CHAPTER-3

Feature Extraction Techniques

3.1 Introduction

For some tasks that a robot performs, vision is the most important source of information about the environment. The recognition of objects, the perception of relation between them and appropriate response to a given scene lie at the foundation of robotics vision. Even though the human vision system is sophisticated and robotic vision crude in comparison, the rudimentary vision system can augment a robot’s capability manifold so as to permit activities, which would otherwise be difficult for it to perform.

An object can be recognized and classified properly if it possesses a number of discriminating properties. The minimum number of such properties or features needed for recognition is called primitive features. The primitive features form a feature vector. Fourier Descriptors can be employed as feature vectors to represent objects of different shapes.

Feature extraction is an important step in achieving best performance of any recognition system. The choice of feature extraction method limits or dictates the nature and output of the preprocessing step. Some feature extraction methods work on gray level sub images of single character while others work on solid four to eight connected symbols, skeleton or symbol counters. Further the type of or format of the extracted features must match the requirement of the chosen classifier. The criteria for selection of feature extraction from, the
character images are.

1. Number of features should be limited.

2. Features should be invariant to the expected distortion and variations in images.

3. The type of extracted features must match the requirement of a chosen classifier.

4. The features extracted from binary images are found to be sufficient for better classification.

Human visual system has been perfectly adapted to handle uncertain information in both data and knowledge. It would be hard to define quantitatively how an object, such as a car, has to look in terms of geometrical primitives with exact shapes, dimensions and colors. Instead we are using a descriptive language to define features that eventually are subject to a wide range of variations. The interrelation of a few such “Fuzzy” properties sufficiently characterize the object of interest. Fuzzy image processing is an attempt to translate this ability of human reasoning into computer vision problem as it provides an intuitive tool for inference from imperfect data.

3.2 Processing of Object for Feature extraction

The picture is captured via a digital camera; it is digitized and stored in the memory of the computer. A binary picture as shown in fig. 3.1 is quite sufficient in most of the applications. Before the features are being extracted from the scanned image certain preprocessing is required to be carried out on these images. These preprocessing steps are performed directly on the image obtained from the scanner. The main aim of the preprocessing methods is to manipulate the image so that feature extraction is simplified and we get required features as expected for recognition.
3.2.1 Binarisation

Any character image \( f(x,y) \) is composed of light objects on a dark background, in such way that object and background pixels have gray levels grouped into dominant modes. One obvious way to extract the objects from background is to select a threshold \( T \) that separates these modes. Point \( (x,y) \) for which \( f(x,y) > T \) is called an object point; otherwise, the point is called a background point. Global binarization methods calculate a single threshold value for the entire image \( f(x, y) \).

Pixels having a gray level darker than the threshold value are labeled black and others are labeled white.

**Algorithm for Binarisation**

<INPUT> Gray scale image  
<OUTPUT> Binary image

1. Find the gray level histogram of input image \( f(x,y) \) on a dark background.
2. Binary image \( g(x,y) \) is defined as

\[
g(x,y) = \begin{cases} 1 & \text{if } f(x,y) > T \\ 0 & \text{otherwise} \end{cases}
\]

where \( T \) is threshold ................. [3.1]
3.2.2 Spatial Filtering

The use of spatial mask for image processing is usually called as the spatial filtering and masks used are called as the spatial filters. The net effect of the low pass filtering is image blurring and high pass filters is the reduction in contrast and average intensity. Figure 3.2 shows a 3x3 mask with arbitrary coefficients and gray levels of pixels under this mask at any location are represented by \( z_1, z_2, z_3, z_4, z_5, z_6, z_7, z_8 \) and \( z_9 \). Then the response of this linear mask is given by

\[
R = W_1 \cdot z_1 + W_2 \cdot z_2 + W_3 \cdot z_3 + \ldots + W_9 \cdot z_9
\]  

[3.2]

If the center of the mask is at location \((x, y)\) in the image, the gray level of the pixel located at \((x, y)\) is replaced by \( R \). The mask is moved to the next pixel location in the image and the process is repeated. This process is continued till all the pixels locations have been covered in the image.

3.2.3 Boundary Detection

The boundary of image can be produced by emphasizing regions containing abrupt dark-light transitions, and de-emphasizing regions approximately homogeneous intensity. Another way of putting this is to say that outlines are edges, and edges are by definition transitions between two dissimilar intensities in the image. In terms of the picture function, an edge is a region of the \(X-Y\) plane where \( g(x, y) \) has a gradient with a large magnitude. Let us approximate the
magnitude of the gradient at image point \((i, j)\) by
\[
\|\delta_x(i, j) \approx R(i, j)\| = \sqrt{[g(i, j) - g(i + 1, j + 1)]^2 + [g(i, j + 1) - g(i + 1, j)]^2} \tag{3.3}
\]

Notice that this corresponds to selecting orthogonal directions of the lines whose slopes are plus and minus one. Diagrammatically, we consider at cell \((i, j)\) a 2X2 window whose diagonal elements are associated by subtraction.

![Figure 3.3: Robert's Filter.](image)

The directional derivative in each direction is approximated by simply subtracting adjacent elements as shown in the figure 3.3. The operator \(R(i, j)\) is sometimes called the Roberts cross operator. The Roberts cross operator is often simplified for computational efficiency by using absolute magnitudes rather than squares and square roots. We define \(f(i, j)\) by
\[
f(i, j) = |g(i, j) - g(i + 1, j + 1)| + |g(i, j + 1) - g(i + 1, j)| 
\]

\[
3.2.4 \text{ Spatial Smoothing}
\]

This operation is carried out to remove the sharp peaks present in the image, which may be present in the image because of any of the following reasons.

- Inherent limitations of scanner.
- Variations of writing pressure.
- Quality of the writing surface and instrument.

Smoothing can be done by computing the sum of object pixels in a 8-pixel neighborhood around each pixel in the image. When this sum is greater than a threshold, the pixel is made an image pixel, else it is replaced by background pixel. This method involves a computation
over the \((L \times W)\) size of character image where \(L\) is the length and \(W\) is the width in terms of pixels. This operation fills out the rough regions of the noisy image. A low pass spatial filter mask is shown in Figure 3.4

\[
\begin{array}{ccc}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1
\end{array}
\]

Figure 3.4: Low pass spatial filter.

**Algorithm for smoothing the image**

<INPUT> ROUGHED CHARACTER IMAGE  
<OUTPUT> SMOOTHED CHARACTER IMAGE

1. Apply a \((3 \times 3)\) mask having all coefficients with value of \("1\"\) to an image \(f(x,y)\)

2. Take average of the all coefficients with the filter response

\[
R = \frac{1}{9} \times (z_1 + z_2 + z_3 + z_4 + z_5 + z_6 + z_7 + z_8 + z_9)
\]
and replace the center pixel value \((x,y)\) with the value of \(R\)

3. Repeat the same process for entire image.

This method is implemented in distance measurement method for eliminating gaps in boundary of character image.

**3.2.5 Noise Elimination**

Preprocessing is performed on the presented word or character in order to minimize the effect of spurious noise in the subsequent processing stages. Normally smoothing filters are used for noise elimination and blurring. The low pass spatial filters are used to blur or rub the noise details rather than complete elimination, so an alternative approach is to use median filter. This method is particularly effective when the noise pattern is strong.
Algorithm for noise elimination

<INPUT> Input Image with Noise
<OUTPUT> Output Image without noise

1. Apply 3x3 mask having all coefficients with value of '1' to image.

2. Sort the values of pixel and its neighbours either in ascending order or descending order.

3. Determine the median, and assign this value to the pixel. Using this algorithm we can eliminate noise.

This algorithm can eliminate noise of small spikes. The median filter is not applicable for skeleton, thin and contour type characters.

3.2.6 Normalization

Normalization is a process of expansion or contraction of all points of image by the same amount. Normalization process cannot distort the significant features in case of binary solid images.

Algorithm for Normalization

<INPUT> Any size of Character
<OUTPUT> 64x64 Size Character

1. Calculate character width and height.

2. If character width is less or more than 64 size.

3. Then expand or compress the image in x-direction.

4. If character height is less or more than 64 size.

5. Then expand or compress the image in y-direction.

3.2.7 Segmentation

In most existing Recognition systems, recognition is performed on individual images. Image segmentation is a technique, which partitions images of lines or words into individual, which poses likely characteristics. The thinning and/or thresholding process in the
binarization step can cause more broken, touching and merged characters. The most common segmentation algorithms are based on vertical projection, pitch estimation or image size, contour analysis. The evaluation of a segment technique is not easy to say there in no universal definition of correct segmentation. In many systems, segmentation is considered as an integral factor in recognition and that recognition results determine the performance of segmentation algorithms.

### 3.3 Feature extraction & selection

In literature survey we have discussed how feature extraction and selection process is important. The methods suggested are time consuming and cannot be used in online application of recognition. We are basically interested in online vision applications. For a robot we wish that the robot should make the decisions by looking at the object, analyzing it and then recognize it for further processing. The conventional methods as suggested in literature are unable to provide this facility hence we have tried to use the well known image processing techniques, Geometrical information of the object and some advance techniques like fuzzy logic to extract features for the system we proposed.

While implementing this system many questions comes to our mind like

1. Which features should be detected and how can they be detected reliably?
2. Most features can be computed in two-dimensional images but they are related to three-dimensional characteristics of objects.
3. How can features in images be matched to models in the database?
4. Effectiveness of features and efficiency of matching technique
must be considered in developing a matching approach.

5. How can a set of likely objects based on the feature matching be selected?

6. How can probabilities be assigned to each possible object?

7. This measure reflects the likelihood of the presence of objects based on the detected features.

8. How can object models are used to select the most likely object from the set of probable objects in the given image?

9. The presence of each likely object can be verified by using their models or not.

We have tried our level best to answer all the above questions by implementing suitable feature extraction, feature selection, and object recognition algorithms, which suits to our requirement and suggested a system which can be full proof depending on application domain where we wish to employ this robot.

We have selected three areas where our robot with vision capacity can be employed.

3.3.1 Objects of known geometrical structure
Type one objects are known geometrical structures; we have operated following features extraction methods.

a. Feature extraction using geometric methods.

b. Feature extraction using moment's methods.

c. Feature extraction using Fourier Descriptors.

3.3.2 Objects of fixed font Character Shape
In type two objects we have considered some fixed font English alphabets type objects: we have operated following feature extraction methods on them.
3.3.3 Complex Images

In type three objects we have considered some complex images like human face and have operated following feature extraction methods.

a. Feature extraction using zoning methods.

b. Feature extraction using distance measurement techniques.

c. Feature extraction using gradient methods.

Once features have been extracted a minimum error rate discriminant function for classification has been selected. Depending on the type of object few of these techniques are employed to calculate the various features, which can describe the object. A covariance matrix is formed using Minimum error rate discriminant function, which has been calculated from conditional density function by making certain assumptions. The actual discriminant function used is

\[ g(x) = \log p(x | \omega_i) + \log p(\omega_i) \] ... (3.5)

The solution for these sets of equations can be evaluated if the Density \( p(x | \omega_i) \) is multivariate normal. If we assume \( p(x | \omega_i) = N(\mu_i, \Sigma_i) \) and if multivariate normal density function which is given by equation

\[ p(x) = \frac{1}{(2\pi)^{d/2} | \Sigma |^{1/2}} \exp[-\frac{1}{2}(x - \mu)^t \Sigma^{-1}(x - \mu)] \] .......................... (3.6)

We can write the discriminant function as

\[ g_i(x) = -\frac{1}{2}(x - \mu_i)^t \Sigma_i^{-1}(x - \mu_i) - \frac{d}{2} \log 2\pi - \frac{1}{2} \log | \Sigma_i | + \log p(\omega_i) \] ....... (3.7)
There are possibilities of getting three types covariance matrices

1. When features are statistically independent and each feature has the same variance $\sigma^2$ we can use the discriminant function

$$g_i(x) = -\frac{\|x - \mu_i\|^2}{2\sigma^2} + \log p(\omega_i)$$ ................................. [3.8]

2. When the covariance matrices for all the classes are identical i.e. $\Sigma = \Sigma_i$

3. When covariance matrices are different for each category.

We have used the first case i.e. assuming that features are statistically independent. This method is also called as minimum distance classifier or linear machine. Using this minimum distance classifier classification is done. Some sequences of steps have been implemented to identify the possible object in the image classification. Basic idea in classification is to recognize objects based on features. Pattern recognition approaches fall in this category and their potential has been demonstrated in many applications. Pattern recognition is the science that concerns the description or classification of measurements. A pattern can be as basic as a set of measurements or observations. They can be represented in matrix or vector forms. Features are any extractable measurements used like signal intensities, It may be symbolic, numerical or both.(e.g. Color, Weight etc.).

3.4 Implementation of feature extraction algorithms

We have implemented algorithms for feature extraction with following techniques.

1. Feature extraction using geometric methods.
2. Feature extraction using moment's methods.
3. Feature extraction using Fourier Descriptors.
4. Feature extraction using zoning methods.
3.4.1 Feature extraction using geometric methods.

The problem of determining the identity of an object from the knowledge of its characteristics is dealt with in the branch of applied science called pattern recognition. In a typical robotic application, the number of shapes to be identified at any given time is quite small. A few features or characteristics of these shapes, which are used for shape discrimination, form the feature set. The representation or templates of various possible shapes are predetermined and stored in the computer. When an unknown object is to be classified, its features are compared with the stored features and the nearest match is taken to indicate the probable shape.

Some important features are found in the

1. Planer area of the object
2. Boundary of the object
3. At the corners of the object

Circularity, corner angle and angular changes around the boundary can also be used as features. One can also use the area information to identify the object by employing the two-second order moments about the two principal axis of the object. The recognition procedure can be made independent of rotation, size and location of the object in the scene.

The important by-products of this scheme are
1. Location of the centre of gravity
2. Orientation of the object.

These two data are useful in guiding the robot's gripper while grasping the identified object.

The information about the object is available in a $N \times N$ binary array. At any pixel the value '1' may represent the presence of the material and value '0' the absence of material i.e. background.

First of all, the centre of area is computed

Let centre be $({C}_x, {C}_y)$

The object pixel is represented by $x(i,j)$ such that,

$x(i,j)=1$ if the material is present
$x(i,j)=0$ if the pixel corresponds to the background.

Now

$$C_x = \frac{\sum_{i=1}^{N} \sum_{j=1}^{N} i \cdot x(i,j)}{\sum_{i=1}^{N} \sum_{j=1}^{N} x(i,j)} \quad \text{...... (3.9)}$$

$$C_y = \frac{\sum_{i=1}^{N} \sum_{j=1}^{N} j \cdot x(i,j)}{\sum_{i=1}^{N} \sum_{j=1}^{N} x(i,j)} \quad \text{...... [3.10]}$$

The center of gravity information is used for positioning the gripper of the robot, with its jaws open, around the object, as a first step in picking it up. The first moments about the center of gravity are given by

$$m_{10} = \sum_{i=1}^{N} \sum_{j=1}^{N} (j - C_y) \cdot x(i,j) \quad \text{...... [3.11]}$$

$$m_{01} = \sum_{i=1}^{N} \sum_{j=1}^{N} (i - C_x) \cdot x(i,j) \quad \text{...... [3.12]}$$
Since the object is balanced about the center of gravity, the first moments $m_{10} = 0 = m_{01}$. However, higher order moments about the center of gravity do exist. The second moments about center of gravity are computed as follows.

\[
m_{20} = \sum_{i=1}^{N} \sum_{j=1}^{N} (j - C_y)^2 * x(i, j) \quad \ldots \quad [3.13]
\]

\[
m_{22} = \sum_{i=1}^{N} \sum_{j=1}^{N} (i - C_x)^2 * x(i, j) \quad \ldots \quad [3.14]
\]

\[
m_{11} = \sum_{i=1}^{N} \sum_{j=1}^{N} (i - C_x)(j - C_y) * x(i, j) \quad \ldots \quad [3.15]
\]

The second moments of a given picture depends on the orientation. To make the moments independent of orientation, we determine the moments about the principal axes. For achieving this, we may rotate the picture so that the principal axes coincide with the x and y-axes as shown in figure 3.5.

Let $\beta$ be the angle between the principal axes frame and the $(x, y)$ frame as shown in figure 3.5.

Then \[ \tan 2\beta = \frac{2m_{11}}{m_{20} - m_{22}} \quad \ldots \quad [3.16] \]

This expression gives the orientation of the object's principal axes w.r.t the $(x, y)$ coordinate system of the scene. The robot can pick the object easily either along or across the principal axes as specified by...
the user.

Other moments recalculated about the principal axes are:

\[ m'_{20} = \frac{m_{20} + m_{02}}{2} + \frac{m_{20} - m_{02}}{2} \cos 2\beta - m_{11} \sin 2\beta \]  \[ \ldots [3.17] \]
\[ m'_{02} = \frac{m_{20} + m_{02}}{2} + \frac{m_{20} - m_{02}}{2} \cos 2\beta + m_{11} \sin 2\beta \]  \[ \ldots [3.18] \]
\[ m'_{11} = 0 \]  \[ \ldots [3.19] \]

Thus we are left with the area and the two-second order moments about the principal axes for object recognition. However to obtain size invariant object recognition we can normalize the second order moments by the square of the area.

\[ F_1 = \frac{m'_{20}}{(\text{area})^2} \quad \text{and} \quad F_2 = \frac{m'_{02}}{(\text{area})^2} \]  \[ \ldots [3.20] \]

the area of the binary picture is simply the sum of all the 'ones' in the binary picture i.e.

\[ \text{area} = \sum_{i=1}^{N} \sum_{j=1}^{N} x(i, j) \]  \[ \ldots [3.21] \]

Calculation of second order of moments for square objects:

![Diagram of a square object](image)

**Figure 3.6: Moments Computations for square objects**

Taking the element strip of a square as shown in the figure 3.6, we see that the element area is \((a/2)dx\) and the second moment about the y axis is

\[ m'_{02} = 4 \int x^2 \left( \frac{a}{2} \right) dx = \left[ \frac{a^2}{4} \right] \]  \[ \ldots [3.22] \]

it may be noted that the moments are being taken about principal
axis. The area of the square is $a^2$

The normalized second order moment about the y axis is

$$F_2 = \frac{m^{'\prime}_{02}}{(a^2)^2} = \left[ \frac{1}{12} \right]$$

......[3.23]

Similarly the normalized second order moment about the x axis is given by

$$F_1 = \frac{m^{'\prime}_{20}}{(a^2)^2} = \left[ \frac{1}{12} \right]$$

......[3.24]

Calculation of second order of moments for circle.

Figure 3.7 : Moments Computation for Circular Objects.

Element area for circle is $\partial A = 2a\cos\theta \partial x$ but $x = a\sin\theta \partial \theta$ and hence $\partial x = a\cos\theta \partial \theta$ and therefore $\partial A = 2a^2 \cos^2\theta \partial \theta$. Taking the second moment about the principal axis $y$,

$$m^{'\prime}_{02} = 2 \int_{\theta=0}^{\theta=\pi} x^2 \partial A = 2 \int_{0}^{\pi} (a\sin\theta)^2 2a^2 \cos^2\theta \partial \theta = a^4 \int_{0}^{\pi} \sin^2 2\theta \partial \theta$$

$$= a^4 \left[ \frac{1 - \cos 4\theta}{2} \right]_{0}^{\pi} = \frac{\pi a^4}{4}$$

. ............[3.25]

The normalised second order moment is:

$$m^{'\prime}_{02} = F_2 = \frac{\pi a^4}{4} \cdot \frac{1}{\pi^2 a^4} = \frac{1}{4\pi}$$

Similarly $m^{'\prime}_{20} = F_1 = \frac{1}{4\pi}$ by symmetry.

3.4.2 Feature extraction using moment's methods.

Imagine a object within a binary image. The object is represented by the pixel that are turned on and the background by the pixels that are
turned off. Considering a general moment equation

\[ M_{a,b} = \sum_{x,y} x^a y^b \] .............. (3.26)

Where \( M_{a,b} \) is the moment of the object within the image with indices \( a \) and \( b \) and \( x \) and \( y \) are the coordinates of each pixel that is turned on within the image raised to power of \( a \) and \( b \) as shown in the figure 3.8. \( M_{0,0} \) is the moment of the object with \( a=0 \) and \( b=0 \). This means that all \( x \) and \( y \) values are raised to a power of 0. \( M_{0,0} \) is same as area of the object. The location of the center of the area relative to the \( x \)-axis and \( y \)-axis can be calculated as

\[ y = \frac{\sum_{area} y}{M_{0,0}} \] and \[ x = \frac{\sum_{area} x}{M_{0,0}} \] .............. (3.27)

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 3.8: Image used for calculations of Moments.

Measuring the distance of each pixel from the \( x \) and \( y \) axes and substituting the measurements into the moment's equations yields the following results.

\[ M_{0,0} = \sum x^0 y^0 = 12(1) = 12 \]

\[ M_{1,0} = \sum x^1 y^0 = \sum x = 2(1) + 1(2) + 3(3) + 3(4) + 1(5) + 2(6) = 42 \]

\[ M_{0,1} = \sum x^0 y^1 = \sum y = 1(1) + 5(2) + 5(3) + 1(4) = 12(1) = 12 \]

\[ y = \frac{\sum_{area} y}{M_{0,0}} = \frac{42}{12} = 3.5 \] and \[ x = \frac{\sum_{area} x}{M_{0,0}} = \frac{30}{12} = 2.5 \]
3.4.3 Feature extraction using Fourier Descriptors.

Fourier descriptors are Fourier coefficients and phase angles obtained from the Fourier analysis of angular or directional changes along the boundary of a pattern. The Fourier descriptors provide a means for identifying different shapes, irrespective of translation, rotation and magnification. The angular rotation of the objects can also be quantitatively estimated using Fourier descriptors.

Boundary Following

Let us assume that it is required to extract the feature vectors of a shape F shown in Figure 3.9. The boundary of the object is uniformly divided into 18 parts. Starting at the point 18, the boundary is followed in the clockwise direction, simultaneously noting down the changes in the directions.

While moving around

at 1 we find \(-90^\circ\) as the change in direction (clockwise)
at 2 the direction changes by \(-90^\circ\).
at 3 there is no change in direction.
at 4 the direction changes by \(+90^\circ\) (anticlockwise).
at 5 the change in direction is again \(+90^\circ\).

Thus a plot of change in direction along the boundary w.r.t. the cumulative distance covered is obtained. The total distance covered is 18 cms and the net angular change is \(-360^\circ\). The pattern is repeated after 18 cm periodically. It is possible to recalibrate the distance 0-18 cms in terms of radians 0-2\(\pi\).

A full sine wave can be fitted between 0 and 2\(\pi\) (or between 0 to 18 cm), which is called the fundamental sine wave. Similarly a fundamental cosine wave can be fitted for the same region. (Fig. 3.10)
Figure 3.9: Angular Changes Vs Cumulative Distance for F shaped object.

The fundamental sine component of the plot (of angular changes) is defined as and shown in Figure No. 3.10 (a)

\[
a_i = \frac{1}{\pi} \sum_{i=1}^{n} y_i \cdot \sin \left(2\pi \frac{x_i}{D}\right)
\]  

The fundamental cosine component of the plot is defined as and shown in Figure No. 3.10(b)
Where $y_k$ is the angular change, $x_k$ is the cumulative distance at a point $k$ on the clockwise traverse of the boundary of the object of known shape and $D$ is the value of $x_k$ at $k = N$ (i.e., $D$ = the total length of the boundary), in a similar manner, we can fit second harmonic sine and cosine waves (Fig. 5.2) and determine the corresponding harmonic components of the pattern $y_k$ as

\[ a_2 = -\frac{1}{2\pi} \sum_{k=1}^{N} y_k \sin\left(\frac{2\pi x_k}{D}\right) \quad \text{.................. (3.30)} \]

\[ b_2 = +\frac{1}{2\pi} \sum_{k=1}^{N} y_k \cos\left(\frac{2\pi x_k}{D}\right) \quad \text{.................. (3.31)} \]

In general the $n$th harmonic component is given by

\[ a_n = -\frac{1}{n\pi} \sum_{k=1}^{N} y_k \sin\left(n\pi x_k\right) \quad \text{.................. (3.32)} \]

\[ b_n = +\frac{1}{n\pi} \sum_{k=1}^{N} y_k \cos\left(n\pi x_k\right) \quad \text{.................. (3.33)} \]

Figure No. 3.10 Fundamental components
The phase angle $\alpha_n$ is defined as

$$\alpha_n = \text{ATAN2}(b_n, a_n)$$

(3.34)

$\text{ATAN2}(b_n, a_n)$ is defined as follows

Let $\beta_n = \tan^{-1}\frac{|a_n|}{|b_n|}$ in degrees

Then if

(i) $a_n > 0$, $b_n > 0$ and $a_n = 90^0 - \beta_n$

(ii) $a_n < 0$, $b_n > 0$ and $a_n = 90^0 + \beta_n$

(iii) $a_n < 0$, $b_n < 0$ and $a_n = 270^0 - \beta_n$

(iv) $a_n > 0$, $b_n < 0$ and $a_n = 270^0 + \beta_n$

Figure 3.11: Phase angle and phasor diagram.

For the shape shown in Figure 3.12 Fourier descriptors have been
calculated and have been given below.

Figure 3.12 (a): Shape for which Fourier Descriptors have been calculated.

We have trace the boundary starting from an arbitrary point (A). Table 3.1 shows the cumulative distance vs. change in the angle.

<table>
<thead>
<tr>
<th>Cumulative Dist (cm)</th>
<th>Change In Angle (rad.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>$-\pi / 2$</td>
</tr>
<tr>
<td>4</td>
<td>$-\pi / 2$</td>
</tr>
<tr>
<td>8</td>
<td>$-\pi / 2$</td>
</tr>
<tr>
<td>10</td>
<td>$-\pi / 2$</td>
</tr>
<tr>
<td>12</td>
<td>$+\pi / 2$</td>
</tr>
</tbody>
</table>

$\alpha = \frac{1}{n}[-(\pi / 2)\sin(2\pi * 3/12) - (\pi / 2)\sin(2\pi * 4/12) - (\pi / 2)\sin(2\pi * 8/12) - (\pi / 2)\sin(2\pi * 10/12) - (\pi / 2)\sin(2\pi * 11/12) + (\pi / 2)\sin(2\pi * 12/12)]$

$= -0.183$

Figure 3.12 (b): Plot of Angular Changes VS Cumulative Distance for Shape shown in Figure 3.12(a).
$$\begin{align*}
b_1 &= \frac{1}{\pi} \left[ (-\pi/2) \cos(2\pi \times 3/12) - (\pi/2) \cos(2\pi \times 4/12) - (\pi/2) \cos(2\pi \times 8/12) - (\pi/2) \cos(2\pi \times 10/12) - (\pi/2) \cos(2\pi \times 11/12) + (\pi/2) \cos(2\pi \times 12/12) \right] \\
&= 0.317
\end{align*}$$

$$A_1 = \sqrt{a_1^2 + b_1^2} = 0.366$$
$$\alpha_1 = \text{ATAN2}(0.317 - 0.183) = 120^0$$

Similarly
$$\begin{align*}
a_2 &= \frac{-1}{\pi} \left[ (-\pi/2) \sin(2\pi \times 6/12) - (\pi/2) \sin(2\pi \times 8/12) - (\pi/2) \sin(2\pi \times 16/12) - (\pi/2) \sin(2\pi \times 20/12) - (\pi/2) \sin(2\pi \times 22/12) + (\pi/2) \sin(2\pi \times 24/12) \right] \\
&= -0.433
\end{align*}$$

$$b_2 = \frac{1}{\pi} \left[ (-\pi/2) \cos(2\pi \times 6/12) - (\pi/2) \cos(2\pi \times 8/12) - (\pi/2) \cos(2\pi \times 16/12) - (\pi/2) \cos(2\pi \times 20/12) - (\pi/2) \cos(2\pi \times 22/12) + (\pi/2) \cos(2\pi \times 24/12) \right]$$

$$= 0.75$$

$$A_2 = \sqrt{a_2^2 + b_2^2} = 0.866$$
$$\alpha_2 = \text{ATAN2}(0.75 - 0.433) = 120^0$$

$$\begin{align*}
a_3 &= \frac{-1}{3\pi} \left[ (-\pi/2) [\sin 270^0 + \sin 360^0 + \sin 720^0 + \sin 900^0 + \sin 990^0 - \sin 1080^0] \right] \\
&= -0.333
\end{align*}$$

$$b_3 = \frac{+1}{3\pi} \left[ (-\pi/2) [\cos 270^0 + \cos 360^0 + \cos 720^0 + \cos 900^0 + \cos 990^0 - \cos 1080^0] \right]$$

$$= 0$$

$$A_3 = \sqrt{(a_3^2 + b_3^2)} = 0.333$$
$$\alpha_3 = \text{ATAN2}(0.0 - 0.333) = 180^0$$

Similarly
$$\begin{align*}
a_4 &= 0 \\
b_4 &= 0.25 \\
A_4 &= 0.25 \\
\alpha_4 &= 90^0
\end{align*}$$

$$\begin{align*}
a_5 &= 0.1333 \\
b_5 &= 0.2366 \\
A_5 &= 0.272 \\
\alpha_4 &= 60^0
\end{align*}$$

3.4.4 Feature extraction using zoning methods.

Bosker describes the commercial OCR systems named as calera that uses zoning on binary characters. The system was designed to recognize machine printed characters of almost any non-decorative
font, possibly severely degraded by photocopying. The zonning method was used to compute the percentage of black pixels in each zone. Additional features were needed to improve performance.

Algorithm for Feature Extraction using Zonning.

<INPUT> Solid Binary Image
<OUTPUT> 8x8 Feature Vector

1. It is assumed that we have 26 different character images.

2. First let us take 20 different images of first character representing them as $A_1, A_2, \ldots, A_{20}$

3. Take first image $A_1$ and rescaled into 64x64 size. Calculate number of pixels in each local zone and we can obtain the 64 values from the image representing them as input vector $[X] = [x_1[N], x_2[N], \ldots, x_{64}[N]]$. Calculate these values for the remaining 19 images represented as $A_2, \ldots, A_{20}$

4. Find mean vector for 20 character images of a letter 'A'.
\[ m(i) = \frac{1}{20} \left( \sum_{i=1}^{64} x_i[N] \right) \text{ where } (i=1,2,3,\ldots,64) \text{ and } N=1\ldots(3.35) \]

5. Finally we can obtain 64 features values for each character. Repeat the same process for next characters.

3.4.5 Feature extraction using some fuzzy image processing methods.

Human visual system has been perfectly adapted to handle uncertain information in both data and knowledge. It would be hard to define quantitatively how an object, such as car, has to look in terms of geometrical primitives with exact shapes, dimensions and colors. Instead we are using a descriptive language to define features that eventually are subject to a wide range of variations. The interrelation of a few such “Fuzzy” properties sufficiently characterize the object of interest. Fuzzy image processing is an attempt to translate this ability
of human reasoning into computer vision problem as it provides an intuitive tool for inference from imperfect data.

Fuzzy logic versus Probability theory

It has been a long-standing misconception that fuzzy logic is nothing but another representation of probability theory. We do not want to contribute to this dispute, but rather try to outline the basic difference.

Probability describes the uncertainty in the occurrence of an event. It allows predicting the event by knowledge about its relative frequency within a large number of experiments. After the experiment has been carried out, the event either has occurred or not. There is no uncertainty left. Even if the probability is very small, it might happen that the unlikely event occurs. To treat stochastic uncertainty, such as random processes (e.g., noise), probability theory is a powerful tool in computer vision. There are, however, other uncertainties that cannot be described by random processes.

As opposed to probability, fuzzy logic represents the imperfection in the informational content of the event. Even after the measurement, it might not be clear whether the event has happened or not. For illustration of this difference, consider an image to contain a single edge, which appears at a certain rate. Given the probability distribution, we can predict the likelihood of the edge to appear after a certain number of frames. It might happen, however, that it appears in every image or does not show up at all. Additionally, the edge may be corrupted by noise. A noisy edge can appropriately be detected with probabilistic approaches, computing the likelihood of the noisy measurement to belong to the class of edges. But how do we define the edge? How do we classify an image that shows a gray-value slope? A noisy slope stays a slope even if all noise is removed. If the slope is
extended over the entire image we usually do not call it an edge. But if the slope is "high" enough and only extends over a "narrow" region, we tend to call it an edge. Immediately the question arises: How large is "high" and what do we mean by "narrow?"

In order to quantify the shape of an edge, we need to have a model. Then, the probabilistic approach allows us to extract the model parameters, which represent edges in various shapes. But how can we treat this his problem, without having an appropriate model? Many real world applications are too complex to model all facets necessary to describe them quantitatively. Fuzzy logic does not need models. It can handle vague information and imperfect knowledge and combine them by heuristic rules —in a well-defined mathematical framework. This is the strength of fuzzy logic!

Fuzzy image understanding

To use fuzzy logic in image processing applications, we have to develop a new image understanding. A new image definition should be established, images and their components (pixels, histograms, segments, etc.) should be fuzzified (transformed in membership plane), and the fundamental topological relationships between image parts should be extended to fuzzy sets (fuzzy digital topology).

Defining image in terms of fuzzy sets

An image $G$ of size $M \times N$ with $L$ gray levels can be defined as an array of fuzzy singletons (fuzzy sets with only one supporting point) indicating the membership value $\mu_{mn}$ of each image point $x_{mn}$ regarding a predefined image property (e.g., brightness, homogeneity, noisiness, edginess, etc.):

$$ G = \bigcup_{m=1}^{M} \bigcup_{n=1}^{N} \mu_{mn} \ x_{mn} $$

The definition of the membership values depends on the specific
requirements of a particular application and on the corresponding expert knowledge. Figure 3.13 shows an example where brightness and edginess are used to define the membership degree of each pixel.

Figure 3.13: Images as an array of fuzzy singletone.
   a. Test image as a fuzzy set regarding
   b. Brightness (Having higher membership)
   c. Edging (having higher membership]

Image fuzzification: From images to memberships

Fuzzy image processing is a kind of nonlinear image processing. The difference to other well-known methodologies is that fuzzy techniques operate on membership values. The image fuzzification (generation of suitable membership values) is, therefore, the first processing step. Generally, three various types of image fuzzification can be distinguished.

1. Histogram-based gray-level fuzzification
2. Local neighborhood fuzzification
3. Feature fuzzification.

As in other application areas of fuzzy set theory, the fuzzification step sometimes should be optimized. The number, form, and location of each membership function could/should be adapted to achieve better results. For instance, genetic algorithms are performed to optimize fuzzy rule-based systems.

Histogram-based gray-level fuzzification.
To develop any point operation (global histogram-based techniques), each gray level should be assigned one or more membership values (such as dark, gray, and bright) with respect to the corresponding requirements.

Local neighborhood fuzzification.
Intermediate techniques (e.g., segmentation, noise filtering etc.) operate on a predefined neighborhood of pixels. To use fuzzy approaches to such operations, the fuzzification step should also be done within the selected neighborhood (Figure 3.14). The local

Figure 3.14 : On local neighborhood fuzzification

be done. Of course, local neighborhood fuzzification requires more computing time compared with the histogram-based approach. In
many situations, we also need more thoroughness in designing membership functions to execute the local fuzzification because noise and outliers may falsify membership values.

For Example, within 3 X 3-neighborhood $U$, we are interested in the degree of membership of the center point to the fuzzy set edge pixel. Here, the edginess $\mu_e$ is a matter of grade. If the 9 pixels in $U$ are assigned the numbers 0, ..., 8 and $G_0$ denotes the center pixel, a possible membership function which can be used is:

$$\mu_e = 1 - \left[ 1 + \frac{1}{\Delta} \sum_{i=0}^{8} ||G_0 - G_i|| \right]^{-1}$$

where $\Delta = \max_u (G_i)$  .................. [3.37]

Feature fuzzification.

For high-level tasks, image features usually should be extracted (e.g., length of objects, homogeneity of regions entropy, mean value, etc.). These features will be used to analyze the results, recognize the objects, and interpret the scenes. Applying fuzzy techniques to these tasks, we need to fuzzify the extracted features. It is necessary not only because fuzzy techniques operate only on membership values but also because the extracted features are often incomplete and/or imprecise.

For example, the length of an object is calculated as fuzzy subsets of very short, short, middle-long, long and very long and can be introduced in terms of the linguistic variable length in order to identify certain types of objects (Figure 3.15).
3.4.6 Feature extraction using contour and Projection Histogram methods.

Projection histogram was introduced in 1956 in a hardware OCR system by glauberman. This is mostly being used for segmenting characters, words and text line or to detect if an input image of a scanned page is rotated. For a horizontal projection \( y(x_i) \) is the number of pixels with \( x = x_i \). The features can be made scale independent by using a fixed number of bins on each axis and dividing by the total number of print pixels in the character image. However the projection histogram are very sensitive to rotation and variability in writing styles. The vertical projection \( x(y_i) \) is slant invariant but the horizontal projection is not. We use the formula

\[
d = \sum_{i=1}^{n} |y_1(x_i) - y_2(x_i)|
\]

...(3.38)

to measure dissimilarity between two histograms. Where \( n \) is the number of bins and \( y_1 \) and \( y_2 \) are the two histograms to be compared. It is more meaningful to compare the cumulative histogram \( y(x) \), the sum of first \( k \) bins using the above equation.
**Algorithm**

<INPUT> Binary image  
<OUTPUT> feature of image

1. Select input image of any size

2. Draw the horizontal and vertical histogram for the given image

3. Find height and width of the character from the horizontal and vertical projections.

4. Calculate the maximum value in horizontal and vertical projections.

5. Calculate the total number of black pixels, center pixels and base line of the given image.

6. Store the six global features in a data file and repeat the process for the next sample of same character.

\[
\text{Center Pixel} = \frac{\sum i \cdot p[i]}{\sum p[i]} \quad \text{where } p[i] \text{ is the horizontal projection.} \quad (3.39)
\]

and \( P_m \) the maximum horizontal projection is \( P_m = \max \{ P_h[i] \} \)

The area of black pixel \( A_v \) reflects the proportion of strokes in a character and is computed as the summation of vertical and horizontal projections. The base line measures as the difference in vertical center of gravities between the left and right halves of the image.

**Feature extraction using contour histogram.**

The contour following algorithm traces boundaries by ordering successive edge points. Contour tracing algorithm used only contour points for extracting features. It causes too many folds reduction of data on which each algorithm should operate. The edge detected binary images contour can be found by two methods namely coarse contour and fine contour.
Coarse contour: It is based on the four connectivity of the edge-detected pixel. It results some of the boundary pixels appearing twice.

Fine contour: It is refinement to the coarse contour, which is based on eight connectivity of the edge-detected pixels. Here fine line of one pixel width is detected. In the present method contour of the character image is traced using this method and is implemented as, starting at any chosen point, the entire contour is traced using eight connectivity and direction information of the previous contour pixel.

Algorithm.

<INPUT> Binary image
<OUTPUT> Contour of image

Search=1

Set current pixel to p

Repeat

Repeat at most eight times.

If there is a black pixel in the search direction then

set current pixel to the pixel in the search direction

if search is 0 or 1 set search to 7
if search is 2 or 3 set search to 1
if search is 4 or 5 set search to 3
if search is 6 or 7 set search to 5

Else

Increment search;

End repeat

Append the current pixel to the contour

Until contour is not completed.

The trace contour points are used for finding the features like maxima, minima, and width of the image and base line of the character image.

Using these two methods about 10 to 12 feature values are calculated and the same are being use as inputs to the neural network. Output is

109
20 nodes corresponding to 20 different characters used for recognition. The total database 200 images are used for training the neural network. If the input to the neural network is features of letter A or Letter Marathi Script letter then only the first neuron will be high and the other neurons will be zero. The following matrix gives the general format.

\[
\text{[Capital English alphabets]} = \begin{bmatrix}
1,0,0,0,0,
0,1,0,0,0,
0,0,1,0,0,
0,0,0,0,0,
\end{bmatrix}
\]

\[
\text{[Devnagri Marathi Script Letters]} = \begin{bmatrix}
1,0,0,0,0,
0,1,0,0,0,
0,0,1,0,0,
0,0,0,0,0,
\end{bmatrix}
\]

3.4.7 Feature extraction using distance measurement techniques.

It is assumed that we have 20 different character images with 20 images of each character. Each image is normalized into 64x64 grid size and it is placed into 64x64 size rectangle. Now, the character is divided into four quadrants. From the center of each quadrant, we are finding distances in eight directions from the center to boundary of a character as shown in figure 3.16

Fig.:3.16(a) 64 X 64 Original image (b) One quadrant of Image 8 X 8

0 represents actual value of a particular pixel and 1 represents the
actual value of the pixel at center of the quadrant. From the four quadrants, we can obtain 32 distances that are used as features. In addition with, we are calculated four distances from each side of rectangle. Finally we can obtain 64 features from the binary character.

Algorithm for Feature Extraction

<INPUT> Solid Binary Image.
<OUTPUT> 64 Features From Image

1. Find boundary of given image using derivative filter.
2. Eliminate any gaps between in the boundary of image.
3. Divide the character into four quadrants.
4. Find center of quadrant and calculate distance between center to boundary character in 8-directions. This will provide 32 different values.
5. Insert this character into a rectangle of 64X64 and find the following distances to each side of a rectangle from the center of the rectangle. The different distances, which are used for this, are.

   a. Euclidean Distance: \( D_e(P, R) = \sum_{n} (x_n - s_n)^2 + (y_n - t_n)^2 \) \[3.40\]
   b. D-4 distance: \( D_4(P, R) = \sum_{n} (x_n - s_n) + (y_n - t_n) \) \[3.41\]
   c. Hamming Distance: \( D_h(P, R) = \sum e_n \) \[3.42\]
      where \( e_n = 1 \) if \( p(x_n, y_n) = r(s_n, t_n) \)
      and \( e_n = 0 \) if \( p(x_n, y_n) = r(s_n, t_n) \)
   d. Jaccard Distance \( d_j = \frac{n_{11}}{n_{11} + n_{10} + n_{01}} \) \[3.43\]
   e. Yule Distance \( d_y = \frac{n_{11}n_{00} - n_{10}n_{01}}{n_{11}n_{00} + n_{10}n_{01}} \) \[3.44\]

6. Store the 64 feature in a data file and repeat the process for next character.

3.4.8 Feature extraction using gradient methods.

As per the task performed by human eyes the variety of objects are
being detected under various environments. For example if an object appears at a particular pose like upside down, it will be correctly identified by the eyes whereas the machine will not be able to identify the same as it will have a different object model for comparison in the memory. Therefore we normally check the whole geometrical configuration instead of the object's details.

Procedure for feature extraction

1. Preprocessing: It is applied to remove small light details and to enhance the contrast.
2. Thresholding for converting image to binary image.
3. Labeling
4. Grouping algorithm for detecting features block by block

This procedure is shown diagrammatically in Figure 3.17

![Figure 3.17: Illustration of main steps of proposed recognition system.](image)

Preprocessing

Sharp peaks and bright noise such as reflections are eliminated from the image using a opening operations. Boost filtering is used to enhance the contrast using the mask as shown in figure 3.18

```
1/9 X
-1 -1 -1
-1 w -1
-1 -1 -1
```

Figure 3.18: A high boost mask
To prevent the side effect of over enhancing background noise a gradient operator is used to calculate the centre weight of the high boost mask. The centre weight $w$ at location $(x, y)$ can be calculated by the following function:

$$w = \left[ z(1-s) + (1-z)(1-g) \right] \times f - 8 + 8 \quad \ldots \quad [3.46]$$

where $s$ = normalised gradient value of the sobel operator at $(x, y)$

$g = \text{normalised gray level at } (x, y)$

$z = \text{weight to be adjusted to prevent over enhancing.}$

$f = \text{boosting factor}$

\[ s = \frac{\text{sobel}(x, y)}{255} \quad \text{and} \quad g = \frac{\text{Gray}(x, y)}{255} \quad \text{and} \quad z = \begin{cases} 0.4 & \text{if } s > 0.5 \\ 0.8 & \text{if } s \leq 0.5 \end{cases} \quad \ldots \quad [3.45] \]