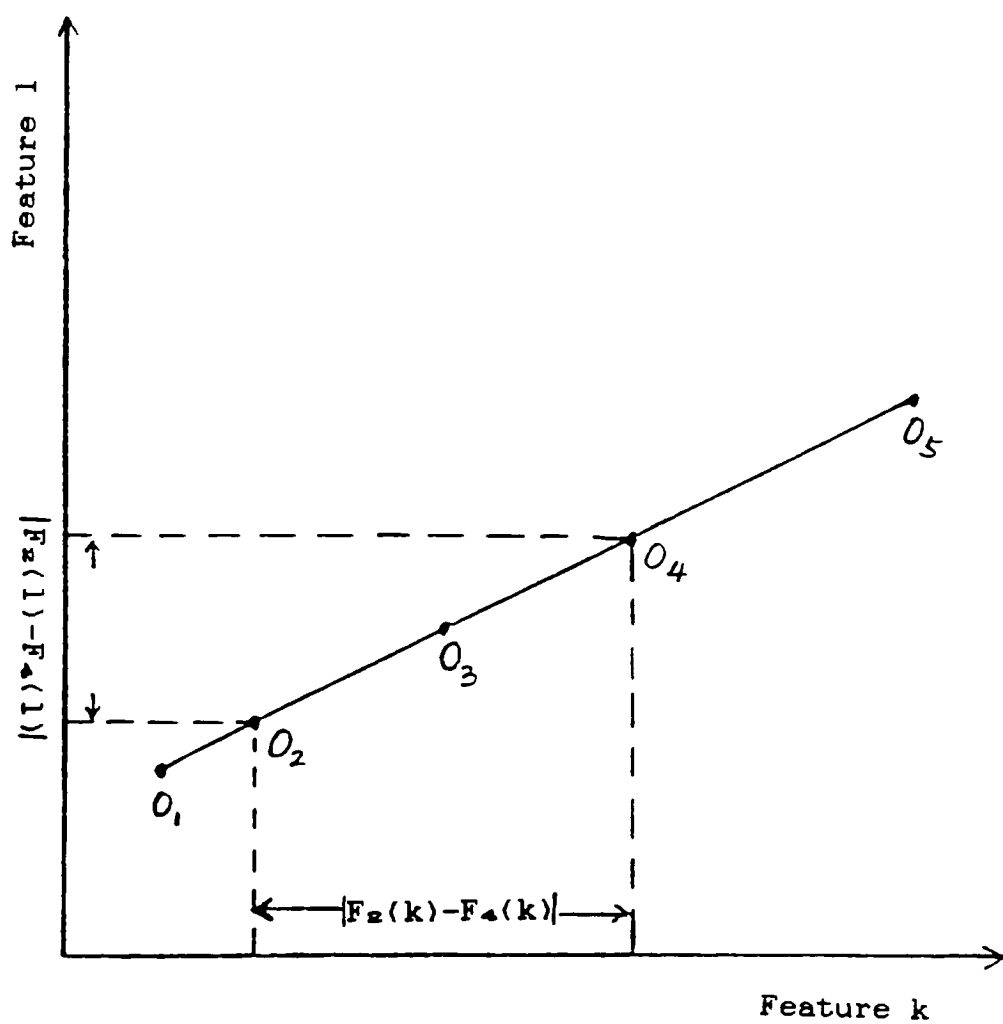


Figure 3.6. Objects in a 2-dimensional feature space.

abilities, $\sum_i^n \sum_{j(j < i)}^n D_{ij}(k)$ and $\sum_i^n \sum_{j(j < i)}^n D_{ij}(l)$, which represent the overall discrimination abilities by feature k and l, respectively. The feature with a smaller sum will be deleted.

As a summary,

$$\begin{aligned} & \text{Overall discrimination ability by feature } k \\ &= DA(k) \\ &= \sum_i^n \sum_{j(j < i)}^n D_{ij}(k) \end{aligned} \quad (3.2)$$

We have explained the fact that 2 features are considered redundant if the objects fall on a straight line in the 2-dimensional feature space. But, how can a computer tell if the objects are on the same line? This can be easily achieved by the method of linear regression. All the points are fitted to a straight line and the correlation coefficient, r , are computed by the following equation [FREU84]:

$$r = \left| \frac{n(\sum xy) - (\sum x)(\sum y)}{\sqrt{n(\sum x^2) - (\sum x)^2} \cdot \sqrt{n(\sum y^2) - (\sum y)^2}} \right| \quad (3.3)$$

where n is the number of points (or objects) and x is the sum of all x coordinates. The definitions for $\sum y$, $\sum xy$, $\sum x^2$ $\sum y^2$ are obvious.

The correlation coefficient is a determination of how closely the data points fits a straight line. At $r = 1$, the

points fall exactly onto a straight line. At $r = 0$, the points can not be approximated at all by a straight line.

In this section, we first defined the discrimination ability by a feature for any object pair. With this definition, it became easier to explain the redundancy of feature pairs.

However, in the implementation of our system, we first examine all feature pairs for redundancy and one feature is deleted in a redundant feature pair. Presently, one feature is deleted if $r > 0.95$. As a rule,

$$\begin{aligned} &\text{delete feature } k \text{ if } r > 0.95 \ \& \ DA(1) > DA(k), \\ &\text{delete feature } l \text{ if } r > 0.95 \ \& \ DA(k) > DA(l). \end{aligned} \quad (3.4)$$

After redundancy is removed, we then use the discrimination ability to select the most discriminating feature for each object pair.

The system deals with one feature or one feature pair at a time. It avoids handling a large number of features simultaneously. Our approach saves time, and the results are good.

3.3.3. Classification

With the selected feature subset, the Mahalanobis distance between an unknown and each chosen reference object class then can be computed. The nearest neighbor algorithm

along with the Mahalanobis distances are used for classification.

3.4. Flow Charts of Implementation

In the beginning of this chapter, we presented a flow chart to show the overall picture of our system. The chart is shown in Figure 3.1. Here, we present five additional flow charts to describe the details of our system implementation.

3.4.1. Hierarchical Process Diagram

Our system was designed to be menu driven. One of three main paths can be selected by the user at one time. They are (1) create and store objects, (2) select reference objects to be compared and then select a feature subset, and (3) identify unknown objects.

A flow chart of the main menu is shown in Figure 3.7. The details of each path are discussed in the succeeding sections.

3.4.2. Create and Store Objects

This is an interactive system, in which the coordinates of key points for an object are provided by the user whenever the system prompts for them. The coordinates come from hand drawn objects on graph paper.

Figure 3.8 shows the flow chart for getting input, creating the digitized edges, plotting the shape, computing

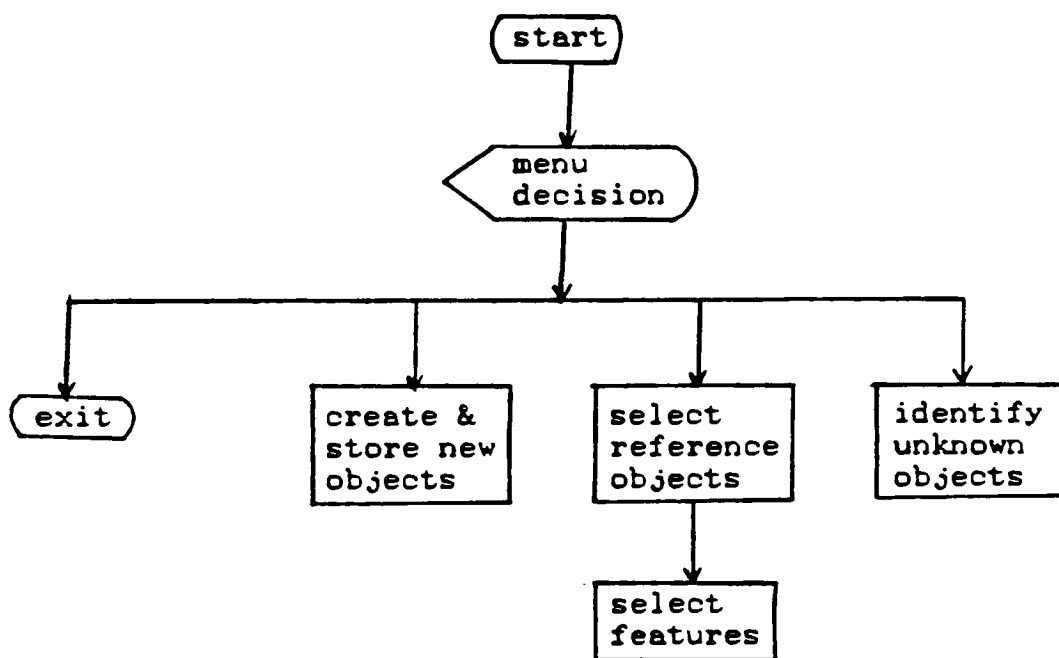


Figure 3.7. Hierarchical process diagram.

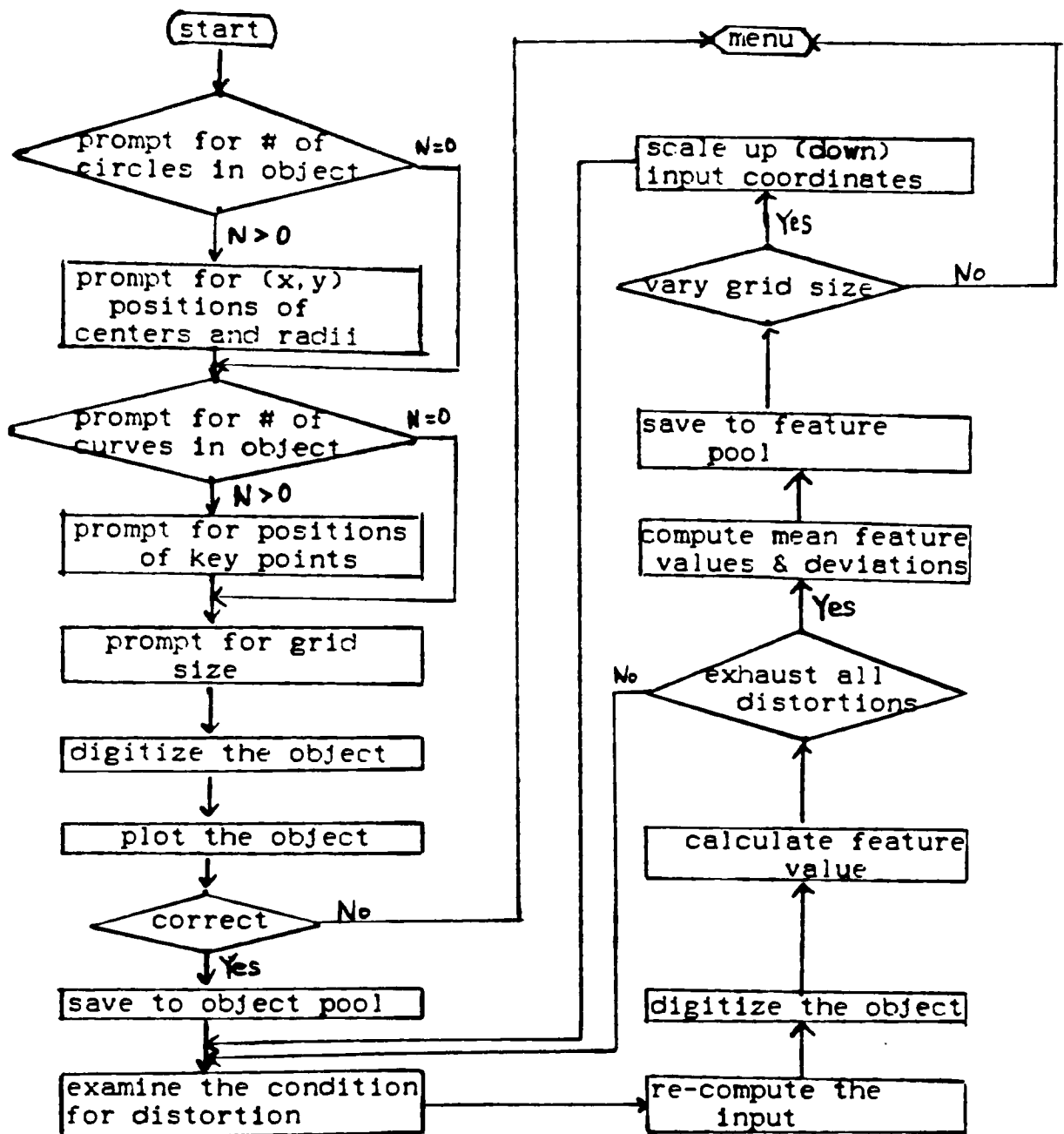


Figure 3.8. Create and store objects.

feature values and their standard deviations, and saving the information in a disk file. The user only needs to provide coordinates at one grid size, and the system will examine the object at several different grid sizes to study the effects of camera resolution.

3.4.3. Select Reference Objects

Figure 3.9 gives the sequence in the selection of reference objects for comparison with the unknown. First of all, the user opens the object file on a disk and retrieves the information about all saved objects. The system first displays a list of object names from which the user can select. It then displays the image of the object selected for the user to confirm. After all the reference objects are chosen, only their feature values and standard deviations are used in the next step, which is to select an effective feature subset.

3.4.4. Select a Feature Subset

The subset of features is automatically selected by the system according to the rules explained in Section 3.3.2. As shown in Figure 3.10, the system provides information on (a) the correlation coefficient of all feature pairs, and (b) discrimination ability of every feature for all object pairs.

Later, only the selected features are used for classification in order to improve recognition speed.

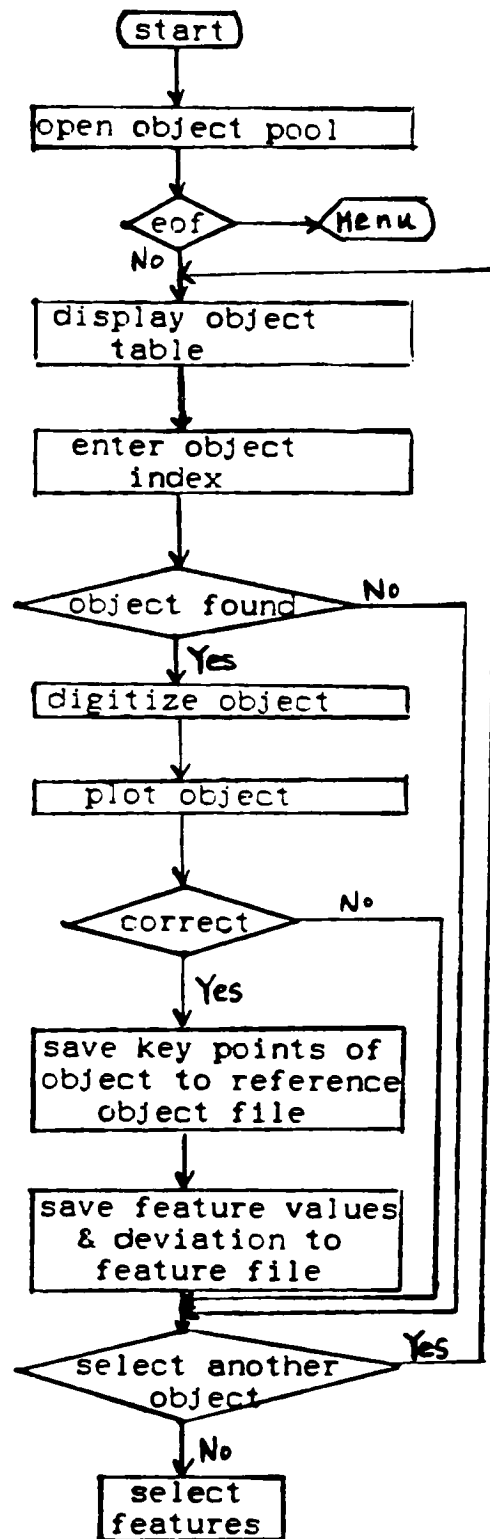


Figure 3.9. Select reference objects.

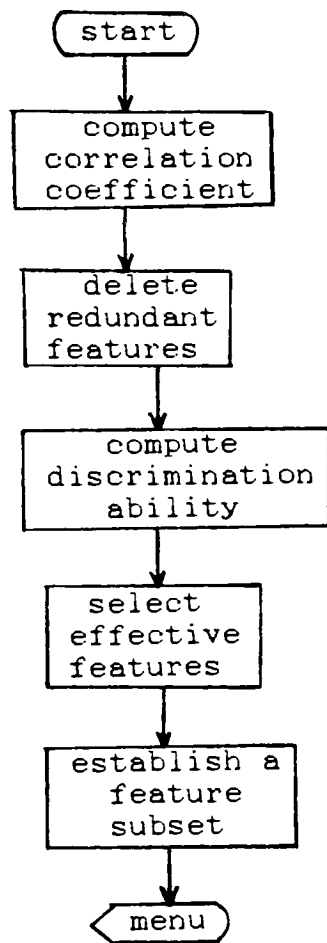


Figure 3.10. Select features

3.4.5. Identify Unknown Objects

Figure 3.11 shows the path for identifying unknown objects. The information to plot the unknown object will be prompted by the system. The system then displays an image of the unknown object on the monitor for the user's confirmation.

The system then computes the values of selected features for the unknown object. These values are used to calculate the Mahalanobis distance to each reference object class in the selected feature space. The unknown then can be identified based on the nearest neighbor classification algorithm.

Along with each identification, it is helpful to define a confidence level, which is shown below;

$$\begin{aligned} \text{Confidence level is high if } D_z > 3 \text{ and } D_1 < 2, \\ \text{Confidence level is low if } D_z < 1 \text{ or } D_1 > 3, \\ \text{Confidence level is average otherwise.} \end{aligned} \quad (3.4)$$

where D_1 is the Mahalanobis distance from unknown to the nearest reference object class and D_z is the distance from unknown to the next nearest reference object class. In addition, if an unknown is identified with a reference object class that has a short Mahalanobis distance (< 1) to another reference object class, then the confidence level is always set at low to warn the user about the possibility of mis-identification. However, this should not let happen in a useful robotic vision system.

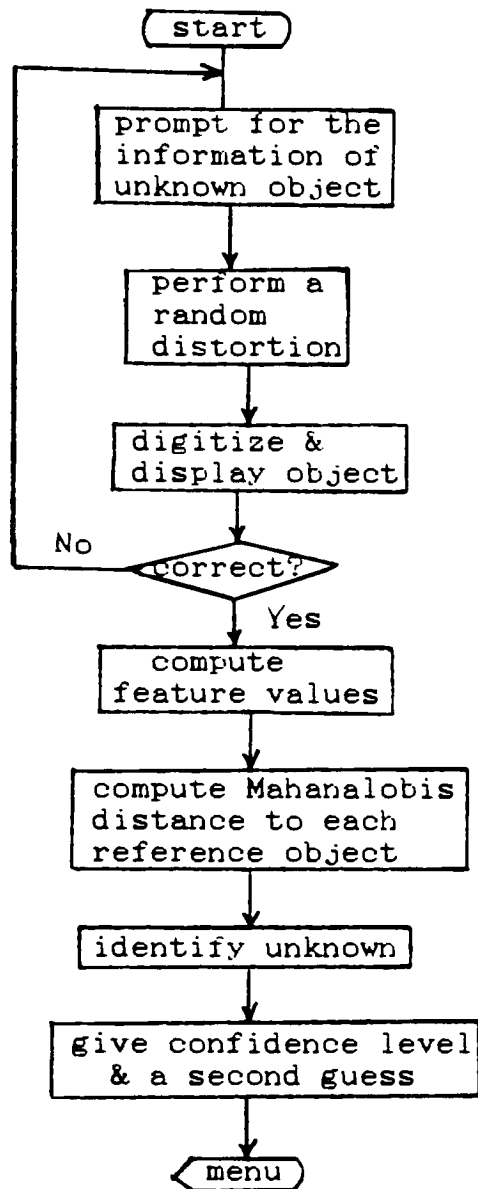


Figure 3.11. Identify unknown objects.

Without involving rigorous mathematical derivations and based on the probabilities shown in Table 3.2, one sees that the probability of mistakenly finding the unknown to be the second nearest object is no more than 0.27% when $D_2 > 3$. The additional condition for a high confidence level, $D_1 < 2$, assures that the unknown is in the neighborhood of the nearest reference object.

One also sees that the probability of mistakenly finding the unknown to be the second nearest reference object can be as high as 68.27% when $D_2 < 1$ and that the probability of correctly finding the unknown to be its nearest neighbor is less than 0.27% when $D_1 > 3$.

Now, we are fully prepared to identify unknown objects. In some cases, the system can identify an unknown with high confidence. In other cases, the system makes identification by its best judgement, but also informs user that the confidence level is low or average.

3.5. System Hardware and Software

The system runs on a super Turbo (10 MHz) IBM-XT compatible computer with 640K RAM, a 30 MB Seagate hard disk and a 360K floppy drive. Our simulated shape recognition program and related data files were installed on the hard disk for fast access. A Seikosha dot matrix printer is used to print text and images.

MS/DOS 3.20 by IBM Corp. and Microsoft, Inc., is the operating system, and Turbo PASCAL is the implementation

language. The Turbo PASCAL compiler, version 4, was used, and the line compile option was chosen.

A large memory, ~110K, was needed to edit our program, which could not be handled by the Turbo PASCAL editor. As a result, a word processing program, Microsoft WORD version 3.0, was selected to do editing. WORD was also used to produce this thesis manuscript.

3.6. Summary

In this chapter, the discussion includes mathematically generating images and also ways of implementing controlled distortions.

There are currently 12 features used by the system. For a given object domain, the system can automatically choose an effective feature subset for improved recognition speed.

Classification was done by the nearest neighbor algorithm and the Mahalanobis distance was adopted.

Five flow charts were presented to show the architecture of the system and the details of the implementation.

The system hardware was described. The operating system, the compiler, and the editor were also discussed.

CHAPTER 4

RESULTS AND DISCUSSION

In this chapter, we will discuss the variables manipulated in testing our system and also the system's performance. We have included 12 features, 10 different objects, 4 types of distortion, and 3 grid sizes to demonstrate various aspects of our work.

The objects, both with and without distortions, were digitized mathematically. This was done to simulate image sensing, digitization, and edge extraction processes in an actual robot vision system.

Feature values and their standard deviations were computed for all 12 features and 10 object classes. Mahalanobis distances in the 12-dimensional feature space between 10 object classes also were computed.

The effect of resolution is discussed by showing results at 3 different grid sizes. Results also are shown to explain the algorithms for selecting an effective feature subset in a given object space. Our method avoids handling many features simultaneously.

First, features were examined pairwise to determine their redundancy, which was measured by the corresponding correlation coefficient. Discrimination abilities of a feature for all object pairs also were computed quantitatively. The feature subset was formed by removing

redundancy and then selecting the most discriminating features from the remaining ones.

Finally, an unknown at its randomly distorted state was identified with one of the reference objects based on the nearest neighbor classification algorithm.

4.1. Graphic Representation of Edge-Extracted Objects

Image sensing, digitization and edge-extraction processes usually are done by hardware in an actual robot vision system. In this study, we tried to simulate the end results of these processes by mathematically generating the edge-extracted images.

Ten geometric objects were included for demonstration. Plots of some digitized objects with and without distortions are shown in Figures 4.1 to 4.4. Four types of controlled distortion and also the magnitude of distortions are tabulated below.

4.1.1. Objects in the Object Pool

We have stored 10 simple geometric objects in our object pool. These objects are listed in Table 4.1 and their undistorted images are shown in Appendix.

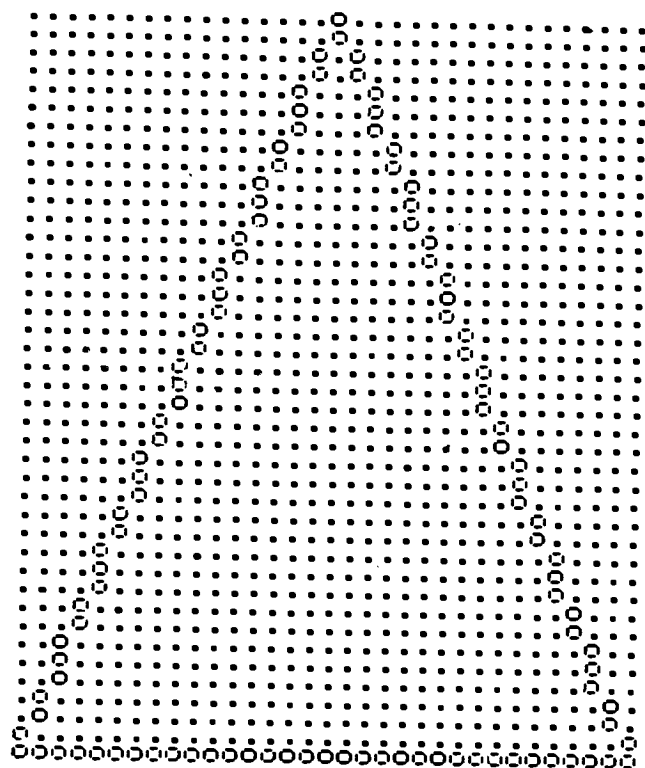


Figure 4.1. A isosceles triangle.

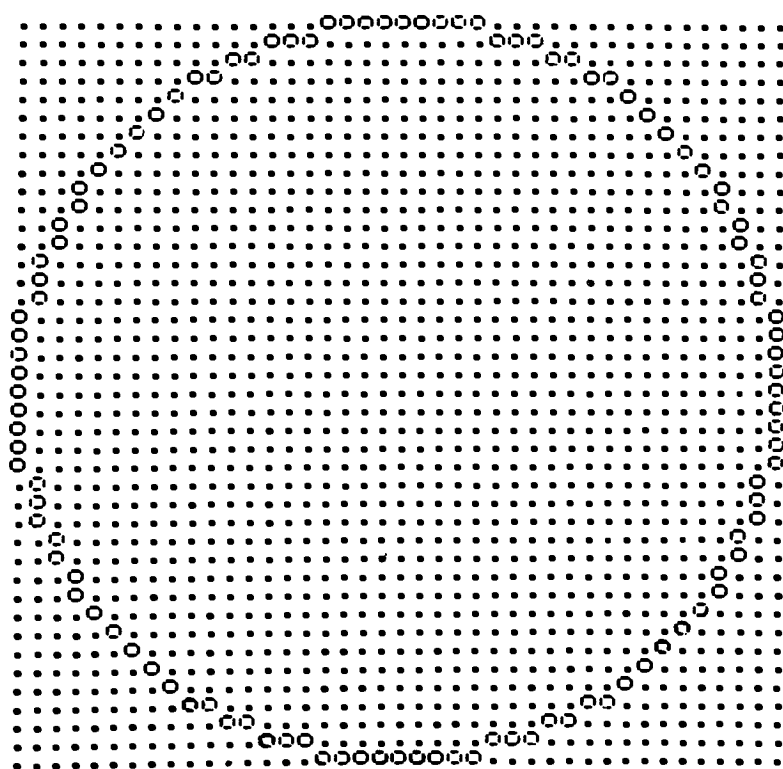


Figure 4.2. A circle

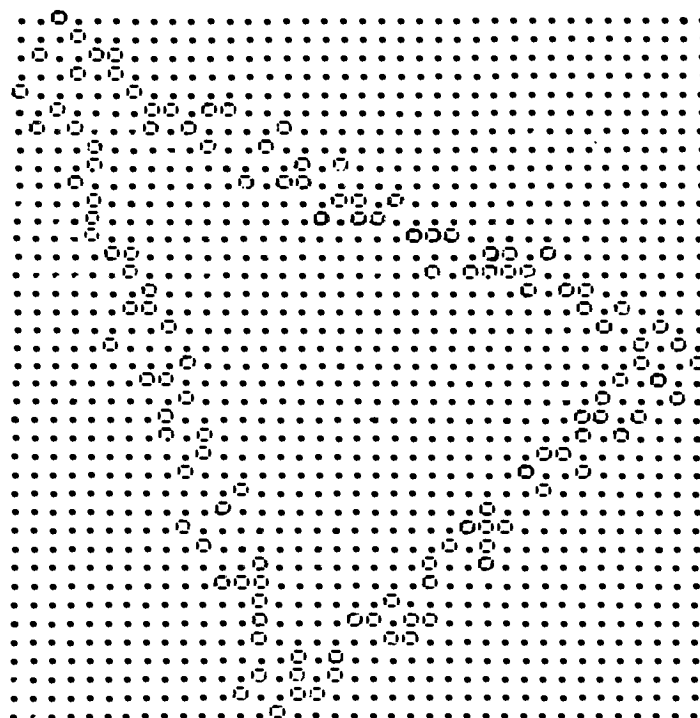


Figure 4.3. A distorted isosceles triangle.

blurring by pixel relocation:	+2 to -2
x-axis compression:	none
rotation angle:	51°
size change:	-10%

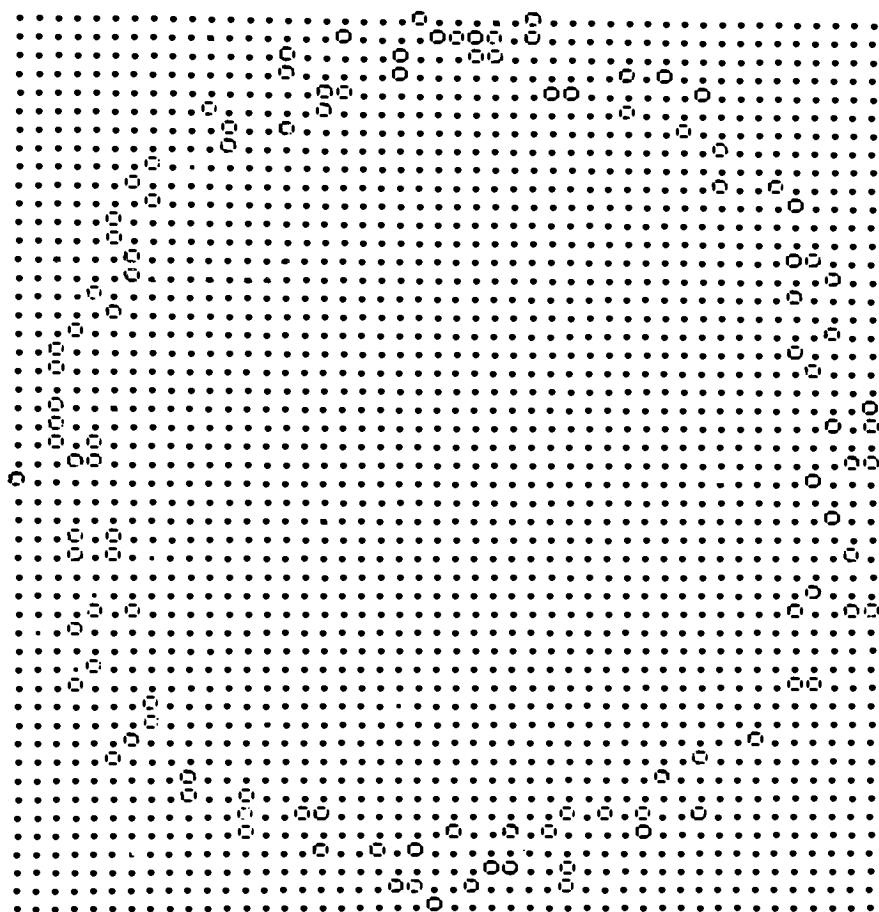


Figure 4.4. A distorted circle.

blurring by pixel relocation:	+2 to -2
x-axis compression:	5%
rotation angle:	51°
size change:	+10%











<u>Index</u>	<u>Object Name</u>	
1	Square	
2	Rectangle	
3	Isosceles Triangle	
4	Right Triangle	
5	Circle	
6	Washer (Ring)	
7	Nut (Circle in Square)	
8	Hexagon	
9	Nut (Circle in Hexagon)	
10	Asymmetric Shape	

Table 4.1. Objects in the Object Pool.

The coordinates of key points for these 10 objects in a 64x64 frame are shown in Table 4.2. Note that the objects do not fully occupy the whole field of view. The system automatically calculated the coordinates for 32x32 and 16x16 frames by multiplying all coordinates with factors 0.5 and 0.25, respectively. The resulted numbers from multiplication were not rounded off.

Object	Key points of lines	Center of a circle & radius
1	(-20, 20)(20, 20)(20, -20) (-20, -20)(-20, 20)	
2	(-24, 12)(24, 12)(24, -12) (-24, -12)(-24, 12)	
3	(-16, -20)(0, 20)(16, -20) (-16, -20)	
4	(-24, -4)(24, 10)(24, -4) (-24, -4)	
5		(0, 0), $r = 20$
6		(0, 0), $r_1 = 20$ (0, 0), $r_2 = 15$
7	(-20, 20)(20, 20)(20, -20) (-20, -20)(-20, 20)	(0, 0), $r = 15$
8	(-10, 17)(10, 17)(20, 0)(10, -17) (-10, -17)(-20, 0)(-10, 17)	
9	(-10, 17)(10, 17)(20, 0)(10, -17) (-10, -17)(-20, 0)(-10, 17)	(0, 0), $r = 12$
10	(-24, 10)(26, 22)(26, -12) (-16, -12)(-24, 10)	(16, 0), $r = 8$

Table 4.2. Coordinates of key points for 10 objects.

As described in Section 3.4.3, the object pool can be expanded easily by accessing the "Create and Store Objects" menu, which guides the user step by step through the process of creating a new object. Once a new object is created, it will remain in the object pool until it is deleted and can be retrieved whenever it is needed.

4.1.2. Digitized Images without Distortion

Figures 4.1 and 4.2 show the undistorted digitized images of an isosceles triangle and a circle in the current object pool. All images have a field of view of 64x64.

In the images, series of "o" characters represent edges and the dots "." indicate pixels that are not part of an edge. Since we chose to use text mode for printing the objects, the line spacing has been re-set to be almost the same as the spacing between characters on the same line. As a result, a square looks like a square on paper.

4.1.3. Types of Distortion

The magnitude and number of each type of distortion, for the current study, are listed in Table 4.3. There are a total of 24 states(variations) for each object class. These states include the undistorted one.

Type of Distortion	Magnitude	Number* of variations
Change of size	$\pm 10\%$	3
x-axis compression	5%	2
Rotation	51°	2
Blurring by pixel relocation	$\sim 5\%$	2

* includes the undistorted state.

Table 4.3. Our controlled distortions.

With our system, the user can ask "what if" questions. For instance, what is the ability to separate a rectangle from a square if the x-axis compression is 20% instead of 5%, or what will happen if the blurring effect becomes twice as severe?

Each type of distortion is controlled by a single parameter, which can be re-set easily for the system. For a real application, the distortion parameters should be determined experimentally in order to closely simulate a given robot vision system. These parameters can be determined by repeatedly taking pictures of the same object.

4.1.4. Digitized Images with Distortions

Figures 4.3 and 4.4 show a distorted isosceles triangle and a distorted circle. The magnitudes of the distortions also are given in the figures along with the images. One can see the effects of blurring, rotation, and x-axis

compression, which simulates the quality of images an actual vision system might produce.

In Figure 4.4, note that the number of x-axis pixels is 46 and the number of y-axis pixels is 49, since the circle is compressed along its x-axis.

4.2. Properties of the Objects in the Object Pool

Since our system recognizes objects by using the method of feature extraction, the first task is to compute various feature values for each object, and the second task is to compute Mahalanobis distances among object classes in a given feature space.

4.2.1. Average Feature Values and their Standard Deviations

Currently, there are 12 features employed in our system. These features were defined in Section 3.3.1 and stored in the feature pool. Feature values were computed for 24 variations of each object class. The average feature values and their standard deviations, for 10 objects with a grid size of 64x64, are shown in Table 4.4. The name of the objects are shown in Table 4.1 and the formulas for computing the features are given in Table 3.1.

As shown in Table 4.4, some of the feature values for higher order moments are near zero. This is expected, since many of our objects are highly symmetrical. Moments are useful to discriminate symmetrical objects from asymmetrical

		Obj 1	Obj 2	Obj 3	Obj 4	Obj 5
Fea	1	1.2560	1.3570	1.6310	1.9430	1.0900
Dev	1	0.0307	0.0310	0.0382	0.0394	0.0593
Fea	2	0.8244	0.5633	0.5331	0.2091	0.9112
Dev	2	0.0378	0.0362	0.0438	0.0463	0.0608
Fea	3	0.6052	0.5408	0.2345	0.2286	0.9928
Dev	3	0.0752	0.0219	0.0397	0.0453	0.0149
Fea	4	0.3948	0.2904	0.4629	0.2879	0.0072
Dev	4	0.0752	0.0208	0.0438	0.0389	0.0149
Fea	5	0.0000	0.1689	0.3026	0.2811	0.0000
Dev	5	0.0000	0.0217	0.0181	0.0157	0.0000
Fea	6	4.0530	4.2490	4.2960	4.9090	4.0090
Dev	6	0.0044	0.0115	0.0193	0.0336	0.0082
Fea	7	0.0205	4.0260	2.2380	18.0200	0.0193
Dev	7	0.0267	0.2707	0.2790	0.5133	0.0207
Fea	8	0.0350	0.0321	30.2900	6.5110	0.0461
Dev	8	0.0798	0.0653	2.5040	1.5690	0.0715
Fea	9	0.0049	0.0066	0.2049	0.4995	0.0079
Dev	9	0.0078	0.0112	0.2094	0.3182	0.0144
Fea	10	0.0001	0.0002	0.6295	1.0480	-0.0002
Dev	10	0.0005	0.0010	0.8856	1.1640	0.0005
Fea	11	-0.0001	0.0063	0.2731	1.9540	0.0001
Dev	11	0.0008	0.0163	0.3273	1.3600	0.0012
Fea	12	0.0000	0.0001	-0.1017	0.2791	-0.0001
Dev	12	0.0002	0.0004	0.2797	0.2700	0.0007

Fea = Average feature value

Dev = standard deviation

Table 4.4. Average feature values and corresponding standard deviation for 10 objects.

grid size = 64x64

		Obj 6	Obj 7	Obj 8	Obj 9	Obj 10
Fea 1		1.2210	1.4250	1.1320	1.2910	1.6890
Dev 1		0.0601	0.0241	0.0291	0.0262	0.0367
Fea 2		0.7477	0.6611	0.8838	0.6596	0.0560
Dev 2		0.0579	0.0590	0.0433	0.0737	0.0275
Fea 3		0.5808	0.3973	0.9830	0.6263	0.2388
Dev 3		0.0140	0.0350	0.0288	0.0229	0.0224
Fea 4		0.4129	0.5353	0.0170	0.3514	0.4521
Dev 4		0.0140	0.0543	0.0288	0.0387	0.0202
Fea 5		0.0000	0.0674	0.0000	0.0223	0.2393
Dev 5		0.0000	0.0518	0.0000	0.0241	0.0272
Fea 6		4.0920	4.1900	4.0150	4.1460	4.4690
Dev 6		0.0085	0.0163	0.0061	0.0072	0.0452
Fea 7		0.0277	0.0176	0.0535	0.0436	2.5510
Dev 7		0.0333	0.0208	0.0483	0.0360	0.3227
Fea 8		0.0340	0.0183	0.0246	0.0210	10.4300
Dev 8		0.0570	0.0308	0.0386	0.0360	1.5570
Fea 9		0.0108	0.0093	0.0145	0.0074	5.2980
Dev 9		0.0173	0.0151	0.0214	0.0116	1.4400
Fea 10		0.0001	0.0001	-0.0001	0.0002	38.6400
Dev 10		0.0007	0.0006	0.0006	0.0004	17.2700
Fea 11		0.0005	-0.0001	0.0010	0.0006	7.6300
Dev 11		0.0021	0.0015	0.0035	0.0019	2.4670
Fea 12		0.0003	-0.0000	-0.0003	-0.0001	14.7000
Dev 12		0.0009	0.0008	0.0007	0.0002	6.8600

Table 4.4. -- continued --

ones. Note that all the moments (features 6 to 12) have non-zero values for our asymmetric object (Obj 10).

4.2.2. Mahalanobis Distances between Object Classes

Given the computed feature values and standard deviations, as shown in Table 4.4, Mahalanobis distances were calculated by using Equation 2.6. Table 4.5 shows the calculated results at a grid size of 64x64. All distances were calculated for our 12-dimensional feature space.

Based on the probabilities shown in Table 3.3, a distance greater than 2 means a successful recognition greater than 95.50% and distance greater than 3 means a correct recognition greater than 99.73%. One can see in Table 4.5 that most of the object pairs can be separated easily. However, objects 5 (circle) and 8 (hexagon) are difficult to discriminate, since the distance is only 0.9485. This seems reasonable, since a circle and a hexagon do look alike.

In order to improve the discrimination ability for certain object pairs, one needs to do one or both of the following: (1) add new features that better discriminate between them; however, this approach is beyond the scope of current study, and (2) use a higher resolution device so that the accuracy of representation is improved. The effect of resolution will be discussed shortly in Section 4.3.2.

	Obj 1	Obj 2	Obj 3	Obj 4	Obj 5
Obj 1	0.0000	20.2922	25.0717	46.0664	7.3146
Obj 2	20.2922	0.0000	14.5795	25.9531	25.1828
Obj 3	25.0717	14.5795	0.0000	24.5178	29.6500
Obj 4	46.0664	25.9531	24.5178	0.0000	47.4236
Obj 5	7.3146	25.1828	29.6500	47.4236	0.0000
Obj 6	3.1850	17.7792	24.2398	43.8471	20.8400
Obj 7	7.9236	14.6707	15.4120	38.3823	16.6716
Obj 8	6.6886	23.0164	28.5858	46.8030	0.9485
Obj 9	8.2966	14.7062	18.4088	40.4302	15.0075
Obj 10	21.6234	15.6869	10.1953	20.4255	31.3193

	Obj 6	Obj 7	Obj 8	Obj 9	Obj 10
Obj 1	3.1850	7.9236	6.6886	8.2966	21.6234
Obj 2	17.7792	14.6707	23.0164	14.7062	15.6869
Obj 3	24.2398	15.4120	28.5858	18.4088	10.1953
Obj 4	43.8471	38.3823	46.8030	40.4302	20.4255
Obj 5	20.8400	16.6716	0.9485	15.0075	31.3198
Obj 6	0.0000	6.3767	14.3880	4.1299	20.7021
Obj 7	6.3767	0.0000	14.8813	5.5470	15.2255
Obj 8	14.3880	14.8813	0.0000	13.5187	27.8225
Obj 9	4.1299	5.5470	13.5187	0.0000	18.2098
Obj 10	20.7021	15.2255	27.8225	18.2098	0.0000

Table 4.5. Mahalanobis distances for 10 object classes
with 12 features at grid size = 64x64.

Note that an adequate distance between object classes does not guarantee an adequate distance between a reference object and an unknown at a randomly distorted state. Recognizing this fact, one can do "image averaging" for improved recognition accuracy. The trade off is a slow down in recognition speed. This "image averaging" feature has not been implemented in our system.

4.3. Selection of Reference Objects from the Object Pool

The goal of our system is to classify unknowns among some predetermined reference objects using the method of feature extraction. The effectiveness of the features depends on the objects to be classified. As a result, effective features need to be selected after the reference objects are specified.

For each given shape recognition task, one begins by selecting reference objects from the object pool. If a desired object is not in the pool, the object should be created by using the "Create and Store Objects" menu.

Table 4.6 shows a set of selected reference objects from our object pool. These objects are used in the current example to further describe our method. Note that the object indices are re-numbered and are different from those in Table 4.1.






<u>Index</u>	<u>Object name</u>	
1	Square	
2	Right Triangle	
3	Circle	
4	Hexagon	
5	Asymmetric Shape	

Table 4.6. Selected reference objects from our object pool.

4.4. Effect of Resolution

Focusing on the above selected objects of which the feature values and standard deviations are shown in Tables 4.7a, 4.7b and 4.7c, Mahalanobis distances are shown in Table 4.8. The results are listed at 3 grid sizes, 64x64, 32x32, and 16x16 to show the effect of resolution.

Comparing Table 4.7a to Tables 4.7b and 4.7c, one can see that the average feature values from one grid size to another do not show much change, because our selected features are independent of size. For instance, the value of feature 1 for object 1 slightly changes from 1.2560 to 1.2310, when the grid size decreases from 64x64 to 16x16.

		Obj 1	Obj 2	Obj 3	Obj 4	Obj 5
Fea 1	1	1.2560	1.9430	1.0900	1.1320	1.6890
Dev 1	1	0.0307	0.0394	0.0593	0.0291	0.0367
Fea 2	2	0.8244	0.2091	0.9112	0.8838	0.0560
Dev 2	2	0.0378	0.4630	0.0608	0.0433	0.0275
Fea 3	3	0.6052	0.2286	0.9928	0.9830	0.2388
Dev 3	3	0.0752	0.0453	0.0149	0.0288	0.0224
Fea 4	4	0.3948	0.2879	0.0072	0.0170	0.4521
Dev 4	4	0.0752	0.0389	0.0149	0.0288	0.0202
Fea 5	5	0.0000	0.2811	0.0000	0.0000	0.2393
Dev 5	5	0.0000	0.0157	0.0000	0.0000	0.0272
Fea 6	6	4.0530	4.9090	4.0090	4.0150	4.4690
Dev 6	6	0.0044	0.0336	0.0082	0.0061	0.0452
Fea 7	7	0.0205	18.0200	0.0193	0.0535	2.5510
Dev 7	7	0.0267	0.5133	0.0207	0.0483	0.3227
Fea 8	8	0.0350	6.5110	0.0461	0.0246	10.4300
Dev 8	8	0.0798	1.5690	0.0715	0.0386	1.5570
Fea 9	9	0.0049	0.4995	0.0079	0.0145	5.2980
Dev 9	9	0.0078	0.3182	0.0144	0.0214	1.4400
Fea 10	10	0.0001	1.0480	-0.0002	-0.0001	38.6400
Dev 10	10	0.0005	1.1640	0.0005	0.0006	17.2700
Fea 11	11	-0.0001	1.9540	0.0001	0.0010	7.6300
Dev 11	11	0.0008	1.3600	0.0012	0.0035	2.4670
Fea 12	12	0.0000	0.2791	-0.0001	-0.0003	14.7000
Dev 12	12	0.0002	0.2700	0.0007	0.0007	6.8600

Table 4.7a. Average feature values and corresponding standard deviations for 5 objects.

grid size = 64x64

		Obj 1	Obj 2	Obj 3	Obj 4	Obj 5
Fea 1	1	1.2580	1.9680	1.1020	1.1400	1.6890
Dev 1	1	0.0324	0.0544	0.0614	0.0294	0.0499
Fea 2	2	0.8256	0.2221	0.8906	0.8675	0.0907
Dev 2	2	0.0366	0.0486	0.0631	0.0569	0.0515
Fea 3	3	0.6076	0.2072	0.9703	0.9768	0.2420
Dev 3	3	0.0792	0.0499	0.0454	0.0338	0.0260
Fea 4	4	0.3924	0.3012	0.0297	0.0232	0.4478
Dev 4	4	0.0792	0.0414	0.0454	0.0338	0.0365
Fea 5	5	0.0000	0.2810	0.0000	0.0000	0.2348
Dev 5	5	0.0000	0.0214	0.0000	0.0000	0.0498
Fea 6	6	4.0560	4.9180	4.0140	4.0170	4.4980
Dev 6	6	0.0052	0.0405	0.0117	0.0071	0.0522
Fea 7	7	0.0436	17.9400	0.0385	0.0451	2.5550
Dev 7	7	0.0789	0.7964	0.0371	0.0487	0.3406
Fea 8	8	0.0202	6.8950	0.0907	0.0447	11.6500
Dev 8	8	0.0448	1.5490	0.1299	0.0649	2.2520
Fea 9	9	0.0143	0.6353	0.0605	0.0293	5.8130
Dev 9	9	0.0315	0.4070	0.1055	0.0393	1.6720
Fea 10	10	0.0002	1.5650	-0.0018	-0.0004	46.0600
Dev 10	10	0.0012	1.6150	0.0077	0.0025	22.9800
Fea 11	11	0.0032	2.5130	-0.0050	0.0007	8.1120
Dev 11	11	0.0107	1.7430	0.0141	0.0054	2.5220
Fea 12	12	0.0002	0.2489	0.0010	-0.0003	20.6900
Dev 12	12	0.0011	0.4549	0.0152	0.0028	10.9000

Table 4.7b. Average feature values and corresponding
standard deviations for 5 objects.

grid size = 32x32

		Obj 1	Obj 2	Obj 3	Obj 4	Obj 5
Fea 1		1.2310	1.9930	1.1050	1.1460	1.6680
Dev 1		0.0384	0.0774	0.0707	0.0364	0.0407
Fea 2		0.8262	0.2450	0.8969	0.8441	0.1073
Dev 2		0.0443	0.0444	0.0846	0.0531	0.0419
Fea 3		0.6563	0.1923	0.9416	0.9315	0.2574
Dev 3		0.0904	0.0655	0.1094	0.0849	0.0300
Fea 4		0.3437	0.2992	0.0584	0.0685	0.4463
Dev 4		0.0904	0.0588	0.1094	0.0849	0.0416
Fea 5		0.0000	0.2949	0.0000	0.0000	0.2162
Dev 5		0.0000	0.0321	0.0000	0.0000	0.0469
Fea 6		4.0590	4.9470	4.0220	4.0290	4.5040
Dev 6		0.0085	0.0604	0.0216	0.0124	0.0466
Fea 7		0.0494	17.6600	0.0144	0.1231	2.2880
Dev 7		0.0733	0.9401	0.0372	0.0861	0.4617
Fea 8		0.1033	8.6910	0.1268	0.1318	11.0800
Dev 8		0.2049	3.7070	0.2265	0.2952	2.4160
Fea 9		0.0293	1.2690	0.0468	0.0365	4.8360
Dev 9		0.0552	1.0530	0.1124	0.0710	1.3010
Fea 10		0.0043	5.8530	0.0042	0.0043	33.2800
Dev 10		0.0174	8.6980	0.0189	0.0151	14.9700
Fea 11		0.0062	5.0990	0.0008	-0.0001	6.2820
Dev 11		0.0132	4.6150	0.0068	0.0169	2.2100
Fea 12		-0.0011	0.8100	-0.0038	-0.0040	13.5000
Dev 12		0.0061	1.0900	0.0393	0.0268	8.9230

Table 4.7c. Average feature values and corresponding standard deviations for 5 objects.

grid size = 16x16

A smaller grid size, however, usually means a less accurate representation of an object, which typically leads to a larger deviation. As shown in Tables 4.7(a, b, c) the

standard deviation of feature 1 for object 1 increases from 0.0307 to 0.0384 while grid size decreases from 64x64 to 16x16.

One also can see from Table 4.8 that Mahalanobis distances increase with the increase in grid size. Judging from Equation 2.6, this increase is mainly due to the decrease in standard deviations.

Table 4.8 shows that the distance between object classes 1 and 2 decreases from 46.0664 to 25.6095, when the grid size decreases from 64x64 to 16x16. It seems that a low resolution camera, 16x16, is more than adequate to discriminate between these two object classes.

A lower resolution increases the speed of recognition. As a rule of thumb, the recognition time is roughly proportional to the number of pixels, so reduction of resolution from 64x64 to 16x16 leads to a 16-fold increase in the speed.

It is generally true that the higher the resolution is, the more accurate the identification is. However, a higher resolution means more expensive hardware and/or a slower recognition speed. It is desired, therefore, to determine a minimum but adequate resolution for a given task.

We believe that our system provides a useful tool for determining a proper resolution for a given pair of object classes.

a. grid size = 64x64

	Obj 1	Obj 2	Obj 3	Obj 4	Obj 5
Obj 1	0.0000	46.0664	7.3146	6.6886	21.6234
Obj 2	46.0664	0.0000	47.4236	46.8030	20.4255
Obj 3	7.3146	47.4236	0.0000	0.9485	31.3198
Obj 4	6.6886	46.8030	0.9485	0.0000	27.8225
Obj 5	21.6234	20.4255	31.3198	27.8225	0.0000

b. grid size = 32x32

	Obj 1	Obj 2	Obj 3	Obj 4	Obj 5
Obj 1	0.0000	33.1086	5.1686	5.9454	16.7848
Obj 2	33.1086	0.0000	33.4545	35.1400	15.2541
Obj 3	5.1686	33.4545	0.0000	0.6847	19.5600
Obj 4	5.9454	35.1400	0.6847	0.0000	21.6939
Obj 5	16.7848	15.2541	19.5600	21.6939	0.0000

c. grid size = 16x16

	Obj 1	Obj 2	Obj 3	Obj 4	Obj 5
Obj 1	0.0000	25.6095	2.7226	2.9320	16.1542
Obj 2	25.6095	0.0000	25.0509	25.7578	12.5170
Obj 3	2.7226	25.0509	0.0000	1.0601	15.2012
Obj 4	2.9320	25.7578	1.0601	0.0000	17.1716
Obj 5	16.1542	12.5170	15.2012	17.1716	0.0000

Table 4.8. Mahalanobis distances with 12 features for
5 objects classes at 3 grid sizes.

4.5. Selection of a Feature Subset

Once the reference objects are selected, the next step is to select an effective feature subset.

We use two criteria to help make an intelligent selection. One is to delete a redundant feature if the correlation coefficient between a feature pair is near unity. The other criterion is to choose the most discriminating feature for each object pair when that feature is used alone.

4.5.1. Elimination of Redundant Features

Table 4.9 shows the correlation coefficients for all feature pairs involving the 5 selected reference object classes. The coefficients were computed according to Equation (3.3). The feature values of each selected object were obtained from Table 4.7a,

If the coefficient for a feature pair is near unity, it means that these two features are highly redundant in the given object space. In our present study, if a feature pair has a correlation coefficient greater than 0.95, which can be re-set by the user, then one of these features will be deleted.

	Fea 1	Fea 2	Fea 3	Fea 4	Fea 5
Fea 1	1.0000	0.9270	0.9391	0.6297	0.9769
Fea 2	0.9270	1.0000	0.9285	0.6877	0.9694
Fea 3	0.9391	0.9285	1.0000	0.8563	0.9062
Fea 4	0.6297	0.6877	0.8563	1.0000	0.5716
Fea 5	0.9769	0.9694	0.9062	0.5716	1.0000
Fea 6	0.9801	0.8591	0.8572	0.4768	0.9541
Fea 7	0.8526	0.6161	0.6584	0.2380	0.7846
Fea 8	0.8483	0.9845	0.8689	0.6577	0.9231
Fea 9	0.4781	0.7706	0.6134	0.6139	0.6031
Fea 10	0.4210	0.7277	0.5687	0.5970	0.5498
Fea 11	0.6153	0.8652	0.7171	0.6480	0.7275
Fea 12	0.4139	0.7223	0.5630	0.5946	0.5432

	Fea 6	Fea 7	Fea 8	Fea 9	Fea 10
Fea 1	0.9801	0.8526	0.8483	0.4781	0.4210
Fea 2	0.8591	0.6161	0.9845	0.7706	0.7277
Fea 3	0.8572	0.6584	0.8689	0.6134	0.5687
Fea 4	0.4768	0.2380	0.6577	0.6139	0.5970
Fea 5	0.9541	0.7846	0.9231	0.6031	0.5498
Fea 6	1.0000	0.9323	0.7668	0.3389	0.2770
Fea 7	0.9323	1.0000	0.4858	0.0214	0.0865
Fea 8	0.7668	0.4858	1.0000	0.8634	0.8287
Fea 9	0.3389	0.0214	0.8634	1.0000	0.9979
Fea 10	0.2770	0.0865	0.8287	0.9979	1.0000
Fea 11	0.4907	0.1454	0.9354	0.9860	0.9731
Fea 12	0.2694	0.0944	0.8243	0.9973	1.0000

	Fea 11	Fea 12
Fea 1	0.6153	0.4139
Fea 2	0.8652	0.7223
Fea 3	0.7171	0.5630
Fea 4	0.6480	0.5946
Fea 5	0.7275	0.5432
Fea 6	0.4907	0.2694
Fea 7	0.1454	0.0944
Fea 8	0.9354	0.8243
Fea 9	0.9860	0.9973
Fea 10	0.9731	1.0000
Fea 11	1.0000	0.9712
Fea 12	0.9712	1.0000

Table 4.9. Correlation coefficients between features.

grid size = 64x64

As described in Section 3.3.2, the feature to be deleted is determined by the overall discrimination ability. The computed overall discrimination abilities of each feature for the 5 selected object classes were computed according to Equation (3.2). The results are listed in Table 4.10.

Feature	Overall discrimination ability
1	59.1476
2	58.5136
3	72.8064
4	43.2413
5	81.0322
6	105.6837
7	139.8460
8	32.7623
9	18.7122
10	11.4993
11	15.2778
12	11.6080

Table 4.10. Overall discrimination abilities of each feature for the selected 5 object classes.

Note that the above table by itself provides information on the overall effectiveness of each feature for the given object domain. One can see that feature 7 has the best overall discrimination ability. Considering both correlation coefficient of linear regression and the overall discrimination ability, the decisions to delete a feature were summarized by Equation (3.4).

One can see in Table 4.9 that the correlation coefficient for features 9 and 11 is 0.9860. We thus can delete one of the two without significantly losing discrimination capability.

From Table 4.10, the overall discrimination ability for feature 9 and feature 11 are 18.7122 and 15.2778, respectively. Therefore, feature 11 was deleted based on Equation (3.4).

The above practice was applied to all feature pairs. After using our algorithm to eliminate redundant features, only features 3, 4, 6, 7, and 9 were retained for classification. The results are shown in Table 4.11 which is, in fact, a condensed version of Table 4.7.

	Obj 1	Obj 2	Obj 3	Obj 4	Obj 5
Fea 3	0.6052	0.2286	0.9928	0.9830	0.2388
Dev 3	0.0752	0.0453	0.0149	0.0288	0.0224
Fea 4	0.3948	0.2879	0.0072	0.0170	0.4521
Dev 4	0.0752	0.0389	0.0149	0.0288	0.0202
Fea 6	4.0530	4.9090	4.0090	4.0150	4.4690
Dev 6	0.0044	0.0336	0.0082	0.0061	0.0452
Fea 7	0.0205	18.0200	0.0193	0.0535	2.5510
Dev 7	0.0267	0.5133	0.0207	0.0483	0.3227
Fea 9	0.0049	0.4995	0.0079	0.0145	5.2980
Dev 9	0.0178	0.3182	0.0144	0.0214	1.4400

Table 4.11. Feature values and deviations for
the selected features after removing
redundancy. grid size = 64x64

4.5.2. Selection of Features by Discrimination Ability

Our second criterion for selecting features is to pick a feature with the highest discrimination ability for a pair of object classes. The discrimination ability was computed according to Equation (3.1).

Table 4.12 shows the discrimination ability for the features from Table 4.11. One can see that, for discriminating object classes 1 and 2, the discrimination abilities of features 3, 4, 6, 7, 9 are 3.1251, 0.9369, 22.5679, 33.3343 and 1.5174, respectively. It is obvious that feature 7 should be chosen to best discriminate between them.

In our system, a second best feature is added if the best feature does not give a discrimination ability greater than 2. For object classes 3 and 4, the discrimination abilities of features 3, 4, 6, 7, 9 are 0.2244, 0.2255, 0.4204, 0.4957, 0.1836, respectively. Both features 7 and 6 were selected, since none of the five has a discrimination ability greater than 2.

The above practice was applied to all pairs of object classes. The selected features for each pair of object classes are checked in Table 4.12. One sees that feature 7 is selected 5 times for objects pairs (1,2), (2,3), (2,4), (2,5) and (3,4). This is consistent with the large overall discrimination ability of feature 7 as shown in Table 4.10.

For the present example, 4 features from Table 4.11 were selected. The properties of selected features, 3, 4, 6, and 7, are shown in Table 4.13.

Feature 3		Obj 1	Obj 2	Obj 3	Obj 4	Obj 5
Obj 1	Obj 2	Obj 3	Obj 4	Obj 5		
Obj 1	0.0000	3.1251	4.3019	3.6316	3.7514	
Obj 2	3.1251	0.0000	12.7049	10.1836	0.1506	
Obj 3	4.3019	12.7049	0.0000	0.2244	20.2091 ✓	
Obj 4	3.6316	10.1836	0.2244	0.0000	14.5238 ✓	
Obj 5	3.7514	0.1506	20.2091	14.5238	0.0000	

Feature 4		Obj 1	Obj 2	Obj 3	Obj 4	Obj 5
Obj 1	Obj 2	Obj 3	Obj 4	Obj 5		
Obj 1	0.0000	0.9369	4.3024 ✓	3.6316	0.6001	
Obj 2	0.9369	0.0000	5.2242	4.0033	2.7774	
Obj 3	4.3024	5.2242	0.0000	0.2255	12.6693	
Obj 4	3.6316	4.0033	0.2255	0.0000	8.8705	
Obj 5	0.6001	2.7774	12.6693	8.8705	0.0000	

Feature 6		Obj 1	Obj 2	Obj 3	Obj 4	Obj 5
Obj 1	Obj 2	Obj 3	Obj 4	Obj 5		
Obj 1	0.0000	22.5679	3.5051	3.6336 ✓	8.3922 ✓	
Obj 2	22.5679	0.0000	21.5605	22.5484	5.5866	
Obj 3	3.5051	21.5605	0.0000	0.4204 ✓	8.6170	
Obj 4	3.6336	22.5484	0.4204	0.0000	8.8520	
Obj 5	8.3922	5.5866	8.6170	8.8520	0.0000	

Feature 7		Obj 1	Obj 2	Obj 3	Obj 4	Obj 5
Obj 1	Obj 2	Obj 3	Obj 4	Obj 5		
Obj 1	0.0000	33.3343 ✓	0.0245	0.4407	7.2431	
Obj 2	33.3343	0.0000	33.7098 ✓	31.9904 ✓	18.5036 ✓	
Obj 3	0.0245	33.7098	0.0000	0.4957 ✓	7.3726	
Obj 4	0.4407	31.9904	0.4957	0.0000	6.7313	
Obj 5	7.2431	18.5036	7.3726	6.7313	0.0000	

Feature 9		Obj 1	Obj 2	Obj 3	Obj 4	Obj 5
Obj 1	Obj 2	Obj 3	Obj 4	Obj 5		
Obj 1	0.0000	1.5174	0.1376	0.3294	3.6561	
Obj 2	1.5174	0.0000	1.4782	1.4281	2.7292	
Obj 3	0.1376	1.4782	0.0000	0.1836	3.6374	
Obj 4	0.3294	1.4281	0.1836	0.0000	3.6153	
Obj 5	3.6561	2.7292	3.6374	3.6153	0.0000	

Table 4.12. Discrimination abilities by each selected feature for object class pairs.

grid size 64x64

	Obj 1	Obj 2	Obj 3	Obj 4	Obj 5
Fea 3	0.6052	0.2286	0.9928	0.9830	0.2388
Dev 3	0.0752	0.0453	0.0149	0.0288	0.0224
Fea 4	0.3948	0.2879	0.0072	0.0170	0.4521
Dev 4	0.0752	0.0389	0.0149	0.0288	0.0202
Fea 6	4.0530	4.9090	4.0090	4.0150	4.4690
Dev 6	0.0044	0.0336	0.0082	0.0061	0.0452
Fea 7	0.0205	18.0200	0.0193	0.0535	2.5510
Dev 7	0.0267	0.5133	0.0207	0.0483	0.3227

Table 4.13. Feature values and deviations for
the final selected features.
grid size = 64x64

We also investigated the best feature subset for a different object set. The objects included a square, a rectangle, a right triangle, a circle in hexagon and an asymmetric shape.

The remaining features after removing redundancy were features 2, 4, 5, 6, 7 and 9. Among them, features 2, 6, and 7 were finally chosen, by judging their discrimination abilities, to form the feature subset.

4.5.3. Mahalanobis Distances between Object Classes

Table 4.14 lists Mahalanobis distances in a 4-dimensional feature space using the features in Table 4.13.

	Obj 1	Obj 2	Obj 3	Obj 4	Obj 5
Obj 1	0.0000	40.3872	7.0216	6.3067	11.7185
Obj 2	40.3872	0.0000	42.3073	40.6393	19.5277
Obj 3	7.0216	42.3073	0.0000	0.7237	26.4107
Obj 4	6.3067	40.6393	0.7237	0.0000	20.3297
Obj 5	11.7185	19.5277	26.4107	20.3297	0.0000

Table 4.14. Mahalanobis distances between object classes in the 4-dimensional feature space.

Table 4.8a shows the corresponding Mahalanobis distances but in the 12-dimensional feature space. Comparing Table 4.8a with Table 4.14, one sees that the distance between object classes 1 and 2 decreases slightly from 46.0664 to 40.3872, when the feature space reduces greatly from 12 to 4.

This demonstrates that the recognition speed can be considerably improved if a smaller but well selected feature subset is employed. Roughly speaking, the recognition time is proportional to the number of features. In the present case, the time is reduced to about one third of the original.

However, the distance from object 3 (circle) to object 4 (hexagon) is 0.7237. This short distance makes the discrimination between circle and hexagon very difficult and also not reliable. As shown in Table 4.8a, the distance increases to only 0.9485 even when all 12 features are used.

In such a case, one should by all means explore other features which can better discriminate circle and hexagon.

4.6. Classification of Unknown Objects

In the preceding section, we have discussed Mahalanobis distance between object classes. For unknown classification, we need to determine the distance between an object class and an unknown. Note that an object class is the average of an object and its many distorted states. However, an unknown is an object at a randomly distorted state. Sometimes, two object classes, A and B, are well separated in a feature space, but object class A and unknown B may not be well separated. In such a case, image averaging can enhance recognition accuracy.

4.6.1. Generation of an Unknown Object at a Randomly

Distorted State

Our computer program uses a random number generator to determine the state of each distortion. Based on the random numbers, blurring, change of size, x-axis compression and rotation will be set either "on" or "off". Figures 4.5 to 4.10 show the images of some unknown objects in their randomly distorted states and the identifications made by the system.

4.6.2. Confidence Level on Identification

An unknown object will be identified with a reference object if their Mahalanobis distance is the shortest among the distances between the unknown and all reference objects. However, the identification may be questionable if the next nearest distance is only slightly longer.

This is why the next nearest distance and the confidence level are also given along with the identification in Figures 4.5 to 4.10. The confidence level is defined by Equation 3.4.

In Figure 4.5, the triangle was correctly identified with high confidence. In Figures 4.6 and 4.7, the asymmetric shapes also were confidently identified at both high and low resolutions. Note that the grid sizes for both reference objects and unknown need to be the same for a fair comparison.

In Figure 4.8, a washer, which was not in the selected reference object set, was intentionally chosen to test the system. The nearest reference object was a square, however, with a long distance of 6.10. The system also warned the user that the confidence level was low.

In Figure 4.9, a circle was mistakenly identified as hexagon and the confidence level was low. This was due to a short Mahalanobis distance between a circle object class and a hexagon object class.

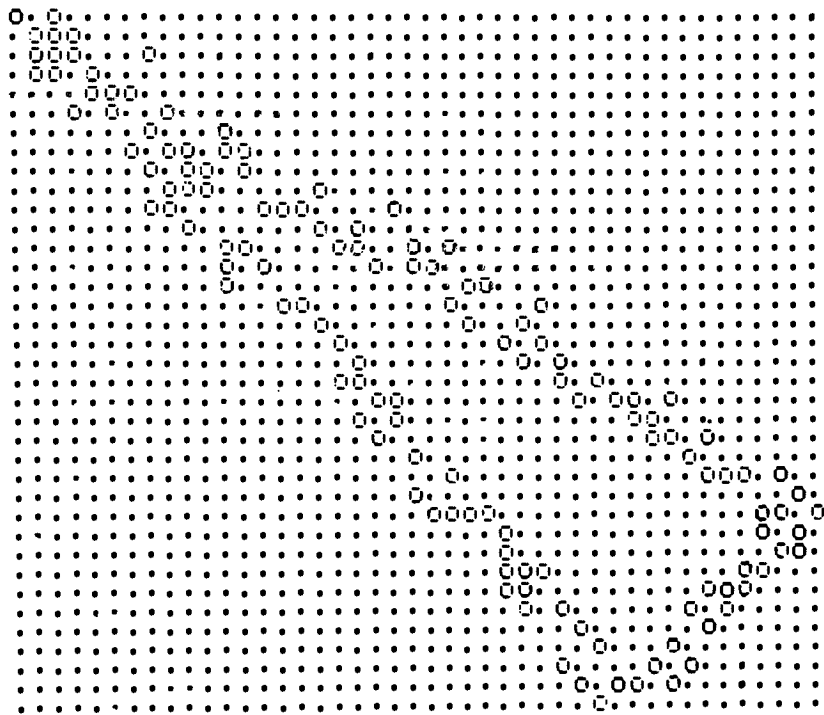


Figure 4.5. unknown object -- a right triangle

blurring by pixel relocation:	+2 to -2
x-axis compression:	no
rotation angle:	51°
size change:	no

Classification:

The unknown object is identified as a right triangle with a Mahanalobis distance of 1.66. The next nearest distance is 49.92 and the corresponding object is an asymmetric shape. Confidence level for identification is high.

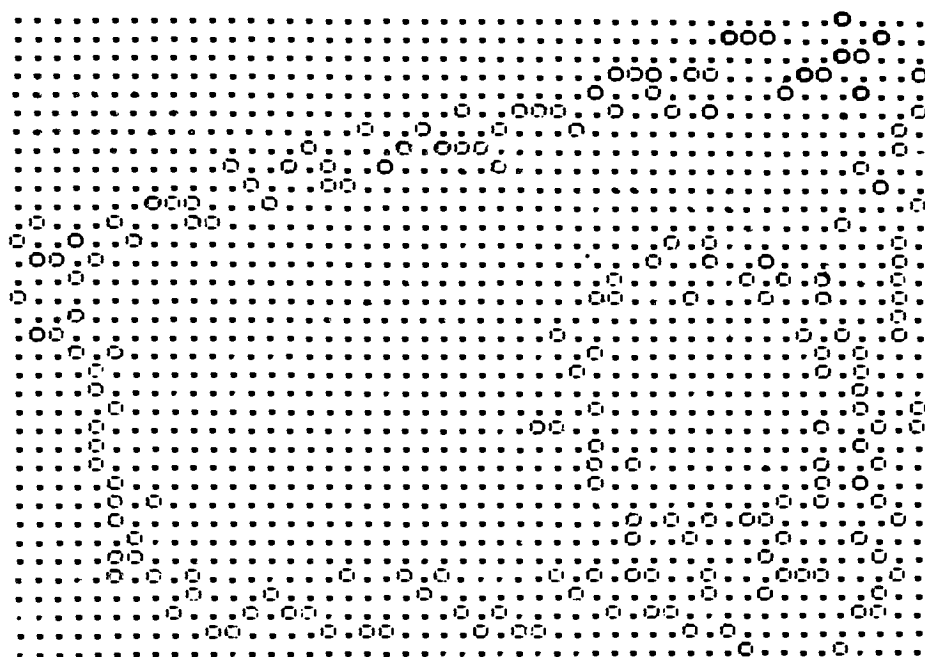


Figure 4.6. unknown object -- asymmetric shape(64x64)

blurring by pixel relocation: +2 to -2

x-axis compression: no

rotation angle: 0°

size change: -10%

Classification:

The unknown object is identified as an asymmetric shape with a Mahanalobis distance of 0.58. The next nearest distance is 33.48 and the corresponding object is a right triangle. Confidence level for identification is high.

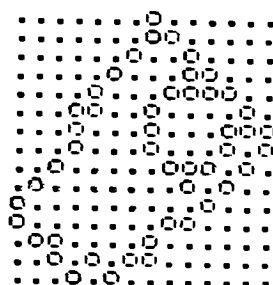


Figure 4.7. unknown object -- asymmetric shape(16x16)

blurring by pixel relocation:	+2 to -2
x-axis compression:	5%
rotation angle:	51°
size change:	+10%

Classification:

The unknown object is identified as an asymmetric shape with a Mahanalobis distance of 1.97. The next nearest distance is 18.72 and the corresponding object is a right triangle.

Confidence level for identification is high.

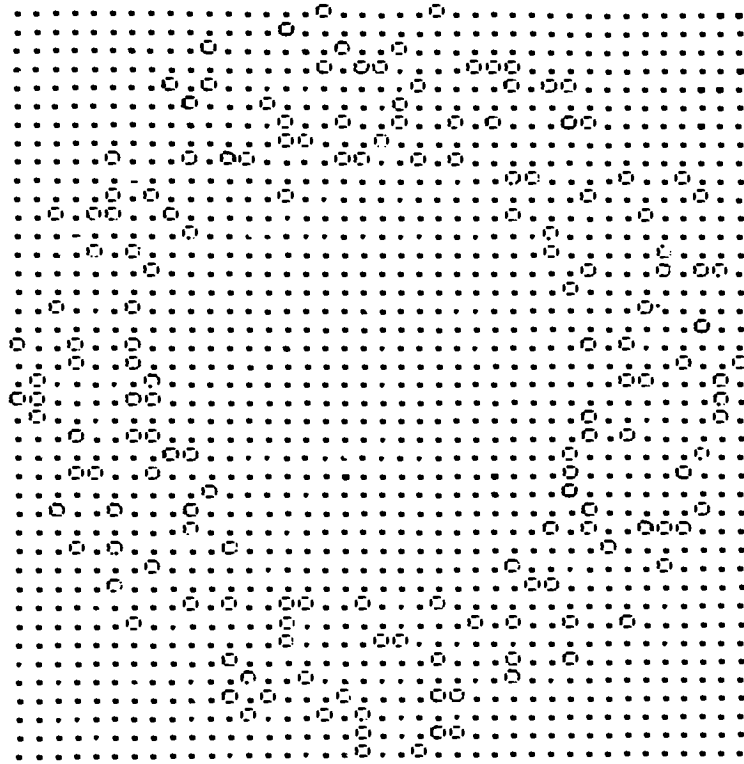


Figure 4.8. unknown object -- a washer

blurring by pixel relocation:	+2 to -2
x-axis compression:	5%
rotation angle:	0°
size change:	-10%

Classification:

The unknown object is identified as a square with a Mahanalobis distance of 6.10. The next nearest distance is 18.15 and the corresponding object is a nut (circle in hexagon). Confidence level for identification is low.

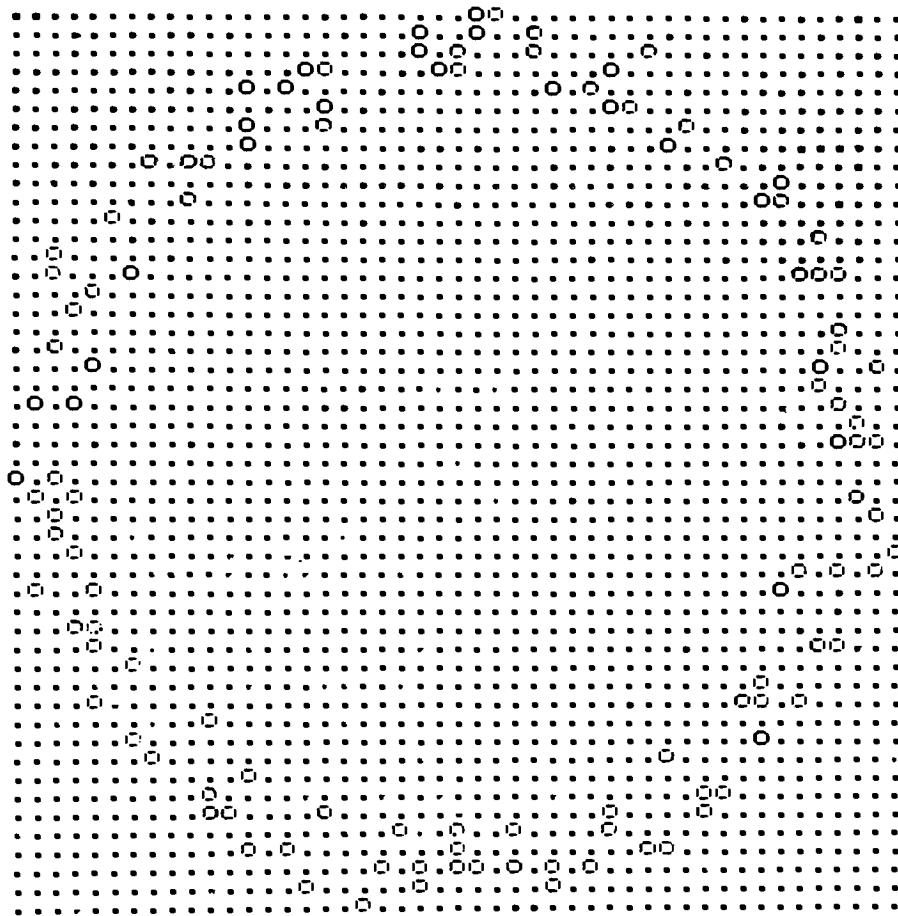


Figure 4.9. unknown object -- a circle

blurring by pixel relocation:	+2 to -2
x-axis compression:	5%
rotation angle:	0°
size change:	+10%

Classification:

The unknown object is identified as a hexagon with a Mahanalobis distance of 0.43. The next nearest distance is 1.87 and the corresponding object is a circle. Confidence level for identification is low.

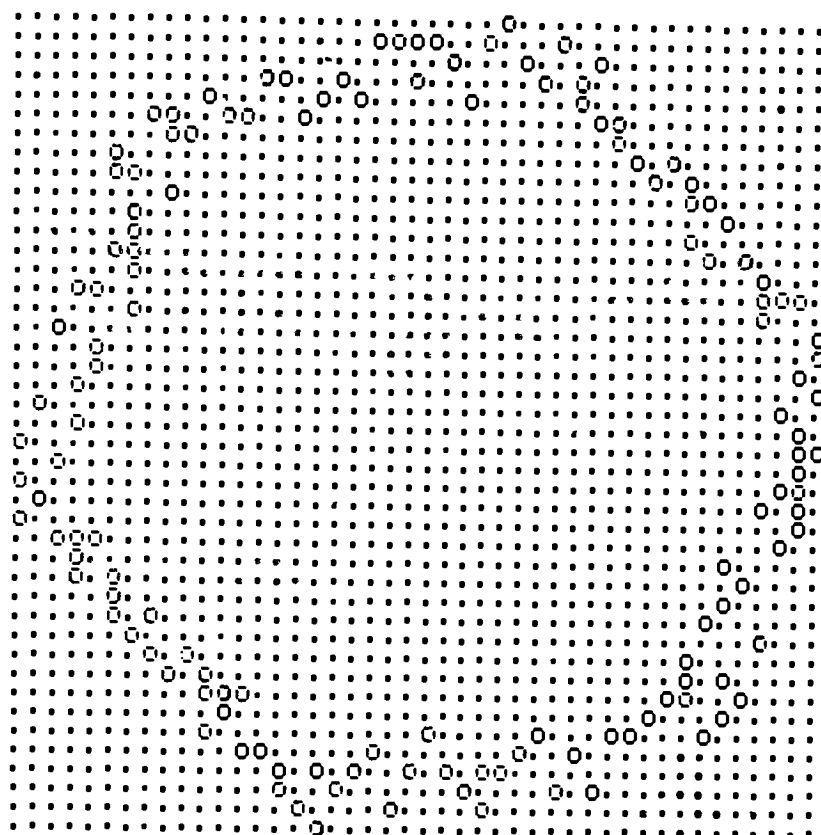


Figure 4.10. unknown object -- a hexagon

blurring by pixel relocation:	+2 to -2
x-axis compression:	5%
rotation angle:	51°
size change:	+10%

Classification:

The unknown object is identified as a hexagon with a Mahanalobis distance of 0.67. The next nearest distance is 1.41 and the corresponding object is a circle. Confidence level for identification is low.

In Figure 4.10, a hexagon was correctly identified as a haxagon but with a low confidence level. Note that an unknown is at a randomly distorted state. As a result, in cases like circle and hexagon, some identifications are correct, but some others are not. However, the system always warns the user about the danger of mis-identification by indicating a low level of confidence.

CHAPTER 5

CONCLUSION, FUTURE WORK, AND OTHER APPLICATIONS

We have successfully constructed a simulated shape recognition system using the method of feature extraction. We believe that the system is a useful tool to do feasibility studies for someone who is interested in buying robot vision equipment. The system also can serve as a framework for future expansion and improvement.

5.1. Future Work

We are serious in comparing this simulated system with an actual robot vision system. We would welcome someone who wishes to test the system with optical and electronic devices to capture images, do digitization and perform edge extraction.

One should start by checking the validity of the simulation of image distortions. One can repeatedly take electronic pictures of the objects in the object pool at various angles and camera-to-object distances. After obtaining the digitized and edge extracted images by hardware, the system can take over from there to compute feature values and standard deviations. Then one needs to adjust the parameters for the controlled distortions currently used by the system (Table 4.3), so that the feature values and the standard deviations can be roughly reproduced by our simulation method. If this can be done, it

implies that our method of simulation is valid. If this can not be done, further work is needed to identify sources of distortion and develop ways to simulate them.

Another extension would be to write an interface between an actual data acquisition device and the system. The interface would enable the system to receive digitized output directly from the hardware device, and perform all the necessary data processing and classification tasks. The system then would become an integral part of an actual robot vision package. Simulation would no longer be needed.

One also could test the system in more complex environment, such as dealing with printed characters, handwritten characters, Chinese characters, machines, machine parts, people and landscapes. One needs to make sure that the computer to be used has adequate memory to handle complex objects.

The current system handles only binary images. One may extend the system to color objects or many gray levels.

The present system deals with 2-D objects. Extending our work to a 3-D environment would be useful. For 3-D objects, three axes are needed for rotating object.

One may expand the feature pool by exploring additional features that are independent of size and orientation. One also may include size and/or orientation dependent features in the feature pool for broader applications.

Currently, confidence levels for object identification is qualitatively classified to high, average or low. It

would be nice to do some mathematical derivation, so that a quantitative confidence level (e.g., 91%) can be given along with each identification. Referring to Equation (3.4), one should be able to calculate a precise confidence level based on the Mahalanobis distances from unknown to the nearest neighbor, D_1 , and to the next nearest neighbor, D_2 .

5.2. Other Applications

We believe that the system can become an inexpensive educational tool to teach students the concept of digitization, feature space, reduction of feature space, classification, etc. The system will be able to generate examples for abstract concept. The students also can work with the system without knowledge of complex hardware.

BIBLIOGRAPHY

[AGIN80]

Agin, G. J., "Robot vision Systems for Industrial Inspection and Assembly", Computer, pp. 11-20, 1980.

[BERT86]

Berthold, Horn., Robot Vision, Cambridge, Mass. MIT Press, 1986.

[BOIE87]

Boie, R. A., and Cox, Ingemar J., "Two Dimensional Optimum Edge Recognition Using Matched and Wiener Filters for Machine Vision", Proceedings of First International Conference on Robot vision, pp. 450-456, 1987.

[BOW84]

Bow, Sing-Tze, Pattern Recognition, Marcel Dekker, Inc., New York and Basel, 1984.

[BRES77]

Bresenham, J. E., "A Linear Algorithm for Incremental Digital Display of Circular Arcs", Communications of the ACM, 20(2), pp. 100-106, 1977.

[CAEL87]

Caelli, Terry and Nagendran, Shyamala, "Fast Edge-Only Matching Techniques for Robot Pattern Recognition", Robot vision, Graphics and Image Processing, Vol. 39, pp. 131-143, 1987.

[CAGN86]

Cagney, F. and Mallon, J., "Real-time feature extraction using moment invariants", from Intelligent Robots and Robot vision, ed. Casasent, David P., SPIE, Bellingham, Washington, Vol. 726, pp. 120-124, 1986.

[CASH87]

Cash, G. L. and Hatamian, M., "Optical Character Recognition by the Method of Moments", Robot vision, Graphics and Image Processing, Vol. 39, pp. 291-310, 1987.

[CHEN86]

Chen, J. S., Huertas, A. and Medioni, G., "Very Fast Convolution with Laplacian-of-Gaussian Masks", Proceedings of IEEE Computer Society Conference on Robot vision and Pattern Recognition, pp. 293-298, 1986.

[DAVI75]

Davis, L. S., "A Survey of Edge extraction Techniques", Computer Graph and Image Processing, Vol. 4, pp. 248-270, 1975.

[DUDA73]

Dura R. O. and Hart P. E., Pattern Classification and Scene Analysis, John Wiley & Sons, Inc., New York, 1973.

[FREU84]

Freund J. E., Modern Elementary Statistics, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1984.

[GOSH83]

Goshorn, L. A., "Vision Systems Eye Real-time Speeds through Multiprocessor Architectures", Electronics, Vol. 56, pp. 137-140, 1983.

[HOOP86]

Hooper, R. P. and Klinger, A., "Artificial Pattern Generation", from Pattern Recognition in Practice II, eds. Gelsema and Kanal, L. N., Elsevier Science Publishers B. V. (North-Holland), pp. 38-46, 1986.

[HU62]

Hu, M. K., "Visual Pattern Recognition by moment invariants", IRE Trans. Inform. Theory, IT-8, pp. 179-187, 1962.

[HUTT87]

Huttenlocher, Daniel P. and Ullman, Shimon, "Object Recognition Using Alignment", Proceedings of First International Conference on Robot vision, pp. 102-111, 1987.

[LI86]

Li, Xiaobo, Ferdous, Mahbuda, Chen, Marie and Nguyen, Thanh, Thny, "Image Matching with Multiple Templates", IEEE Computer Society Conference on Robot vision and Pattern Recognition, pp. 610-613, 1986.

[MUCC71]

Mucciardi, Anthony N. and Gose, Earl E., "A comparison of Seven Techniques for Choosing Subsets of Pattern Recognition Properties", IEEE Transactions on Computers, pp. 1023-1031, 1971.

[MUND83]

Mundy, J. L., "Image Input Systems", from Proc. Industrial Applications of Image Analysis, eds. Zimmerman, N. J. and Oosterlinck, A., D. E. B. Publishers, Antwerp, pp. 69-95, 1983.

[PCVI86]

PCVISION Frame Grabber Manual, Imaging Technology Inc., 1986.

[RAPA88]

Rapaport, Steven J., "The Secret Art of Frame Grabbing",
Photonics Spectra, pp. 137-140, 1988.

[ROSE66]

Rosenfeld, A. and Pfaltz, J. L., "Sequential Operations
in Digital Picture Processing," J. ACM, Vol. 14, No. 4, pp.
471-494, 1966.

[SHIO86]

Shiozaki, A., "Edge Extraction Using Entropy Operator".
Robot vision, Graphics and Image Processing, Vol. 36, pp. 1-
9, 1986.

[SUET86]

Suetens, P. and Oosterlinck, A., "Industrial Pattern
Recognition", from Pattern Recognition in Practice II, eds.
Gelsema, E. S., and Kanal, L. N., Elsevier Publishers, B. V.,
North Holland, pp. 345-360, 1986.

[TECH86]

Technical Specifications of Palantir Compound Document
Processor, The Palantir Corporation, Santa Clara,
California, 1986.

[VANG86]

Van Gool, L., Vermeyen, P. and Oosterlinck, "Vision-
Based Object Recognition and Acquisition", from Intelligent
Robots and Robot vision, ed. Casasent, D. P., SPIE,
Bellingham, Washington, Vol. 726, pp. 306-313, 1986.

[WONG78]

Wong, R. Y. and Hall, E. L., "Scene Matching with
Invariant Moments", Comput. Graphics Image Process, Vol. 8,
pp. 16-24, 1978.

[WU86]

Wu, R. and Stark, H., "Rotation-Invariant Pattern
Recognition Using Optimum Feature Extraction", from Pattern
Recognition in Practice II, eds. Gelsema, E. S. and Kanal,
L. N., Elsevier Science Publishers, B. V. (North-Holland),
pp. 401-410, 1986.

[ZAKA87]

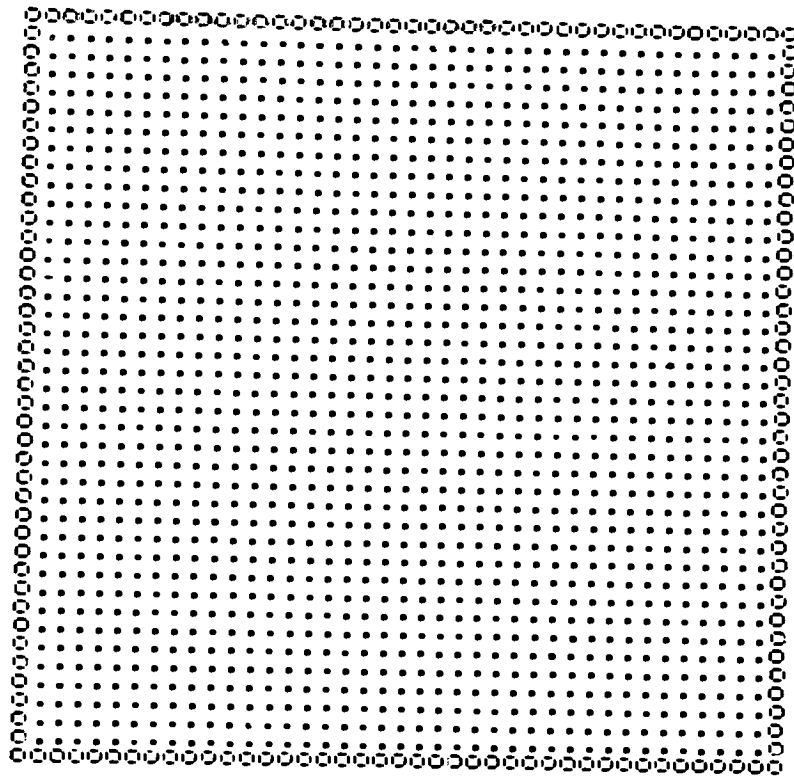
Zakaria, M. F., Zsombor-Murray, P. J. A. and Van Kessel,
J. M. H. M., "Fast Algorithm for the Computation of Moment
Invariants", Pattern Recognition, Pergamon Journals Ltd.,
Vol. 20, No. 6, pp. 639-643, 1987.

[ZIMM83]

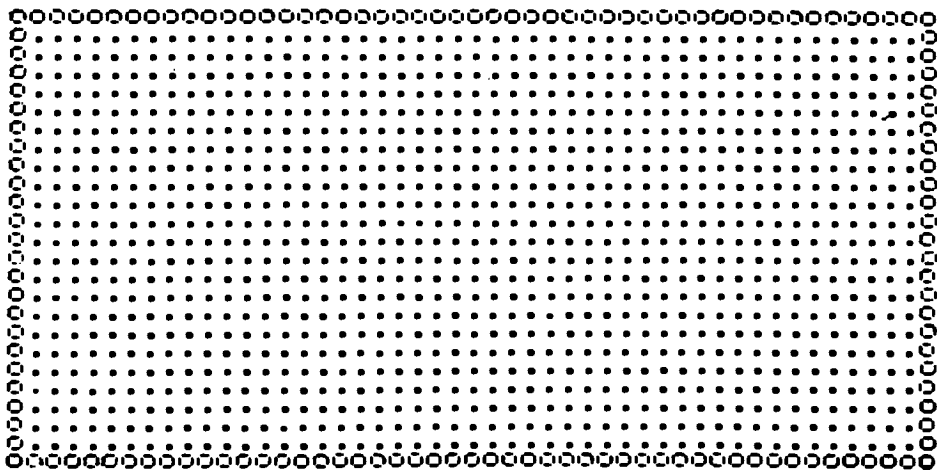
Zimmerman, N. J., Van Boven, G. J. R. and Oosterlinck,
A., "Overview of Industrial Vision Systems", from Proc.
Industrial Applications of Image Analysis, eds. Zimmerman,
N. J. and Oosterlinck, A., D. E. B. Publishers, Antwerp. pp.
193-231, 1983.

APPENDIX

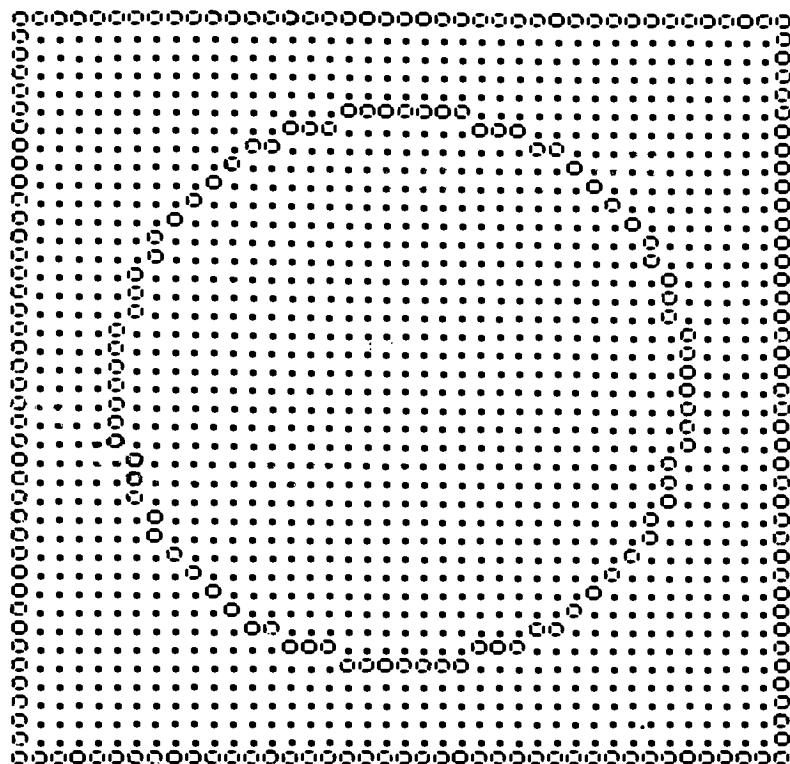
Ten Undistorted Objects in the Object Pool



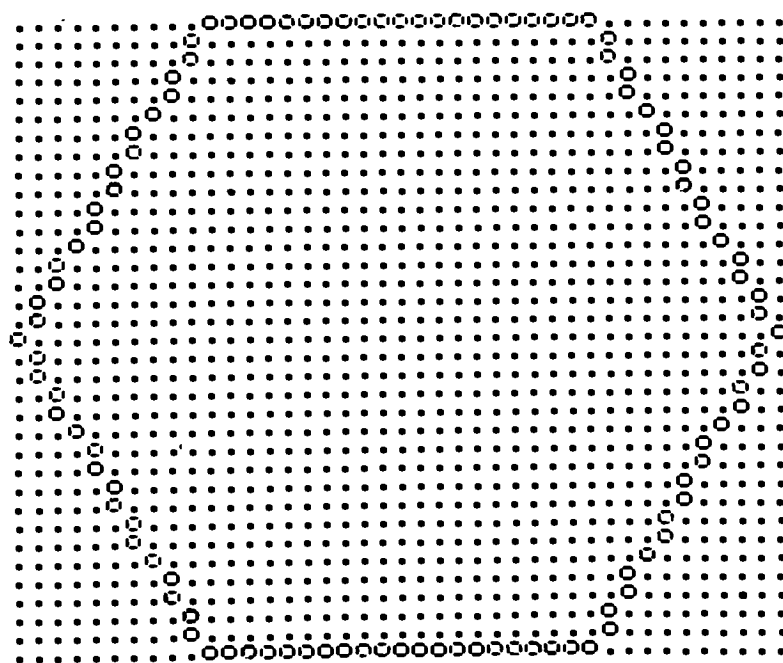
1. A square.



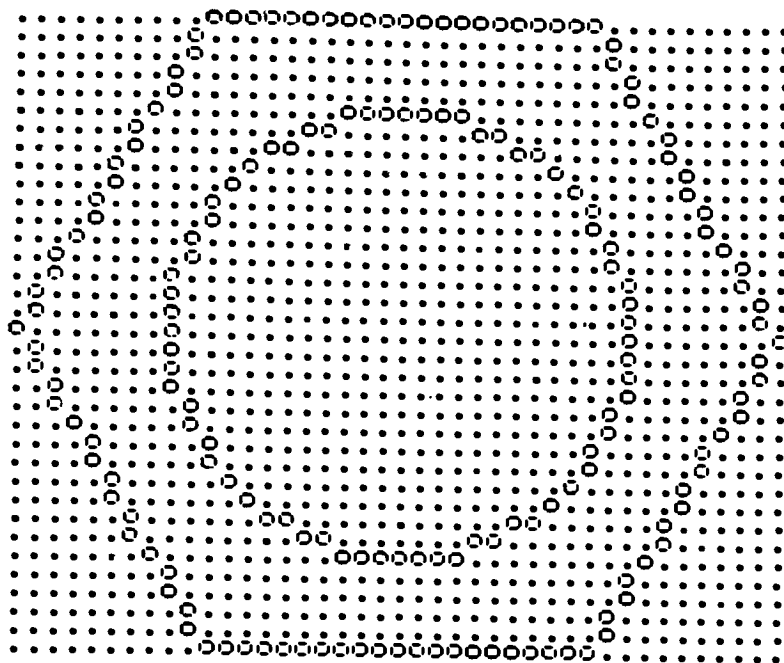
2. A rectangular.



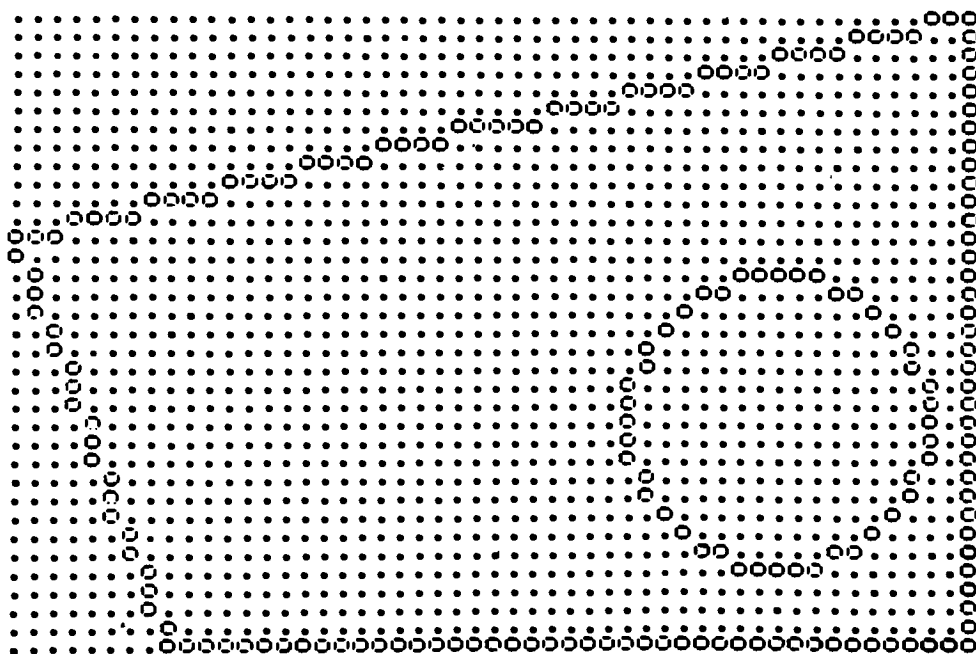
7. A nut (circle in square).



8. A hexagon.



9. A nut (circle in hexagon).



10. An asymmetric shape.