Shape Recognition by Curvature Changing Points and Fourier Transform

Jianzhong Li
Western Michigan University
SHAPE RECOGNITION BY CURVATURE CHANGING POINTS
AND FOURIER TRANSFORM

by

Jianzhong Li

A Thesis
Submitted to the
Faculty of The Graduate College
in partial fulfillment of the
requirements for the
Degree of Master of Science
Department of Computer Science

Western Michigan University
Kalamazoo, Michigan
December 1987
SHAPE RECOGNITION BY CURVATURE CHANGING POINTS AND FOURIER TRANSFORM

Jianzhong Li, M.S.
Western Michigan University, 1987

Computer vision is a relatively new and fast-growing field. Shape representation and recognition is a very important but difficult branch in this field.

One method of using curvature changing points and Fourier transform to recognize the shape is described in this thesis. Also an important feature in computer vision, multiscaling, is achieved by this method.
ACKNOWLEDGMENTS

I would like to express my profuse gratitude to my advisor, Dr. Kountanis. His patience, guidance, and encouragement will never be forgotten. My appreciation is also extended to Dr. Natour and Dr. Pinkoski for their time and consideration.

I would also like to mention my thanks to my friends and my brother for their indispensable help and support: James Lee, Tom Sprague and Wang Tao. And finally, my special thanks go to my parents. Without their support and encouragement I would have never survived the ordeal.

Jianzhong Li
INFORMATION TO USERS

This reproduction was made from a copy of a document sent to us for microfilming. While the most advanced technology has been used to photograph and reproduce this document, the quality of the reproduction is heavily dependent upon the quality of the material submitted.

The following explanation of techniques is provided to help clarify markings or notations which may appear on this reproduction.

1. The sign or "target" for pages apparently lacking from the document photographed is "Missing Page(s)". If it was possible to obtain the missing page(s) or section, they are spliced into the film along with adjacent pages. This may have necessitated cutting through an image and duplicating adjacent pages to assure complete continuity.

2. When an image on the film is obliterated with a round black mark, it is an indication of either blurred copy because of movement during exposure, duplicate copy, or copyrighted materials that should not have been filmed. For blurred pages, a good image of the page can be found in the adjacent frame. If copyrighted materials were deleted, a target note will appear listing the pages in the adjacent frame.

3. When a map, drawing or chart, etc., is part of the material being photographed, a definite method of "sectioning" the material has been followed. It is customary to begin filming at the upper left hand corner of a large sheet and to continue from left to right in equal sections with small overlaps. If necessary, sectioning is continued again—beginning below the first row and continuing on until complete.

4. For illustrations that cannot be satisfactorily reproduced by xerographic means, photographic prints can be purchased at additional cost and inserted into your xerographic copy. These prints are available upon request from the Dissertations Customer Services Department.

5. Some pages in any document may have indistinct print. In all cases the best available copy has been filmed.

University Microfilms International
300 N. Zeib Road
Ann Arbor, MI 48106

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Shape recognition by curvature changing points and Fourier transform

Li, Jianzhong, M.S.
Western Michigan University, 1987
PLEASE NOTE:

In all cases this material has been filmed in the best possible way from the available copy. Problems encountered with this document have been identified here with a check mark ✓.

1. Glossy photographs or pages ____
2. Colored illustrations, paper or print _____
3. Photographs with dark background ____
4. Illustrations are poor copy _____
5. Pages with black marks, not original copy ✓
6. Print shows through as there is text on both sides of page____
7. Indistinct, broken or small print on several pages ✓
8. Print exceeds margin requirements _____
9. Tightly bound copy with print lost in spine ______
10. Computer printout pages with indistinct print ______
11. Page(s) _______ lacking when material received, and not available from school or author.
12. Page(s) _______ seem to be missing in numbering only as text follows.
13. Two pages numbered 53 __. Text follows.
14. Curling and wrinkled pages ______
15. Dissertation contains pages with print at a slant, filmed as received _____
16. Other ____________________________________________

_____________________________________________________

_____________________________________________________

U-M-I
Table of Contents—Continued

CHAPTER

V. EXAMPLES ................................... 44

VI. CONCLUSIONS AND RECOMMENDATIONS ................. 50

APPENDICES ............................................ 52

A. PROGRAM I: Image Creation ......................... 53

B. PROGRAM II: Image Representation and Recognition 62

BIBLIOGRAPHY ............................................. 84
LIST OF FIGURES

1. The Shape Containing Two Different Kinds of Curvature Changing Points ......................... 17
2. The Corner Point and Its Input Function .......... 18
3. The Smooth Point and Its Input Function ............ 20
4. The Corner Point and Its Fourier Transform ...... 25
5. The Smooth Point and Its Fourier Transform ...... 30
6. Block Diagram of a Transform Function .............. 33
7. Two Trees Having the Same Shape ................... 45
8. Two Different Triangles With Reconstructed Image . 47
9. The Reconstructed Images Under Different Scale ... 48
CHAPTER I

INTRODUCTION

One of the most important things to be determined from an image is the shape of the objects in it. Shape is an intrinsic property of the vision from which many other factors can be derived.

Shape recognition is doubtless one of the most important abilities of the mammalian vision system. Usually the shapes of the different objects are recognized through the outlines of their surfaces. Therefore, different outlines of the surfaces can represent different shapes of objects.

For example, let us draw the shape of human eyes and a mouth with a black pen on a piece of white paper. It is clear that these two objects, eyes and mouth, have different shapes. Their differences can be easily determined by their different lengths, different curvatures and orientation along the contour. Therefore, intuitively, we could use these parameters to describe the shape of the objects and, indeed, this is the main topic of this paper: to recognize and to reconstruct the image of different shapes.
objects by their curvature and orientation changes.

Chapter II deals with reviewing the relevant literature. Generally, there are four mathematical models used to recognize the shape of objects. These models are template matching model, feature model, structure description model and Fourier model. The pros and cons of each model have been discussed briefly. It is concluded from these discussions that using both curvature changing points and Fourier transform technique is an ideal method to recognize the shape of objects, which can take full advantage of both feature model and Fourier model.

Theoretically, the relationship between the curvature changing point and its Fourier transform has been explained in detail in Chapter III. The results show that not only can the shape of the objects be reconstructed and recognized by the Fourier transform but also another important feature—multiscaling—can be achieved during this recognition process. This feature is consistent with the human recognition process and is a very important feature in the computer vision.

Chapter IV describes the concrete process to achieve the principle in Chapter III. The Z-transform and FFT algorithm have been used to obtain the feature points of the image. Also, the time complexity needed for the above method has been discussed.
Finally, in Chapter V, several examples are presented to show the results and the advantage and disadvantage of the proposed method.
People have dreamed of intelligent machines for a very long time. Its origin may be traced back to antiquity. However, the first substantial contribution in the context of digital computers was made by Turing in 1950 (Ballard and Brown, 1982). Since then, this field, called artificial intelligence, has been developing rapidly. The main goal for artificial intelligence was to endow computers with information-processing capabilities comparable to those of human organisms. From the outset, one of the important goals of artificial intelligence is to equip machines with the capability of dealing with sensory inputs. Computer vision is just such a field to achieve this goal.

Computer Vision and Shape Representation

Computer vision is a relatively new and fast growing field. The first experiment was conducted in the late 1950s, and many of the essential concepts, methods and theories have been developed during the last ten years.
Now computer vision has a variety of applications in artificial intelligence, computer graphics and image processing and, therefore, has become a very important field in computer science.

Computer vision is the construction of explicit, retrievable and meaningful descriptions of physical objects from images. This description is quite different from image processing, which studies image-to-image transformation and has no explicit description building. Description is a prerequisite for recognizing, manipulating and thinking about objects.

When an object has been observed, the incoming visual data do not exhibit corresponding coherence or invariance; they contain much irrelevant or even misleading variation. Therefore, the major challenge for computer vision is to decide what intrinsic information can be extracted from the image. Among this information, one of the most important things to be determined from an image is the SHAPE of the objects in it.

There are several factors which hamper the effort to represent the shape of an object. First, shapes are often complex. Whereas motion or intensity is relatively simply quantified by a few well understood parameters, shape is much more subtle. Common natural shapes are too complex to be described by parameters. Worse, it is not
clear what aspects of shapes are important for applications such as recognition. Further, an explicit and complete representation may be computationally intractable for such basic uses as matching. Secondly, there is a lack of means to describe the shape. Why is shape recognition so easy for human beings and description so hard? The fact that we have no precise language for shape may argue for the inaccessibility of our shape-processing algorithms or data structures. This lack of cognitive leverage is a trifle daunting, especially when taken with the complexity of everyday shapes. Last, the discipline is young. Mathematics traditionally has not concerned itself with shape. Until very recently, people communicated about complex shapes mainly through words, gestures, and two-dimensional drawings. It was not until the advent of the digital computer that it became of interest to represent complex shapes so that they could be specified to machine, manipulated, computed with, and represented as output graphics. Also, no generally accepted single representation scheme is available for all shapes. Each scheme has its own advantages and disadvantages. Algorithms for manipulating shapes are surprisingly complex, and are research topics. Often the representations good for one application, such as recognition, are not good for other computations.
Traditional Theories of Shape Recognition

Traditionally, there are three mathematical models for the shape recognition—templates, feature and structural descriptions (Lindsay and Norman 1977). In the 1980s, Fourier analysis was introduced to the vision cognition. This formed the fourth mathematical model (Cornsweet 1980). In this section, a brief review will be given to each model and also the pros and cons of each model will be discussed.

Template Matching

This is the simplest and the most straightforward class of model for shape recognition. In this model, there is a template in the memory. The input array would be simultaneously superimposed with all the templates in the memory. The one with the closest above-threshold match (e.g., the largest ratio of matching to nonmatching points in corresponding locations in the input array) would indicate the shape that is present. One of the standard measurement methods for the degree of matching is by using correlation coefficients. The higher the value of correlation coefficient, the more likely the object would be matched.
Theoretically, this method is the most simple one. But in order to match all the possible changes for a particular object, such as a change in distance, location, and orientation, it is necessary to have multiple templates corresponding to each of its possible transformation, rotation and combination in the memory. Or, the input image could be rotated, displaced and scaled to a canonical set of values before matching against the templates. In this case, however, it needs either a very large memory to store all these templates or to decide in advance which transformation should be applied. Unfortunately, neither is the memory large enough for the current computer nor anyone knows in advance which transformation should be used. Therefore, only in a very limited simple situation can this method be actually used.

Feature Model

This class of models is intuitively based on the early "Pandemonium" model of shape recognition (Selfridge and Neisser, 1960). In these models, there are no templates for the entire shapes; rather, there are "mini-templates" or "feature detectors" for simple geometric features such as lines, curves, curvature, angles, T-
junction, etc. There are detectors for every feature at every location in the input array. These detectors send out a graded signal which corresponds to the degree of match between the target feature and the part of the input array they are "looking at." The number of each occurrence of the feature is counted and recorded with its corresponding graded signal. In this feature model, the shape is represented by the feature and if the feature in the input array can be matched with that in the memory, they could be considered the same.

The advantage of this model is that the characterization of the feature is, in general, invariant to the location, translation, rotation, and even scale changes, since the feature belongs to the shape itself but not the location.

The disadvantage is that for a natural shape, sometimes it is very difficult to extract the feature correctly and efficiently. Consider how one would define the shape of a horse. Naturally, one could define a horse by choosing some special parts as features, such as "mane," "horse head," and so on, but then detecting these features would be no less difficult than detecting the horse itself. To improve this, sometime this model is combined with the third model, as we will discuss later, to get a more efficient algorithm.
**Structural Description Model**

A structure description is a data structure that can be thought of as a list of propositions whose arguments correspond to parts and whose predicates correspond to properties of the parts and to spatial relationships among them. Often these propositions are depicted as a graph whose edges link the nodes which correspond to the spatial relations. The explicit representation of spatial relations is one aspect of these models that distinguishes them from feature models and allows them to escape from some of the problems we pointed out before.

One of the chief advantages of this structure description is that it can factor apart the information in a scene without losing information in it. For a good recognition system, it is not sufficient simply to supply a list of labels for the objects that are recognized, for we need to know not only what things are but also how they are oriented and where they are with respect to us and each other. For example, when we are reaching for an object, we need to know how far we are from the object. Also, a good recognition process must have the ability to factor apart unnecessary information so that it can access the information relevant to a particular task without becoming overloaded, distracted, or misled by the irrelevant information.
Another advantage is that the structural description model can break up the recognized objects into smaller subparts, which are well-suited to simulate human being's visual system (Hoffman and Richards 1984).

Fourier Model

A new class of shape recognition methods, the Fourier model, was first proposed by Persoon and Fu (1974), and then improved by Ballard and Brown (1982). Many researchers adopted this model implicitly as the most likely candidate for shape recognition, since it is consistent with the human recognition process. In this model, the two-dimensional input array is subjected to a spatial trigonometric Fourier analysis. And in such an analysis, the array is decomposed into a set of components. For example, a Fourier transform of the intensity array would consist of two numbers for each of these components. The first number would specify the degree of contrast in the image corresponding to the frequency at that orientation (such as, the degree of difference in brightness between the bright areas and the dark areas of that image for that frequency in that orientation). The full set of these numbers is the amplitude spectrum corresponding to the image. The second number would specify where in the image the peaks and troughs of the
intensity change defined by that component lies. The full set of these numbers is the phase spectrum corresponding to the image. The amplitude spectrum and the phase spectrum together define the Fourier transform of the image, and the transform contains all the information in the original image.

What is the advantage of the Fourier model? When there is an exact match between the image and one of the stored templates, there are neither advantages nor disadvantages to doing the match in the transform domain because no information is lost in the transformation. But, when there is not an exact match, the Fourier model does have an advantage. For example, to a first order approximation the amplitude spectrum corresponding to a shape is the same regardless of where the object is located. Therefore, a shape could be recognized across all its possible translations, such as rotation, transfer and even scaling. Another important advantage of the Fourier transform is that it segregates information about sharp edges and small details from information about gross overall shape. The latter is specified primarily by the lower frequency components of the transform, while the former, by the higher frequency components. Thus if the algorithm for recognition could selectively ignore the higher frequency of the amplitude spectrum when comparing input and template transforms, a shape could be
recognized even if its boundaries are blurred. This feature of recognition is consistent with the human recognition process and therefore is a significant feature for most recognition processes.

The method used in this paper to recognize the shape of objects is based on a combination of the Fourier model and feature model. More specifically, the curvature changing points are used as shape features of objects and the Gaussian function and its Fourier transform are employed to achieve the benefits of the Fourier model. These will be discussed in detail in the following chapters.
CHAPTER III

CURVATURE CHANGING POINT AND ITS TRANSFORMATION

As mentioned in Chapter II, using some spatial feature points to represent and recognize the shape of objects is a classical idea. Many methods have been put forward to describe the shape of objects in the past 20 years. Chain Code (Freeman, 1974) is one of the earliest way to represent the shape of objects. Quad tree (Samet, 1980) is another effective method used to describe the spatial occupation, through which the shape of the object can be determined directly. In recent years, much emphasis has been put on the Fourier model by many researchers, since this model reveals precious information about the nature of the human recognition process. One of the key technique which has been used in the Fourier model is called multiscaling, which is based on the following idea. In the representation function of the shape of the object, there is parameter, called scale, which reflects the coarseness of the reconstructed object. Scale usually denoted by $\sigma$. The larger the $\sigma$, the more the number of segments (or curves) needed to fit the outline of the object. Multiscaling investigates the variations of the
number and location of the feature points which are required to identify and represent the object. Usually, the larger the $\sigma$, the more precise and sharper the shape of the object can be represented and recognized.

Curvature changing point is one of the feature points which not only preserves the necessary information about the shape of the original object but also fulfills the requirement of multiscaling technique. Therefore, this kind of feature point has received much emphasis in the area of computer vision and was recommended as one of the most important feature points (Pavlidis, 1986). In this Chapter, a detailed description of curvature changing points and their properties will be given and also their relationship with multiscaling will be discussed.

The basic idea of using curvature changing points to depict the shape of objects is similar to the idea of using chain code to describe the outline of the objects. In Chain Code theory, the outline of objects is approximated by a set of unit straight line segments (Ballard and Brown, 1982). These line segments must lie on a fixed grid with a fixed set of possible orientations. Since the grid is uniform, the direction is sufficient to characterize each segment. By successively being given a direction number to each segment, the shape of the object can be well represented by these numbers. Curvature is another kind of property to describe
the shape of objects just like the direction number of Chain Code. Curvature at a point on the curve is the reciprocal of the radius of the curve at that point. So curvature will be changed along the contour as the shape of the object is changed. The smaller the curvature, the more smooth the curve. The point at which the curvature has been changed is called curvature changing point. By successively recording each curvature changing point along the contour of the object, a set of curvature changing points will be obtained. Clearly, different shapes have different sets of curvature changing points. Therefore, a particular set of curvature changing points can be used to represent the shape of the object. It is noted from its nature that curvature is an intrinsic property of that shape; it does not depend on the orientation and position of that object (Brady and Ponce, 1982). In other words, no matter how an object appears, whether in vertical or horizontal position, on the left corner or right corner, the set of curvature changing points will not change. This set of curvature changing points depend only on the geometric property of the shape itself. In this sense, the curvature changing point is a very nice feature point used to describe the shape of objects.

Generally, there are two different kinds of curvature changing points (Asada and Brady, 1986). The first kind of curvature changing points is the kind of points at
which not only has the curvature been changed but also the orientation has been changed discontinuously. This kind of point is called a corner point. Corner points are shown by point A and point B in Figure 1.

![Figure 1. The Shape of a Mouth Which Contains Two Different Kinds of Curvature Changing Points are Illustrated.](image)

The second kind of curvature changing points is the type of point at which the curvature changes are discontinuous but the orientation is not changed. This kind of point is called a smooth joint point, and is shown by point C and point D in Figure 1. There are no sudden changes at this kind of curvature changing points and, therefore, these points look much more smooth than the corner points.

Input Function of Curvature Changing Point

As mentioned before, the set of curvature changing points can be used to describe the shape of objects, therefore, input functions are needed to represent these
curvature changing points. These input functions should preserve all the information about the shape of objects and should be easy for further manipulation. The input function in orientation space is just such a candidate which well represents the relationship among orientation, curvature and the length of the outline of the object. Furthermore, in this space it is easy to achieve the multiscaling feature as we will see later. The input function in the orientation space is the relationship between orientation of the tangent to the curve to the arc length along the contour at the curvature changing point. An example of the input function in the orientation space describing the corner point is shown in Figure 2.

![Diagram](image)

Figure 2. A Corner Point and its Input Function in the Orientation Space. (a) A Corner Fragment Consisting of Two Circular Fragments of Curvature $m_1$ and $m_2$, Enclosing an Angular Discontinuity of $k$. (b) An Input Function Corresponding to the Corner Point on the Left.

where $s$-axis represents the arc length and $\theta$-axis represents...
the orientation of the tangent to the curve. The curvature changing point in Figure 2 is a typical corner point. The analytical form of this corner point can be defined by the following piece-wise function:

\[
f_C(s) = \begin{cases} 
  m_1s + b & \text{if } s < 0 \\
  m_2s + b + k & \text{if } s > 0 
\end{cases}
\]

where \(m_1\) and \(m_2\) are curvatures of the curves and both are constants; \(b\) is the initial value of the orientation and is a constant also; \(k\) is the degree of angular discontinuity. The above piece-wise function obtained from Figure 2 (b) is under the following assumption: the curvature between the two curvature changing points is not changed. Note that this is only a local assumption and has no effect on the neighborhood points, even if these points are very close to each other. By using the above assumption for the input function, a series of important results can be revealed without using complicated mathematical manipulation, as we will see in the next section.

Similar analysis can be applied to the smooth joint point. An example of the input function in the orientation space describing the smooth joint point is shown in Figure 3.
where $s$-axis represents the arc length and $\theta$-axis represents the orientation of the tangent to the curve.

Similarly, under the same assumption used for a corner point, the above input function can be defined by the following piece-wise function.

$$f_c(s) = \begin{cases} 
  m_1s + b & \text{if } s < 0 \\
  m_2s + b & \text{if } s > 0 
\end{cases}$$

where $m_1$, $m_2$, and $b$ are the same as before, but notice that $k$ has disappeared since there is no sudden changes in the orientation of tangent to the curve.

So far two input functions which describe the two different kinds of curvature changing points, i.e., corner...
point and smooth point, have been obtained. The next question is how to get the multiscaling feature and how to restore the original image of the object by these input functions. These will be achieved by Gaussian function and its Fourier transform which will be discussed in the following sections.

Multiscaling and Gaussian Function

As we mentioned in Chapter II, the Fourier transform is a spatial-to-frequency transformation. The sharper the outline of the shape of the object in the spatial domain, the more the high frequency components in the frequency domain. In other words, the sharp portion of, or the sudden change in the outline represents the high frequency in its corresponding Fourier transform, while the smooth portion, or the flat portion of the outline represents the low frequency in its corresponding Fourier transform. Therefore, if the high frequency components can be removed in the frequency domain and the remaining parts can be transformed back to the spatial domain by using inverse Fourier transform, the shape of the object will become more smooth than the original one. Intuitively, the more high frequency components are removed, a more smooth image of the original object will be obtained and, therefore, the so called coarse-to-fine representation
of the object can be achieved just by removing more or less high frequency components in the frequency domain.

In the filter design theory (Andrews and Hunt, 1977), many filters are available to achieve the above function simply by changing the scope of band-limited filter, or by using different types of filters. In practical, not all the filters can be used for the image representation and recognition since after transform and inverse transform under different scaling many filters will alter the zero-crossing point; i.e., the position of curvature changing point in this paper. As pointed out by Marr and Hildreth (Brady 1982), "... a major difficulty with natural image is that changes can and do occur over a very wide range of scales, so it follows that one should seek a way of dealing with changes occurring at different scales." A way to do this, which is strongly advocated by many specialists on computer vision (Pinker 1985), is to use the Gaussian filter. They have shown that the Gaussian filter is essentially unique in having the property that location of the feature points is not changed as one moves from fine to coarse scale. Therefore the Gaussian filter is adopted for the shape representation process and will be given detailed discussion in this paper. In this section, some properties of convolution of the Gaussian function will be presented and the results will show that the standard deviation in the
Gaussian function plays just the role of scaling in the multiscaling of image representation and recognition. The results show that the Fourier transform of function is just the Gaussian filter we discussed above.

Let's first look at the Gaussian function \( g_\sigma(s) \) and its convolution with the input function \( f_C(s) \). Gaussian function is usually denoted by \( g_\sigma(s) = \frac{1}{\sqrt{2\pi}\sigma} \cdot \exp(-s^2/2\sigma^2) \). (Brady and Ponce, 1985). Note that the subscription \( \sigma \) has been specially marked out, since the standard deviation \( \sigma \) works as a very important parameter, scaling, in the multiscaling recognition process and its value will be frequently changed. The convolution will be dealt with separately for the corner point and smooth point since their input functions are different.

**Convolution for Corner Point**

First consider the convolution of the corner point, where input function is denoted by

\[
f_C(s) = \begin{cases} 
  m_1s + b & \text{if } s < 0 \\
  m_2s + b + k & \text{if } s > 0 
\end{cases}
\]

Calculating the convolution of \( g_\sigma(s) \) and \( f_C(s) \), we find

\[
g_\sigma(s) * f_C(s) = \int_{-\infty}^{+\infty} g_\sigma(t) \cdot f_C(s-t) \cdot dt
\]

\[
= \int_{-\infty}^{+\infty} \left( \frac{1}{\sqrt{2\pi}\sigma} \cdot \exp(-t^2/2\sigma^2) \right) \cdot f_C(s-t) \cdot dt
\]
\[
\begin{align*}
&= \int_{-\infty}^{\infty} \left( \frac{1}{\sqrt{2\pi}\cdot\sigma} \right) \cdot \exp\left( -t^2 / 2\sigma^2 \right) \cdot [m_2(s-t)+b+k] \cdot dt \\
&\quad + \int_{-\infty}^{\infty} \left( \frac{1}{\sqrt{2\pi}\cdot\sigma} \right) \cdot \exp\left( -t^2 / 2\sigma^2 \right) \cdot [m_1(s-t)+b] \cdot dt \\
&= (m_2/\sqrt{2\pi}\cdot\sigma) \cdot s \cdot \int_{-\infty}^{\infty} \exp\left( -t^2 / 2\sigma^2 \right) \cdot dt \\
&\quad + (m_1/\sqrt{2\pi}\cdot\sigma) \cdot s \cdot \int_{-\infty}^{\infty} \exp\left( -t^2 / 2\sigma^2 \right) \cdot dt \\
&\quad \quad \quad \left[ + (b/\sqrt{2\pi}\cdot\sigma) \cdot s \cdot \int_{-\infty}^{\infty} \exp\left( -t^2 / 2\sigma^2 \right) \cdot dt \right. \\
&\quad \quad \quad \quad \left. + (k/\sqrt{2\pi}\cdot\sigma) \cdot \int_{-\infty}^{\infty} \exp\left( -t^2 / 2\sigma^2 \right) \cdot dt \right] \\
&\quad \quad \quad \quad \quad \left[ - (m_2/\sqrt{2\pi}\cdot\sigma) \cdot s \cdot \int_{-\infty}^{\infty} \exp\left( -t^2 / 2\sigma^2 \right) \cdot dt \\
&\quad \quad \quad \quad \quad \quad \quad \quad + m_1 \cdot \int_{-\infty}^{\infty} \exp\left( -t^2 / 2\sigma^2 \right) \cdot dt \right] + b \\
&\quad \quad \quad \quad \quad \quad \quad \quad \quad \left[ + (m_2-m_1) \cdot \int_{-\infty}^{\infty} \exp\left( -t^2 / 2\sigma^2 \right) \cdot dt \right]
\end{align*}
\]

Take the first derivative of \( g_\sigma(s) \cdot f_\sigma(s) \) and multiply the result by \( \sqrt{2\pi}\cdot\sigma \), that is

\[
\sqrt{2\pi}\cdot\sigma \cdot \left( g_\sigma(s) \cdot f_\sigma(s) \right)'
\]

\[
= k \cdot \exp\left( -s^2 / 2\sigma^2 \right) + m_2 \cdot s \cdot \int_{-\infty}^{\infty} \exp\left( -t^2 / 2\sigma^2 \right) \cdot dt + m_1 \cdot \int_{-\infty}^{\infty} \exp\left( -t^2 / 2\sigma^2 \right) \cdot dt \quad (1)
\]

Then take the second derivative of (1),

\[
\sqrt{2\pi}\cdot\sigma \cdot \left( g_\sigma(s) \cdot f_\sigma(s) \right)''
\]

\[
= (-k/\sigma^2) \cdot s \cdot \exp\left( -s^2 / 2\sigma^2 \right) + (m_2-m_1) \cdot \exp\left( -s^2 / 2\sigma^2 \right) \quad (2)
\]
The first and the second derivatives of the above convolution are shown in Figure 4 (b) and (c), respectively.

For the first derivative of the above convolution, it has the shape of a typical Gaussian function which is offset from the line $s = 0$ by the scale $\sigma$. The height of the Gaussian curve will decrease dramatically with the small scale $\sigma$.

If $m_1 = m_2 = m$, the first and the second derivatives of the formula (1) and (2) will be in a much simpler form. In this case, $f_c(s)$ is denoted as follows:

$$f_c(s) = \begin{cases} \text{ms} + b & \text{if } s < 0 \\ \text{ms} + b + k & \text{if } s > 0 \end{cases}$$
By using the same procedure as above, we find

\[ \sqrt{2\pi} \cdot \sigma \cdot (g_\sigma(s) \ast f_\sigma(s))' = k \cdot \exp(-s^2/2\sigma^2) + \sqrt{2\pi} \cdot \sigma \cdot m \quad (3) \]

The formula (3) consists of two parts. The first part is \( k \cdot \exp(-s^2/2\sigma^2) \). The second part is \( \sqrt{2\pi} \cdot \sigma \cdot m \). Note that the second part contains no variable \( s \), so it is just a constant for the fixed value \( m \) and \( \sigma \). The first part is the product of \( k \) and \( \exp(-s^2/2\sigma^2) \). Consider the integral of \( k \cdot \exp(-s^2/2\sigma^2) \), i.e.

\[ I_\sigma(s) = \int_0^{s_1} k \cdot \exp(-t^2/2\sigma^2) \cdot dt \]

Clearly, the larger the value of \( \sigma \), the larger the value of this definite integral. Also note that the integral of formula (3) is the convolution of \( g_\sigma(s) \) and \( f_\sigma(s) \), which is just the inverse Fourier transform in the spatial domain. Therefore, the larger value of this function means the easier the corresponding corner point is detected. In this sense, the value of \( \sigma \) determines the detectability of the corresponding corner point. In other words, whether a particular corner point can be detected depends on the value of scale \( \sigma \). Therefore, a coarse-to-fine image representation and recognition process can be achieved by adequately choosing scale and the so called multiscaling technique has been achieved.

Taking the derivative for the (3) again, we obtain
\[
\sqrt{2\pi\cdot\sigma^2}(g_\sigma(s) \ast f_c(s))'' = (-k/\sigma^2) \cdot s \cdot \exp(-s^2/2\sigma^2)
\]  
(4)

Note that in the formula (4), if \( s = 0 \), the function value equals zero, which means that the second derivative of convolution of Gaussian function and input function has a zero-crossing. This zero-crossing indicates where the original curvature changing point lies and also tells us that this point is a corner point since there exists no zero-crossing points for the smooth point. Besides that, the distance along the \( s \)-axis between the two peaks in the formula (4) is

\[
d_{pp} = 2\sigma
\]  
(5)
which is obtained by finding the local extreme in formula (4). \( d_{pp} \) is exactly trice the value of standard deviation and therefore can be used to reflect the nature of scale \( \sigma \).

From the above discussion about the corner point, the following conclusions can be made.

1. The value of the first derivative of the convolution of Gaussian function and input function is proportional to the \( k \) at point \( s = 0 \), which indicates the discontinuous changes in the orientation at the corner point.

2. The second derivative of the convolution of Gaussian function and input function has a zero-crossing at \( s = 0 \), i.e. the location of curvature changing point. Therefore, this parameter can be used to distinguish the corner point and smooth point.
3. Most important of all is that the multiscaling feature can be achieved by using the scale $\alpha$. This feature is a distinguished one in the Fourier model and is consistent with the human being's recognition process.

**Convolution of Smooth Point**

Now let's consider the case of smooth point, whose input function is denoted by

$$f_C(s) = \begin{cases} m_1s + b & \text{if } s < 0 \\ m_2s + b & \text{if } s > 0 \end{cases}$$

Similar to the case of corner point, by calculating the convolution of $g_\alpha(s)$ and $f_C(s)$, we have

$$g_\alpha(s) * f_C(s) = \int_{-\infty}^{+\infty} g_\alpha(t) \cdot f_C(s-t) \cdot dt$$

$$= \int_{-\infty}^{+\infty} (1/\sqrt{2\pi}\cdot\alpha) \cdot \exp(-t^2/2\alpha^2) \cdot f_C(s-t) \cdot dt$$

$$= \int_{-\infty}^{S} (1/\sqrt{2\pi}\cdot\alpha) \cdot \exp(-t^2/2\alpha^2) \cdot [m_2(s-t)+b] \cdot dt$$

$$+ \int_{S}^{+\infty} (1/\sqrt{2\pi}\cdot\alpha) \cdot \exp(-t^2/2\alpha^2) \cdot [m_1(s-t)+b] \cdot dt$$

$$= (m_2/\sqrt{2\pi}\cdot\alpha) \cdot S \cdot \int_{-\infty}^{S} \exp(-t^2/2\alpha^2) \cdot dt + (m_1/\sqrt{2\pi}\cdot\alpha) \cdot S \cdot \int_{S}^{+\infty} \exp(-t^2/2\alpha^2) \cdot dt$$

$$+ \alpha/\sqrt{2\pi} \cdot [m_2 \cdot \exp(-S^2/2\alpha^2) - m_1 \cdot \exp(-S^2/2\alpha^2)] + b$$
Take the first derivative of \( g_0(s) * f_C(s) \) and multiply the result by \( \sqrt{2\pi} \cdot \phi \), that is
\[
2\pi \cdot \phi \cdot (g_0(s) * f_C(s))'
\]
\[
= m_2 \int_{-\infty}^{\infty} \exp(-t^2/2\sigma^2) \cdot dt + m_1 \int_{-\infty}^{\infty} \exp(-t^2/2\sigma^2) \cdot dt
\]
\[
= \begin{cases} 
  m_1 \int_{-\infty}^{\infty} \exp(-t^2/2\sigma^2) \cdot dt + (m_2-m_1) \int_{-\infty}^{\infty} \exp(-t^2/2\sigma^2) \cdot dt & m_2 > m_1 \\
  m_2 \int_{-\infty}^{\infty} \exp(-t^2/2\sigma^2) \cdot dt + (m_1-m_2) \int_{-\infty}^{\infty} \exp(-t^2/2\sigma^2) \cdot dt & m_1 > m_2
\end{cases}
\]

Then take the second derivative of (6), we have
\[
\sqrt{2\pi} \cdot \phi \cdot (g_0(s) * f_C(s))'' = (m_2-m_1) \cdot \exp(-s^2/2\sigma^2)
\]

The first and the second derivatives of the above convolution are shown in Figure 5 (b) and (c), respectively.

The first derivative of the convolution of Gaussian function with input function shown in Figure 5 (b) has the form of a smooth step whose height is proportional to \((m_1-m_2)\), or \((m_2-m_1)\). And this difference is just the
difference in curvature flanking the curvature changing point. If the difference in flanking curvatures is small enough, this step may not be perceived. In other words, the less the changes in curvature, the more difficult to perceive it. This property is consistent with the nature of human perception of the changes. The second derivative of the convolution is shown in Figure 5 (c). Instead of having zero-crossing point, it has a single peak whose height is proportional to \((m_2 - m_1)\). Therefore, this information not only can be used to decide where the curvature changing point locates but also can tell corner points from smooth points.
As a summary of the smooth point, the following conclusions can be made.

1. The height of the step in the first derivative of the above convolution is just the difference of curvature changes at the smooth point. The larger the difference in flanking curvatures, the easier the detection of this curvature changing point, which is consistent with human perception of changes.

2. There is no zero-crossing for the second derivative of the above convolution, but there is a peak at \( s = 0 \) which can be used to tell smooth points from corner points.

Fourier Transform and Gaussian Filter

In the last section, it has been shown that there exists many nice properties in the convolution \( g_{\omega}(s) * f_c(s) \). Now the questions are: what is the physical meaning behind the convolution of the Gaussian function with its input functions? How does the Gaussian function work in the frequency domain. These are the questions to be discussed in this section.

It is known in the previous chapter that the image signal can be represented in the frequency domain by Fourier transform, and it also can be transformed back to the spacial domain by the inverse Fourier transform. Physically, it is clear that if some high frequency components can be removed
in the frequency domain, the image in the spatial domain will become more smooth than before. As we will see later, the Fourier transform of the Gaussian function under the convolution with its input function works just like the filter for removing the high frequency components. Before applying the Fourier transform directly to the Gaussian function, it is better to establish the concept of transfer function first.

The Transfer-Function Concept

Generally, for any system having input and output, transfer function of the system is the ratio of its output to its input (Ziemer and Tranter, 1983). For example, if a low-passing filter is being considered as a system, where \( R(w) \) denotes the input signal of the filter and \( C(w) \) denotes the output signal through this filter, then the transfer function, denoted by \( G(w) \), of this filter is \( G(w) = \frac{C(w)}{R(w)} \).

Transfer function reflects the intrinsic properties of that system. Usually, the transfer function can be represented by a block diagram as in Fig. 6, and each block has a one-to-one corresponding relation with the physical device such as the filter in the example. Now let's see what will happen if the Fourier transform is applied to the convolution of the Gaussian function and its input function.
Fourier Transform of the Convolution

By definition (Stearns, 1975), Fourier transform is denoted as

$$F(w) = \int_{-\infty}^{\infty} f(t) \cdot \exp(-jwt) \cdot dt$$

where $f(t)$ is any continuous or piece-wise continuous absolutely integral function in the time or spatial domain, $w$ is the variable in the frequency domain.

By taking the Fourier transform to $g_o(t) * f_c(t)$, we find

$$F(w) = \int_{-\infty}^{\infty} [g_o(t) * f_c(t)] \cdot \exp(-jwt) \cdot dt$$

$$= \int_{-\infty}^{\infty} \left[ \int_{-\infty}^{\infty} g_o(t-s) \cdot f_c(s) \cdot ds \right] \cdot \exp(-jwt) \cdot dt$$

By changing the order of integration, we find

$$F(w) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} g_o(t-s) \cdot \exp(-jwt) \cdot dt \cdot f_c(s) \cdot ds$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
By replacing \( t \) for \( t-s \) in the inner integral, then

\[
F(w) = \int \left[ \int g_\phi(\tau) \cdot \exp(-j\omega \tau) \cdot \exp(-j\omega s) \cdot d\tau \right] \cdot f_C(s) \cdot ds
\]

\[
= \int \left[ \int g_\phi(\tau) \cdot \exp(-j\omega \tau) \cdot d\tau \right] \cdot f_C(s) \cdot \exp(-j\omega s) \cdot ds
\]

\[
= \int g_\phi(\tau) \cdot \exp(-j\omega \tau) \cdot d\tau \cdot \int f_C(s) \cdot \exp(-j\omega s) \cdot ds
\]

\[
= G_\phi(w) \cdot F_C(w)
\]  

(8)

where \( G_\phi(w) = \int g_\phi(\tau) \cdot \exp(-j\omega \tau) \cdot d\tau \) is the Fourier transform of the Gaussian function, and \( F_C(w) = \int f_C(s) \cdot \exp(-j\omega s) \cdot ds \) is the Fourier transform of the input function, i.e. input function in the frequency domain.

It is known from formula (8) that the Fourier transform of \( g_\phi(t) f_C(t) \) is the multiplication product of \( G_\phi(w) \) and \( F_C(w) \). Since \( F_C(w) \) is the input function in the frequency domain, \( G_\phi(w) \) can be treated as a transfer function or be considered as a filter of some system. So, \( G_\phi(w) \) is also called Gaussian Filter. The output, \( F(w) \), of this system is the Fourier transform of the convolution of Gaussian function and the input function. Therefore, simply using the inverse Fourier transform, the desired convolution of \( g_\phi(t) \) and \( f_C(t) \) can be obtained.

So far, the relationship of the convolution of the Gaussian function and input function in both frequency domain
domain and spatial domain has been revealed. Now the question is how to get this convolution. Since the actual signal obtained from the computer is a series of sequences instead of a continuous signal, the above method must be modified to handle the discrete signal. This will be discussed in the next chapter.
CHAPTER IV

FFT IMPLEMENTATION OF CONVOLUTION FUNCTION

As discussed in the previous chapter, the convolution of the input function, \( f_c(s) \), with the Gaussian function \( g_\alpha(s) \) is our desired result. This result can also be obtained by the inverse Fourier transform of the product of \( F_c(w) \) and \( G_\alpha(w) \). But in practice, the signal obtained from sampling the input function is a series of discrete sequences. So it is better to use the discrete mathematical model directly. In this chapter, the relationship between the continuous and discrete Fourier models is explained, and a practical algorithm based on this discrete model has been put forward to solve the recognition problems.

Z-Transform and Periodic Property

Let \( g(nT) \) be the sequence of the Gaussian function after adequate sampling \( g(s) \). Similarly, denotes \( f(nT) \) as the sequence of input function \( f(s) \). \( T \) is the sampling period. From the sampling theory (Ziemer and Tranter, 1983), it is known that a suitable model of the sampled signal is
\[
    f_s(t) = \sum_{n=-\infty}^{\infty} f(t) \cdot \delta(t-nT) \tag{9}
\]

where \( f_s(t) \) is the input function after sampling and \( f(t) \) is the continuous input function. Note that \( \delta(t-nT) \) is a delta function which is identically zero except at the sampling instants, \( t=nT \), so \( f(t) \) can be replaced by \( f(nT) \). Thus formula (9) will be written as

\[
    f_s(t) = \sum_{n=-\infty}^{\infty} f(nT) \cdot \delta(t-nT) \tag{10}
\]

Taking the Fourier transform, yields

\[
    F(w) = \int_{-\infty}^{\infty} \sum_{n=-\infty}^{\infty} f(nT) \cdot \delta(t-nT) \cdot \exp(-jwt) \cdot dt
\]

which is, upon interchanging integration and summation

\[
    F(w) = \sum_{n=-\infty}^{\infty} f(nT) \cdot \int_{-\infty}^{\infty} \delta(t-nT) \cdot \exp(-jwt) \cdot dt
\]

Using the sifting property of the delta function, this integrates to

\[
    F(w) = \sum_{n=-\infty}^{\infty} f(nT) \cdot \exp(-jwnT)
\]

Finally, defining the parameter \( z \) as

\[
    z = \exp(jwT)
\]

yields

\[
    F(z) = \sum_{n=-\infty}^{\infty} f(nT) \cdot z^{-n} \tag{11}
\]

\( F(z) \) is known as the \( z \)-transform of the sequence of samples.
Similarly, for the sequence of Gaussian function \(g(nT)\), its corresponding \(z\)-transform is

\[
G(z) = \sum_{n=-\infty}^{\infty} g(nT) \cdot z^{-n}
\]  

(12)

By the previous discussion, \(G(z)\) and \(F(z)\) can be considered as the Fourier transforms for the discrete models. So, their product is the Fourier transform of \(f(nT) \ast g(nT)\). To get back the convolution of \(f(nT)\) and \(g(nT)\), simply by applying inverse \(z\)-transform to the product of \(G(z)\) and \(F(z)\), which is denoted as

\[
y(nT) = Z^{-1}(G(z) \cdot F(z))
\]

In order to speed up the calculation, Fast Fourier Transform (FFT) algorithm may be applied to \(g(nT) \ast f(nT)\). The necessary condition for this algorithm is that both \(g(nT)\) and \(f(nT)\) must be finite-duration, or periodic. Fortunately, after adequate approximation, both of them could be considered as periodical sequences (Stearn, 1975).

First look at the Gaussian function

\[
g(t) = \frac{1}{\sqrt{2\pi} \cdot \sigma} \cdot \exp(-t^2/2\sigma^2).
\]

It is clear that \(g(t)\) is nearly zero when the absolute value of \(t\) is greater than \(5\sigma\). So it is quite reasonable to consider that duration of \(g(t)\) is only within \(-5\sigma < t < 5\sigma\), since the value outside of this range is close to zero and the contribution to the final convolution can be neglected.
On the other hand, f(n) is actually a periodic function, since we already assume in Chapter III that the contour is closed. Therefore, both of them can be considered as the finite-duration function. For convenience, let N be the length of this finite-duration and z substituted by \( \exp(2\pi jk/N) \), we have

\[
F(\exp(2\pi jk/N)) = \sum_{n=0}^{N-1} f(n) \cdot \exp(-2\pi jk/N)
\]

(13)

where \( f(n) \) is a short notation for \( f(nT) \).

It is not difficult to see that formula (13) is in the form of Discrete Fourier Transform (DFT) which can be calculated by the FFT algorithm directly (Brigham, 1974). Therefore, by the above discussion, the convolution of the input function and the Gaussian function can be achieved by the following steps and each of these steps is simply calculated by the FFT algorithm.

1. Calculate the DFT for \( f(n) \)

\[
F(k) = \sum_{n=0}^{N-1} f(n) \cdot W_N^k, \quad k = 0, 1, ..., N-1
\]

(14)

where \( F(k) \) is a short notation for \( F(\exp(2\pi jk/N)) \); and \( W_N \) is the short notation for \( \exp(-2\pi j/N) \); \( f(n) \) is the input sequence.

2. Calculate the DFT for \( g(n) \)
where $G(k)$ is the short notation for $G(\exp(2\pi jk/N));$ and $W_N$ is the short notation for $\exp(-2\pi j/N);$ $g(n)$ is the Gaussian sequence.

3. Calculate inverse DFT (IDFT)

$$y(n) = \text{IDFT}[F(k) \cdot G(k)]$$

$$= \frac{1}{N} \sum_{k=0}^{N-1} F(k) \cdot G(k) \cdot W_{kn}, \quad n = 0, 1, \ldots, N-1$$

(16)

where $y(n)$ is convolution of $f(n) * g(n)$.

FFT Algorithm and Time Complexity

The FFT algorithm we use here is the so called decimation-in-time algorithm (Brigham, 1974). The principle of decimation-in-time is most conveniently illustrated by considering the special case of $N$ as an integer power of $2$, i.e.,

$$N = 2^V$$

Since $N$ is an even integer, we can consider computing $F(k)$ by separating $f(n)$ into two $N/2$-point sequences consisting of the even-numbered points in $f(n)$ and the odd-numbered points in $f(n)$. With $F(k)$ given by

$$F(k) = \sum_{n=0}^{N-1} f(n) \cdot W_{kn}, \quad k = 0, 1, \ldots, N-1$$

(17)
and separating \( f(n) \) into its even and odd-numbered points we obtain

\[
F(k) = \sum_{n \text{ even}} f(n) \cdot w^{kn_N} + \sum_{n \text{ odd}} f(n) \cdot w^{kn_N},
\]

or with the substitution of variables \( n = 2r \) for \( n \) even and \( n = 2r + 1 \) for \( n \) odd,

\[
F(k) = \sum_{r=0}^{(n/2)-1} f(2r) W_N^{2rk} + \sum_{r=0}^{(n/2)-1} f(2r+1) W_N^{(2r+1)k}.
\]

But \( W_N^2 = W_n/2 \), since

\[
W_N^2 = \exp(-2j\pi/N) = \exp(-2j\pi/(N/2)) = W_N^N/2.
\]

Consequently, Eq. (18) can be written as

\[
F(k) = \sum_{r=0}^{(n/2)-1} f(2r) W_N^{rkN/2} + W_N^k \sum_{r=0}^{(n/2)-1} f(2r+1) W_N^{rk2N/2}.
\]

\[
= G(k) + W_N^kH(k).
\]

Each of the sums in Eq. (19) is recognized as an \( N/2 \)-point DFT of the even-numbered points of the original sequences and the second being the \( N/2 \)-point DFT of the odd-numbered points of the original sequences. Although the index \( k \) ranges over \( N \) values, \( k = 0, 1, \ldots, N-1 \), each of the sums need only be computed for \( k \) between 0 and \( N/2-1 \), since \( G(k) \) and \( H(k) \) are each periodic in \( k \) with period \( N/2 \). After the two DFT's corresponding to the two sums in Eq. (19) are computed, they are then combined to yield the \( N \)-point DFT, \( F(k) \).
With the computation restructured according to Eq. (19), we can compare the number of multiplication and addition required with those required for a direct computation of the DFT. We know that for direct computation without exploiting symmetry, \( N^2 \) complex multiplications and additions were required (Ziemer and Tranter, 1983). By comparison, Eq. (19) requires the computation of two \( N/2 \)-point DFTs, which in turn requires \( 2(N/2)^2 \) complex multiplications and approximately \( 2(N/2)^2 \) complex additions. Then the two \( N/2 \)-point DFTs must be combined, requiring \( N \) complex multiplications, corresponding to multiplying the second sum by \( W_N^k \) and then \( N \) complex additions, corresponding to adding that product to the first sum. Consequently, the computation of Eq. (19) for all value of \( k \) requires \( n+2(N/2)^2 \) or \( N+(N/2)^2 \) complex multiplications and complex additions. It is easy to verify that for \( N>2 \), \( N+(N/2)^2 \) will be less than \( n^2 \).

Equation (19) corresponds to breaking the original \( N \)-point computation into two \( N/2 \)-point computations. If \( N/2 \) is even, as it always is when \( N \) is equal to a power of 2, then we can consider computing each of the \( N/2 \)-point DFTs in Eq. (19) by breaking each of the sums in Eq. (19) into two \( N/4 \)-point DFTs, which would then be combined to yield the \( N/2 \)-point DFTs. Thus \( G(k) \) and \( H(k) \) in Eq. (19) would be computed as indicated below:

\[
G(k) = \sum_{l=0}^{N/2-1} g(r) w_r^k N/2 = \sum_{l=0}^{N/2-1} g(2l) w_{lN/2} + w_{N/2}^k \sum_{l=0}^{N/2-1} g(2l+1) w_{lN/4}^k
\]
Similarly,

\[ H(k) = \sum_{l=0}^{N/2-1} h(2l)W_k^{lN/4} + W_k^{N/2} \sum_{l=0}^{N/2-1} h(2l+1)W_k^{lN/4} \quad (20) \]

For the more general case with \( N \) a power of 2 greater than 3, we would proceed by decomposing the \( N/4 \)-point transforms in Eq. (20) into \( N/8 \)-point transforms, and continue until left with only two-point transforms. This requires \( v \) stages of computation, where \( v = \log_2 N \). Previously we found that in the original decomposition of an \( N \)-point transforms, the number of complex multiplication and additions required was \( N+2(N/2)^2 \). When the \( N/2 \)-point transforms are decomposed into \( N/4 \)-point transforms, then the factor of \((N/2)^2\) is replaced by \( N/2 + 2(N/4)^2 \), so the overall computation then requires \( N+N+4(N/4)^2 \) complex multiplication and additions. If \( N=2^V \), this can be done at most \( v = \log_2 N \) times, so that after carrying out this decomposition as many times as possible the number of complex multiplications and additions is equal to \( N \log_2 N \).
CHAPTER V

EXAMPLES

The software package using PASCAL language has been developed to test the previous algorithm. The whole package is divided into two parts.

The first part of this package (see Program I, Appendix A) is to create the image information by moving the cursor on the screen. When the cursor moves, the picture is drawn on the screen by the track of the cursor. The coordinates of all the points on the outline of this picture are the data representing the picture and are stored on the disk for later use.

The second part of this package (see Program II, Appendix B) is the real program for shape recognition and representation, which executes all the functions mentioned in the previous chapters. The program first inputs all the data already stored on the disk by the program I, and then converts them into the adequate input functions in the orientation space. By taking DFT for both input function and Gaussian function and taking IDFT for their product in the frequency domain, the curvature changing points and their
related parameters, such as curvature $m_1$ and $m_2$, angular discontinuity $k$, and so on, can be obtained. By using these points and their parameters, the original image can be reconstructed or be recognized.

Several data representing different images have been tested by this software package. The first example is to show that this algorithm can recognize objects irrelevant to their position and orientation. The original data represent two trees having the same shape but in different orientation, as shown in Figure 7.

![Figure 7. Two Trees Having the Same Shape But in Different Orientation.](image)

The parameters of the curvature changing points for the first tree are first calculated under some fixed scale by the above algorithm and stored in the memory as a template. Then the parameters which represent the second tree are calculated under the same scale and then are compared to that of the first tree. Theoretically, if all the parameters are the same, these two trees are considered the same one. Since these two trees have the same shape, the number of corner
points and the degree of angular discontinuity should be the same. The following is the actual result which is executed by the above algorithm and displayed on the screen. The result shows that over a very large range of scale $\alpha$. These two trees are being considered as the same one.

Enter Name of the File to be Listed:
B:Dual_Tree.dat (CR)

( Input the file name which has two trees of the same shape but in different orientation. )

Your Purpose? (Compare, Show detail, or reconstruction)
Compare (CR)

Running

Sample Period (Scale): 4 (CR)

( Result shows that these two trees are the same. )

Running

Sample Period (Scale): 6 (CR)

( Result shows that these two trees are the same. )

Running

Sample Period (Scale): 140 (CR)

( Result shows that these two trees are the same. )

The underlined part is typed by the computer. When the computer considers that these two objects are the same, it will say that they are the same and show only one of them on
the screen. If the computer considers that they are different, it will display both of them on the screen which means that they are different. In fact, these two trees are being considered as the same over a very large range of scaling, i.e. from 4 to 140.

On the other hand, look at the two triangles in Fig. 8 (a) and (b). They have different shapes. Though they have the same number of corner points, i.e., points A, B, and C in Figure 8 (a) and points A', B', and C' in Figure 8 (b). The results of the above algorithm show that only within a small range of scale, i.e., from 3 to 5, they are considered as the

![Image of two triangles](image)

**Figure 8.** (a) and (b) The Original Triangles. (c) The Reconstructed Image on the Screen Under the Small Scale 4. (d) and (e) The Reconstructed Image on the Screen Under the Scale 12, Which Also Shows that They Are Different Objects.
same triangle. In fact, under such a scale, the reconstructed image is very rough and is shown in Figure 8(c). Also this result gives the reason why they are considered as the same under such a scale. By increasing the value of the scale, the computer can "see" more detailed parts of the two triangles and can tell the difference between them. There are two reconstructed images in Figure 8 (d) and (e) under the scale 12. They look very close to their original images, respectively. But at same time, it is easy to see that they are different. This process of recognition is just like a human being looking at an object from far to near by and seeing more clearly with the decrease of distance in the object. This advantage is due to the multiscaling technique as we mentioned in previous chapters.

This multiscaling feature can be seen more clearly from another example. The image data for this example is a hammer as shown in Figure 9 (a). The original image has many

![Figure 9](image-url)

**Figure 9.** (a) The Image of the Original Hammer.  
(b) The Reconstructed Image of the Hammer Under Scale 12.  
(c) The Reconstructed Image of the Hammer Under Scale 5.
sharp angles, i.e., there are many corner points. Under the different scale $\alpha$, the reconstructed images are different. Figure 9 (b) is the reconstructed image of the hammer under scale 12, and Figure 9 (c) is the reconstructed image of the hammer under scale 5. Also, under the scale 12, the result of the above algorithm shows that a total of twelve corner points have been found. But under the scale 5, it is shows that only 5 corner points have been found.

It is clear from these examples that this algorithm can not only recognize the object but also can achieve the multiscaling feature during the recognition process. The higher the value of scale $\alpha$, the more precise the image of the original object can be obtained.
CHAPTER VI

CONCLUSIONS AND RECOMMENDATIONS

Curvature changing points have been suggested by many researchers as important feature points and have been used widely in pattern recognition. Fourier transform is the classic technique for digital signal processing. In this paper, a method combining curvature changing points with discrete Fourier transform has been used successfully in the process of shape recognition and representation. An important feature in computer vision, multiscaling, is achieved by this method. Several examples have also shown that the process of recognizing objects by this method is matched with the human recognition process. Besides that, since DFT and IDFT are the dominant calculation in the whole recognition process, FFT algorithm can be used to reduce the time complexity dramatically.

A problem exists in that the above algorithm works only under the assumption that the shape of the object must be closed and connected (see Chapter III). This assumption will exclude many application areas. A simple modification can be made to improve this weakness so that the same algorithm can
be used to recognize disconnected but closed shapes. This modification has actually been achieved in this software package. The principle is simple. Since each component is still closed and connected, they can be dealt with separately by using the same algorithm. Then, simply by combining all the results of each component will produce the desired image.

Another problem is that the relative position of each component can not be determined by the above algorithm. In order to know their relative position some more sophisticated structure description technique must be used. Therefore, it is recommended that some structure description technique, such as semantic nets or a decision tree, be used to determine the mutual relationship among each component so that more sophisticated objects can be recognized. The main trouble in doing so is that the Fourier transform does not preserve this property. In order to know the relative position, some knowledge must be obtained before the Fourier transform. Though there are several ways to do so (Winston, 1984), all of them have nothing to do with the Fourier transform and, therefore, were included in this paper.
Appendix A

Program I: Image Creation
PROGRAM I: IMAGE CREATION

THE PURPOSE OF THIS PROGRAM IS TO PRODUCE THE IMAGE INFORMATION FOR FURTHER ANALYSIS. WHEN YOU MOVE THE CURSOR ON THE SCREEN, A PICTURE WILL BE DRAWN ON THE SCREEN BY THE TRACK OF THE CURSOR. ALL THE POINTS ON THIS TRACK WILL BE CONVERTED INTO THE CORRESPONDING COORDINATES AND THEN BE STORED ON THE DISK.

{* c-}
Program Image_c(input,output);
{* I graph.p}

VAR
FilVar: Text;
Filename: String[14];
Px,py,ctx,cty:ARRAY[1..40] OF integer;
Property: ARRAY[1..40] OF char;
CUR,ORF,ORB: ARRAY[1..40] OF real;
First,Second,Third,Last,FunctionKey:Boolean;
x,y,i:integer;
InKey:char;
PROCEDURE DRAWING

THIS PROCEDURE PERFORM THE TASK OF DRAWING THE IMAGE ON THE SCREEN BY MOVING THE CURSOR.

Procedure Drawing(InKey:char; FunctionKey:boolean);
Const
  NorthEast=45;
  SouthEast=135;
  SouthWest=225;
  NorthWest=315;
PROCEDURE DOFUNCTION

THIS PROCEDURE IS USED TO IDENTIFY THE DIFFERENT
FUNCTION KEYS ON THE KEYBOARD.

Procedure DoFunction(FunctionKey:char);

begin
CASE FunctionKey OF
  'H': SetHeading(North); { UP ARROW KEY }
  'P': SetHeading(South); { DOWN ARROW KEY }
  'M': SetHeading(East); { LEFT ARROW KEY }
  'K': SetHeading(West); { RIGHT ARROW KEY }
  'I': SetHeading(NorthEast); { PGUP KEY }
  'Q': SetHeading(SouthEast); { PGDN KEY }
  'G': SetHeading(NorthWest); { HOME }
  'O': SetHeading(SouthWest); { END }
  '<': SetHeading(Heading+5); { F1 }
  ';': SetHeading(Heading-5); { F2 }
end;
Forwd(1);
IF start=true THEN
begin
  x:=xcor+158; y:=96-ycor;
  writeln(FilVar, 'P ',x,' ',y);
  plot(x,y,3);
end;
end;
begin
    IF FunctionKey then DoFunction(upcase(InKey))
ELSE
    begin
        IF (first=false) OR (Upcase(InKey)='F') Then
        begin
            CASE Upcase(InKey) OF
            'F': begin { TO START DRAWING }
                x:=xcor+158; y:=96-ycor;
                i:=1; property[i] := 'F';
                plot(x,y,3);
                px[i]:=x; py[i]:=y;
                first:=false;
                i:=i+1;
            end;
            'P': begin { POINT OF SMOOTH }
                X:=xcor+158; y:=96-ycor;
                px[i]:=x; py[i]:=y;
                plot(x,y,1);
                smoothp;
                property[i] := 'P';
                i:=i+1;
            end;
            'C': begin { TO START CORNER POINT }
                X:=xcor+158; y:=96-ycor;
                px[i]:=x; py[i]:=y;
                plot(x,y,1);
                smoothp;
                property[i] := 'C';
                second:=true; third:=true;
                i:=i+1;
            end;
            END;
        end;
    end;}
'E': begin {TO END DRAWING FOR SMOOTH POINT}
px[i]:=px[l]; py[i]:=py[l];
property[i]:'P';
smoothp;
i:=i+1;
end;

'D': begin {TO END DRAWING FOR CORNER POINT}
px[i]:=px[l]; py[i]:=py[l];
property[i]:'C';
smoothp;
i:=i+1;
end;
end; {of case}
end;
end;
FUNCTION Getkey(Var FunctionKey:Boolean):char;
Var ch: char;
begin
   read(kbd,Ch);
   If (Ch=#27) AND KeyPressed THEN  (it must be a function key)
      begin
         read(kbd,Ch);
         FunctionKey:=true;
      end
      ELSE FunctionKey:=false;
      GetKey:=ch;
   end;
PROCEDURE NEWSCREEN

PROCEDURE NEWSCREEN INITIALIZES THE PROGRAM IN THE
GRAPHIC MODE AND RESET ALL THE NECESSARY VARIABLES. ALSO
A SQUARE OUTLINE (OR CALLED WINDOW) IS DISPLAYED ON THE
SCREEN TO DEFINE THE SCOPE OF DRAWING.

Procedure NewScreen;
Const
  CharHeight=4;
VAR
  k: integer;

PROCEDURE DRAWBOX

THIS PROCEDURE IS TO DEFINE THE WINDOW ON THE SCREEN
AND THEN DRAW THE OUTLINE OF THIS SCREEN.

************
Procedure DrawBox(a,b,w,h:integer);
begin
  Draw(a,b,a+w,b,1);                         (top)
  Draw(a,b,a,b+h,1);                         (left side)
  Draw(a,b+h,a+w,b+h,1);                     (bottom)
  Draw(a+w,b+h,a+w,b,1);                     (right side)
end;  {draw box}

begin
  Graphics;
  DrawBox(0,0,319,199-charheight);
  TurtleWindow(159,99-(CharHeight DIV 2), 318, 198-CharHeight);
  SHowturtle;
  Home;
  Wrap;
  Penup;
  i:=1;
end;
BEGIN
  WRITE('Enter name of file to create:');
  READLN(Filename);
  ASSIGN(FilVar,Filename);
  REWRITE(FilVar); {initialize disk file}
  First:=true; second:=true; third:=true;
  Last:=false;
  NEWSCREEN;
  REPEAT
    Inkey:=getKey(FunctionKey);
    Drawing(Inkey,FunctionKey);
    Until UpCase(Inkey) in [#27,'c'];
  ClearScreen;
  CreateFile;
END.
Appendix B

Program II: Image Representation and Recognition
PROGRAM II: IMAGE REPRESENTATION AND RECOGNITION

The purpose of this program is to recognize and represent the shape of objects. The information or data which describe the shape of object has already been stored in the memory by another program.

This program performs the following tasks:

1. Fetch the data in the database and convert them into the form which is acceptable by this program.
2. Calculate the convolution of Gaussian function and input function and get the feature points by using FFT.
3. Reconstruct the image according to the following choices.
4. There are several choices in this program: reconstruction, comparison and showing details.
REDRAW THE IMAGE ON THE SCREEN BY THE DATA IN THE DATABASE
ACCORDING TO THE DIFFERENT SCALING. COMPARISON PERFORM THE
TASK OF COMPARING TWO DIFFERENT IMAGES AFTER RECOGNITION.
IF TWO IMAGES ARE SAME, ONLY ONE OBJECT WILL BE SHOWN ON THE
SCREEN. IF THEY ARE DIFFERENT, BOTH OF THEM WILL BE SHOWN ON
THE SCREEN AND SAY THAT THEY ARE DIFFERENT. SHOWING
DETAILS IS TO DISPLAY THE SUBIMAGE OF THE WHOLE IMAGE, WHICH
MEANS THAT THIS RECOGNITION PROCESS CAN SEE "DETAILS" JUST
LIKE HUMAN BEING.

Program Cog_pas(input,output);
($I graph.p)

VAR
FilVar,Cm: Text;
Filename: String[14];
Px,py,ctx,cty:ARRAY[1..140] OF integer;
Property,cha:ARRAY[1..140] OF char;
CUR,ORF,ORB:ARRAY[1..140] OF real;
max:ARRAY[1..10] OF integer;
cpx,cpy,ct:ARRAY[1..10,1..140] OF integer;
curl,curr:ARRAY[1..5,1..140] OF real;
First,Second,Third:Boolean;
mmx,ix,mm,scmode,index,x,y,i,tmax,segment:integer;
fi,pm,inkey: char;
PROCEDURE CURVE

THIS PROCEDURE DRAW THE CURVES BETWEEN TWO DIFFERENT CURVATURE CHANGING POINTS.

*********************************************************************}

Procedure curve(ax,ay,bx,by,cx, cy:integer);
VAR
orab, orbf, orbb, orcf, ml, m2, b1, b2, ex, ey, r, s:real;
ss:char;
{******************************************************************

PROCEDURE CORNER TRUE

THIS PROCEDURE DEALS WITH THE CORNER POINTS

******************************************************************

Procedure thtrue;
VAR
 dx, dy, s1, s2:real;
 tx, ty, ptx, pty, xincr, yincr:integer;
ST:CHAR;
begin
  dx:=ax-cx; dy:=ay-cy;
  tx:=ax; ty:=ay;

  {* DECIDE THE DIRECTION OF INCREMENT OF X AND Y *}
  If dx<0.0 THEN xincr:=1 ELSE xincr:=-1;
  IF dy<0.0 THEN yincr:=1 ELSE yincr:=-1;
  IF abs(dx)<abs(dy) THEN
    begin
      WHILE ty<>cy DO
        begin
          ty:=ty+yincr;
        end;
    end ELSE
    begin
      WHILE tx<>cx DO
        begin
          { read(st);  }
          tx:=tx+xincr;
          sl:=r-(tx-ex)*(tx-ex);
          { writeln('NOW THE ARGUMENT OF SQRT: ',SL); }
          IF ty>ey THEN ty:=trunc(ey+sqrt(sl))
            ELSE ty:=trunc(ey-sqrt(sl));
          gotoxy(2,24);
          writeln('tx=',tx,'ty=',ty);
        end;
    end;
  end
end
ptx:=trunc(tx+320); pty:=trunc(ty/2);
Plot(ptx, pty, 1);
{ read(st);  }
end;
third:=false;
end;

{******************************************************************************
   *
   * PROCEDURE SMOOTH TRUE
   *
   * THIS PROCEDURE DEALS WITH THE CASE OF SMOOTH POINTS.
   *
   ******************************************************************************}

Procedure thfalse;
VAR
  dx, dy: real;
  tx, ty, ptx, pty, xincr, yincr: integer;

begin
  dx:=bx-cx; dy:=by-cy;
  tx:=bx; ty:=by;

  {* DECIDE THE DIRECTION OF THE INCREMENT ON THE SCREEN. *}
  If dx<0.0 THEN xincr:=1 ELSE xincr:=-1;
  IF dy<0.0 THEN yincr:=1 ELSE yincr:=-1;
  IF abs(dx)<abs(dy) THEN
    begin
      WHILE ty<>cy DO
        begin
          ty:=ty+yincr;
        end
    end
  end;
(* IF INCREMENT OF X > Y, CHOOSE Y, THEN X. *)

IF tx > ex THEN tx := trunc(ex + sqrt(r - (ty - ey) * (ty - ey)))
ELSE tx := trunc(ex - sqrt(r - (ty - ey) * (ty - ey)))
ptx := trunc(tx + 320); pty := trunc(ty / 2);
plot(ptx, pty, 1);
end;
end ELSE

begin
WHILE tx < cx DO
begin
  tx := tx + xincr;

(* IF INCREMENT OF Y > X, CHOOSE X, THEN Y. *)

IF ty > ey THEN ty := trunc(ey + sqrt(r - (tx - ex) * (tx - ex)))
ELSE ty := trunc(ey - sqrt(r - (tx - ex) * (tx - ex)))
ptx := trunc(tx + 320); pty := trunc(ty / 2);
plot(ptx, pty, 1);
end;
end;

begin
gotoxy(42, 2);
writeln('The Original Image:');
FOR i := 3 TO index DO
begin
  IF property[i - 1] = 'C' THEN third := true
ELSE
begin
  Ax := px[i - 2]; ay := py[i - 2];
  bx := px[i - 1]; by := py[i - 1];
cx:=px[i]; cy:=py[i];
ex:=ctx[i-1]; ey:=cty[i-1];
r:=sqr(1.0/cur[i-1]);
{ writeln('AX=',AX,'AY=',AY);
 WRITELN('BX=',BX,'BY=',BY);
 WRITELN('CX=',CX,'CY=',CY);
 WRITELN('EX=',EX,'EY=',EY);
 WRITELN('R=',R); READ(ST); }
IF third=true THEN thtrue ELSE thfalse;
end;
end;

(******************************************************************************
 *                                                                             *
 *             PROCEDURE SMOOTH POINT                                           *
 *                                                                             *
 *             THIS PROCEDURE IS TO GET SMOOTH POINT                            *
 *                                                                             *
 ******************************************************************************
 Procedure smoothp;
 VAR
   ax,ay,bx,by,cx,cy:integer;
begin
    IF second=true THEN second:=false
  ELSE
    begin
        Ax:=px[i-2]; ay:=py[i-2];
        bx:=px[i-1]; by:=py[i-1];
        cx:=px[i]; cy:=py[i];
        curve(ax,ay,bx,by,cx,cy);
    end;
end;

BEGIN

{******************************************************************************************
*/
*/
*/    PROCEDURE DRAWING
*/
*/
*/    THIS PROCEDURE DRAW THE LINES ON THE SCREEN ACCORDING TO
*/    THE DIFFERENT NATURE OF POINTS ON THE SCREEN.
*/
*/
******************************************************************************************}/

Procedure Drawing(InKey:char; FunctionKey:boolean);
Const
    NorthEast=45;
    SouthEast=135;
    SouthWest=225;
    NorthWest=315;
begin
  IF FunctionKey then DoFunction(upcase(InKey))
ELSE
  begin
    IF(first=false)OR(Upscase(InKey)='F') Then
      begin
        CASE Upscase(InKey) OF
          {* FOR THE FIRST POINT. *}
          'F': begin
            x:=xcor+158; y:=96-ycor;
            i:=1; property[i]:='F';
            plot(x,y,3);  
            px[i]:=x; py[i]:=y;
            first:=false;
            i:=i+1;
            end;
          END;
          {* FOR THE MIDDLE POINT *}
          'P': begin
            x:=xcor+158; y:=96-ycor;
            px[i]:=x; py[i]:=y;
            plot(x,y,1);
            smoothp;
            property[i]:='P';
            i:=i+1;
            end;
          END;
          {* FOR THE CORNER POINT *}
          'C': begin
            x:=xcor+158; y:=96-ycor;
            px[i]:=x; py[i]:=y;
            plot(x,y,1);
            smoothp;
            property[i]:='C';
            i:=i+1;
            end;
          END;
          END;
        END;
      end;
    end;
  end;
end;
'C': begin
  x:=xcor+158;  y:=96-yco111;
  px[i]:=x; py[i]:=y;
  plot(x,y,1);
  smoothp;
  property[i]:'C';
  second:=true; third:=false;
  i:=i+1;
  end;

(* FOR THE END POINT *)

' E ': begin
  px[i]:=px[i];  py[i]:=py[i];
  property[i]:'E';
  smoothp;
  i:=i+1;
  end;

{ * FOR THE END POINT AND ALSO A CORNER POINT * }

'D': begin
  px[i]:=px[i];  py[i]:=py[i];
  property[i]:'D';
  smoothp;
  i:=i+1;
  end;
  end; {of case}
end; end;
FUNCTION GETKEY

THID PROCEDURE DEALS WITH THE SPECIAL CASES IN DRAWING PICTURE.

FUNCTION GetKey(Var FunctionKey:Boolean):char;
Var ch: char;
begn
  read(kbd,Ch);
  If (Ch=#27) AND KeyPressed THEN {it must be a function key}
    begin
      read(kbd,Ch);
      FunctionKey:=true;
    end
  ELSE FunctionKey:=false;
  GetKey:=Ch;
end;

PROCEDURE NEWSCREEN

THIS PROCEDURE IS TO CLEAR ALL THE EXISTING INFORMATION ON THE SCREEN AND CREATE A NEW SCREEN.

Procedure NewScreen;
Const
  CharHeight=4;
{**************************************************}
*               PROCEDURE   DRAW-BOX
*               THIS PROCEDURE IS DRAW THE WINDOW ON THE SCREEN.
**************************************************}

Procedure DrawBox(a,b,w,h:integer);
begin
  Draw(a,b,a+w,b,1);       {top}
  Draw(a,b,a+b+h,1);       {left side}
  Draw(a,b+h,a+w,b+h,1);   {bottom}
  Draw(a+w,b+h,a+w,b,1);   {right side}
  Draw(trunc(w/2),b,trunc(w/2),b+h,1);
  Draw(trunc(w/2),trunc(h/2),a+w,trunc(h/2),1);
end; {draw box}

begin
  HiRes;
  DrawBox(1,1,638,200-charheight);
  i:=1;
end;

Procedure readin;
  VAR j:integer;
      ss:char;

begin
  j:=1;
  WHILE NOT EOF(FilVar) Do
    begin
readln(FilVar, px[j], py[j], ctx[j], cty[j], cur[j],
orf[j], orb[j], ss, property[j]);
  writeln(px[j], ',py[j], ',ctx[j], ',cty[j], ',cur[j], ',
orf[j], ',orb[j], ',property[j]);

j:=j+1;
end; {of while}
index:=j-1;
{ writeln('INDEX=',index); } 
close(FilVar);
{ read(ss); } 
end;

{*******************************************************************************
 *                        PROCEDURE K-POINT                                      *
 *                        *                                                  *
 *                        THIS PROCEDURE IS USED TO PICK UP ALL THE SMOOTH     *
 *                        POINT AND CORNER TO BE DRAWN ON THE SCREEN. AND LIST *
 *                        ALL THE RESULTS OF THESE POINT, SUCH AS CURVATURE AND *
 *                        POSITION.                                          *
 *                        *******************************************************************************
}

Procedure kpoint;
  CONST lim=5.0;
  VAR
    abscur,Ave,T1,T2: real;
    T,j: integer;
    S: char;
begin
ave:=0.0; s:=' ';
FOR i:=1 TO index DO
  ave:=ave+cur[i];
ave:=ave/(index-1.0); {calculate the average of curvature.}
cur[index]:=cur[1];
FOR i:=1 TO index-1 DO
  begin
    IF(property[i+1]='C') THEN cha[i+1]='V'
    ELSE
      begin
        abscur:=abs(cur[i]-cur[i+1])/ave;
        IF(abscur>lim) THEN
          cha[i+1]='I'
        ELSE cha[i+1]='V';
        end;
  end;
{ arrange the data to be display on the screen. }
{ write(ave);
FOR i:=1 TO INDEX DO
  write(' ',cha[i]); writeln; }
gotoxy(2,3);
writeln(' X  Y  M1  M2  D'); gotoxy(2,4);
j:=5;
FOR i:=2 TO index DO
  begin
    IF cha[i]='V' THEN (list all the key point)
      begin
        IF property[i]<> 'C' THEN
          begin
            gotoxy(2,j);
          end;
      end;
  end;
writeln(s:2,px[i]:3,s:2,py[i]:3,cur[i-1]:8:5,
s:2,cur[i]:8:5,s:2,0:3);
end ELSE
begin
  gotoxy(2,j);
  t1:=arctan(orf[i]);
  t2:=arctan(orb[i]);
  t:=trunc((t2-t1)*180.0/3.14159);
  writeln(s:2,px[i]:3,s:2,py[i]:3,cur[i-1]:8:5,
          s:2,cur[i]:8:5,s:2,t:3);
  end;
  j:=j+1;
end;
end;
PROCEDURE B-SPLINE

IN THIS PROCEDURE THE TECHNIQUE OF B-SPLINE HAS BEEN USED TO SMOOTH THE CURVE OR THE OUTLINE OF THE OBJECT TO BE DISPLAYED.

Procedure b_spline;

CONST dt=0.05;
VAR
  j,k,nmax:integer;
  px,pcy,ct: real;
  XVer,YVer:ARRAY[0..41] OF integer;
  XV,YV:ARRAY[0..41] OF real;
  XC,YC,C0,cok,cok1: ARRAY[1..4] OF real;
  C: ARRAY[1..4,1..4] OF real;

begin
  j:=1;
  assign(cm,'b:c.dat');
  FOR i:=1 TO index DO
    begin
      IF cha[i]='V' THEN
        begin
          XVER[j]:=px[i]; yver[j]:=py[i]; j:=j+1;
        end;
    end;
  Xver[j]:=XVer[1]; Yver[j]:=YVer[1];
  XVer[j+1]:=XVer[2]; YVer[j+1]:=YVer[2];
  Xver[0]:=XVer[j-1]; Yver[0]:=YVer[j-1];
\begin{verbatim}
\{ FOR i:=0 TO j DO write(xver[i]," "); writeLN; \}
nmax:=j-1;
reset(cm);
FOR i:=1 TO 4 DO
begin
readln(cm,c[i,1],c[i,2],c[i,3],c[i,4]);
{ writeln(c[i,1]," ",c[i,2]," ",c[i,3]," ",c[i,4]); }
end;
close(cm);
FOR i:=1 TO nmax DO
begin
    FOR j:=1 TO 4 DO
    begin
        xc[j]:=0.0; yc[j]:=0.0;
        FOR k:=1 TO 4 DO
        begin
            xc[j]:=c[j,k]*xver[i+k-2]+xc[j];
            yc[j]:=c[j,k]*yver[i+k-2]+yc[j];
        end;
        { writeln(xc[j]," "); }
    end;
    co[4]:=1.0; co[3]:=dt;
    co[2]:=co[3]*dt; co[1]:=co[2]*dt;
    FOR j:=1 TO 20 DO
    begin
        cok[4]:=1.0; cok[3]:=j;
        cok[2]:=cok[3]*j; cok[1]:=cok[2]*j;
        FOR k:=1 TO 4 DO
        begin
            cok1[k]:=co[k]*cok[k];
            { write('cok1==',cok1[k]); writeln; }
        end;
        pcx:=0.0; pcy:=0.0;
        FOR k:=1 TO 4 DO
        begin
            pcx:=pcx+cok1[k]*xc[k];
            pcy:=pcx+cok1[k]*yc[k];
        end;
    end;
end;
\}
\end{verbatim}
begin
  pcx:=pcx+cok1[k]*xc[k];
  pcy:=pcy+cok1[k]*yc[k];
end;
  x:=trunc(pcx)+320; y:=trunc(pcy/2)+100;
  writeln('x=',x,'y=',y); 
  plot(x,y,1);
  read(f1); 
end;
end;

{******************************************************************************
 * PROCEDURE CREATE-FILE
 * THIS PROCEDURE IS TO READ THE DATA FROM CORRESPONDING INPUT FILE.
 *}

Procedure createfile;
VAR
  j,index: integer;
  is: char;
begin
  textmode;
  index:=i-1; j:=1;
  writeln('index=',index); { read(is); }
  FOR j:=1 TO index DO
    writeln(FilVar,px[j],',py[j]',',ctx[j]',',cty[j]',',cur[j]',
            ',orf[j]',',orb[j]',',property[j]');
  close(FilVar);
end;
PROCEDURE COMARE
* THIS PROCEDURE COMARE THE DIFFERENCE BETWEEN THE CORNER POINTS.

Procedure compare(tmax:integer; VAR ni:integer);
    Label c1;
    Const lim2=5.0;
    VAR
        n:integer;

    begin
        ni:=2;
        writeln('tmax',tmax);
        For i:=1 TO Tmax DO
            begin
                IF abs(curl[1,i]-curl[2,i])>lim2 THEN goto c1;
                IF abs(curr[1,i]-curr[2,i])>lim2 THEN goto c1;
                IF abs(ct[1,i]-ct[2,i])> THEN goto c1
            end;
        ni:=1;
    C1: end;
(***************************************************************************

* *

* MAIN PROGRAM

*

***************************************************************************)

begin
write('Enter name of file to list:');
readln(Filename);
writeln('Your purpose? (Comparison, Show detail or Reconstruction)');
read(pm); scmode:=1;
NewScreen(scmode);
createfile;
IF pm='S' THEN
begin
gotoxy(10,24);
Showturtle; Home; Penup;
write('Which part? (move cursor to decide)');
choose(index, lx);
mmx:=lx;
b_spline(mmx,max[mmx]);
end ELSE
begin
Clearscreen; Newscreen(scmode);
kpoint(index);
createfile;
IF pm='C' THEN
begin
compare(tmax, lx);
FOR l:=1 TO lx DO b_spline(l,max[l]);
read(fi);
end ELSE
begin
  b_spline(1, max[1]);
  read(f1);
  end;
  end;
  clearscreen;
  textmode;
  end.


Brady, M. "Computational Approaches to Image Understanding." Computer Surveys 14,1 (March 1982): 3-64.


