Rochester Institute of Technology RIT Scholar Works

Theses

7-1-1994

## Human face profile recognition

Vincent Wong

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

#### **Recommended Citation**

Wong, Vincent, "Human face profile recognition" (1994). Thesis. Rochester Institute of Technology. Accessed from

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

# **Human Face Profile Recognition**

by

Vincent Wong

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of MASTER OF SCIENCE in Computer Engineering

Approved by:

Graduate Advisor - Tony H. Chang, Professor

Ronald G. Matteson, Professor

Department Head - Roy S. Czernikowski

DEPARTMENT OF COMPUTER ENGINEERING COLLEGE OF ENGINEERING ROCHESTER INSTITUTE OF TECHNOLOGY ROCHESTER, NEW YORK

July, 1994

## **Thesis Release Permission Form**

## **Rochester Institute of Technology College of Engineering**

Title: Human Face Profile Recognition

I, Vincent Wong, hereby grant permission to the Wallace Memorial Library of RIT to reproduce my thesis in whole or in part.

\_\_\_\_\_ Signature:

Date: <u>7/20/94</u>

This document was produced using Lotus AmiPro 3.0 and Microsoft Windows Paintbrush. All programs for the human face profile recognition system were developed using Microsoft QuickC version 2.5 and they were run on an AT&T 386-class PC compatible machine with a math co-processor. All profile images were captured using ITEX PCVISIONplus frame grabber board and Panasonic WV-BL204 CCD camera.

The following names used here are registered trademarks of the respective companies:

AmiPro	Lotus Development Corporation,
Paintbrush	Microsoft Corporation,
QuickC	Microsoft Corporation,
PCVISIONplus	Imaging Technology Incorporated.

Copyright © 1994 by Vincent Wong All rights reserved.

# Abstract

The purpose of this thesis is to implement an automatic person identification system based on face profiles. Each person's face profile can be quite unique within a small sample population and therefore it can be used as the basis of an automatic person identification system. To quantify human face profiles for use in the recognition system, Fourier descriptors are used to describe the open curve extracted from a face profile. Fourier descriptors in the low-frequency range are shown to be useful for human face profile recognition. By using 16 Fourier coefficients, a correct recognition rate of 92% for 60 subjects was achieved.

# Contents

Abstract	•••••••••••••••••••••••••••••••••••••••	iv
Contents	•••••	v
List of Fi	gures	vi
List of Ta	ables	vii
Chapter 1	Introduction	1
Chapter 2	System Overview	4
Chapter 3	Face Profile Curve Extraction	10
	3.1 Chain Code and Curve Extraction	16
	3.2 <b>Turning Point Detection Algorithm</b>	20
Chapter 4	Fourier Descriptors	24
	4.1 Curve sampling	35
Chapter 5	Matching	38
Chapter 6	Analysis of the System Performance	47
Referenc	es	56
Appendix	κ	58

# **List of Figures**

Figure 2.1	Block diagram of the system	4
Figure 2.2	System setup	5
Figure 3.1	Tip of nose	10
Figure 3.2	Bottom of nose	11
Figure 3.3	Top of nose	12
Figure 3.4	Chin position	13
Figure 3.5	Terminating positions of the face profile curve	14
Figure 3.6	Complete face profile curve	15
Figure 3.1.1	Eight-connectedness in binary images	17
Figure 3.1.2	Contour-tracing mask	17
Figure 3.1.3	Ambiguity in curve length calculation	18
Figure 3.1.4	Right-angled silhouette in contour-tracing	19
Figure 3.2.1	Turning point detection algorithm block diagram	20
Figure 3.2.2	Bottom of nose for turning point detection algorithm	
	demonstration	22
Figure 3.2.3	Intermediate output of the turning point detection algorithm	
	demonstration	22
Figure 4.1	Contour function in a complex space	24
Figure 4.2	A rotated coordinate system	27
Figure 4.3	Closed boundary obtained from an open curve	30
Figure 4.4	Example of a way of open curve sampling	32
Figure 4.1.1	Different ways of sampling the boundary curve of a line object	36
Figure 5.1	Fourier descriptors vectors of 4 people's face profiles	41
Figure 5.2	Euclidean distance comparisons of the Fourier descriptors vectors	
	of 4 people's face profiles	42
Figure 5.3	Fourier descriptors vectors of 7 face profiles of the same person	44
Figure 5.4	Euclidean distance comparisons of the Fourier descriptors vectors	
	of 7 face profiles of the same person	46
Figure 6.1	Recognition rate versus sample population with various vector	
	sizes	48
Figure 6.2	Confusion matrix	49
Figure 6.3	Five look-alike face profiles	50
Figure 6.4	Relative Euclidean distances of one vector to the others	51
Figure 6.5	Four very different face profiles from a person's face profile	52
Figure 6.6	Recognition rate versus the number of coefficients used in the	<b>_</b>
	matching process	53

# **List of Tables**

Table 4.1	Some basic properties of Fourier coefficients	29
Table 6.7	Euclidean distance comparisons of the face profiles with angular	
	deviations	54

# Human Face Profile Recognition

#### Chapter 1. Introduction

Using computers to identify human faces has always been an attractive field to scientists and engineers. There are other person identification systems based on fingerprinting, iris-scanning, or retina-scanning available nowadays; however, none of them is more natural than identifying a person with his/her own face. This type of person identification system using the human face as the basis can be used for identification of criminals. It can also be used for authentication in secure systems. For example, it can add additional security on top of the required personal identity number (PIN) at the automatic teller machines (ATM), or it can automate the personnel check-in/check-out at the entrances of office buildings. Probably one of the most unique features to this kind of person identification systems is that the person being examined may not even be aware of the examination taking place. The person's face can be zoomed in with a video camera hidden in a place where nobody can see. This type of non-contact and non-interactive automatic person identification system is most valuable in areas of surveillance and security.

There are two types of human face images that can be used in a person identification system.<sup>(12)</sup> Of course, the most natural way to identify a person is from the frontal image (i.e. the camera is focused on the front of the face.) Many studies have been devoted to this area.<sup>(13,14,15,16)</sup> However, the frontal image of a person can be very difficult to analyze because of its complexity and variability. In addition, the amount of

information that one can extract from a frontal image can be overwhelming, which means it can take quite an amount of computing time to perform a single identification.

On the other hand, it is possible to identify a person from the face profile (i.e. the camera points towards one side of the face.) Work in this area dates back to the last century when Francis Galton proposed algorithmic techniques for quantifying normalized profile traces with characteristic lengths and angles.<sup>(9,10)</sup> Each person has a unique face profile. We often can recognize a person from the face profile with very casual inspection. We do that in many social occasions. The profile image of a person's face is easier to analyze than the frontal image since the only information that we need to process is the shape of the profile.

Researches on face profile recognition using fiducial points are commonly found.<sup>(6, 7, 8, 11)</sup> Systems with this approach use fiducial marks such as chin, nose, forehead, bridge, mouth and so forth. The distances between the fiducial points, angles between them, and areas of some triangles formed by the fudicial points are used as the features.

Aibara *et al.* used a different approach to identify face profiles.<sup>(4, 5)</sup> Fourier descriptors in the low-frequency range were shown to be useful for human face profile recognition. A correct recognition rate of 93.1% for 130 subjects had been reported.

In this thesis, a person identification system based on the face profile is implemented. Fourier descriptors are used to characterize face profile curves. The input to the system is assumed to be an image containing predominantly a human face profile. Otherwise, detection of face profiles becomes an additional problem. It is important to distinguish between face profile detection and face profile identification. In face profile detection, an algorithm is devised to search for the presence of one or more face profiles in an image. If a face is present, its size and location in the image must also be determined. This problem is fairly complex and computational intensive. On the other hand, face profile identification assumes that the profile curve of a face has been perceived, and the next step is to associate a name to the face.

This thesis concerns face profile identification only. Therefore there are some strict rules on how a person may pose in front of the camera. First of all, it is required that the size of the face profile be at least half the vertical size of the image. Secondly, the face should be upright although a little tilt is tolerable. Moreover, there should be no occlusion (i.e. glasses are not permitted.) Lastly, mouths should also be closed naturally to ensure consistency in the face profiles. To simplify the problem, it is also required that the left face profile be captured.

# Chapter 2. System Overview

Like most of the recognition systems, the human face profile recognition system has two modes of operations:

- 1) collect and store data in a database, and
- 2) match input data against stored data in a database.

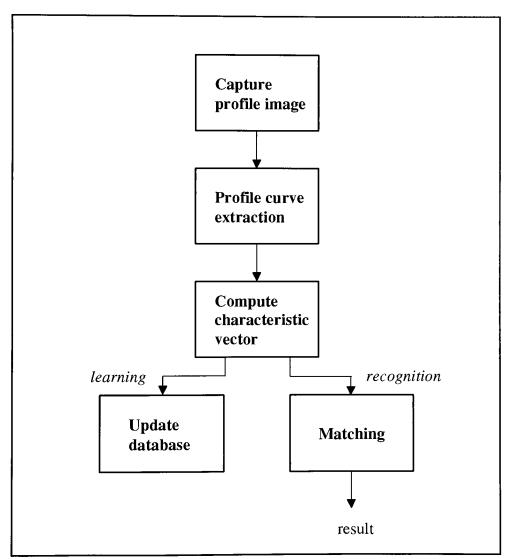


Figure 2.1 Block diagram of the human face profile recognition system.

Figure 2.1 shows the block diagram of the human face recognition system. The input data to the database are the feature vectors extracted out of the profile images captured with a CCD camera. Input data are stored in a database for future reference in the training mode, or matched against the stored data in a database in the recognition mode. The human face profile recognition is divided into five major functional units, each of which will be briefly discussed in this section.

#### A. Capture profile image

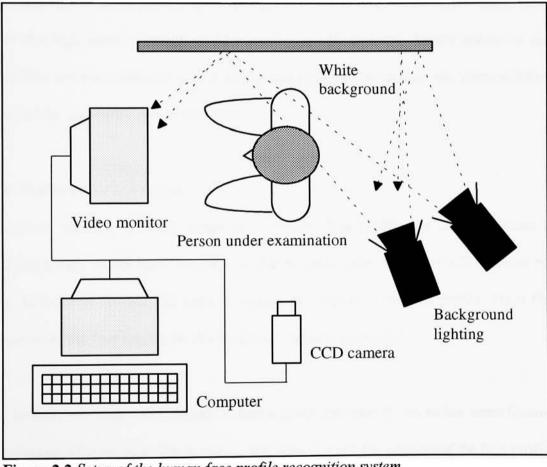


Figure 2.2 Setup of the human face profile recognition system.

The domain of our recognition system is the human face profile. Therefore, the input to the recognition system are profile images of faces. The input device for capturing images is a CCD camera. In **Figure 2.2**, the setup of the experimental human face profile recognition system is shown. The person under examination sits beside a white background. In front of the person is a video monitor which is connected to a CCD camera and a frame-grabber board resides in a computer. The CCD camera captures the profile image of the person's face. The live video image of the person's face profile provides a feedback to the person so that adjustment of the posture can be made accordingly. A set of photographic spotlights is used to brighten the white background so that high contrast images of face profiles can be obtained. Binary images of the profiles are then obtained with a simple threshold operation with the frame-grabber board for the profile curve extraction.

#### **B.** Profile curve extraction

Once we have a binary image of a person's face profile, we have to obtain a characteristic vector from the image so that all subsequent operations will be based on it. In order to do that, we have to extract the outline of the face profile. From the outline of the face profile, the characteristic vector is computed.

In order to extract the outlines of face profiles consistently, we utilize some feature points on a face profile. The feature points help to mark the position of the face profile and give clues to the size of the face profile in an image. There are six feature points that we are interested in. Among them, two are derived from the others. These feature points are:

- 1) the tip of nose,
- 2) the top of nose,
- 3) the bottom of nose,
- 4) the chin position,
- 5) the upper terminating point (derived), and
- 6) the lower terminating point (derived).

The feature point extraction relies on the general shape of human face profiles. For instance, there must be a nose protrusion and a chin protrusion in a face profile, regardless of who the person is. Once we obtain the upper and the lower terminating points, the outline of the face profile is simply the boundary curve running from one terminating point to the other. The details of the curve extraction process will be discussed in **Chapter 3. Face Profile Curve Extraction**.

#### C. Compute characteristic vector

Once the processor has determined the face profile curve of a person, it is not far from being able to compare it with others. The comparison should concern only the shape difference between two profile curves. Therefore we need to get a characteristic vector from the profile curve which can represent the curve independent of its size, position, and orientation in the image plane. We have chosen Fourier descriptors to represent face profile curves since Fourier descriptors are invariant of size, position, and orientation of any closed boundary. In our case, a face profile curve can be viewed as a line object with a closed boundary. The mathematics of Fourier descriptors and how we use them to describe open curves such as face profile curves will be discussed in detail in **Chapter 4. Fourier Descriptors**.

#### **D.** Update database

The characteristic vector, in which we use Fourier descriptors to represent face profile curves, is invariant of size, position, and orientation. Therefore it is useful for comparison purpose. As we have seen in **Figure 2.1**, there are two operations that we can perform after we obtain a characteristic vector from a face profile. We can either store the characteristic vector in a database for future reference or we can match it with the stored vectors in a database.

A database is simply a collection of characteristic vectors which represent face profiles of people. The vectors in a database are identified by names. To learn a new face profile is to merely put a new entry in the database with the person's name associated with it. Usually more than one characteristic vector are obtained from the same person in various poses. So, we take the average and store the averaged values in the database.

#### **D.** Matching

The human face profile recognition system identifies an unknown person by comparing the characteristic vector obtained from the person's face profile to the set of known vectors in a database. Distance measurement is made between the vector of the unknown person and each vector in the database. The unknown face profile is identified to be the person whose characteristic vector yields the shortest distance in the distance measurement.

The performance of the human face profile recognition system greatly depends on how the well the Fourier descriptors can resolve the differences in the face profiles of different persons. A complete analysis of the system and the test results will be presented in **Chapter 5. Analysis of the System Performance**.

#### **Chapter 3. Face Profile Curve Extraction**

Before we can perform any kind of comparisons on human faces, we have to extract the face profile curve from the image. To automate the extraction process, we have to successfully locate the upper and the lower terminating positions on the face profile. The face profile curve is then defined as the curve running from the upper to the lower terminating positions along the face-to-background boundary.

To help locate the upper and the lower terminating positions, a few feature points on the face profile must be identified. These feature points mainly identify the positions of the nose and the chin.

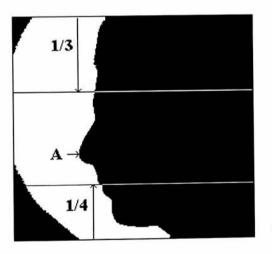


Figure 3.1 Tip of nose.

The first feature point that we identify is the *tip of nose* (**Point A** in **Figure 3.1**). Since the tip of nose is a protrusion that is most highlighted in a human face profile, we choose that as the point of reference in our face profile curve extraction. To locate the tip of nose, we search for the extreme left point in the face silhouette in a window that is 1/3 down from the top of the image and 1/4 up from the bottom of the image. The upper and the lower positions of the window are determined by examining a large number of images posed by a large number of people.

In order to successfully extract the face profiles from the face images of various sizes, some kind of distance measurement on the face profile is necessary. The nose distance, defined as the distance between the top and the bottom of nose, is used as a reference. The upper and the lower terminating positions will be determined based on this reference distance.

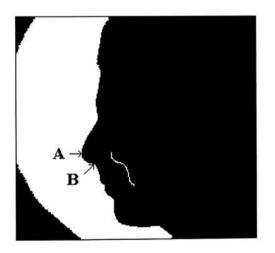


Figure 3.2 Bottom of nose.

The second feature point that we identify is the *bottom of nose* (**Point B** in **Figure 3.2**). First, a small portion of the face profile curve is extracted out from the tip of nose and downward. The extracted curve is encoded in chain code. The details of the

chain code and the curve extraction will be discussed in **Chapter 3.1 Chain Code and Curve Extraction**. For now, we have a small portion of the face profile curve. The bottom of nose is defined as the position of the first clockwise turn on the curve, starting from the tip of nose. The details of the *turning point detection algorithm* will be discussed in **Chapter 3.2 Turning Point Detection Algorithm**.

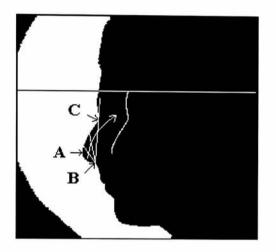


Figure 3.3 Top of nose.

The third feature point that we identify is the *top of nose* (**Point C** in **Figure 3.3**). Unlike the bottom of nose, it is not easy to use the turning point detection algorithm to locate the top of nose because of the large variation that is found in the shapes of noses and eyes of people. In the turning point detection algorithm, only the chain code of the curve is examined. The actual shape of the face profile is not taken into consideration. Therefore, a different approach is used to locate the top of nose.

Since the position of the top of nose is a point on the face profile curve above the tip of nose, we extract a portion of the face profile curve as shown in **Figure 3.3**. Then,

the line of sight from the bottom of nose to each point on the curve is examined, starting from the tip of nose. If the immediate extension of the line of sight from the bottom of nose does not belong to the background, the top of nose is said to be found. The immediate extension is a line of a few pixel units in length, an extension that is sufficient to distinguish the nose-to-background boundary and the peak at the top of nose seen from the inside of the face silhouette.

Now that we have clearly defined the position of the nose with three feature points, we can compute the distance between the top and the bottom of nose. We will use this distance, called a "reference distance", d, in our chin position definition.

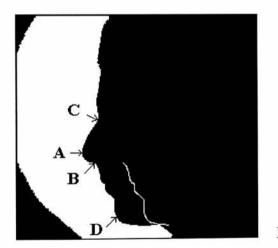


Figure 3.4 Chin position.

The next feature point that we identify is the *chin position* (**Point D** in **Figure 3.4**). For the chin position, the turning point algorithm is used along with a distance constraint. First, the curve of the face profile below the bottom of nose is extracted. Then the chin position is defined as the first counterclockwise turn on the curve that is at least a reference distance away from the bottom of nose. This distance constraint provides the discrimination against the feature positions at the lips that may be mistaken by the turning point detection algorithm.

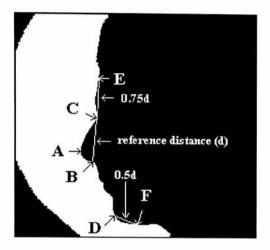


Figure 3.5 Terminating positions.

The last two points that we have to locate are the *upper and the lower terminating positions* (**Points E and F** in **Figure 3.5**). These two points are defined by the distances from the other feature points on the face profile. These distances are defined in terms of the reference distance d that we computed from positions of the top and the bottom of nose. The upper terminating position is defined as the point that is 0.75d above the top of nose and the lower terminating position is defined as the point that is 0.5d below the chin position. The ratios defined here also come from the observation of a large number of human face samples.

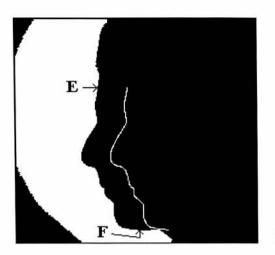


Figure 3.6 Complete face profile curve.

Finally, the complete face profile curve is extracted from the upper to the lower terminating positions as shown in **Figure 3.6**.

#### **Chapter 3.1 Chain Code and Curve Extraction**

To get the face profile curve is to get the contour of the face silhouette. The curve is encoded in chain code. Eight-connectedness is used to represent the positions of the neighboring pixels. The 8 directions are numbered in a clockwise modulo-8 fashion as shown in **Figure 3.1.1**. To minimize the noise from the rough edges that may be found in the face silhouette during the contour-tracing, a large contour-tracing mask is used. The contour-tracing mask is analogous to a ball rolling on a hill surface. The larger the ball is, the less bumpy is the resulting locus of the center of the ball. This technique is similar to the erosion operation in morphological image processing except that a constraint is applied to the movement of the mask in our case. Using a mask in contour-tracing is essentially applying a low-pass filter to the curve. All the high-frequency components such as the noisy edges are eliminated in the resulting curve. However, if we use too large a mask, the critical curvature information may also be removed. With a working image size of 512×480 pixels, the size of the contour-tracing mask is chosen to be 7 pixels in diameter. The digitized version of the contour-tracing mask is shown in Figure 3.1.2. During the contour-tracing, the contour-tracing mask is moved along the inside edge of the face silhouette. The locus of the center of the contour-tracing mask is then encoded in chain code.

NW	N	NE
0	1	2
w		Е
7		3
SW	S	SE
6	5	4

Figure 3.1.1 Eight-connectedness.

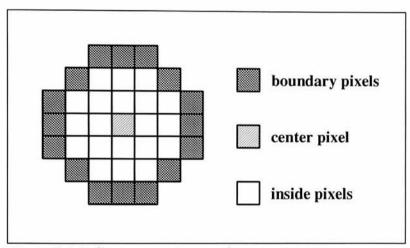


Figure 3.1.2 Contour-tracing mask.

One constraint we impose on the movement of the curve is that the maximum directional change from pixel to pixel is limited to 1 unit. In other words, the angular change from one pixel to the next pixel is limited to 45 degrees. The reason for imposing such constraint on contour-tracing is to eliminate the ambiguity found in the curve length calculation.

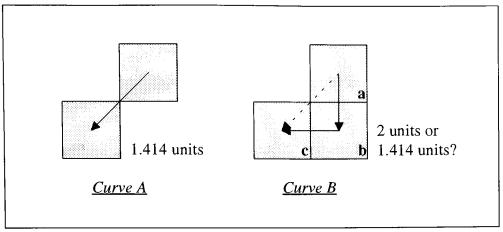


Figure 3.1.3 Ambiguity in curve length calculation.

Consider the two situations in **Figure 3.1.3**. Let the displacement from any pixel to its horizontal or vertical neighbors be 1 unit and 1.414 units (root 2) for its diagonal neighbors. The curve length for *curve* A and *curve* B are 1.414 units and 2 units respectively. *Curve* B has a directional change of 2 units from *pixel* b to *pixel* c. As we can see, *pixel* b and *pixel* c are both neighbors of *pixel* a. Therefore *curve* B could have been 1.414 units in length if it went from *pixel* a to *pixel* c directly. Since the curve length directly affects how we sample the curve, we simply impose a constraint to limit the movement from pixel to pixel. The constraint allows the contour-tracing mask to be moved in 3 directions only. They are:

- 1) 45 degrees clockwise,
- 2) no change in direction since the last movement, and
- 3) 45 degrees counterclockwise.

The decision on the movement from pixel to pixel is made by examining the 3 possible moves in the order listed above. The order of the examination is important because it ensures that the contour-tracing mask always leans towards the edge of the face silhouette. The path of a move is considered clear when none of the edge pixels of the contour-tracing mask overlaps with the background. When the path of a move being examined is clear, the position of the current pixel is advanced with that move. If none of the 3 possible moves is legal, the position of the current pixel is advanced with that move #3 (45 degree away from the edge.) This condition may occur with a right-angled turn in a face silhouette. An example is shown in **Figure 3.1.4**. The right-angled turn in a face silhouette does not cause any problem in contour-tracing. However, it should be handled properly. It should also be mentioned that it is a rare case since a human face profile is usually smooth. A sharp angle on a face profile like this is very unlikely to exist.

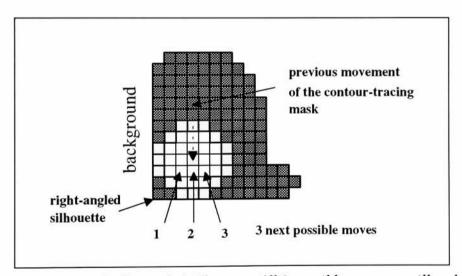


Figure 3.1.4 Right-angled silhouette. All 3 possible moves are illegal.

# **Chapter 3.2 Turning Point Detection Algorithm**

The bottom of nose and the chin position are located with the *turning point detection algorithm*. The algorithm is to apply a series of filters to the chain code of a curve and find the points on the curve which represent the points of inflections. The filtering process is shown in the block diagram of **Figure 3.2.1**.

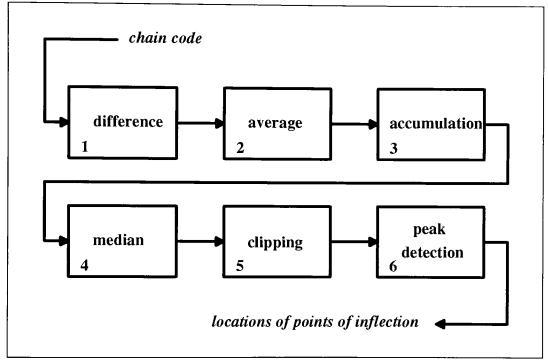


Figure 3.2.1 Turning point detection algorithm block diagram.

There are 6 steps in the process. The chain code is encoded with 8 allowable directions as discussed in **Chapter 3.1 Chain Code and Curve Extraction**. The directions are numbered in a modulo-8 fashion so that the incremental change in direction from pixel to pixel can be computed. To simplify the chain encoding and eliminate the ambiguity

found in the curve length calculation, the maximum directional displacement cannot be greater than 1 unit between neighboring pixels.

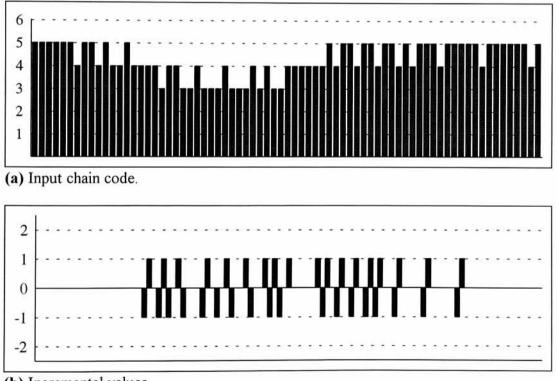
In **Figure 3.2.1**, the first filter converts the chain code to its incremental values. The output is then passed to an averaging filter. The averaging filter performs a convolution on the input data with a kernel of all 1's. Only the portion of the curve with a large net directional change would yield a large magnitude. The choice of the kernel size is dependent upon the feature size on the curve. A kernel size of 7 is sufficient for our case. The output of the averaging filter is passed to an accumulation filter which is essentially the same filter as the averaging filter but with a larger kernel size. The accumulation filter generates large magnitudes at positions in which the curve has apparent directional changes. The next two filters, the median and the clipping filters, are used to process the data before the peak detection.

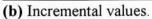
In the accumulation filter, a small kernel is used when the target is a sharp turning position. In the case of detecting the bottom of nose, a kernel size of 11 is used. When the filter is used to detect a slow turn on the curve, a larger kernel is used. In the case of detecting the chin position, a kernel size of 31 is used. The reason for using a larger kernel for a slower turn is obvious. If a small kernel was used on a slow turn, each segment of the curve as seen by the kernel would become nearly a straight line and it would be impossible to detect the turn.

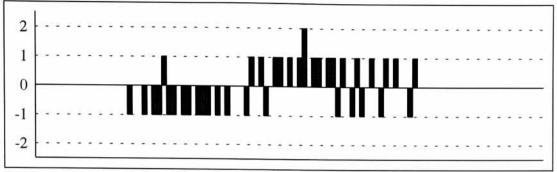
In Figure 3.2.2, we are trying to detect and locate the bottom of nose. The intermediate output for each stage of the filtering process is shown in Figure 3.2.3.



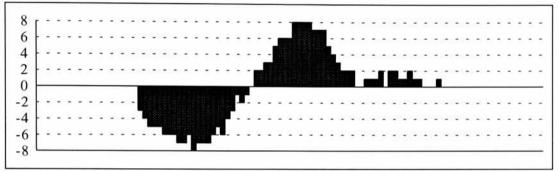
Figure 3.2.2 Bottom of nose.



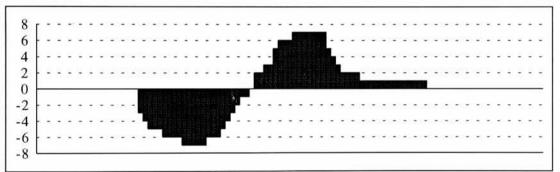




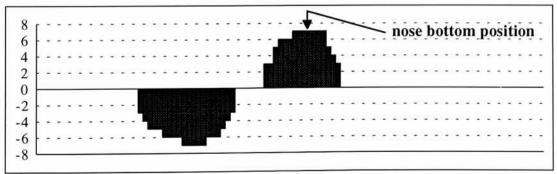
(c) Average. (kernel size = 7)



(d) Accumulation. (kernel size = 11)



(e) Median. (filter size = 11)



(f) Clipping. (threshold value = 3)

**Figure 3.2.3** Using the turning point detection algorithm to locate the bottom of nose.

### **Chapter 4. Fourier Descriptors**

After we obtain the outline of the face profile of a person in the form of chain code, we compute a characteristic vector of the profile curve using Fourier descriptors.

There are many ways to define Fourier descriptors that represent closed curve contour functions. Two of them were reviewed by Persoon and Fu.<sup>(1)</sup> The Fourier descriptors presented by Zahn and Roskies are based on the function of arc length by the accumulated change in direction of the curve since the starting point.<sup>(2)</sup> Granlund defined the Fourier descriptors in the complex space that is immediately related to the Cartesian image plane.<sup>(3)</sup> The Fourier descriptors used in the human face profile recognition system is based on it. The following represents the mathematical considerations of the technique used by Granlund.

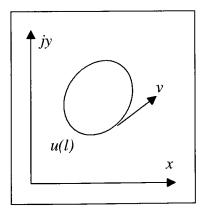


Figure 4.1 A contour function in a complex space.

A contour function, a closed-curve C, is included in a complex space as shown in Figure 4.1. A complex-valued function u is generated by moving a point around the contour. Assume that the point is moving at a constant speed along C. Let the parameter of length covered by the movement of the point at every time t be l. Then the complex function is represented by

$$u(l) = x(l) + jy(l).$$

Let the total arc length of the closed-curve C be L and let the complex function u be periodic with period L. Now the complex function u can be expressed as a Fourier series. The Fourier coefficients become

$$a_n = \frac{1}{L} \int_0^L u(l) e^{-jn2\pi l/L} dl$$

and

$$u(l) = \sum_{n \to \infty}^{\infty} a_n e^{jn2\pi l/L}$$

For simplicity, we let  $L = 2\pi$ . Then the formulas become

$$a_n = \frac{1}{2\pi} \int_0^{2\pi} u(l) e^{-jnl} dl$$

and

$$u(l)=\sum_{n=-\infty}^{\infty}a_{n}e^{jnl}.$$

The Fourier coefficients here are not unique for a specific contour. They are dependent upon the starting position. We are also interested in the effect on the Fourier coefficients when the contour undergoes translation, rotation, and dilation.

#### A. Starting position

There is a set of Fourier coefficients for each starting position of the contour-tracking. That means that there is a set of Fourier coefficients for each  $\delta$  of the function

$$u = u(l + \delta)$$
.

We now assume that there exists a function

$$u(l)=u^{(0)}(l),$$

and let  $a^{(0)}$  be the set of Fourier coefficients of this specific contour function. All the other functions are given by

$$u(l) = u^{(0)}(l+\delta).$$

The resulting Fourier coefficients become

$$a_{n} = \frac{1}{2\pi} \int_{0}^{2\pi} u^{(0)}(l+\delta)e^{-jnl}dl$$
$$= \frac{1}{2\pi} \int_{0}^{2\pi} u^{(0)}(l)e^{-jn(l-\delta)}dl$$
$$= \frac{1}{2\pi} \int_{0}^{2\pi} u^{(0)}(l)e^{-jnl}e^{jn\delta}dl$$
$$= e^{jn\delta}a_{n}^{(0)}.$$

Therefore, the Fourier coefficients differ from that of the specific contour function by a factor of  $e^{in\delta}$ .

#### **B.** Translation

When the specific contour function is translated with a complex vector Z, it becomes

$$u(l)=u^{(0)}(l)+Z_{\perp}$$

The Fourier coefficients then become

$$a_{n} = \frac{1}{2\pi} \int_{0}^{2\pi} [u^{(0)}(l) + Z] e^{-jnl} dl$$
  
$$= \frac{1}{2\pi} \int_{0}^{2\pi} u^{(0)}(l) e^{-jnl} dl + \frac{1}{2\pi} \int_{0}^{2\pi} Z e^{-jnl} dl$$
  
$$= a_{n}^{(0)} \qquad \text{for } n \neq 0, \text{ or}$$
  
$$= a_{n}^{(0)} + Z \qquad \text{for } n = 0.$$

Therefore, all coefficients except  $a_0$  are invariant of translation.

#### C. Rotation

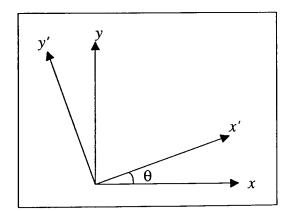


Figure 4.2 A rotated coordinate system.

In Figure 4.2, the original coordinate system is rotated counterclockwise by  $\theta$ . From the elementary trigonometry, the new and old axes are related by the equations

$$x' = x \cos \theta + y \sin \theta$$
$$y' = -x \sin \theta + y \cos \theta$$

or in matrix format,

$$\begin{bmatrix} x'\\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta\\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x\\ y \end{bmatrix}.$$

The 2×2 matrix that transforms the original coordinate system to the new system can be written as a complex vector  $e^{j\theta}$  if the image plane is viewed as a complex space. Therefore, when the specific contour function is rotated counterclockwise by  $\theta$  in the complex image plane, the contour function become

$$u(l) = e^{-j\theta} u^{(0)}(l),$$

and hence the Fourier coefficients become

$$a_n = e^{-j\theta} a_n^{(0)}$$

#### **D.** Dilation (Scaling)

The size of the specific contour can be scaled with a factor R. Similarly, it can be shown that the Fourier descriptors are simply multiplied with R.

$$a_n = Ra_n^{(0)}$$

#### General form of Fourier coefficients

As a result, the general form of the Fourier coefficients generated by translation, rotation, dilation, and changes in the starting position can be expressed as

$$a_n = e^{-jn\delta} \cdot e^{-j\theta} \cdot R \cdot a_n^{(0)} + Z\delta(n),$$

where  $\delta(n)$  is a delta function.

 Table 4.1 summarizes the Fourier coefficients for a contour function that undergoes

 all possible geometric transformations and changes in starting position.

Transformation	Contour function	Fourier coefficients
Translation	$u(l) = u^{(0)}(l) + Z$	$a_n = a_n^{(0)} + Z\delta(n)$
Rotation	$u(l) = e^{-j\theta}u^{(0)}(l)$	$a_n = e^{-j\theta} a_n^{(0)}$
Dilation	$u(l) = Ru^{(0)}(l)$	$a_n = Ra_n^{(0)}$
Starting position	$u(l) = u^{(0)}(l+\delta)$	$a_n = e^{jn\delta} a_n^{(0)}$

**Table 4.1** Some basic properties of Fourier coefficients.

In order to make the Fourier coefficients useful in shape discrimination, the coefficients of a contour function must be independent of translation, rotation, dilation, and the starting position. Consider the following set of coefficients, derived from the Fourier coefficients in their general form  $a_n$ :

$$b_{n} = (a_{1+n}a_{1-n})/a_{1}^{2}$$
  
=  $\left[a_{1+n}^{(0)}e^{j(1+n)\delta}Re^{-j\theta} \cdot a_{1-n}^{(0)}e^{j(1-n)\delta}Re^{-j\theta}\right]/\left[a_{1}^{(0)}e^{j\delta}Re^{-j\theta}\right]^{2}$   
=  $\left[a_{1+n}^{(0)}a_{1-n}^{(0)}\right]/\left[a_{1}^{(0)}\right]^{2}$  for  $n \neq 1$ .

Both sets of coefficients are complex numbers. The difference in these two sets of coefficients is that the new coefficients of the contour function are invariant of the

starting position, rotation, and dilation. Because the new coefficients do not contain  $a_0$ , they are also independent of translation.

#### Fourier descriptors for open curves

In order to use the Fourier descriptors discussed above on open curves, we trace the line pattern once and then retrace it so that a closed boundary is obtained. An example is shown in **Figure 4.3**.

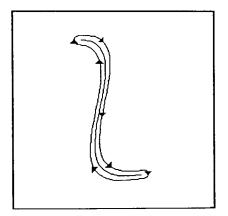


Figure 4.3 Trace the line pattern once and retrace it.

Since the curve is an open curve, the positions of the two terminating points are known. We can define the Fourier descriptors based on the magnitudes of the general Fourier coefficients. We have

$$c_{n} = |a_{n+1}|/|a_{1}|$$

$$= \left[ \left| a_{n+1}^{(0)} \right| \cdot \left| e^{j(n+1)\delta} \right| \cdot R \cdot \left| e^{-j\theta} \right| \right] / \left[ \left| a_{1}^{(0)} \right| \cdot \left| e^{j\delta} \right| \cdot R \cdot \left| e^{-j\theta} \right| \right]$$

$$= \left[ \left| a_{n+1}^{(0)} \right| \cdot \left| e^{j(n+1)\delta} \right| \right] / \left[ \left| a_{1}^{(0)} \right| \cdot \left| e^{j\delta} \right| \right] \quad \text{for } n \ge 0$$

The resulting coefficients are independent of translation, rotation, and scaling. Although the coefficients are still sensitive to the starting position, we can let the starting position be one of the terminating points of the open curve.

#### Implementation

The mathematics of the Fourier descriptors we discussed so far assumed the contour function was continuous in space. The Fourier transform we applied was the Fourier Series. In reality, the curve is represented by a discrete number of points in the complex image plane. Since we trace the curve and retrace it back to the starting point during the curve sampling, the sample values represent a periodic discrete-time signal. Therefore, we can express the signal in a discrete-time Fourier series. Obviously, the sample points of a boundary curve should be evenly spaced in curve length. The details of the curve sampling will be discussed in **Chapter 4.1 Curve Sampling**.

Let u(k) be a complex-valued function that represents the coordinates of the points on the boundary curve of a line object sampled at a fixed arc length. Let N be the number of data points in u(k). The discrete-time Fourier series becomes

$$a(k) = \frac{1}{N} \sum_{n=0}^{N-1} u(n) e^{-j2\pi k n/N} \quad \text{for } 0 \le k < N,$$

and

$$u(k) = \sum_{n=0}^{N-1} a(n)e^{2\pi k n/N}$$
 for  $0 \le k < N$ 

The normalized Fourier coefficients are then given by

$$c_k = |a_{k+1}|/|a_1|$$

Due to the symmetry in the Fourier coefficients, that is

$$|a_k| = |a_{N-k}|$$

there are N/2 normalized Fourier coefficients for an N-point transformation. So, the Fourier descriptors for an open curve are given by  $c_0$  to  $c_{N/2-1}$ .

## Another way of looking at Fourier descriptors for open curves

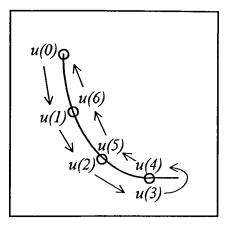


Figure 4.4 An example of a way of open curve sampling.

Consider the open curve as shown in Figure 4.4. Let u(k) be a complex-valued function that represents the coordinates of the points on the boundary curve of a line object sampled at a fixed arc length. Let N be the number of samples in u(k) and let N be an odd number as well. Moreover, let the starting point be one of the open curve terminals. Then, we have

$$u(k) = u(N-k)$$
 for  $0 < k < N$ .

The discrete-time Fourier series becomes

$$\begin{split} a(k) &= \frac{1}{N} \sum_{n=0}^{N-1} u(n) e^{-j2\pi kn/N} \\ &= \left[ u(0) + u(1) e^{-j2\pi k/N} + u(2) e^{-j2\pi (2k/N)} + \dots + u(\frac{N-1}{2}) e^{-j2\pi (\frac{N-1}{2})k/N} \right. \\ &+ u(N-1) e^{-j2\pi (N-1)k/N} + u(N-2) e^{-j2\pi (N-2)k/N} + \dots + u(\frac{N-1}{2}+1) e^{-j2\pi (\frac{N-1}{2}+1)k/N} \right] / N \\ &= \left[ u(0) + u(1) e^{-j2\pi k/N} + u(2) e^{-j2\pi (2k/N)} + \dots + u(\frac{N-1}{2}) e^{-j2\pi (\frac{N-1}{2})k/N} \right. \\ &+ u(1) e^{+j2\pi k/N} + u(2) e^{+j2\pi (2k/N)} + \dots + u(\frac{N-1}{2}) e^{+j2\pi (\frac{N-1}{2})k/N} \right] / N \\ &= \left[ u(0) + 2u(1) \cos(2\pi k/N) + 2u(2) \cos(2\pi 2k/N) + \dots \right. \\ &+ 2u(\frac{N-1}{2}) \cos(2\pi (\frac{N-1}{2})k/N) \right] / N \end{split}$$

The formula implies that the discrete-time Fourier series of the closed boundary can be computed without explicitly closing the open curve. If we separate the real part and the imaginary part of the complex-valued function, u(k), the Fourier transformation changes from a discrete-time Fourier series to a discrete Fourier transform (DFT). The Fourier coefficients become

$$\operatorname{Real}[a(k)] = \left\{ 2 \operatorname{Real} \left\{ \sum_{n=0}^{(N-1)/2} \operatorname{Real}[u(n)]e^{-j2\pi k n/N} \right\} - \operatorname{Real}[u(0)] \right\} / N,$$
$$\operatorname{Imag}[a(k)] = \left\{ 2 \operatorname{Real} \left\{ \sum_{n=0}^{(N-1)/2} \operatorname{Imag}[u(n)]e^{-j2\pi k n/N} \right\} - \operatorname{Imag}[u(0)] \right\} / N.$$

The equations show that the Fourier coefficients can be computed with the sample points of an open curve. In order to use discrete Fourier transform in our computation, the input signal to the transformation must be in real numbers. Therefore, it is necessary to separate the complex-valued function, u(k), into its x and y components.

We have seen two ways of computing Fourier descriptors for open curves. In the human face profile recognition system, we choose the first method in our implementation. By explicitly closing an open curve, the Fourier transformation is straight forward since it is rather natural to represent the sample points in complex values.

## Chapter 4.1 Curve Sampling

After the face profile curve extraction, the face profile curve is represented by a chain code. The characteristic vector of the face profile curve is then obtained using Fourier descriptors. The input signal to the discrete-time Fourier series transformation is a complex-valued function. It represents the periodic sequence of the locations of the sample points of a face profile curve. The open curve is viewed as a line object which has a boundary curve that can be traced and retraced back to the starting position. Therefore, the boundary curve is a function of (x, y) coordinates and its independent variable is the arc length since the starting position. To obtain a uniform sampling of the boundary curve, we have to sample the curve at a fixed arc length interval. The arc length interval is equal to the perimeter of the boundary curve divided by the number of sample points. Obviously, the perimeter of the boundary curve is 2 times the arc length of the open curve. Since the open curve that we obtained is represented by a sequence of pixels, the arc length of the open curve is approximated by summing the incremental distances from one pixel to the next pixel, from one end to the other end of the open curve. The incremental distance of the horizontal moves and the vertical moves are counted as 1 pixel unit while the diagonal moves are counted as 1.414 pixel units (root 2). To retain the fidelity of the original curve, all sample points are linearly interpolated from the original pixel coordinates of the open curve.

The positions of the sample points closest to the terminals of the open curve can affect the overall layout of the samples. Consider the three different sets of samples of the same curve in **Figure 4.1.1**. In all three cases, the sampling arc length interval is *l*. In cases (a) and (b), the sample points of the retrace overlap with those of the first trace. In case (c), the starting point is neither at the terminal of the open curve nor half way of the sampling interval from the terminal of the curve. As a result, the sampling points in the first trace do not overlap with those of the retrace.

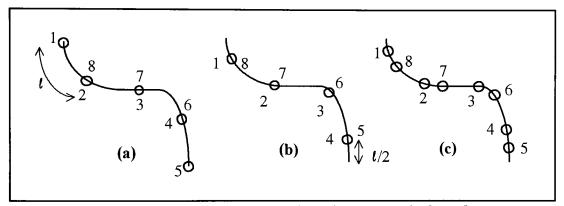


Figure 4.1.1 Different ways of sampling the boundary curve of a line object.

Obviously, it is easier to sample the boundary curve with cases (a) and (b) as shown in **Figure 4.1.1**. Sample points of the retrace of the curve can be duplicated from those obtained in the first trace.

In the human face profile recognition system, the open curves of face profiles are sampled as in case (a) of Figure 4.1.1. The total number of points that we obtain from the boundary curve is a power of 2. In that case, we can use the fast Fourier transform

to calculate our Fourier coefficients. The arc length interval for the curve sampling is therefore the total length of the open curve divided by  $2^{n-1}$  where  $2^n$  is the number of sample points on the boundary curve.

#### Chapter 5. Matching

In the human face profile recognition system, matching is the final process that determines the identity of an unknown person. Preceding the matching process, a Fourier descriptors vector is computed from the face profile of the person under examination. This vector is called a *test vector*, from which a distance measurement is made to each vector (*template vector*) in a database. A match is said to be found when the shortest distance falls below a certain threshold value. The threshold value is a maximum allowable distance for the system to consider a match. It is used to discriminate against people that are not registered in the database.

To quantify the difference between the Fourier descriptors vectors of the face profiles of two persons, we use Euclidean distance measurement. The m-value Euclidean distance between two n-dimensional vectors is given by

$$d_m = \sum_{j=0}^{m-1} \left[ v_1(j) - v_2(j) \right]^2,$$

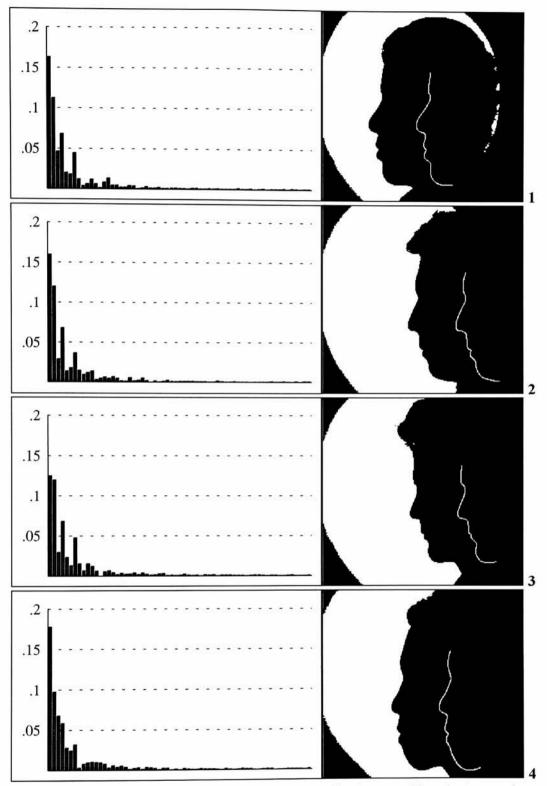
where  $m \le n$  and n is the size of vectors  $v_1$  and  $v_2$ . The smaller the Euclidean distance is, the closer the two vectors are in the *n*-dimensional space. To ensure that the Fourier descriptors can truly represent a person's face profile, the Euclidean distance between the Fourier descriptors vectors of two persons should be large. In other words, there should be a significant difference between the two Fourier descriptors vectors. In the following, we will examine the differences in the Fourier descriptors vectors of people's face profiles as well as the consistency in the Fourier descriptors vectors of the face profiles of the same person. The size of the Fourier descriptors vector, of course, affects the effectiveness of Fourier descriptors in describing human face profiles. For now, we will use a vector size of 64 coefficients for the following tests. Formal evaluations on how the vector size affects the performance of the system will be presented in **Chapter 6. Analysis of the System Performance**.

#### A. Differences in the Fourier descriptors vectors of people's face profiles

Figure 5.1 shows 4 face profiles and their Fourier descriptors vectors. The size of the Fourier descriptors vector is 64. (i.e. 65 points are sampled from the open curve.) The Euclidean distances between the vectors are shown in Figure 5.2. There are 4 tables in Figure 5.2. Different numbers of values in the Fourier descriptors vectors are used in the Euclidean distance calculation. In an *m*-value Euclidean distance calculation, only the first *m* values of the Fourier descriptors vectors are used. Since the profile of the Fourier descriptors vector diminishes as the index increases, the *m*-value Euclidean distance between two vectors converges as *m* increases. For this reason, it is not necessary to use all the values in the Fourier descriptors vector for our Euclidean distance calculation. We can see that the Euclidean distances increase by a fair amount when 16 values are used in the calculation instead of 8. However, the distances do not increase as much when the calculation is switched from 16 to 32

values and from 32 to 64 values. Using only a small fraction of the Fourier descriptors vector for the Euclidean distance calculation also reduces the computation time in the matching process. The effect will become more noticeable if a large database is used.

In **Table (b)** of **Figure 5.2**, a typical value of the Euclidean distance between the Fourier descriptors vectors of two persons is about  $1000 \times 10^{-6}$ . We can choose our threshold value based on this number.



**Figure 5.1** Fourier descriptors vectors of size 64 of 4 face profiles. ( $c_0$  is not shown, which is equal to 1.0.)

Vectors	vI	v2	v3	v4
v1		469	1,857	1,238
v2	469		1,447	2,659
<i>v3</i>	1,857	1,447		5,223
v4	1,238	2,659	5,223	

(a) 8 values are used in the calculations.

Vectors	vI	v2	v3	v4
νI		648	2,013	1,532
v2	648		1,506	2,889
v3	2,013	1,506		5,531
v4	1,532	2,889	5,531	

(b) 16 values are used in the calculations.

Vectors	v1	v2	v3	v4
v1		688	2,035	1,576
v2	688		1,554	2,965
v3	2,035	1,554		5,570
v4	1,576	2,965	5,570	

(c) 32 values are used in the calculations.

Vectors	v1	v2	v3	v4
v1		693	2,041	1,584
v2	693		1,561	2,973
v3	2,041	1,561		5,577
v4	1,584	2,973	5,577	

(d) 64 values are used in calculations.

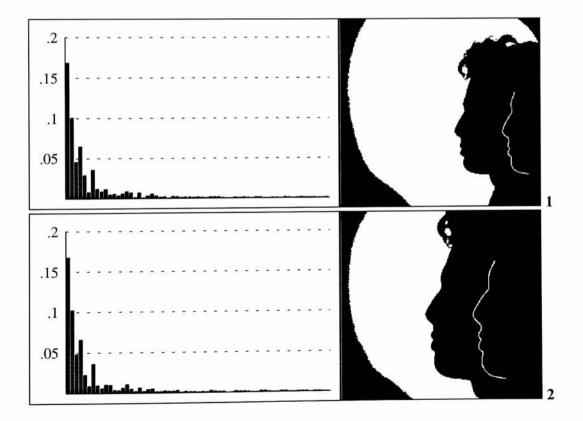
**Figure 5.2** Euclidean distances of 4 Fourier descriptors vectors of size 64. Various numbers of coefficients are used in the calculations. (Values are in  $10^{-6}$ .)

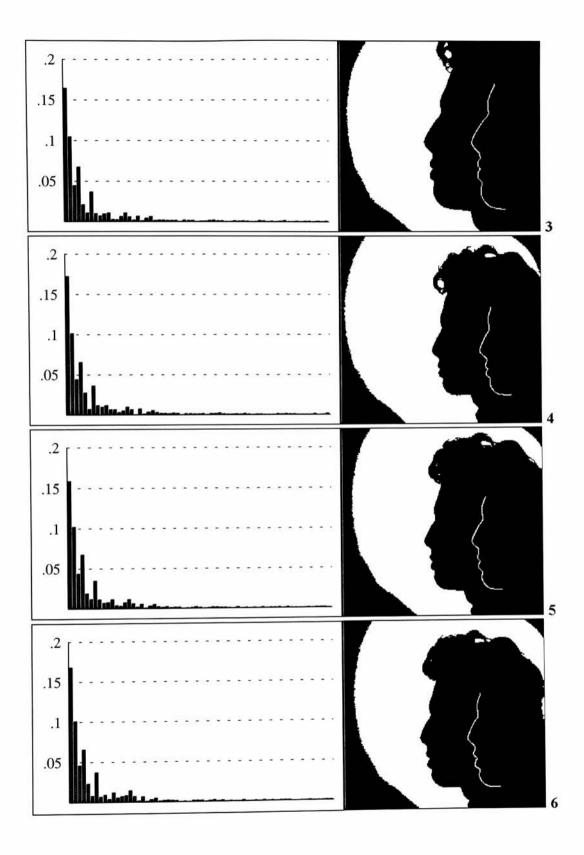
# B. Consistency in the Fourier descriptors vectors of the face profiles of the same person

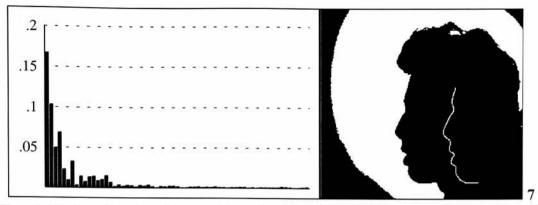
In order to show that Fourier descriptors can be used as the basis for identifying human face profiles, consistent results must be obtained from different images of the same person. In other words, small Euclidean distances must be obtained among different images of the same person.

Figure 5.3 shows 7 face profiles of the same person and their corresponding Fourier descriptors vectors. In images 1, 2, 3, and 4, the person was asked to keep his mouth closed naturally. In images 5, 6, and 7, the person was asked to open his mouth gradually. The Euclidean distances between the vectors are shown in Figure 5.4. The size of the Fourier descriptors vector is 64. In Table (a) of Figure 5.4, 16 values are used in the Euclidean distance calculations, while in Table (b), 64 values are used. Once again, we can see that the Euclidean distances do not increase as much when the number of values used in the distance calculation switched from 16 values to 64 values. Further investigation on this respect will be detailed in Chapter 6. Analysis of System Performance. Now, recall from the distance comparison of the Fourier descriptors vectors of different people, the typical value of the Euclidean distance of a 16-value comparison using Fourier descriptors of size 64 is 1000×10<sup>-6</sup>. If we choose  $300 \times 10^{-6}$  as our threshold value, images 1-4 will all fall below this value when comparing with one another. The Euclidean distances from the vectors of images 5, 6, and 7 to the vectors of images 1 to 4 are larger since there are subtle changes in the face profiles when the person's mouth is allowed to open widely. With the open mouth, the curve length of the face profile becomes longer. Since a fixed number of points are sampled from the face profile curve for the vector calculation, the positions of the sampled points are spread out more on the profile curve, causing the Fourier descriptors vector to vary slightly.

In **Figure 5.4**, we can see that there is certainly a consistency in the Fourier descriptors vectors of the face profiles of the same person with the mouth closed naturally. The open mouth images have some impact on the resulting Fourier descriptors, depending on the degree of openness. Therefore, the Euclidean distances from these vectors to the vectors of the naturally posed images are slightly larger.







**Figure 5.3** Fourier descriptors of size 64 of the face profiles of the same person. ( $c_0$  is not shown, which is equal to 1.0.) The person's mouth is open in pictures 5, 6, and 7. Picture 5 is the least open and picture 7 is the most open.

Vectors	v1	v2	v3	v4	v5	v6	v7
vI		101	178	31	306	224	464
v2	101		44	109	147	112	341
v3	178	44		189	59	141	386
v4	31	109	189		352	199	415
v5	306	147	59	352		206	424
v6	224	112	141	199	206		184
v7	464	341	386	415	424	184	

(a) 16 values are used in the calculations.

Vectors	vI	v2	v3	v4	v5	v6	v7
vI		122	192	39	317	243	508
v2	122		52	131	163	132	376
v3	192	52		206	71	159	428
v4	39	131	206		370	218	461
v5	317	163	71	370		217	455
v6	243	132	370	218	217		209
v7	508	376	218	461	455	209	

(b) 64 values are used in the calculations.

**Figure 5.4** Euclidean distances between the Fourier descriptors vectors of the face profiles of the same person. Vectors v5, v6, and v7 are computed from images with the person's mouth open. 16 and 64 values are used in the calculations. (Values are in  $10^{-6}$ .)

## Chapter 6. Analysis of the System Performance

The correct recognition rate of the human face profile recognition system decreases as the sample population increases. Besides that, there are other parameters that can affect the performance of the system as well. The two most important ones are the size of the Fourier descriptors vector and the number of coefficients in the vector that are used in the matching process.

#### A. Sample population and vector size

We begin our investigation by seeing how the sample population affects the recognition rate. At the same time, we investigate the effect of the size of the Fourier descriptors vector on the system performance. First, 24 databases are constructed based on the images obtained from 60 people. Each database is characterized by the number of samples it contains and the size of the Fourier descriptors vector representing each sample. Each vector in the databases is constructed using 3 different images of the same person. To test the system, 4 different images of the same person, which differ from those used to construct the databases, are presented to the system for each corresponding entry in each database. The results of the tests are plotted in **Figure 6.1**.

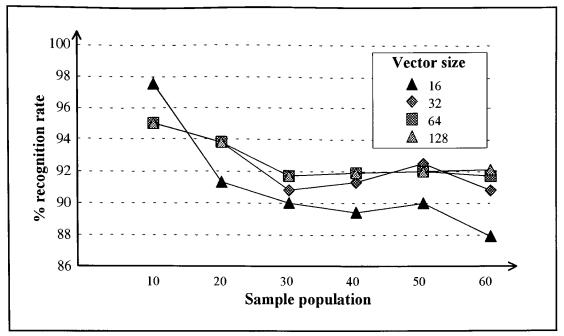


Figure 6.1 Recognition rate versus sample population with various vector sizes.

As expected, the correct recognition rate decreases as the sample population increases. Moreover, the larger the vector size, the better the system performance with large sample populations. The recognition rate also seems to converge as the vector size increases. As we can see, there is not a big difference in the system performance between vector sizes 64 and 128.

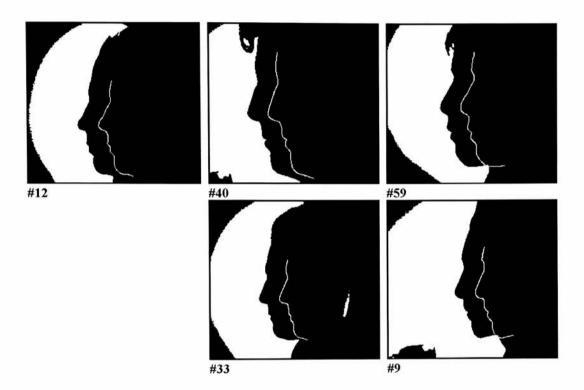
Now, let us take a close examination of the above tests. Specifically, we choose the test with the vector size of 64 coefficients and the sample population of 60 people. In **Figure 6.2**, the confusion matrix of the test results is shown.

	n	0	0 r																	~			~			(	0	u	t	p	u	t																									
	ŏ	ĩ	0 0 2 3	4	1 5	5 6	7	8	9			2	3	4	15	6	7	8	9	0	1	2	3	4	5	6	7	8	2 9	э: 0:	3 3	3 3	33 34		33	5 7	33 76	33	4	14	4	4	4	4 5	4 6	47	4 8	4 9	5 0	5 1	5 2	5 3	5 ! 4 !	5 ! 5	55	5 7	5 6
01	1	2																					1																																		
	·		4 .	. •	•	•		•												•																																					:
1	·	•	. 4	٠.																																																					
5	·	•	• •	4																																																					
6	:		: :			```																																																			
7							4		•		•	•		:		•	•	•	•	•	•	•	•	•	•	•	•	·	•	•	•	•	• •	• •	•	•		•	•	•	•			٠	٠	•	•	·	•	•	•	·	•				
8								4				÷		:	÷	÷			2	:		:	1	÷.			•	•	•	•	•	•	• •		•	•	• •	•	•	•	•	•	•	•	•	•	٠	·	·	٠	·	•	• •	• •	• •	•	
	•								3	ι.													2	2	:	:	:				•	•	•		•		•	• •	•			•	•	·	•	•	•	•	·	•	·	•	•	• •	• •	•	
10	٠	•			•	•		•		4	ι.			•																																											
11	•	•	• •	•	•	•	•	•	•		4	•	•																																												ļ
12	•	•				•					•	2	1	•						•																			1																		
4			: :								•		4	:	•	·	•	•		•		•																																			
5	•	•									:			4		•	•	•	•	•			•	•																																	
6	:	:		•		•	•	•	•		•	•		•	3	÷	•	•	•	•	•	•	•	•	•	•	•	•	•	• •	•	•			•	•		•	•	•	•	1		-							•	•					•
7	÷	÷.							:			:					à	•	•	•	•	•	•	•	•	•	:	•	•	•	•	•	• •	•	•	•	• •	•	•	•	•	•	•	٠	•	٠	•	•	•	·	•	•	• •		•		•
8																÷		ŝ	2	:		:	1	:	:	•	:	•	•	•	•		• •			•	• •	•	•	•	•	•	•	•	•	•	·	•	•	•	•	•	• •	• •	•	•	•
9					•														4				÷		:	:	÷	:	:							•	• •		•	•	•	•	•	•	•	•	•	•	•	·	•	•	• •		•	•	1
	•																			з																:						Ċ	•	1								: '					1
	•													•							4																						÷			÷		2	÷	:	:						ļ
	÷	•												•				•	•	•		4					•																														
3	2	•	• •	•	•	•	•	•	•		•	•	•	:	•	•	•	٠	۰.	٠.	•	٠	2	•	•	•	•	•	•	•	•							•																			,
	:		: :																					4	2	·	•	•	•	•	•	• •		•	•	•	•	•		•	•	•	•	•	•	•			·	•	•	•			•		,
6	2	:																							4	÷	•	•	•	•	•	• •			•	•	•	•	•	•	•	•	٠	·	•	٠	·	٠	·	·	•	•	• :	•	•		
7												:			:	•			•	•	•	•	•	•	•	3																															
8	1						1	L.										1	:	2	÷.	:	2	:	:		۰.	i	•	•	•		• •		•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	٠.	i		•			
9		•				•								•															4																												ļ
0	•	•	• •	•		•		•	•		•			•																4																											
11	:	•	• •	•	• •	•	٠	•	•		•	•	-	-	٠	٠	•	٠	٠	٠	•	·	•	•	•	•		•	•	. 4	٩.																										
			: :		• •	• •		•	•	•	•	٠	•	•	·	•	٠	·	·	•	•	•	•	•	·	·	•	•	•	• •	. 1	٩.																				•					
4	•	•	: :	•		• •	•	•	•		•	•	•	•	•	٠	•	•	·	•	•	•	•	·	•	•	•	•	•	•	•	. 3	• •	•																		• •					
5	:	:								1					•	•		•	•	•	•	•	•	•	•	•	•	•	•	•	•	•		• ;		•																• •					
16																÷			:	:	1	:	1	2	÷.		:			•		• •			' .	•																: :					
7																											÷.									4	1																				
88		•																																			4	11																			
9	•	•	• •	•		• •	٠	•	•		•	-		•	-								•				•											4														•					
0	•	·	• •		• •	•	•	•	•		•	•	•	•	•	٠	•	٠	٠	·	•	·	•	•	•	•	•	•	•	• •	• •			•			•		4																		
2	•	•	• •		•	• •	•	•	•		•	•	•		•	·	•	•	·	•	٠	٠	·	•	•	•	•	•	•	• •	•	• •	•	•	•	•	•	•	٠	4												• •					
13		:						•		1	•	•	•		·	·	•	·	·	·	•	•	•	•	•	•	•	•	•	•	• •	• •	•	•	•	•	•	•	•	•	4	Å										: :					
44																											:																3	•								: :					
15																	2	2	2			1	1	:	:	:	:	:															3	à	÷	÷			÷.		•						Ì
6																																													4	÷		1		:	:						Ì
7	•	•						•														•			•																					4											
8	•	·												•																																	4										
19	•	•												•																																•	·	4	:	٠	•	•	• •	•	•		•
1		:																																												•	·	·	4	;	•	•	• •	•	•		•
		:	: :								•	•	•		1	:	1	1	÷	1	:	:	1	1	1		:	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	·	•	•	•	•	٠	4	Å	• •	• •	•	•		1
3																																																		:		4			:		ļ
54																																																		2		. 4					
5	•	•																									•																										. 4	۱.			•
56	•	•	. 1																								•																											3	ι.		•
			• •																																																				4		•
			: :											:																																									•	4	4
0			• •			• •	•	•	1	• •	•			:																																											
									•		•	•	•	•	•	•	•	•	•	•		•	•	•	•	•		•	•	• •				•	•			,			•																•

**Figure 6.2** Confusion matrix of the test results for the vector size of 64 coefficients and the sample population of 60 people. (Output 0 represents the unidentifiables.)

The confusion matrix shows the output of the system for the presented input. The numbers along the diagonal of the matrix represent the numbers of correct recognitions for the given input. Let us take a look at person #12. The confusion matrix shows that the system has mistaken him as another person for 2 times, one as person #40 and another one as person #59. Person #33 has also been mistaken by the system as person #40. Similarly, person #9 has also been mistaken by the system as person #59. We shall examine the face profile curves of these people and see what

similarities they possess. In Figure 6.3, the face profile curves of these people are shown.



**Figure 6.3** Look-alike face profiles. Person #12 was mistaken as persons #40 and #59. Person #33 was mistaken as person #40. Person #9 was mistaken as person #59.

At the first glance, these face profile curves may not look alike. However, they all have similar features that make the computer mistake one as the other. First of all, while we are looking at these images, we should remember that the system makes comparisons based on face profile curves only. The face silhouettes of these images obviously do not come close to one another. So, when we compare these images, we should pay more attention to the flow of the profile curves. One of the similarities that may be found among these people is the proportions of the main features on the face profile. These may include the ratio of the distance between the eye and the bottom of nose to the distance between the bottom of nose and the chin position. On the other hand, the errors made by the system can also be coming from the angular position of the face. Therefore, it is difficult to conclude what causes the faults.

In Figure 6.4, the relative distances of all the vectors to that of person #12 are shown. Four people with their vectors which are farthest away in the Euclidean space to that of person #12 are chosen for close examination. The face profile curves for these people are shown in Figure 6.5.

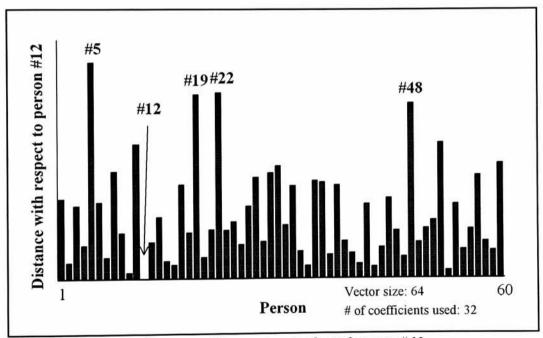
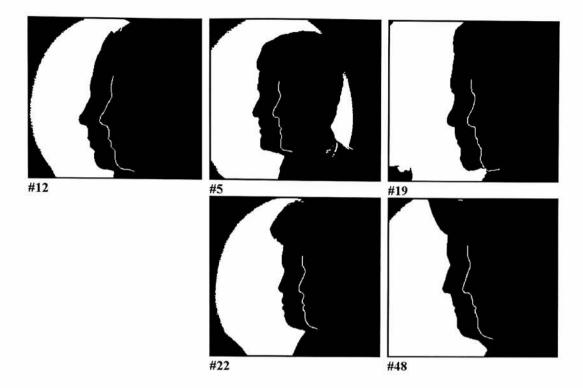


Figure 6.4 Relative distances of the vectors to that of person #12.



**Figure 6.5** Vectors of the face profiles #5, #19, #22, and #48 are far away from that of the face profile #12 in the Euclidean space. (Euclidean distances are calculated from the database of sample population of 60 people and vector size of 64 coefficients.)

Now, let us compare these face profile curves. The face profile curve of person number #12 is quite rounded compared with the others. The area covered by the curve and the straight line joining the two terminating points is obviously larger than those of the other face profile curves. Also, the eyebrow area of person #12 is not as distinctive as the rest. Person #5 has a sharp chin and person #19 has a big round chin. The chin of person #12 looks like those of persons #22 and #48. However, the noses of person #22 and #48 are flatter than that of person #12. With all these differences, there is no doubt these 4 face profile curves yield quite different Fourier descriptors vectors compared to that of person #12.

#### B. Number of coefficients used in the matching process

Another system parameter which directly affects the recognition rate is the number of coefficients in the Fourier descriptors vector that are used in the matching process. Since the profile of the Fourier descriptors vector diminishes as the index of the vector increases, the Euclidean distance between two vectors converges to a stable value as the number of coefficients used in the calculation increases. Therefore, we can find out the number of coefficients that is adequate for computing the Euclidean distance between two vectors. In **Figure 6.6**, the correct recognition rate is plotted against the number of coefficients used in the matching process. We have chosen the vector size of 64 coefficients for the test since it is most appropriate for a large sample population size as determined earlier.

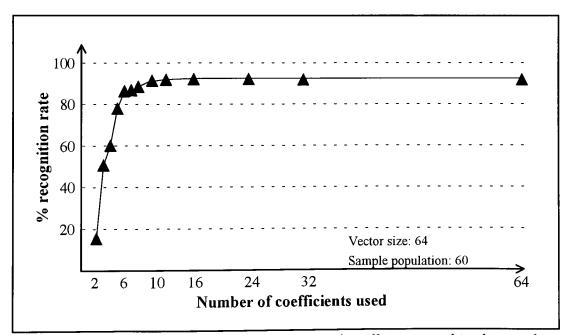


Figure 6.6 Recognition rate versus the number of coefficients used in the matching process.

As we can see, the recognition rate increases as the number of coefficients used in the matching process increases and it reaches its maximum at a quarter of the total number of coefficients in the vector. In other words, the recognition rate of the system with a vector size of 64 coefficients does not get any better with more coefficients used in the matching process than that obtained with 16 values.

#### C. Face direction

In general, the human face profile recognition system requires that the person under examination be facing absolutely perpendicular to the view of the camera. Any deviation from the perpendicular position causes subtle changes or sometimes big changes in the shape of the face profile curve, depending on person to person. In **Table 6.7**, the Euclidean distances between the deviated face profile curves and the straight face profile curves for a few people are shown.

Angle (deg.)	-10	-7.5	-5	-2.5	0	2.5	5	7.5	10
Person #1	2,170	1,186	619	827	0	128	129	71	356
Person #2	3,178	1,345	345	231	0	78	110	164	403
Person #3	2,048	809	509	574	0	109	124	289	558
Person #4	3,809	1,348	467	157	0	89	78	147	438

**Table 6.7** The Euclidean distances between the deviated face profile curves and the straight face profile curve. (The vector size is 64 and 16 values are used in the distance calculation. Values shown are in  $10^{-6}$ .)

The angular position of the straight face profiles is at zero degrees. Faces that turn towards the camera have positive angular positions and negative for the opposite direction. Notice that there is a difference between the two directions of angular deviation. The face profile curves of people that turn towards the camera are much more tolerable than those of the opposite direction. Intuitively, we think that the face profiles captured at two different directions with the same magnitude of angular deviation should be very comparable. However, as the person's head turns away from the camera, part of the face begins to block or distort some of the essential features found in the straight face profile. These include the nose and the lips. On the other hand, the general shape of the straight face profile suffers minimum distortion as the person's head turns towards the camera at a small angle since both sides of the face hardly obstruct those essential features.

#### Conclusion

The human face profile recognition system presented in this thesis has clearly shown that it is possible to construct a system to identify human individuals through their face profiles. The use of Fourier descriptors to represent face profile curves has demonstrated its practical application in this domain. Although considerable effort is still required to construct a more robust and convenient system, this thesis has confirmed the ideas and feasibility of building such systems that researchers have developed over the past two decades.

## References

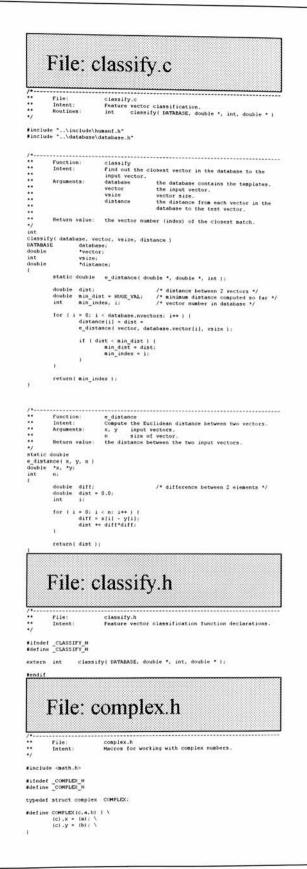
- [1] E. Persoon and K. S. Fu, "Shape discrimination using Fourier descriptors," *IEEE Trans. Systems, Man, Cybern.*, 7, 170-179 (1977).
- [2] C. T. Zahn and R. Z. Roskies, "Fourier descriptors for plane closed curves," *IEEE Trans. Computers.* **21**, 269-281 (1972).
- [3] G. H. Granlund, "Fourier preprocessing for hand print character recognition," *IEEE Trans. Computers.* **21**, 195-201 (1972).
- [4] T. Aibara, K. Ohue, and Y. Oshita, "Human face profile recognition by a P-Fourier descriptor," *Optical Engineering*. **32**(4), 861-863 (1993).
- [5] T. Aibara, K. Ohue, and Y. Matsuoka, "Human face profile recognition by a P-type Fourier descriptor," in *Visual Communications and Image Processing* '91: Image Processing, Proc. SPIE. 1606, 198-203 (1991).
- [6] C. J. Wu and J. S. Huang, "Human face profile recognition by computer," *Pattern Recognition*. **23**(3/4), 255-259 (1990).
- [7] L. D. Harmon, S. C. Kuo, P. F. Ramig and U. Raudkivi, "Identification of human face profiles by computer," *Pattern Recognition*. **10**, 301-312 (1978).
- [8] L. D. Harmon, M. K. Khan, R. Lasch and P. F. Ramig, "Machine identification of human faces," *Pattern Recognition*. 13, 97-110 (1981).
- [9] F. Galton, "Numeralized profiles for classification and description," *Nature*.
   83(2109), 31 March, 127-130 (1910).
- [10] F. Galton, "Personal identification and description," *Nature*. 21 June, 173-177 (1888); 28 June, 201-202 (1888).
- [11] L. D. Harmon and W. F. Hunt, "Automatic recognition of human face profiles," *Computer Graphics and Image Processing*. 6(2), 135-156 (1977).
- [12] A. Samal and P. A. Iyengar, "Automatic recognition and analysis of human faces and facial expressions: A survey," *Pattern Recognition*. 25(1), 65-77 (1992).

- [13] A. J. Goldstein, L. D. Harmon and A. B. Lesk, "Identification of human faces," *Proc. IEEE*. 59, 748-760 (1971).
- [14] Y. Kaya and K. Kobayashi, "A basic study on human face recognition," Frontiers of Pattern Recognition. 265-289. Academic Press, New York (1971).
- [15] K. H. Wong, H. H. M. Law and P. W. M. Tsang, "A system for recognizing human faces," Proc. ICASSP 89, 1638-1642 (1989).
- [16] R. J. Baron, "Mechanisms of human facial recognition," Int. J. Man-Mach. Stud., 15, 137-178 (1981).

## Appendix Program Listings

This appendix contains the program listings of the human face profile recognition system. The two main modules are matching.c and training.c for the identification process and the registration process respectively. The rest of the supporting modules are listed below with brief descriptions provided.

Module name	Description
classify.c	Matching process.
classify.h	
complex.h	Macros for working with complex numbers.
costable.c	Table to use for cosine and sine table lookups.
curvelib.c curvelib.h	Chain code arithematics.
database.c database.h	Database manipulations.
extract.c extract.h	Curve extraction with chain code.
feature.c feature.h	Feature detection and location.
fft.c fft.h	Fast Fourier transform.
fgrabber.c fgrabber.h	Frame-grabber board operations.
filters.c filters.h	Filters for the Turning point detection algorithm.
global.c global.h	Global variables.
humanf.h	Useful macros and definitions for the system.
image.c image.h	Image file operations for non-realtime processing.
sampling.c sampling.h	Curve sampling and display.
vector.c vector.h	Highest in the program hierarchy besides the main programs.



```
idefine COMPLEX ASSIGN(a,b) ( \
    (a) x = (b) x ; \
    (a) y = (b) y; \
    define COMPLEX_CADJ(a) (a) y = -(a) y

define COMPLEX_ADJ(c,a,b) ( \
    (c) x = (a) x + (b) x; \
    (c) x = (a) y + (b) y; \

define COMPLEX_SUB(c,a,b) ( \
    (c) x = (a) x + (b) x; \
    (c) y = (a) y + (b) y; \

define COMPLEX_SUB(c,a,b) ( \
    (c) x = (a) x + (b) x; \
    (c) y = (a) y + (b) y; \

define COMPLEX_SUB(c,a,b) ( \
    (c) x = (a) x + (b) x; \
    (c) y = (a) y + (b) y; \

define COMPLEX_SUB(c,a,b) ( \
    (c) y = (a) y + (b) y; \

define COMPLEX_SUB(c,a,b) ( \
    (c) y = (a) x + (b) x; \
    (d) define COMPLEX_SUB(c,a,b) ( \
    (c) y = (a) y + (b) y; \
    (d) define COMPLEX_SUB(c,a,b) ( \
    (d) y = (a) x + (b) x; \
    (d) define COMPLEX_SUB(c,a,b) (b) = sqct( (a) x + (a) y + (a)
```

# File: costable.c

#### double cos\_table || =

Pendit

double cos_table[] =
1.000000000000000,0.999981175282601,0.9999924701839145,0.999830581795823,
D.999698818696204.0.999529417501093.0.999322384588350.0.999077727752645.
0.998795456205172.0.998475580573295.0.998118112900149.0.997723066644192.
0.997290456678690.0.996820299291166.0.996312612182778.0.995767414467660.
0.995184726672197,0.994564570734255,0.993906970002356,0.993211949234795,
0.992479534598710,0.991709753669100,0.990902635427780,0.990058210262297,
0.989176509964781,0.988257567730750,0.987301418157858,0.986308097244599,
0.985277642388941,0.984210092386929,0.983105487431216,0.981963869109555,
0.980785280403230,0.979569765685441,0.978317370719628,0.977028142657754,
0.975702130038529,0.974339382785576,0.972939952205560,0.971503890986252,
$\begin{array}{l} 0.970031253194544, 0.968522094274417, 0.966976471044852, 0.965394441697689, \\ 0.963776065795440, 0.962121404269042, 0.960430519415566, 0.958703474895872, \end{array}$
0.956940335732209, 0.955141168305771, 0.953306040354194, 0.951435020969008,
0.949528180593037,0.947585591017741.0.945607325380521,0.943593458161960,
0.941544065183021,0.939459223602190,0.937339011912575,0.935183509938948,
0.932992798834739,0.930766961078984,0.928506080473216,0.926210242138311,
0.923879532511287.0.921514039342042.0.919113851690058.0.916679059921043.
0.914209755703531,0.911706032005430,0.909167983090522,0.906595704514915,
0.903989293123443,0.901348847046022,0.898674465693954,0.895966249756185,
0.893224301195515,0.890448723244758,0.887639620402854,0.884797096430938,
0.881921264346355,0.879012226428634,0.876070094195407,0.873094978418290,
0.870086991108712,0.867046245515693,0.863972856121587,0.860866938637767,
0.857728610000272.0.854557988365401.0.851355193105265.0.848120344803297. 0.844853565249707.0.841554977436898.0.838224705554838.0.834862874986380.
0.831469612302545,0.828045045257756,0.824589302785025,0.821102514991105,
0.817584813151584, 0.814036329705948, 0.810457198252595, 0.806847553543799,
0.803207531480645.0.799537269107905.0.795836904608884.0.792106577300212.
0.788346427626606.0.784556597155575.0.780737228572095.0.776888465673232.
0.773010453362737,0.769103337645580.0.765167265622459,0.761202385484262,
0.757208846506485,0.753186799043613,0.749136394523459,0.745057785441466,
0.740951125354959,0.736816568877370,0.732654271672413,0.728464390448225,
0.724247082951467,0.720002507961382,0.715730825283819,0.711432195745216,
0.707106781186548,0.702754744457225,0.698376249408973,0.693971460889654,
0.689540544737067,0.685083667772700,0.680600997795453,0.676092703575316,
0.671558954847018, 0.666999922303638, 0.662415777590172, 0.657806693297079, 0.653172842953777, 0.648514401022113, 0.643831542889792, 0.639124444863776,
0.634393284163646.0.629638238914927.0.624659488142387.0.620057211763289.
0.615231590580627.0.610382806276310.0.605511041404326.0.600616479383869.
0.595699304492434,0.590759701858874,0.585797857456439,0.580813958095765,
0.575808191417845,0.570780745886967,0.565731810783613,0.560661576197336,
0.555570233019602.0.550457972936605.0.545324988422047.0.540171472729893.
0.534997619887097,0.529803624686295,0.524589682678469,0.519355990165590,
0.514102744193222,0.508830142543107,0.503538383725718,0.498227666972782,
0.492898192229784.0.487550160148436.0.482183772079123.0.476799230063322.
0.471396736825998,0.465976495767966,0.460538710958240,0.455083587126344,
0.449611329654607,0.444122144570429,0.438616238538528,0.433093818853152, 0.427555093430282,0.4220082707998800,0.416429560097637,0.410843171057904,
0.405241314004990.0.399624199845647.0.393992040061048.0.388345046698826.
0.382683432365090,0.377007410216418,0.371317193951838,0.365612997804774,
D.359895036534988, 0.354163525420491, 0.348418680249435, D.342660717311994,
0.336889853392220,0.331106305759876,0.325310292162263,0.319502030816016,
0.313681740398892,0.307849640041535,0.302005949319228,0.296150888243624,
0.290284677254462,0.284407537211272,0.278519689385053,0.272621355449949,
0.266712757474898,0.260794117915276,0.254865659604515,0.248927605745720,
0.242980179903264,0.237023605994367,0.231058108280671,0.225083911359793,
0.219101240156870,0.213110319916091.0.207111376192219.0.201104634842092.
0.195090322016128,0.189068664149806,0.183039887955141,0.177004220412149,
0.170961888760301, 0.164913120489970, 0.158858143333861, 0.152797185258443, 0.146730474455362, 0.140658239332849, 0.134580708507126, 0.128498110793793,
0.122410675199216.0.116318630911905.0.110222207293883.0.104121633872055.
0.098017140329561,0.091908956497133.0.085797312344440,0.079682437971430,
0.073564563599667,0.067443919563664,0.061320736302209,0.055195244349690,
0.049067674327418,0.042938256934941,0.036807222941359,0.030674803176637,
0.024541228522912,0.018406729905805,0.012271538285720,0.006135884649155,
0.000000000000000,-0.006135084649154,-0.012271538285720,-0.010406729905005,
-0.024541228522912,-0.030674803176636,-0.036807222941359,-0.042938256934941
-0.049067674327418,-0.055195244349690,-0.061320736302209,-0.067443919563664
-0.073564563599667,-0.079682437971430,-0.085797312344440,-0.091908956497133
-0.098017140329561, -0.104121633872055, -0.110222207293883, -0.116318630911905
-0.122410675199216,-0.128498110793793,-0.134580708507126,-0.140658239332849 -0.146730474455362,-0.152797185258443,-0.158858143333861,-0.164913120489970
-0.170961888760301,-0.177004220412149,-0.183039867955141,-0.184913120489970
-0.170961888760301,-0.177004220412149,-0.183039887955141,-0.189068664149806 -0.195090322016128,-0.201104634842092,-0.207111376192218,-0.213110319916091
-0.219101240156870,-0.225083911359793,-0.231058108280671,-0.237023605994367
-0.242980179903264, -0.248927605745720, -0.254865659604515, -0.260794117915276
-0.2667127574748980.2726213554499490.2785196893850530.284407537211272

0.00000.00000.0000
-0.290284677254462,-0.296150888243624,-0.302005949319228,-0.307849640041535,
6.3126617403466, 0.25613049644, 0.302005549319228, 0.30744664004133, 0.31668740396891, 0.23155020816016, 0.325310292462651, 0.311106305759876, -0.33668953392220, -0.342660717311994, 0.348418680249434, -0.354163252420490, -0.35689515510680, 0.36613000000000000000000000000000000000
-0.382683432365090,-0.388345046698826,-0.393992040061048,-0.399624199845647, -0.405241314004889,-0.41864313105202
-0.402341314004990.0.4186343046098226,-0.393992040061048,-0.199624199804547, -0.405241314004990.0.418043111057904,-0.416429560097637,-0.422000270799800, -0.427555093430282,-0.431093818853152,-0.438616238538527,-0.444122144510429,
-0.4496113296546070.45508358712644.0.46053838527,-0.444122144570429,
-0.471396736825998,-0.476799230063322,-0.482183772079123,-0.487550160148436, -0.492898192229784,-0.499272666372792,-0.482183772079123,-0.487550160148436,
-0.514102744193222, -0.519355990165590, -0.52458968267869, -0.529803624686295, -0.52458968267869, -0.529803624686295, -0.51499761882097, -0.540121432328803, -0.52458968267869, -0.529803624686295, -0.51499761882097, -0.540121432328803, -0.52458968267869, -0.529803624686295, -0.51499761882097, -0.540121432328803, -0.5245886826786969696969696969696969696969696969696
<ul> <li>-0.534997619887097, -0.54011472129893, -0.54532998422046, -0.550459722956055,</li> <li>-0.55557023019602, -0.560661576197336, -0.565731810783613, -0.570780745886667,</li> <li>-0.55557023019402, -0.560661576197336, -0.565731810783613, -0.570780745886667,</li> </ul>
-0.653172842953777,-0.657806693297079,-0.662415777590172,-0.666999922313637, -0.671558954847018,-0.676092703575316,-0.6806609997795453,-0.685083667772700,
-0.724247082951467,-0.72846439044225,-0.7125120825283819,-0.72000250795182, -0.724247082951467,-0.72846439044225,-0.712654271672411,-0.738616568877270, -0.740951125354959,-0.745057785441466,-0.749116394521459,-0.751186799043612, -0.752720845506485,-0.74120735442745,-0.755127047042,-0.7551186799043612,
-0.757208846506485,-0.761202385404262,-0.76516394523459,-0.753186799043612, -0.757208846506485,-0.761202385404262,-0.765167265622459,-0.769103337645580, -0.773010453362737,-0.712888464262,-0.765167265622459,-0.769103337645580,
-0.817594813151584,-0.821102514991105,-0.824589302785025,-0.828045045257756, -0.831467612302545,-0.834862874986380,-0.838224705554838,-0.841554977436898,
-0.881921264348355,-0.884797098430936,-0.887633620402854,-0.890448723244758, -0.893224301195515,-0.895966249756185,-0.898674465693954,-0.901348847046022,
-0.903909293123443,-0.906595704514915,-0.909167987000522 _0.911706032005430
~U.9146U97557U3531,~U.916679059921043,-D.919113851600058 _0 021414030343045
-0.923879532511287,-0.926210242138311,-0.928506080473216,-0.930766961078084
$\begin{array}{c} -0.932992798834739, -0.935183509938948, -0.937339011912575, -0.939459221602190, \\ -0.941544065183021, -0.943593458161960, -0.945607325380521, -0.947585591017741, \\ \end{array}$
-0.949528180593037,-0.951435020969008,-0.953306040354194,-0.955141168305771.
-0.956940335732209,-0.958703474895872,-0.960430519415566,-0.962121404269042
-0.963776065795440,-0.965394441697689,-0.966976471044852,-0.968522094274417,
-0.970031253194544,-0.971503890986252,-0.972939952205560,-0.974339382785576, -0.975702130038529,-0.977028142657754,-0.978317370719628,-0.979569765685441,
-U.98U/8528U4U323U,-D.981953869109555,-D.983105487431216,-D.984210092386920
-0.983211042388941,-0.986308097244599,-0.987301418157858,-0.988257567730750.
-0.999176509964781,-0.990058210262297,-0.990902635427780,-0.991709753669100,
-0.992479534598710,-0.993211949234795,-0.993906970002356,-0.994564570734255, -0.995184726672197,-0.995767414467660,-0.996312612182778,-0.996820299291166,
-0.997290456678690,-0.997723066644192,-0.998118112900149,-0.998475580573295.
-0.998795456205172,-0.999077727752645,-0.999322384588350,-0.999529417501093.
-0.999698818696204,-0.999830581795823,-0.999924701839145,-0.999981175282601, -1.0000000000000000,-0.999981175282601,-0.999924701839145,-0.999830581795823,
=0.999698818696204, -0.999529417501093, -0.999322384588350, -0.999077727757645.
-0.998795456205172, -0.998475580573295, -0.998118112900149, -0.997723066644192.
-0.997290456678690,-0.996820299291166,-0.996312612182778,-0.995767414467660,
-0.995184726672197,-0.994564570734256,-0.993906970002356,-0.993211949234795, -0.992479534598710,-0.991709753669100,-0.990902635427780,-0.990058210262297,
-0.989176509964781,-0.988257567730750,-0.987301418157858,-0.986308097244599,
-0.985277642388941, -0.984210092386929, -0.983105487431216, -0.981963869109555.
-0.980785280403230,-0.979569765685441,-0.978317370719628,-0.977028142657754,
-0.975702130038529,-0.974339382785576,-0.972939952205560,-0.971503890986252, -0.970031253194544,-0.968522094274417,-0.966976471044852,-0.965394441697689,
-0.963776065795440,-0.962121404269042,-0.960430519415566,-0.958703474895872,
-0.956940335732209,-0.955141168305771,-0.953306040354194,-0.951435020969008,
-0.949528180593037,-0.947585591017741,-0.945607325380521,-0.943593458161960,
-0.941544065183021,-0.939459223602190,-0.937339011912575,-0.935183509938948, -0.932992798834739,-0.930766961078984,-0.928506080473216,-0.926210242138312,
-0.923079532511207,-0.921514039342042,-0.919113851690058,-0.916679059921043,
-0.914209755703531,-0.911706032005430,-0.909167983090523,-0.906595704514915,
-0.903969293123443,-0.901346847046022,-0.898674465693954,-0.895966249756185, -0.893224301195515,-0.890448723244758,-0.887639620402854,-0.884797098430938,
-0.881921264348355,-0.879012226428634,-0.876070094195407,-0.873094978418290,
-0.870086991108712,-0.867046245515693,-0.863972856121587,-0.860866938637767,
-0.857728610000272,-0.854557988365401,-0.851355193105265,-0.848120344803297,
-0.844853565249707,-0.841554977436898,-0.838224705554838,-0.834862874986380, -0.831469612302546,-0.828045045257756,-0.824589302785025,-0.821102514991105,
-0.817584813151584, -0.814036329705949, -0.810457198252595, -0.806847553543799.
-0.803207531480645,-0.799537269107905,-0.795836904608884,-0.792106577300212,
-0.708346427626606,-0.784556597155575,-0.780737228572095,-0.776888465673232,
-0.773010453362737,-0.769103337645580,-0.765167265622459,-0.761202385484262, -0.7572088465064850.753186799043613,-0.749136394523459,-0.745057785441466,
-0.757208846506485,-0.753186799043613,-0.749136394523459,-0.745057785441466, -0.740951125354959,-0.736816568877370,-0.732654271672413,-0.728464390448225,
-0.757208846506485,-0.753186799043613,-0.749136394523459,-0.745057785441466, -0.740951125354959,-0.736816568877370,-0.732654271672413,-0.728464390448225, -0.724247082951467,-0.722002507961382,-0.715730825283819,-0.711432195745217,
<ul> <li>-0.751700846506485,-0.751186199043613,-0.749136394521459,-0.745057785441466,</li> <li>-0.74095112534959,-0.718618568973770,-0.725654271672413,-0.72644709448225,</li> <li>-0.724247082951467,-0.72002507961382,-0.715730825283819,-0.711432195745217,</li> <li>-0.72424708295148118654,-0.702754744477222,-0.698376249408737,-0.639371460899654,</li> </ul>
$\begin{array}{c} -0.757208846506485, -0.753186799043613, -0.749116394521459, -0.745057785441466, \\ -0.74095115354959, -0.756161568877370, -0.735654271672411, -0.726443904825, \\ -0.72424708253467, -0.72002597961282, -0.735120225283819, -0.711421495745217, \\ -0.70710671186548, -0.702734744473725, -0.69837624490973, -0.69871460289564, \\ -0.6895460344737067, -0.685083667172700, -0.686509977395433, -0.67605703575116, \\ -0.67153059447019, -0.60999922303629, -0.66841577590172, -0.6870669270797, \\ -0.761556954474700, -0.6699992230362, -0.66841577590172, -0.870669270797, \\ -0.7615569544731067, -0.685083667172700, -0.6861577590172, -0.87066927079, \\ -0.7615569544731067, -0.685083667172700, -0.6861577591172, -0.78066927079, \\ -0.751569547019, -0.78069992230362, -0.66841577590172, -0.87066927079, \\ -0.751569547019, -0.78069992230362, -0.66841577590172, -0.87066927079, \\ -0.751569547019, -0.78069992230367, -0.68615775700, -0.6870670, \\ -0.751569547019, -0.78069992230362, -0.68641577570172, -0.7806927079, \\ -0.7515695470547419, -0.78069992230362, -0.68641577570172, -0.7806927079, \\ -0.751559547047019, -0.78069992230369, -0.68641577570172, -0.7806927079, \\ -0.75156954705747445705, -0.6864157779172, -0.7806927079, \\ -0.75156954705747445705, -0.6864157779172, -0.7806927079, \\ -0.75156954705747445705, -0.6879523705474457005, -0.687057057, \\ -0.75156954705747445705, -0.6879523705, -0.780692707005, \\ -0.7515954705, -0.780547005, -0.78054705, -0.78054705, -0.78054, \\ -0.751595470, -0.7805470, -0.780547005, -0.78054705, -0.7805470, -0.7805, \\ -0.7515795470, -0.7805470, -0$
-0.75720884506485,0.753186799043613,-0.749126394521459,-0.745057785441466, -0.74095112514959,0.758616568877370,-0.73265427162411,-0.7464439048225, -0.742421082951467,-0.720002597961382,-0.715310825281819,-0.711432195745217, -0.707106781186548,-0.702754744457225,0.69837624940873,-0.63917146088654, -0.689546544737067,-0.65508266772700,-0.680560997795453,-0.6540270375116, -0.651558954487019,-0.6659952203638,-0.66241577759012,-0.5870669327079, -0.5317249253777,-0.6851401022112,-0.63831812889752,-0.58124444683776
-0.75720884550645,-0.753186799043613,-0.74912639452145,-0.745057785441466, -0.74059115534595,-0.750816568677370,-0.735654271672413,-0.726443904825, -0.72424708253467,-0.72002597961282,-0.73512022283819,-0.711472195745217, -0.70710671186548,-0.72754744457252,-0.64983764490973,-0.6397146088954, -0.669546544737067,-0.65058267172700,-0.686500997795453,-0.67605703575116, -0.6751595447019,-0.6695992230368,-0.66241577590172,-0.6381054288972,-0.63912444863776, -0.63129249163646,-0.26761828194292,-0.62831542889722,-0.39124444863776, -0.641293294163646,-0.26761828194292,-0.62831942889722,-0.39124444863776, -0.641293294163646,-0.26761828194292,-0.26835948142389,-0.86205721175289,
-0.75720884550645,-0.753186799043613,-0.74912639452145,-0.745057785441466, -0.7405915254859,-0.750816586877370,-0.732565271672413,-0.726443904825, -0.72424708253467,-0.72002597961282,-0.735127022283819,-0.711421495745217, -0.70710671186548,-0.72754744457252,-0.64983764490973,-0.6397146089565, -0.649546544737067,-0.65698267172700,-0.686500997795433,-0.67605703575116, -0.64153728447019,-0.6699952303681,-0.66241577590172,-0.639164288972, -0.641547294163646,-0.62651826149427,-0.62851978142389,-0.63017211750127, -0.6125317264163646,-0.62761828914227,-0.628519481142789,-0.6307711578147, -0.612531924163646,-0.630757012189671,-0.62851948114278,-0.6307711751876, -0.555693914624231,-0.5075970185947,-0.5557964543,-0.56017957054639,-0.59013796439,-0.59013796434,-0.56777978746439,-0.5907897879545439,-0.56013958797546439,-0.56013958797546439,-0.56013958673546439,-0.5557978546439,-0.55579787978546439,-0.560139587978546439,-0.5557978546439,-0.5557978546439,-0.555797879546439,-0.5557978546439,-0.5557978546439,-0.5556439546439,-0.5557978546439,-0.55579786439,-0.5557978546439,-0.5557978454439,-0.5557978546439,-0.5557978546439,-0.5557978546439,-0.5557978546439,-0.5557978546439,-0.5557978546439,-0.5557978546439,-0.5557978546439,-0.555797845449,-0.5557978546439,-0.5557978546439,-0.5557978546439,-0.55579945444,-0.5557978546439,-0.5557978546439,-0.5557978546439,-0.5557978546439,-0.5557978546439,-0.5557978546439,-0.5557978546439,-0.5557978546439,-0.5557978546439,-0.5557978546439,-0.5557945444,-0.5557978546439,-0.5557978546439,-0.5557978546439,-0.5557978546439,-0.5557978546439,-0.5557978546439,-0.5557978546439,-0.5557978546439,-0.5557978546439,-0.5557978546439,-0.555797846439,-0.555797846439,-0.555797846439,-0.555797846439,-0.555797846439,-0.555797846439,-0.5557978464394,-0.5557978464394,-0.5557978464394,-0.555797846439,-0.555797846439,-0.55579978464394,-0.5557978464394,-0.5557978464394,-0.5557978464394,-0.555797845444,-0.555797845444978,-0.555797845444,-0.555797845444,-0.555797845444,-0.555797845444,-0.555797844,-0.55579784444
-0.75720884500485.0.7531867990436130.7491163945214590.745057785441460. -0.74005115354959.0.7531867990436130.73265427167411.0.726443904825. -0.7244470829514670.720002579613820.73256427167411.0.726443904825. -0.767106711806480.7270474445725.0.6483764499710.59711460489634. -0.6895465447176070.8650816677727000.6887644997145.0.97671460489634. -0.6895465447176070.8650816677727000.6887644997145.0.97671460489634. -0.6895465447176070.8650816677727000.68857648972.0.26787146489634. -0.643172445317770.6451444102210.6483146499720.261214444681715248. -0.6129214241631660.6274182148144720.463815449716270.861124444681715248. -0.6129214950106270.01828602762000.60551104147260.60661471818064. -0.55596930445241310.597057018586740.5557718574564390.966511679514
-0.75720884550645,-0.753186799043613,-0.74911639452145,-0.745057785441466. -0.74059115354959,-0.750816586877370,-0.732565271672411,-0.726443904825,- -0.72424708253467,-0.72002597961282,-0.735126721672411,-0.72694439048025,- -0.7010671180548,-0.70754744457252,-0.69837624490973,-0.6397146089565,- -0.689546544737067,-0.65699922303658,-0.6624157795012,-0.6580669270793,- -0.6135978947019,-0.6059992230365,-0.6624157795012,-0.658056927097,- -0.6317284295377,-0.648514401022112,-0.64831942889782,-0.6312444463776,- -0.64193924163646,-0.62761828914927,-0.62859481412387,-0.62057211763289,- -0.5559691463464,-0.50795701050897,-0.55511041404326,-0.00001647913876,- 0.55596913014274423,-0.57780745886967,-0.555731810735614,-0.506015767114727898,- -0.55559731010622,-0.504577250669,-0.54531394224047,-0.56778757854543,-0.55578578422047,-0.555787857842047,-0.5557878578542047,-0.5557878542047,-0.5557878578542047,-0.5557878578542047,-0.5557857854204720,-5.555784204722047,-0.5557878578542047,-0.5557878578422047,-0.5557878578422047,-0.555787854204772,-0.555785785420472047,-0.555785785420472047,-0.5557878542047,-0.555787854204772,-0.5557857854204778236422047,-0.55578578545437,-0.5557842204778,-0.555787854204778,-0.555786422047,-0.5557898542047,-0.5557898542047,-0.5557898545437,-0.555789854543778545437,-0.5557845437,-0.555784542047,-0.555784542047,-0.555784542047,-0.5557845437,-0.5557845437,-0.5557845437,-0.55578454379,-0.5557845437,-0.5557845437,-0.5557845437785442247,-0.555784543,-0.555784543778545437,-0.55578454379,-0.5557845437,-0.5557845437,-0.555784542047,-0.555784544442247,-0.555784543,-0.555784543,-0.555784543,-0.55578454244,-0.5557845454,-0.55578454247,-0.55578454247,-0.55578454247,-0.55578454247,-0.55578454247,-0.55578454247,-0.55578454247,-0.55578454247,-0.55578454247,-0.55578454247,-0.55578454247,-0.55578454247,-0.55578454247,-0.55578454247,-0.55578454444247,-0.55578454444247,-0.55578454444247,-0.555784544444247,-0.555784544444247,-0.555784544444247,-0.555784544444247,-0.55578444444247,-0.55578578444444474274,-0.5557
-0.157208846506485,-0.753186799043613,-0.149116394521459,-0.745057785414466, -0.74005115354859,-0.7501861656877370,-0.732654271672411,-0.7464439044825, -0.72424108251467,-0.72002597961382,-0.73256271672411,-0.72644397146089545, -0.70710671186548,-0.7254744457252,-0.64837624449971,-0.6397146089545, -0.64954654471300,-0.65081647172700,-0.68600987795453,-0.67606720375316, -0.6155892447019,-0.6495992230436,-0.646215771970172,-0.53016643270797, -0.6311789441019,-0.649992230430,-0.646215771970172,-0.53016643270797, -0.6311789441019,-0.049542301022112,-0.645149112790172,-0.53016643270797, -0.6311789441019,-0.049542301022112,-0.6451411414122,-0.600643270797, -0.6311789441019,-0.049542301022112,-0.645514144122,-0.60061370797, -0.53596910449241,-0.590759710854874,-0.5557745745649,-0.5961396109514,-0.55590615769137, -0.55550611411454,-0.259012570805,-0.55532498422047,-0.301141252993, -0.555570231011962,-0.5504577230665,-0.55532498422047,-0.3011412529793, -0.555702301191485,-0.2790102546627,-0.5532498422047,-0.3011412529943, -0.55197051498072,-0.5504577230665,-0.5532498422047,-0.30114152594065359
-0.75720884500485.0.7531867990436130.749136394321459.0.745057785441460. -0.740051125345959.0.7531867990436130.73265271276724130.7264439748250. -0.724247082514670.7200025079613820.7325052716724130.72694397416089654. -0.70710671811665480.7027547444572250.6483764409730.6397146089654. -0.6415592447130670.645144010221120.648315421897300.63971460895654. -0.64155924471300.645144010221120.642154715901720.6397640571405714668. -0.64159124416146.0.6.2556214281494270.638571691421870.64051218784637. -0.64159124416464.0.5075917610260726000.551104140412500.60161497183866. -0.555691401424131.0.539751012106271600.551104140412500.60161497183866. -0.55569121105086027.0.53951012108647.0.535710511464390.8401940199199193737. -0.555570212101801620.530457192186640.5357185124844220470.3401142727893. -0.53557021210180120.530457192186640.5357185124844220470.34011145727893. -0.53559021210190120.530457192186640.53524984220470.34011145727893. -0.53557021210190120.530457192186640.53524984220470.3401114515990155558. -0.514017441912220.504519732186640.5453124814220440.442766677782. -0.53597021210190120.530457192186640.53524984220470.344114515590155558. -0.514017441912220.504519732186640.545324984220470.347979815590155590155590.
-0.757208846506485,-0.753186799043613,-0.743163794521459,-0.745057785441466. -0.74095112534959,-0.750816586877370,-0.732565271672411,-0.7264439748252, -0.72424708253467,-0.72002597961282,-0.735126721672413,-0.726941397416028954, -0.69346544737067,-0.655081667712700,-0.686500997395433,-0.67602703575116, -0.67155087447019,-0.60599922303629,-0.66241577950127,-0.5870669270797, -0.63172842953177,-0.648514401022112,-0.64831542889782,-0.6312444463776, -0.64193294163646,-0.6276415279,-0.62859549142389,-0.650215718738,-0.65511041404326,-0.600616479183866, -0.5550931041244231,-0.50759701508967,-0.54537949422047,-0.3400139218366, -0.5555093101417445,-0.57708075806697,-0.54537349422047,-0.34013958257643,-0.555378422047,-0.34114272983, -0.5555073101602,-0.505379216605,-0.54532499422047,-0.3415359910137805, -0.555509310142741322,-0.50577921508967,-0.54537394822047,-0.341135227616,-0.90506157611147272983, -0.5555073101602,-0.5059103125411,-0.5553782125718,-0.4962766977278, -0.55139976198707,-0.52980312454137,-0.50537827518,-0.4952766977278, -0.5141027413222,-0.505303124524137,-0.505378125718,-0.4952766977278, -0.5141027413222,-0.5053031245211,-0.50539,-0.55531981078511,-0.516397, -0.514102741322,-0.555701803976,-0.55531981078511,-0.517518,-0.4952766977278, -0.514102741322,-0.5053031245211,-0.50539,-0.5553198107851,-0.3452976697278,-0.555378216997571,-0.505398,-0.555398,-0.515329976134976,-0.555398,-0.5
-0.157208846506485,-0.753186799043613,-0.149116394521459,-0.745057785441460, -0.74095112534959,-0.750161568877370,-0.732654271672411,-0.749443904825, -0.72424708251467,-0.7200257961382,-0.735730827673413,-0.72694397146089654, -0.70710671186548,-0.72504744457225,-0.64837644499713,-0.6397146089654, -0.649540544713007,-0.685081667727003,-0.6881042949713,-0.6397146089654, -0.64155794447019,-0.665081925210136,-0.6624157759012,-0.65106691257079, -0.631517284295177,-0.648514401022112,-0.6481142288972,-0.3511244486376, -0.641595744161646,-0.62504128914227,-0.6285348412139,-0.6205721165284, -0.64159124416466,-0.5270812614294,-0.628514941239,-0.62065721165284, -0.6315215934816646,-0.527081764594,-0.55571181474391,-0.540611395900572, -0.55557021011845,-0.550457972316664,-0.55571181474391,-0.54061576197377, -0.55557021011845,-0.550457972316664,-0.5557181074391,-0.5406157619737, -0.55557021011845,-0.550457972316664,-0.5557181074391,-0.5406157619737, -0.55557021011845,-0.550457972316664,-0.555312498422047,-0.5401134599155901555, -0.51402744192227,-0.508810142541107,-0.50531810372518,-0.4822766077782, -0.4229227844,-0.459515201624441224417,-0.540531810395240,-0.4822766077782, -0.42292694607,-0.45951601644412244127,-0.540531810395240,-0.4822766677782, -0.429512954607,-0.459515016244412244127,-0.540531810395251,-0.3605195721,-0.54053972123,-0.47952105032,-0.45031872518,-0.48023767312,-0.4795210532,-0.4795210534,-0.4905139525,-0.4503937039524,-0.450318725312,-0.47031848,-0.450139520,-0.35031848,-0.557021,-0.3403934,-0.30031848,-0.557021,-0.3003948,-0.3003948,-0.430534,-0.557021,-0.300394,-0.30034,-0.300394,-0.30034,-0.300394,-0.30034,-0.300394,-0.30034,-0.300394,-0.30034,-0.30034,-0.300394,-0.30034,-0.300394,-0.30034,-0.300394,
-0.757208846506485,-0.753186799043613,-0.74316394521459,-0.745057785441460. -0.74059115534595,-0.750816586877370,-0.732565271672411,-0.7264439748252, -0.72710671186548,-0.727007257961282,-0.73517270127,-0.5879146089654, -0.689546544737067,-0.655081667172700,-0.68650997395433,-0.67605703575116, -0.67155087447019,-0.6699922303628,-0.66241577590127,-0.5870669270797, -0.63172842953177,-0.648514401022112,-0.64831942889782,-0.6312444463776, -0.64193294163646,-0.67057812809472,-0.6285954811342889782,-0.6312444463776, -0.64193294163646,-0.670578101808974,-0.6587198142389,-0.86051211781289, -0.5555073101622,-0.50757010508974,-0.55757045439,-0.580139581254439,-0. -0.5555073101602,-0.50557970150695,-0.54532498422047,-0.34013785746439,-0.5801395807556, -0.55550813114174452,-0.57780745868667,-0.565731810735614,-0.59661357619147272981, -0.5513199761987077,-0.5695702150897,-0.5453294822047,-0.34013422766697727, -0.55130612321149222,-0.5657782366657,-0.5453294822047,-0.34013422766697727, -0.5513061232114272298,-0.46755016148436,-0.46128772079123,-0.4769220660372, -0.5141027413222,-0.567649577697,-0.55537811810785614,-0.5805687663589, -0.5141027413222,-0.567649577697,-0.55537811810785214,-0.36797821660538, -0.5141027413222,-0.567649577697,-0.55537811810785219,-0.47692206607327, -0.441292765980,-0.4675501614843,-0.461283772079123,-0.4769206537621660538, -0.441292765980,-0.4575045457649,-0.463531852719,-0.48058276669778,-0.46053189769240,-0.350587811547649431,-0.595697625766977676,-0.46053187079123,-0.47692050057821,-0.441216437044,-0.46353185097643,-0.46123779123,-0.47692005779126069721,-0.4605387814917055940,-0.461317579144171659540,-0.45053871518,-0.441211655940,-0.461317557916,-0.461317557916,-0.4603178579143176,-0.46053387538,-0.4603387538,-0.441317657940,-0.461317657940,-0.46033875398,-0.440317657940,-0.46033875398,-0.4603387538,-0.4603387538,-0.4603387538,-0.4603387538,-0.4603387538,-0.4603387538,-0.46033875387,-0.440317855940,-0.40338758398,-0.4603387538,-0.46037763176676437767920077799407940767,-0.46038
-0.757208846506480.7511667990436130.7431657964234590.745057785441460. -0.74005112534595.0.7511661568877700.73265427167411.0.7264439.042825 -0.724447082514600.720002579613820.73250527167413.0.7264439044825. -0.767106711664480.72747444572250.6483764499710.63714140.027657146048964 -0.6895465447317050.6550816971727000.683164097155151. -0.645172492531770.651491465720.6483764997150.312142446813715116. -0.645172445351770.1032602750900.6551161497920.0.31214444681371 -0.64517244531770.10326026726090.6553161497520.31214444681371 -0.64517249530720.507597018580740.5635778574564390.569014978605765. -0.5556993044524310.597057018580740.55571817785146.4300.66151619731. -0.5535702330160220.5504577572366050.555324984220470.3041142727983. -0.554997014174520.5045757012366050.355324984220470.3041314272983. -0.51409761227840.46575040124341070.305319381257180.46227666677782. -0.45296493044224310.5949125216624431070.3053193837257180.4622766677782. -0.51499761227840.465750451674650.4821397279120464831412627.0. -0.46296492278460.46575467516740.48213727912046481314272212344. -0.46227666077820.524984220470.304979721204648134270422.0.00615787782. -0.514997612278480.465754675167460.48213972791204648131265902.00. -0.55299304422431.0.599012.2.0.52498422047.0.309007120.4481272693. -0.514997612279480.465754675167460.48213972971204641347270420502172. -0.4711967366259880.4657645787850.2459896774120484171827942. -0.441027444192227480.46576457878676760.4410179721204714417227893. -0.51297480.482764270490.4821740979120484171827942. -0.441017441922274840.48476427099067400.441174579120484171827942. -0.441191274940.48576459784770.44114171827942. -0.441718579420.484774492270420.441744979120.441744979120.484171825944. -0.405241114049400.19974449794449440.41647799120.441717857944. -0.405241144049400.21974440.441740479799120.441717857944. -0.405241144049400
- 0.75720884500485.0.7531867990436130.7481163945214590.745057785441400. 0.74005115354959.0.7531867990436130.73265271672413.0.72644039048253. -0.724247082514670.7200025079613820.7325027167413.0.7264439048253. -0.707106781180548.0.75754744457252.0.048316424897320.93971460889644. -0.6895405447370670.6850826671727000.68050997354330.676057375116. -0.67155984470190.64508216271727000.684514401230.62415775901720.63706691297079. -0.633172842953770.6485144010221120.643815428897920.63172444468776. -0.641952944161464.0.637057611850877.0.62635484124790.62057711872894. -0.6315231590580270.610282062761090.6353110414042560.600615479138866. -0.555569130114174452.0.5707807458866670.5453718107730140.560615761737. -0.555567310106020.550457725160600.4531248420140.54661376517387. -0.555569130106020.550457725160600.45324894220140.540613761737. -0.55556731010620.550457725160600.4532484220140.540613761737. -0.55556731010620.55045772516060.4532489720120.4477972106132627. -0.45296912272740.4535160148420240.452167972160.452578677782. -0.422766677782. -0.4292012272740.453516014842240.42815172791210.44822766677782. -0.42920192274740.453516014842240.42815172791210.44821746217782. -0.42920194564070.4595167857676670.4635187109592.00.45035877782. -0.42920194564070.459747856070.4482187571210.47827786677782. -0.429791210.478921172277440.429742124457457676670.463518710952000.4500158195721. -0.42975460777421051220.44412124576767670.438578792120.4483174274234. -0.42975460773100.4396442210.53516014419420.4281675792120.4184317427434. -0.42975460773102.0.209421998456770.4397421994554.0.0209763770.41843174274472. -0.4295641224574740.421244576767670.43987597120.4484317453054649777. -0.46964142145554070.4397421994554.0.0210.43979120.44130457676677782. -0.45956412142716570490.3397420400014490.34844744745. -0.45956497321492.0.021.200277998000.4317179395189520.41130546694774.
-0.757208846506480.7511667990436130.7431657964234590.745057785441460. -0.74005112534595.0.7511661568877700.73265427167411.0.72464139.042524 -0.742447082514600.7200257961182.0.67125022818190.71142145745217. -0.70710671160448.0.7274744457225.0.648376449971.2.00.747141.0.278547146089645 -0.649546544717051.0.6.6510816971727000.68316409715.0.5178547140689545 -0.649546544717051.0.6.6510816971727000.683164199712.0.2.017114214575411.0.0000 -0.65117249251777.0.624514441522.0.0648314548972.0.2.0171144446617151248. -0.645117244515177.0.61082606276090.6655110414726.0.06061516197317. -0.61241950500627.0.61082606276090.6655110473614.0.2.0648131454466617161248. -0.555969304492431.0.590759701858074.0.5573161703614.0.2.0606151619371. -0.55130916117445.0.2590162740662.0.55731817073614.0.2.0566151619317. -0.5513091612745.0.2590162740662.0.55312498422047.0.3.04011412722983. -0.5514997162954607.0.459516124643107.0.55312498422047.0.3.0401141272983. -0.551499716954607.0.459516124643107.0.50318183725718.0.48622766677782. -0.44951252784.0.465751521614643.0.4861387279131.0.48622766677782. -0.44951252784.0.46575487516124643.0.4861387297121.0.4862776637782. -0.44951252784.0.46575487516124643.0.4861387295121.0.478622766677782. -0.4495112954600.0.4597649516746.0.4481214579121.0.0.4791212344. -0.4495112954600.0.459764576767.0.46313973957240051045.0.352398.0.499121.0.49617382344. -0.449511292784.0.34421245510104.0.4481214551019121.0.4191227812.0.4863170792121.0.496177822040. -0.44951124527846.0.0.444122145510104.0.4481405979121.0.0.478612766677782. -0.44951124527840.0.0.429749454.0.426142744570104.0.41846279121.0.0.4786127844. -0.04951124527840.0.0.219742121.0.478792121.0.0.478612789212.0.0.47861278244. -0.04951124527840.0.0.219742124454.0.4791422145510144.0.418412214551019144.0.418462766677782. -0.4495112455000.0.0.719704122545470.0.4484184502979121.0.0.478612784544. -0.03261114004959974.0.0.3110199595974.0.0.31195952104010414.0.311117955020148603154. -0.032603114088.0.0.71412325520481.0.31711959
-0.757208846506480.7511667990436130.743165796423430.740577854414.00. -0.74005112534595.0.7511661568877700.73265427167411.0.72464130.01254 -0.742447082514670.72002579613820.7326527167414.0.72764130.724647034125 -0.76710671166448.0.7274744457220.648376449971.0.6397146048964 -0.6495465447130670.650816649922300360.6462157191031720.530497146048964 -0.6415599465447131000.6508164792210030.6462157191031720.53049716048954 -0.64155195010510.610368067261000.6462157191031720.53046421801716 -0.61550950105005270.610386067261000.65551410443260.4066157119175. -0.55269730442241631660.6254182169146270.66351410442560.6066157191731. -0.55596930442241.0.101826067261000.557316110432560.0.606515191731. -0.55596930442241.0.79700745866960.55573161795161.4.0.56615719153590 -0.55596930442241.0.79700745869670.55573161795161.4.0.566157191731. -0.55596930442241.0.79700745869670.55573161795161.4.0.566157191731. -0.55596930442241.0.79700745869670.5557316179514.0.0.566157191731. -0.55199701171450.52790124264250.55324984220470.30417147272983 -0.51499716259880.4657469516740.4821937709130.47979201631250.479792016312. -0.42286126710.5457102475161040.482194179712310.479792103120.479792103120.479792103120.4797921034 -0.42926149257480.40571049516740.4821947091230.479792103120.479521204 -0.42926149257480.42412144570100.4842154510791230.479521234 -0.4295013201310227480.40571095167460.33930431755110.4061711147212484 -0.42950134012210.4292012954400.418412145510010.418417110592400.5139243107. -0.4495113957013400.4290027099800.43841421445101010.44811410542.0.40101318483112 -0.4275500134012810.42200027099800.4184275409703710.41641371039184 -0.43954119673645990.43944194410.3393044019440.339524444410.4395149444444444444444444444444444444444
- 0.157208846506485.0.15318619601251.0.152162785141460.57254214.0.2764414.0.2764414.0.2764414.0.2764419.0.2764419.0.2764419.0.2764419.0.2764419.0.2764419.0.276419.0.2764419.0.276419.0.2764419.0.276419.0.2764419.0.276419
- 0. T37208846506485, -0. 753186799043613, -0. 74316394521459, -0. 745057785441466, 0. 740951125345959, -0. 7508161568877730, -0. 7326527167411, -0. 72644397414625, -0. 72444108251467, -0. 72002507961382, -0. 73370625283819, -0. 711422145745217, -0. 70710671186548, -0. 72754744457252, -0. 64837624449971, -0. 6297146048954, -0. 64954654471300, -0. 6450816267172700, -0. 64860397195453, -0. 6740770357316, -0. 61355989440719, -0. 64654992230362, -0. 646214571970172, -0. 65016643270709, -0. 613517894416134, -0. 64854821022112, -0. 646214571970172, -0. 65016643270709, -0. 613517895416346, -0. 64854821022112, -0. 645141571970172, -0. 650613701707, -0. 61521795036627, -0. 7070074586667, -0. 645514571705746, -0. 5555693104492411, -0. 500759710854874, -0. 53577167145649, -0. 9606137619781, -0. 55550703104492411, -0. 500759710854874, -0. 53577167345649, -0. 94061395690758, -0. 53559693104492411, -0. 5007597108548674, -0. 53537405745649, -0. 94061395690758, -0. 5359693104492411484, -0. 645514279236605, -0. 545312498422047, -0. 34011142727993, -0. 5349971692709, -0. 64755161244824107, -0. 505313033725718, -0. 48622766677782, -0. 4529819227984, -0. 46557649576767, -0. 465313703572913, -0. 476922106312577162, -0. 4796210593122, -0. 50983104254107, -0. 503313033725718, -0. 486217696277782, -0. 479621059312, -0. 479621059312, -0. 4769210631252, -0. 479501075954607, -0. 44557162751764, -0. 31939240401342, -0. 500318185517, -0. 31093189521, -0. 479621059322, -0. 47961139576625998, -0. 4655767867, -0. 31981940410412, -0. 300318189511, -0. 30031818951, -0. 31039781, -0. 30031818951, -0. 31039781, -0. 30031818951, -0. 31039781, -0. 30031818951, -0. 3103978443, -0. 31981944134, -0. 31981944134, -0. 31981944134, -0. 30031818951, -0. 3103031818951, -0. 3103031818951, -0. 31056159712, -0. 310612057443, -0. 3198189431, -0. 3198189431, -0. 3198189431, -0. 3198189431, -0. 319818943, -0. 319818943, -0. 3198189441, -0. 31981894443, -0. 31981894443, -0. 31981894443, -0. 31981894443, -0. 3198184443, -0. 3198
- 0.757208846506485.0.7531867990436130.743165795423459.0.745057785441460. -0.74005112534595.0.7531867990436130.73265271672413.0.7264430948253 -0.72424708253467.0.7200625079613820.73265271672413.0.7264439048253 -0.767106781186548.0.75754744457252.0.698376449873.0.6997146089564 -0.689540544737067.0.6450426471727000.69856494973.0.63297146089564 -0.687559444731063.0.6654157727000.648514401237.0.627550172.0.637564453 -0.64159294416146.0.63264128914927.0.628517814428797.0.63057181620771187289 -0.63317284245377.0.648514401022112.0.64831432889792.0.63112444486376. -0.64195294161646.0.63057570186097.0.638571814773546439.0.5305711116278 -0.55556731010627.0.53057570186097.0.5435748442074.7.54054738 -0.55556913101627.0.530517018074886667.0.545371810773514.0.7467979210651272. -0.55190813101627.0.530516126466275.0.54324899822017.0.4312484978. -0.55556913101627.0.530516126466275.0.54324899822017.0.431248978200772. -0.54197154084231.0.45374686575.0.543248998230735591.0.44797921061325. -0.5556913101627.0.530516126466275.0.543248998230735591.0.448439. -0.5545991510180797.0.530516126486775.0.54324998230735591.2.448439. -0.5545912106200535161014442207.0.545378551559.0.43124948207. -0.541927694.0.5355161014442140.0.545786785757951.2.44899230755912.0.0.4757921005127. -0.54197561931022.0.535161014442140.0.44211457504001701512. -0.475756073312023.0.45334460444221.0.45518498230755912.0.0.4757921005127. -0.4767920105127. -0.4767920105127. -0.4767920105127. -0.4767920105127. -0.4767920105127. -0.47575607341223.0.741421245767667. -0.435956791341224.5. -0.47555673412023.0.741421445704000.41861371055520.0.410139181855127. -0.41964137155994.0. -0.355647234402.0.0.7170074102154180.319321000101460.4621845049433. -0.31066149427050.0.5111001995075.0.0.201494473122.9.425605431.0.21560894473. -0.31066149427050.0.51110019950526.0.201421244575045431.0.215619842470473. -0.31066149427256447.0.23702265054732.0.23442555054431.0.2261555494473. -0.2261555744899.0.21101019950567.0.2340555045441.0.275615558
- 0.757208846506485.0.7531867990436130.74316578754214.0.764241.0.76443944862. -0.7402515354859.0.7531867990436130.73265271672414.0.7264439448263. -0.742447082514670.72002579613820.73265271672414.0.726443944825. -0.767106781186548.0.7275474445725.0.6483164248973.0.6397146088654. -0.68754954954473106.3.8.6569952303638.0.6624157759017.2.0.5305695214068 -0.64195284473106.3.8.6569952303638.0.6624157759017.2.0.5706527311. -0.64349724416646.0.6264157270.0.62859484124790.6205771167284. -0.64349724416646.0.6507570186097.0.63857484124790.6205771167284. -0.6555693101417465.0.57078014586667.0.55571811073014.0.56708578546439. -0.555569310140274.0.5057570186097.0.54357484637. -0.555569310140274.0.5057570186097.0.54352489422074.0.54357845. -0.55569310140274.0.5057570186097.0.54352489422074.0.54357845. -0.555693101417452.0.57078014586667.0.54537181073514.0.53578. -0.5545973101062.0.5459730186097.0.54352489422047.0.54579248045. -0.55459751018097.0.5435910.545349. -0.55459751018097.0.5459701.5454494.0.54577855781. -0.5416274419222.0.768911425244022.0.545348942074.0.545792879. -0.5416274419222.0.768911424214.0.54570192571.0.54538. -0.541627419222.0.7980112524104.0.545398. -0.541627419222.0.7980112524104.0.545398. -0.541627419222.0.7980112524104.0.545398. -0.45298119222.0.7980112524104.0.545398. -0.45298119222.0.7984107.0.5259911254844. -0.4139515924.0.01391191721541. -0.421942124297484. -0.413951244421445701.0.4139515924.0.01391195524.0.013911951944. -0.45298411922.0.01391191521541. -0.445298411922.0.0219984421. -0.41396119524.0.0.139199844. -0.41396119524.0.0219984.0.01391195524.0.0.139198484444. -0.4139611924290.0.021919845157. -0.42298411922.0.0219442414545701094.0.0.319199519524.0.01484117151934. -0.452554192220.0.011110613558677.0.0.31919826901643.0.14641191024970.0.015986941044.0.0159894443.0.0159894443.0.0159894443.0.0159894443.0.0159894443.0.0159894443.0.0159894443.0.0159894443.0.0159894443.0.00144444205.0.001444442054.0.001444442054.0.00144444205.0.00144444205.0.001444442054.0.
- 0. T37208846506485, -0. 7.3186799043613, -0. 748116594521459, -0. 745057785414 466, -0. 74016, -0. 73204871767414, -0. 7264419 6426, -0. 740419042851467, -0. 7320672797414, -0. 7370471767414, -0. 7367414, -0. 7464419049425, -0. 74704710375416, -0. 7370672175414, -0. 7464419042851467, -0. 73006721974140548, -0. 70710671166548, -0. 70720671974140547, -0. 64811421499713, -0. 64917140049954, -0. 6491497139012, -0. 6491497149014, -0. 6491497139012, -0. 649149714914, -0. 64950454471190, -0. 649504547139012, -0. 649146127, -0. 6491497149, -0. 64950421471914, -0. 6495042147194, -0. 6495142041022, -0. 649141242, -0. 649141242, -0. 649141242, -0. 649141242, -0. 649141242, -0. 649141242, -0. 649141242, -0. 649141242, -0. 649141242, -0. 649141242, -0. 649141242, -0. 649141242, -0. 649141242, -0. 649141242, -0. 649141242, -0. 649141242, -0. 649114242, -0. 649114424, -0. 649141242, -0. 6491141424, -0. 6491141424, -0. 6491141424, -0. 6491141424, -0. 6491141424, -0. 6491141424, -0. 6491141444, -00. 555507105146449, -0. 555307105146449, -0. 555507105146449, -0. 555507105146449, -0. 555507105146449, -0. 55514444442, -00. 649114124, -0. 6491141253991053, -0. 55516911411484, -0. 7550161444444, -00. 555149444, -0. 557145144, -0. 5574444, -0. 557444, -0. 557444, -0. 557444, -0. 557444, -0. 557444, -0. 557444, -0. 557444, -0. 557444, -0. 557444, -0. 557444, -0. 557444, -0. 5574444, -0. 557444, -0. 5574444, -0. 5574444, -0. 5574444, -0. 5574444, -0.
- 0. T3720884500485, -0. 73186799043613, -0. 748116594521459, -0. 745057785441460, -0. 7404515354959, -0. 736816568877370, -0. 73265427167414, -0. 7246437944250, -0. 7444308251460, -0. 720002597961382, -0. 71320627167414, -0. 724741, -0. 746439044825, -0. 7444308251460, -0. 72000259712700, -0. 488020947935431, -0. 7464720357316, -0. 649316244971, -0. 649514649712700, -0. 64801649713, -0. 6497141604896, -0. 72714144512752, -0. 64837644971, -0. 6497141604896, -0. 7271414451257, -0. 64837649713, -0. 7491740375316, -0. 64931624453177, -0. 64514441022, -0. 648514489712, -0. 2481514444681712, -0. 648514449712, -0. 648514449712, -0. 648514244468171, -0. 649112444468171, -0. 64911244468171, -0. 64911244468171, -0. 64911244468171, -0. 64911244468171, -0. 64911244468171, -0. 64911244468171, -0. 64911244468171, -0. 54911244468171, -0. 5491244468171, -0. 553509210442241, -0. 553509210442241, -0. 553509210442241, -0. 553509210442241, -0. 5535092104422, -0. 550457172304665, -0. 5532498422047, -0. 548512498422047, -0. 548512498422047, -0. 548512498422047, -0. 548512498422047, -0. 54891259121, -0. 747952006127, -0. 549124, -0. 5491212, -0. 747952006127, -0. 54929121, -0. 747952006127, -0. 5492914114272981, -0. 6485131059240, -0. 5531049134227844, -0. 54927844, -0. 54927844, -0. 5491134579121, -0. 747952006124, -0. 54929142, -0. 7491134272981, -0. 54927844, -0. 5491134579121, -0. 747952006127, -0. 5491134579121, -0. 747952006127, -0. 549121, -0. 747952006124, -0. 54914411212344, -0. 54914279121, -0. 749227848, -0. 54140274441922, -0. 50681312659140, -0. 31939724040146, -0. 549132125914, -0. 54927844, -0. 549144445, -0. 54914279121, -0. 747952006127, -0. 5491444444444, -0. 549142784444, -0. 5491427844, -0. 5491427844, -0. 5491427844, -0. 5491427844, -0. 5491427844, -0. 5491427844, -0. 5491427844, -0. 5491427844, -0. 5491427844, -0. 5491427844, -0. 5491427844, -0. 5491427844, -0. 5491427844, -0. 5491427844, -0. 5491427844, -0. 5491427844, -0. 5491427844, -0. 54914278444, -0. 5491427844, -0. 5491427844, -0. 54927865785, -0. 248665
- 0. T37208846506485, -0. 7.3186799043613, -0. 748116594521459, -0. 745057785414 466, -0. 74016, -0. 73204871767414, -0. 7264419, -0. 7324419, -0. 734
- 0. T37208846506485, -0. 7.3186799043613, -0. 7481165984521459, -0. 745057785414 +0. (24647904825), -0. 74041057785414 +0. (24647904825), -0. 750615687172700, -0. 7326527167414, -0. 72644797176413, -0. 7507167414, -0. 7267141, -0. 72644797176420, -0. 74647904825, -0. 7507067186487, -0. 73206257715716, -0. 6485164409714, -0. 62671416648, -0. 750716714649714, -0. 6267141648, -0. 726714649714, -0. 6267141648, -0. 726714649714, -0. 6267141648, -0. 74647149714, -0. 62667952301634, -0. 648514401971, -0. 626971460489644, -0. 64954054471190, -0. 64651497147012, -0. 64651497147012, -0. 64651497141019, -0. 6206615761717, -0. 648514401022, -0. 6491145248972, -0. 639114244486776, -0. 6535967034416149, -0. 62661576172710, -0. 64851440109, -0. 620697211105164, -0. 555577057456430, -0. 64611357601571116144, -0. 555577057456430, -0. 6481142109, -0. 62060157619731, -05555670104492641, -0. 55071057456640, -0. 555377057456430, -0. 64601157619731, -05060157619731, -0555570111110460, -0. 550457702164664, -0. 555312498422047, -0. 5469113559010558, -0. 555312498422047, -0. 5469113559010558, -0. 555312498422047, -0. 5469113559010558, -0. 555312498422047, -0. 5469113559010558, -0. 555312498422047, -0. 546911359010558, -0. 555312498422047, -0. 5469113559010558, -0. 555312498422047, -0. 5469113559010558, -0. 555312498422047, -0. 5469113559010558, -0. 555312498422047, -0. 54691131559010558, -0. 555312498422047, -0. 54691131559010558, -0. 555312498422047, -0. 54691131559010558, -0. 555312498422047, -0. 546911711559010558, -0. 545312498422047, -0. 54691171159591, -0. 4669117115952, -0. 566915167131, -05669111155902, -0. 556911016952, -055691121, -0. 746921069132, -0. 746911991259, -0. 746911991259, -0. 746911991259, -0. 746911991259, -0. 746911991259, -0. 746911991259, -0. 746911991259, -0. 746911991259, -0. 746911991259, -0. 746911991259, -0. 746911991259, -0. 746911991259, -0. 75691141252, -0. 7579115551144, -0. 737914495, -0. 2369124276654475, -0. 2369124276654475, -0. 2369124276654475, -0. 2369124276654475, -0. 236912475
- 0. T37208846506480. 7.31867990436130. 74316578574214.0. 745787414.0. 7264430.04252 0. 74441082514600. 720002507961382.0. 7320527167414.0. 72767414.0. 72764430944252 0. 7444108251460.4.0. 720002507961382.0. 6132510274314.0. 7276413.0. 7276443094425 0. 707106718106548.0. 7275414445725.0. 64837644971.0. 63071414028754521 0. 648374643914252 0. 648374643914052 0. 648374643914025 0. 648374643914025 0. 648374643914 0. 64897465443914025 0. 648374643914 0. 648974654 0. 648974654 0. 648974654 0. 64897465 0. 6483746 0. 64857468 0. 64857468 0. 64857468 0. 6485746 0. 6485746 0. 6485746 0. 6485746 0. 6485746 0. 6485746 0. 6485746 0. 6485746 0. 648574 0. 648574 0. 648574 0. 648574 0. 648574 0. 6485 0. 6485 0. 6485 0. 6485 0. 6485 0. 6485 0. 6485 0. 6485 0. 6485 0. 648 0. 64 0. 648 0. 648 0. 64 0. 648 0. 64
<ul> <li>0. 157208846506485, -0. 753186799043613, -0. 74316578523459, -0. 745057785441460.</li> <li>0. 7409511534959, -0. 736816568877370, -0. 7326527167414, -0. 7264439744825,</li> <li>0. 74244102851467, -0. 72002597961382, -0. 73375025203819, -0. 71142219574463904825,</li> <li>0. 767106781186548, -0. 72500259781382, -0. 64381642449731, -0. 6297146089654,</li> <li>0. 669540544731067, -0. 68508166772700, -0. 68605097735433, -0. 6740750375116,</li> <li>0. 6715509744410129, -0. 664505972310362, -0. 648316421849732, -0. 63701642957316,</li> <li>0. 664159734161645, -0. 67250412491427, -0. 624815441239, -0. 62506721165278,</li> <li>0. 634397244161645, -0. 65264121914927, -0. 624815441239, -0. 62507211165278,</li> <li>0. 63439724161645, -0. 65264121914927, -0. 624815441239, -0. 62507211165278,</li> <li>0. 63439724161645, -0. 672901674586657, -0. 6648514401239, -0. 6205711165278,</li> <li>0. 63439724161645, -0. 672901674586657, -0. 6355110414041259, -0. 630512114848,</li> <li>0. 61439724111485, -0. 73901674586665, -0. 55571101731614, -0. 54066137617317,</li> <li>0. 555570210118042, -0. 550653972316664, -0. 55571011731614, -0. 5406137617317,</li> <li>0. 5555702101189102, -0. 550653972316664, -0. 555312488422047, -0. 3406131815590165588,</li> <li>0. 514102744191222, -0. 50881014254107, -0. 5035131831725718, -0. 4862766677782,</li> <li>0. 4298012927894, -0. 405576495767967, -0. 46523870956274, -0. 34501384125452,</li> <li>0. 429801299027, -0. 2599016246852, -0. 3451314698422047, -0. 3450138972113, -0. 4796210631322,</li> <li>0. 429801299027, -0. 2599016246852, -0. 412184779123, -0. 4796210631257,</li> <li>0. 429801299027, -0. 239012924697, -0. 4350128976142, -0. 339103901381953152, -0. 41218452466977, -0. 3398920400164, -0. 23910320412, -0. 3391030138253, -0. 41218452466977, -0. 3398920401048, -0. 238135528, -0. 4138155426447, -0. 3391292041352, -0. 2680137511937, -0. 348198541352, -0. 2680137511937, -0. 33819831522, -0. 768613046442766477, -0. 3</li></ul>
- 0.757208846506485.0.753186799043613.0.76.7326527167241.0.7267441.0.72644708251469.0.726741.0.7276741.0.7276741.0.7276741.0.7276741.0.7276741.0.7276741.0.727641.0.727641.0.727641.0.727641.0.727641.0.727641.0.727641.0.727641.0.727641.0.727641.0.727641.0.727641.0.727641.0.7276741.0.7277741.0.7276
- 0. T37208846506480.731867990436130.73265271674130.74057785414406. 0. 74095.12534959.0.731861958073700.73265271674130.726443074282 0. 72444082514600.7200025079613820.7320627167413.0.72764310.746430944825 0. 7070106718106548.0.727414445722.0.4083074897140049954 0. 6495465447130610.655081671727000.808009477954530.87404720337316 0. 613559744440120.80548942017271000.4085009471954530.87404720337316 0. 613559744445120.6101826067261000.462451719701720.87004720337316 0. 613559741445120.101826067261000.462451719701720.87004720337316 0. 6135597414643720.001826067261000.462451719701720.87004720337316 0. 6135597053066270.101826067261000.4055137197193614.00.5066137619310 0. 61351991091070.557010214586070.55731017051614.00.5066137619310 0. 53540931044224310.5070570124586070.557312071930.4707921030.47079210300377 0. 5355702330160220.50457723466090.553124984220470.304011412727983 0. 6344011414520.40066137619310 0. 63440114197020.55290162466627 0. 4352410274419222748.00.46551444214250.40263197103514.05066137619310 0. 5344971049000.553124984220470.305131031559016.558 0. 6354993104422410.505310342431070.3053130337257180.4822766677782 0. 4429811222784.00.405310424341070.4842114257040.0.00137183414 0. 53449714419222748.00.40531498341070.3149074010140.3149274053791230.47079210300.
-0. 737208846506485, -0. 73186799043613, -0. 748136394521459, -0. 745057785441460, -0. 74029515354959, -0. 73061566877370, -0. 7326527167414, -0. 7464439044825, -0. 74244302851467, -0. 72002539781382, -0. 733573025283819, -0. 731674397146088654, -0. 671505971460548, -0. 725002539712700, -0. 68805099735433, -0. 67402703573116, -0. 671559974447019, -0. 668508267172700, -0. 68805099735433, -0. 67402703573116, -0. 671559974447019, -0. 668508267172700, -0. 68805099735433, -0. 67402703573116, -0. 6715599744161464, -0. 652504128414227, -0. 6248142849702, -0. 631124444863776, -0. 6135211540546127, -0. 61397206276109, -0. 6035110414041269, -0. 600516711163278, -0. 6135211540546127, -0. 6139787101290874, -0. 63857185746439, -0. 8409139784639, -0. 5155971011019672, -0. 530457197216664, -0. 53571957464439, -0. 640914978778, -0. 5355971210110402, -0. 530457197216664, -0. 53571957464639, -0. 8409149784639, -0. 53559721210119012, -0. 530457197216664, -0. 54537195749463, -0. 6409119579123, -0. 53597021210119012, -0. 530457197216664, -0. 45837149797123, -0. 4797921063123, -0. 47992106312, -0. 530457197216664, -0. 4583714973743, -0. 4402766677742, -0. 42989129279 4, -0. 447531612444412, -0. 4381142385784, -0. 44822766677742, -0. 42980129279 4, -0. 4441221445740160, -0. 438124787351834, -0. 44822766677742, -0. 4496112924794, -0. 444212445740104, -0. 4381143795183, -0. 4642376097617, -0. 410441114271943, -0. 4796112924794, -0. 444212445740104, -0. 438143735182, -0. 41315459411539704, -0. 4496112924794, -0. 444212445740104, -0. 43814372571, -0. 410441114271943, -0. 4496112924794, -0. 444212445740104, -0. 438143797041, -0. 4404131457427489, -0. 44961129421, -0. 140900, -0. 37007410216418, -0. 37111713795183, -0. 4661297607473, -0. 329801740290, -0. 37007410216418, -0. 37111713795183, -0. 3661297970471, -0. 33980530634888, -0. 4341552402947, -0. 3148480000164, -0. 2786135944944, -0. 3106617354462, -0. 24440535712127, -0. 7381948938054, -0. 72621355449444, -0. 246017314945, -0. 20704119155547, -0. 310850054014, -0. 7282155984444, -0.
- 0.757208846506485.0.753186796043613.0.76.7326527167413.0.7264430.0265785441466. 0.74005112534595.0.75318679604361568877370.0.7326527167413.0.7264430944225. 0.72444708251467.0.7200257981382.0.64831674499713.0.6307146048954 0.670106781186548.0.725474445722.0.6483164499713.0.6307146048954 0.68154894654471190.0.685081667727002.0.6483164499713452.0.5016643270797. 0.66151593954612046.0.6484992230308.0.64621577919172.0.6306643270797. 0.66151593504627.0.01032606273050.0.64651577919172.0.6306643270797. 0.6152159350627.0.01032606273050.0.6465157791971546.430.0.6497181950. 0.555569304492411.0.50058701629627.0.56371867436497.0.3006157679371. 0.535969304492411.0.50058701629627.0.56373181793614.0.5006157679371. 0.535969304492411.0.550758701858674.0.55377857456439.0.3006157679371. 0.5359569304492411.0.550758701858674.0.55377857456439.0.3006157679371. 0.535969304492411.0.550758701858674.0.55377857456439.0.300615767372782. 0.534987612922798.0.4655764957.0.55373810793514.0.5006157677372. 0.543987612954640.0.4575501624484.0.4381249797121.0.0.4767920160312. 0.54298761292798.0.4655764057.0.4532498422047.0.50318135590.0.5539. 0.544967105956400.0.44421244570400.0.488121635590.0.5339. 0.5429876956400.0.4242124545040.0.442514079121.0.0.476792106312. 0.479511396736625998.0.46557649576767.0.5338383725718.0.4862376039181855157. 0.45264142245040.0.0.1700740216418.0.31917979121.0.0.476792100312834.0. 0.4555411404490.0.0.19064198864001.0.448111057914. 0.3589693041888.0.13441224420457049.0.0.31924940401048.0.348140544409774. 0.3589693041888.0.13441524504094.0.319389404139.0.4864197815904.0. 0.35989503130220.0.1344422442144510.0.31949404001048.0.34844144504449427. 0.35895033412821.0.4200270799800.0.43442956097637.0.4184411145540449427. 0.35895033412821.0.42002701998004.0.44847.0.31938940401048.0.3484411545444940.0. 0.35989503341282.0.131101393195514.0.217051284404.0.019481374440.0.00194413744050.0.00194441442032. 0.35989503341282.0.13110419394029.0.231111173751845.0.45642976003744203.0. 0.3598950341886.0.013712314990.0.02314419420202.0.21751
- 0. 737208846506485, -0. 73186799043613, -0. 748116594521459, -0. 745057785414 460, -0. 74095112534595, -0. 730616568877370, -0. 7326527167414, -0. 7246430744252, -0. 72444302534526, -0. 72002597961382, -0. 7373025283819, -0. 7114214574430, -0. 707106781166548, -0. 72504744457252, -0. 648364449971, -0. 62071440689654, -0. 6495465447130, -0. 64508164772700, -0. 64806039773543, -0. 764067305735116, -0. 6151539544470139, -0. 64508164772700, -0. 6480509773543, -0. 76407210575316, -0. 6152173944416346, -0. 64524821042167, -0. 64631547190172, -0. 6500643270709, -0. 61521173944416346, -0. 64524821042167, -0. 645316471441262, -0. 64066137671836, -0. 5555693104492411, -0. 500758710854867, -0. 64551371073614, -0. 5066137671874, -0. 5555693104492411, -0. 500758710854867, -0. 55577167145649, -0. 9450113956807, -0. 5355093104492411, -0. 500758710854867, -0. 555731617436449, -0. 340811395890155, -0. 53559693104492411, -0. 500758710854867, -0. 5553124973614, -0. 50661376171836, -0. 5555093104492411, -0. 500758710854867, -0. 555371671545649, -0. 3153590105589, -0. 51449761292794, -0. 4457512016246437, -0. 5553716173614, -0. 50661376171836, -0. 5555093104492411, -0. 500758710854867, -0. 555312498422047, -0. 34011142729931, -0. 479501129227494, -0. 4457102454107, -0. 5053130133725718, -0. 49622766677782, -0. 429614252402, -0. 50493712454107, -0. 503313013725718, -0. 49621769210631, -0. 4296142165400, -0. 4441214451064, -0. 448114058429, -0. 30031848651152, -0. 430541139654401, -0. 444121445104, -0. 4481471059244, -0. 4562414145105449482, -0. -0. 456241140499, -0. 2090241998545776, -0. 314941240401048, -0. 3481416304649427, -0. -0. 456241140499, -0. 2090241998545776, -0. 34846555003541, -0. 41046417145103, -0. -0. 4555003130220, -0. 4041512420467, -0. 3139814001048, -0. 34814504649427, -0. -0. 35989503140222, -0. 131104402190, -0. 201941197279113, -0. 41641314054649427, -0. -0. 35989503149222, -0. 20194411915376, -0. 2348655804541, -0. 4481514504649427, -0. -0. 359805031488, -0. 3144152540039421, -0. 3148
- 0. 757208846506480.731867990436130.743165785421490.74057785441460. 0. 74005.15534595.0.731867590436130.732645717674110.7484439044825. 0. 724441082514600.720025079613820.7326527167411.0.72544130.002543 0. 707106711864.84.0.727414445722.0.648316449971.2.0.34711402495472421. 0. 6495465447310700.655081697127000.848104997154530.74847203375316. 0. 6495465447310700.655081697127000.84857484142710.6471140214574 0. 6495465447310700.655081697127000.64851464197320.2311244446817154 0. 6495465447310700.85081691727000.64851464192720.2311244446817154 0. 6453172441530770.8574514445720.6485145419720.23112444468171544 0. 655569330445243170.5007597018586740.5637180736140.5066157617371. 0. 5556933044524130.597087045866670.55531201736140.5066157617371. 0. 5556933044524130.597087045866670.55531201736140.5066157617371. 0. 555702330160220.550457725366650.54539804720130.4842766677782. 0. 6429241227184.0.48457501621446341070.5053193817257180.4842276666777782. 0. 64292841227184.0.4845750162144634.0.482187791230.4787952006312. 0. 64296841227184.0.484575016214463.0.4821877912310.478792006372. 0. 64295841227184.0.4845750162144634.0.0.3121797912310.4785276820657782. 0. 64295841227184.0.48457501621446370.0.486137102920071210.486177822066377772. 0. 64295841227084.0.484575016214463.0.44121877912310.4785270863783. 0. 6326411402490.0.2.9942419985647.0.3319372912310.4785270863783. 0.3189675012312231.0.20170714221481.0.37311779315184.0.366177711230.48661777123.0.4861778220663777712. 0.3189675012312231.0.2017041221420.0.311137939184.0.30181779520063137. 0.31896950126314988.0.33416352420847.0.331937939184.0.366177711319. 0.318667420312230.0.0.0131463420407.0.70132192183.0.0.1266279704744. 0.319667202244620.20794445352121.0.0.371137939184.0.30262791.0.20184644444444444444444444444444444444444
- 0. 737208846506485, -0. 73186799043613, -0. 748116594521459, -0. 745057785414 460, -0. 74095112534595, -0. 730616568877370, -0. 7326527167414, -0. 7246430744252, -0. 72444302534526, -0. 72002597961382, -0. 7373025283819, -0. 7114214574430, -0. 707106781166548, -0. 72504744457252, -0. 648364449971, -0. 62071440689654, -0. 6495465447130, -0. 64508164772700, -0. 64806039773543, -0. 764067305735116, -0. 6151539544470139, -0. 64508164772700, -0. 6480509773543, -0. 76407210575316, -0. 6152173944416346, -0. 64524821042167, -0. 64631547190172, -0. 6500643270709, -0. 61521173944416346, -0. 64524821042167, -0. 645316471441262, -0. 64066137671836, -0. 5555693104492411, -0. 500758710854867, -0. 64551371073614, -0. 5066137671874, -0. 5555693104492411, -0. 500758710854867, -0. 55577167145649, -0. 9450113956807, -0. 5355093104492411, -0. 500758710854867, -0. 555731617436449, -0. 340811395890155, -0. 53559693104492411, -0. 500758710854867, -0. 5553124973614, -0. 50661376171836, -0. 5555093104492411, -0. 500758710854867, -0. 555371671545649, -0. 3153590105589, -0. 51449761292794, -0. 4457512016246437, -0. 5553716173614, -0. 50661376171836, -0. 5555093104492411, -0. 500758710854867, -0. 555312498422047, -0. 34011142729931, -0. 479501129227494, -0. 4457102454107, -0. 5053130133725718, -0. 49622766677782, -0. 429614252402, -0. 50493712454107, -0. 503313013725718, -0. 49621769210631, -0. 4296142165400, -0. 4441214451064, -0. 448114058429, -0. 30031848651152, -0. 430541139654401, -0. 444121445104, -0. 4481471059244, -0. 4562414145105449482, -0. -0. 456241140499, -0. 2090241998545776, -0. 314941240401048, -0. 3481416304649427, -0. -0. 456241140499, -0. 2090241998545776, -0. 34846555003541, -0. 41046417145103, -0. -0. 4555003130220, -0. 4041512420467, -0. 3139814001048, -0. 34814504649427, -0. -0. 35989503140222, -0. 131104402190, -0. 201941197279113, -0. 41641314054649427, -0. -0. 35989503149222, -0. 20194411915376, -0. 2348655804541, -0. 4481514504649427, -0. -0. 359805031488, -0. 3144152540039421, -0. 3148

	.242980179903264,0.248927605745720,0.254865659604514,0.260794117915276,
0	.266712757474898,0.272621355449949,0.278519689385053,0.284407537211272,
	.290284677254462,0.296150888243623.0.302005949319228,0.307849640041535.
	.313681740398891.0.319502030816015.0.325310292162263.0.331106305759876.
Ó	.336889853392220,0.342660717311995,0.348418680249435,0.354163525420490,
	.359895036534988,0.365612997804774,0.371317193951837,0.377007410216418,
	.382683432365090,0.388345046698826,0.393992040061048,0.399624199845646.
	.405241314004990, 0.410843171057904, 0.416429560097637, 0.422000270799799,
	.427555093430282,0.433093818853152,0.438616238538527,0.444122144570429,
	.449611329654607,0.455083587126344,0.460538710958240,0.465976495767966,
	.471396736825998,0.476799230063322,0.482183772079122,0.487550160148436,
	.492898192229784,0.498227666972782,0.503538383725718,0.508830142543107,
	.514102744193222.0.519355990165589.0.524589682678469.0.529803624686295.
	.534997619887097,0.540171472729892,0.545324988422047,0.550457972936605,
	.555570233019602,0.560661576197336,0.565731810783613,0.570780745886967,
	.575808191417845,0.580813958095765,0.585797857456439,0.590759701858874,
	.595699304492433.0.600616479383869.0.605511041404325.0.610382806276309.
	.615231590580627,0.620057211763289,0.624859488142386,0.629638238914927,
	.634393284163646,0.639124444863776,0.643831542889791,0.648514401022112.
	.653172842953777,0.657806693297079,0.662415777590172,0.6669999922303638.
	.671558954847018,0.676092703575316,0.680600997795453,0.685083667772701.
	.689540544737067,0.693971460889654,0.698376249408972,0.702754744457225, .707106781186547,0.711432195745216,0.715730825283819,0.720002507961382,
	.724247082951467,0.728464390448225,0.732654271672413,0.736816568877370,
	.740951125354959.0.745057785441466.0.749136394523459.0.753186799043612.
	.757208846506484,0.761202385484262,0.765167265622459,0.769103337645580, .773010453362737,0.776888465673233,0.780737228572094,0.784556597155575,
	.788346427626606.0.792106577300212.0.795836904608883.0.799537269107905.
	.803207531480645.0.806847553543799.0.810457198252595.0.814036329705948.
	.017584813151584.0.021102514991105.0.024509302785025.0.020045045257755.
	.031469612302545,0.834862074986300,0.830224705554838,0.841554977436898,
	.844853565249707,0.848120344803297,0.851355193105265,0.854557988365401, .857728610000272,0.860866938637767,0.863972856121586,0.867046245515693,
	.870086991108711,0.873094978418290.0.876070094195407.0.879012226428633,
	.001921264340355,0.004797090430930,0.007639620402054,0.090440723244750,
	.893224301195515,0.895966249756185,0.898674465693954,0.901348847046022,
	.903989293123443,0.906595704514915,0.909167983090522,0.911706032005430,
	.914209755703531,0.916679059921043,0.919113851690058,0.921514039342042,
	.923879532511287,0.926210242138311,0.928506080473216,0.930766961078984,
	.932992798834739,0.935183509938948,0.937339011912575,0.939459223602190,
	.941544065183021,0.943593458161960,0.945607325380521,0.947585591017741,
	.949528180593037,0.951435020969008,0.953306040354194,0.955141168305771,
	.956940335732209,0.958703474895872,0.960430519415566,0.962121404269042.
	.963776065795440,0.965394441697689,0.966976471044652,0.968522094274417
	.970031253194544,0.971503890986252,0.972939952205560,0.974339382785576.
	.975702130038529,0.977028142657754,0.978317370719628,0.979569765685441
	.980785280403230,0.981963869109555,0.983105487431216,0.984210092386929
	.985277642388941,0.986308097244599.0.987301418157858.0.988257567730750.
	.989176509964781,0.990058210262297,0.990902635427780,0.991709753669099
	.992479534598710,0.993211949234795,0.993906970002356,0.994564570734255
	,995184726672197,0.995767414467660,0.996312612182778,0.996820299291166
	.997290456678690.0.997723066644192.0.998118112900149.0.998475580573295
	.998795456205172,0.999077727752645,0.999322384588349,0.999529417501093
	.999698818696204.0.999830581795823.0.999924701839145.0.999981175282601
1	I contract the second

## File: curvelib.c

/		
	File:	curvelib.c
**	Intent:	Curve arithmetic functions.
••	Routines:	void chain info( int *, int, double *, COORD *)
		<pre>void position( double *, double, int * )</pre>
•/		0

#include "..\include\humanf.h" #include "..\global\global.h"

```
function: chain_info
func
```

```
fscanf( fp, "ld\n", idatabase->vsize ): /* vector size */
void
position( runlen, clen, index )
double *runlen;
double clen;
int *index;
                                                                                                                                                                   database->vector =
    (double *)=alloc( sizeof(double *)*database->nvectors 1;
    database->vector_id =
    (char *=malloc( sizeof(char *)*database->nvectors 1;
    database->vector_ctr =
        (inr *)malloc( sizeof(int)*database->nvectors 1;
             int 1 = 0;
             ::
                         Determine which point is closer the desired curve length.
                                                                                                                                                                    fclose( fp );
             return( 1 ):
                                                                                                                                                       1
             *index = 1;
                                                                                                                                                      load vector
To load a vector from a currently opened database file
into the memory.
fp file pointer
database printer to the database in memory.
vnum vector number
                                                                                                                                                                    Function:
Intent:
          File: curvelib h
                                                                                                                                                                    Arguments:
                                                                                                                                                       */
static void
load_vector(fp, database, vnum)
FILE * fp;
DATABASE *database;
int vnum;
 /•-
                                        curvelib.h
Curve arithmetic function declarations.
             File:
Intent:
 #ifndef _CURVELIB_H
#define _CURVELIB_H
                                                                                                                                                                                 but(80);
                                                                                                                                                                     char
                                                                                                                                                                    int
int
                                                                                                                                                                               len:
 #define ROOT2 1.4142136
                                                                 /* value of root 2 */
                                                                                                                                                                    fgets( buf, 80, fp );
len = strien( buf );
buf[len-1] = '\0';
                                                                                                                                                                                                                      /* get the vector id */
/* string length included the newline */
/* replace newline with a null */
 Directions defined by neighbor[] in global.c, which defines the
neighboring pixel positions and their distances. They are also used
as chain code in a clockwise manner.
                                                                                                                                                                    database->vector_id[vnum] = (char *)malloc( sizeof(char)*len );
strcpy( database->vector_id[vnum], buf );
             0 1 2
7 X 3
6 5 4
                                       X is the current pixel.
                                                                                                                                                                     fscanf( fp, "ld\n", idatabase->vector_ctr(vnum) );
                                                                                                                                                                    */
#define NWEST
#define NORTH
#define NEAST
#define EAST
#define SEAST
#define SEAST
#define SWEST
#define WEST
                                                                                                                                                                     £
                                                                                                                                                        Function: save database
Intent: To save a face profile database from memory into
a file.
Arguments: database database in memory.
 ::
              Function declarations.
 */
extern void chain_info( int *, int, double *, COORD *):
extern void position( double *, double, int *):
                                                                                                                                                                                                                      database in memory.
name of the output file.
                                                                                                                                                                                              outfile
  sendit
                                                                                                                                                        void
save_database( database, outfile )
DATABASE database:
char *outfile:
           File: database.c
                                                                                                                                                                     FILE *fp:
int i, j/
                                                                                                                                                                     fp = topen( outfile, "w" );
                                        database.c
Database.d
Database maipulation functions,
int lead database( DATABASE *, int )
int lead database( DATABASE *, char *)
void save database( DATABASE *, char *)
void save database( DATABASE *, double *, int *)
int bind vector( DATABASE *, double *, int )
int find_vector( DATABASE *, double *, int )
int find_vector( DATABASE *, double *, int )
                                                                                                                                                                     tprintf( fp, "ld\n", database.nvectors );
fprintf( fp, "ld\n", database.vsize );
  File:
Intent:
Routines:
                                                                                                                                                                     for ( i * 0; i < database.nvectors; i++ ) [
    fprint( fp, "is\n", database.vector_id[i] );
    fprintf( fp, "id\n", database.vector_ctr[i] );</pre>
                                                                                                                                                                                  for ( j = 0; ) < database.vsite: j++ )
fprintf( fp, ^\-8.6f\n*, database.vector[i]]) );</pre>
                                                                                                                                                                     j)
  finclude "..\include\humanf.h"
                                                                                                                                                                     fclose( fp );
 /:----
::
:/
                                                                 .....
                                                                                                                                                        \mathbf{P}
                                         create_database
To create a new database.
database pointer to the database in memory.
waize vector sile of the new database.
                Function:
                Intent
               Acquments:
                                                                                                                                                        add vector
To add a new vector into the database.
database pointer to the database in memory.
vector the vector to be added.
vid vector id.
                                                                                                                                                                     Function:
Intent:
Arguments:
  void
  void
create_database( database, vsize )
DATABASE *database;
int vsize;
   int
                                                                                                                                                        void
add_vector(_database, vector, vid )
DATABASE == *database;
                database->nvectors = 0;
database->vsize = vsize;
  a,
                                                                                                                                                         double
char
                                                                                                                                                                                   *vector:
*vid:
                                        load database
To load a face profile database from a file into
the senory.
database pointer to the database in memory.
infile name of the input file.
0 unable to open input file.
1 operation second
                                                                                                                                                                     double "'vector_array:
char "'vector_id_array:
int 'vector_ctr_array:
int nv:
int i:
  ......
                Function:
Intent:
                                                              pointer to the database in memory.
name of the input file.
unable to open input file.
operation successful.
                Arguments:
                                                                                                                                                                                                                    /* increment the number of vectors */
                                                                                                                                                                      nv = ++database->nvectors;
               Return value:
   •
                                                                                                                                                                      ...
                                                                                                                                                                                   Save the array pointers before allocating a new one.
   int
  int
load_database( database, infile )
DATREASE *database:
char *infile:
                                                                                                                                                                      vector_array = database->vector:
vector_id_array = database->vector_id;
vector_ctr_array = database->vector_ctr;
                static void load_vector( FILE *, DATABASE *, int ):
                                                                                                                                                                      ::
                                                                                                                                                                                  Allocating space for the new arrays of pointers.
                FILE *fp:
                                                                                                                                                                      */
database->vector = (double **)malloc( sizeof(double *)*nv );
database->vector_id = (char **)malloc( sizeof(char *)*nv );
database->vector_ctr = (int *)malloc( sizeof(int)*nv );
                int
                         nv:
i:
               if ( (fp = fopen( infile, "r" )) == NULL ) return( D ): /* unable to open input file */
                                                                                                                                                                                   Transferring the data from the arrays to the new arrays.
                fscanf( fp, *ld\n*, idatabase->nvectors ): /* #vectors in the file */
```

for ( i = 0; i < nv-1; i++ ) {
 databas=-vector[i] = vector\_array[i];
 databas=-vector\_idi] = vector\_id\_array[i];
 databas=-vector\_ctr[i] = vector\_ctr\_array[i];
}</pre> :: Allocating space for the new vector and storing it. /\* Allocating space for the new vid and storing it. '/database->vector\_id[nv-1] \*
 (char \*]malloc( sizeof(char)\*(strlen(vid)+1) );
strcpy( database->vector\_id[nv-1], vid ); ... Setting new vector counter to 1. database->vector\_ctr[nv-1] = 1: /····· Free memory space from old arrays. free( vector\_array );
free( vector\_id\_array );
free( vector\_ctr\_array ); i 
 Function:
 blend\_vector

 Intent:
 To blend a vector into the existing one in the database.

 Blending here amag averaging.

 Arguments:
 database pointer to the database in memory.

 vector in the new vector to be blended.

 vnum
 vector number of the stiming vector.

 Return value:
 0

 1
 operation successful.
 ...... if ( vnum >= database->nvectors )
 return( 0 ); /\* invalid vector number \*/ vector\_ctr = database->vector\_ctr[vnum]; vector = database->vector[vnum]; database->vector\_ctr[vnum]++; return(1); à Punction: find vector\_id Intent: To find the vector given a vector id. Arguments: database pointer to the database in memory. vector\_id the vector id. Return value: >= 0 the vector number. -1 vector not found. int vector\_id( database, vector\_id ) AASE database: \*vector\_id; find DATABASE char int i: return -1; File: database.h /\*-File: Intent: database.h Database manipulation function declarations. #ifndef \_DATABASE\_H #define \_DATABASE\_H #include "..\include\humanf.h" extern void create database(DATABASE \*, int ); extern int load\_database(DATABASE \*, char \*); extern void save\_database(DATABASE \*, char \*); extern void add\_wector(DATABASE \*, double \*, char \*); extern int blend\_wector(DATABASE \*, double \*, int ); extern int find\_wector(DATABASE \*, char \*); sendit

File: extract c / ........ extract.c Curve extraction from binary images. Chain code is used for curve representation. int where is boundary( int, int, int, double, void get\_chain\_code( COORD, int, int, int, double, int \*, int, int \*, COORD \*, double \*) file: Intent: Routines: .. #include \*..\include\humanf.h\*
#include \*..\fft\complex.h\*
#include \*..\global\global.h\* ... Chain code generation traverse directions. #define CM 0 #define CCM 1 /\* clochwise \*/ /\* counter-clochwise \*/ 1 .... The outer pixel relative positions of the template used in the check\_pixel() function. the chect\_pliel() funct: state CookD \_ circle16[] \* ( ( -2 - 2 - 1), ( -2 - 2 - 1), ( -2 - 3 - 1), ( 10, -3 - 1), ( 1, -3 - 1), ( 3, 0 - 1), ( 3, 0 - 1), ( 3, 0 - 1), ( 3, 0 - 1), ( 2, 2 - 2), ( 1, 3 - 1), ( 2, 2 - 2), ( -3, 1), ( -3, 1), ( -3, 1), ( -3, 1), ( -3, 1), ( -3, 1), ( -3, 1), ( -3, 1), ( -3, -1)
}; /\* north-west \*/ /\* north \*/ /\* north-east \*/ /\* east \*/ /\* south-east \*/ /\* south \*/ /\* south-west \*/ /\* west \*/ 1. Function: Where is boundary
 Furpose: Given the initial position in the image plane
 of a binary image, search horizontally for a boundary
 pixel. A boundary pixel is a pixel belongs to the set of pixels of solid face profile. Its value should be 0.
 Note: This procedure assumes that the person is facing
 towards x = 0.
 initial position.
 Beturn value: the x-coordinate of the boundary pixel found. int where \_is\_boundary( init\_x, init\_y ) Init\_x, init\_y; int int initial\_value: int x = init\_x; int i; initial\_value = READ\_PIXEL( init\_x, init\_y ); if (initial value == 0) ( /\* initially within the face profile \*/
for (i = ); i < init x: i++)
if ( READ\_PUTCL init x-i, init\_y; ) )
return(init\_x-i+1);</pre> 1 а. get\_chain\_code To extract the face profile curve from a binary image of a human face. The difference code of the chain code generated contains only -1, 0, and 1. init\_pos the storing pixel. Is should be the storing pixel is should be a pixel of the boundary. normal the contail direction of pixel given in init pos. It does not have to be exact, but it should be close to the reality. dif traverse direction. It is either cu for clochuise or CCE for counter-clochuise. y\_limit terminate the chain when the curve reaches this y value. distance the staring position and the current position maches this position. the init ethe chain when the current position matches this position. the init code. max\_size maximum size of the chain code. may size maximum size of the chain code. clem the total curve length. Function: Purpose: ..... Arguments: yoid get\_chain\_code( init\_pos, normal, dir, y\_limit, distance, stop\_pos, chain, max\_mize, npoints, end\_pos, clen ) COORD init\_pos: int normal: int dir: int y imit: double distance: COORD stop por int

```
int
int
COORD
double
                     max_size:
    npoints:
    end_pos:
    clen;
                                                                                                                                                                                                                                                                   #define CCW
                                                                                                                                                                                                                                                                                                           1
                                                                                                                                                                                                                                                                   extern int where_is_boundary( int, int );
extern void get_chain_code( COORD, int, int, int, double, COORD,
int *, int, int *, COORD *, double * );
                      static int
                                                                 check_pixel( int, int, int );
                      int durr_x, durr_y;
int next_pos:
int next_x, next_y;
int next_normal;
int np;
double len = 0.0;
double dust = 0.0;
int i;
                                                                                                                                                                                                                                                                    sendif
                                                                                                               /* next neighbor position to check */
/* coordinates of the next position */
/* normal of the next pixel */
/* array pointer for the chain code */
/* initial curve length */
/* distance between two pixels */
                                                                                                                                                                                                                                                                                     File: feature.c
                       \begin{array}{l} {\rm chain}[0] = {\rm curr}_X = {\rm init}_{100,Xi} \; /^* \; {\rm initial} \; {\rm coordinates} \; {\rm in} \; {\rm chain} \; {\rm code} \; */ \\ {\rm chain}[1] = {\rm curr}_Y = {\rm init}_{100,Yi} \; /^* \; ... \; */ \\ {\rm np} = 2i \\ {\rm orightarrow} \; {\rm coordinates} \; {\rm orightarrow} \; {\rm coordinates} \; {\rm orightarrow} \; {\rm orightarrow} \; {\rm coordinates} \; {\rm orightarrow} \; {\rm coordinates} \; {\rm orightarrow} \; {\rm orightarrow} \; {\rm orightarrow} \; {\rm coordinates} \; {\rm orightarrow} \; {\rm orightarrow} \; {\rm orightarrow} \; {\rm coordinates} \; {\rm orightarrow} \; 
                                                                                                                                                                                                                                                                   .....
                                                                                                                                                                                                                                                                                                                                     feature.c
feature.c
feature.c
from.chain.code. The person is assumed to be
from.chain.code. The person is assumed to be
from.chain.code. The person is assumed to be
void nome_tip(COORD *)
void nome_top(COORD, COORD *)
void uper_termination(COORD, COORD *)
void chin_position(COORD, COORD, COORD, COORD,
void lower_termination(COORD, COORD, COORD,
COORD *)
                                                                                                                                                                                                                                                                                           File:
Intent:
                   Routines
                                                                   if (check_pixel(next_x, next_y, next_normal) ))
    i += 3 ) (
    chain[np++] = next_pos;
    curt_x = next_x;
    curt_y = next_x;
    normal = next_normal;
    len += neighbor[next_normal].len;

                                                                                                                                                                                                                                                                   finclude *..\include\humanf.h*
finclude *..\vincelib\filters.h*
finclude *..\vurvelib\curvelib.h*
finclude *..\guturelib\filtylobai.h*
finclude *..\extract\extract.h*
                                                                                                                                                                                                                                                                   #define NOSE_WIN_TOP
#define NOSE_WIN_BOTTOM
#define RATIO_A
#define RATIO_B
                                                                                                                                                                                                                                                                                                                                                   160 /* 1/3 down from the top */
360 /* 1/4 up from the bottom */
0.75 /* ratio for the upper termination */
0.5 /* ratio for the lower termination */
                                                                                            ·····
                                                                                                                  Compute the distance only if the curve
length is longer than the given value.
Note that a curve line is always
longer than a straight line.
                                                                                                                                                                                                                                                                   static void sark_feature( int *, int, int *, int *);
                                                                                           nose_tip
Find the nose tip position within a window area.
The nose tip is found by scanning a vertical line
across the window.
pos nose tip position.
                                                                                                                                                                                                                                                                                          Function:
Intent:
                                                                    1
                                              ¥.
                                     Arguments:
                                                                                                                                                                                                                                                                      biov
                                                                                                                                                                                                                                                                    nose_tip( pos )
COORD *pos:
                                                                                                                                                                                                                                                                                                               x, y, y2:
curr_win_top = NOSE_WIN_TOP:
not_yet: /* not yet reach the nose tip */
                                                                                                                                                                                                                                                                                            int
                                                                                                                                                                                                                                                                                            int
                                                                                                                                                                                                                                                                                           x = where_is_boundary( DAGE_DX/2, (NOSE_WIN_TOP+NOSE_WIN_BOTTOM)/2 );
                                                                                                                                                                                                                                                                                          chein[np] = 255;
*npoints = np - 1;
                                                                                          /* chain termination */
/* -2 (start_x, start_y) + 1 ( starting pt ) */
                                                                                                                                                                                                                                                                                                                  /:
...
                                                                                                                                                                                                                                                                                                                                       Adjust window top position so that the vertical
scan starts from a background area. The lowest window
top position is the center of the image.
                       end_pos->x = curr_x: /* last pixel coordinates */
end_pos->y = curr_y: /* .. */
                                                                                                                                                                                                                                                                                                                 if ( READ_PIXEL( x, curr_win_top )
curr_win_top = IMAGE_DY/2;
                                                                                   /* total curve length */
                        "clen = len:
1
                                                                                                                                                                                                                                                                                                             Function: check_pixel
Purpose: Using a technique similar to a morphological image
processing basic operation. It uses a template of
16 outer pixels to smooth out the face profile.
Arguments: x, y the location of the pixel to be checked.
normal the mormal direction of the pixel.
Return value: 0 not a valid pixel.
1 valid pixel.
                                                                                                                                                                                                                                                                                                                if ( not_yet ) x--;
                                                                                                                                                                                                                                                                                                               else
break;
static int
check_pixel( x, y, normal )
                                                                                                                                                                                                                                                                                           1
 int
                       x, y;
normal:
                                                                                                                                                                                                                                                                                           x++;
                        int
                                         p:
11
                                                                                                                                                                                                                                                                                            /·
··
                                                                                                                                                                                                                                                                                                                Find out the center position of the nose tip.
                        x -= circle16[normal*2].x;
y -= circle16[normal*2].y;
                                                                                                                                                                                                                                                                                            y = NOSE WIN BOTTOM;
while ( READ_PIXEL( x, y ) 1= 0 ) (
                       y -- 1
                                                                                                                                                                                                                                                                                            1
                                                                                                                                                                                                                                                                                           y2 = y;
while ( READ_PIXEL( x, y2 ) == 0 ) (
y2--;
                                              p = MOD16( p-1 );
                                                                                                                                                                                                                                                                                            )
y2++:
                        í
                       return 1:
                                                                                                                                                                                                                                                                                            pos->x = x:
pos->y = (y+y2)/2:
                                                                                                                                                                                                                                                                     ŧ.
                  File: extract.h
                                                                                                                                                                                                                                                                      nose_bottom
Find the nose bottom position given the nose tip
position. The nose bottom is defined as the first
positive curvature change.
start_pos nose tip position.
pos nose bottom position.
                                                                                                                                                                                                                                                                                            Function:
Intent:
 .....
                                                                      extract.h
Curve extraction function declarations.
                                                                                                                                                                                                                                                                                            Arguments:
                        File:
Intent:
                                                                                                                                                                                                                                                                      */
void
nose_bottom(start_pos, pos)
COORD start_pos;
COORD *pos;
 Hifndef _EXTRACT_H
 ...
                      Chain code generation traverse directions.
                                                                                                                                                                                                                                                                                            cooRD out_pos:
int *chain:
                                                                                                                                                                                                                                                                                            int
  #define CW 0
                                                                                                                /* clockwise */
                                                                                                                                                                                                                                                                                                                  npoints:
```

/\* counter-clockwise \*/

) -= 0 )

/\* just pass the nose tip \*/

/\* move back to the nose tip \*/

/\* a position outside of the image \*/
/\* chain code \*/
/\* number of pixels in the chain \*/

```
COORD end_pos;
double clen;
int l;
                                                                                                                   /* last pixel position in the chaln */
/* curve length of the chaln */
                                                                                                                                                                                                                                                                                                                     */
done = 1;
                                                                                                                                                                                                                                                                                                                    int *deta;
double *runlen;
COORD *location;
Int fpoint{8{;
int nfp;
                                                                                                                  /* choin codo w/o the starting coord */
/* run-length of the curve */
/* coordinates of eoch pixel */
/* feoture points 'indexes in data{{ */
/* number of feature points found */
                                                                                                                                                                                                                                                                                                                                          lf { READ_PIXEL{ x, y } != 0 } {
    done = 0;
    breok;
                      out_pos.x = IMAGE_DX;
out_pos.y = IMAGE_DY;
                      chain = {int *)mailoc{ sizeof{int}*256 };
                                                                                                                                                                                                                                                                                                                     ł
                     get_chain_code{ start_pos, %EST, CCW, start_pos.y+TMAGE_DY/8,
9999., out_pos, choin, 256, &npoints, &end_pos, &cien );
                                                                                                                                                                                                                                                                                                                    if { done == 1 ) {
    pos->x = location{l{.x;
    pos->y = location{l}.y;
    break;
                      dato = {int *}molloc{ sizeof(int)*npoints );
runlen = {double *}molloc( sizeof(double)*npoints );
location = {COORD *}malloc{ sizeof(COORD)*npoints );
                                                                                                                                                                                                                                                                                                                    )
                                                                                                                                                                                                                                                                                               )
                       /*
                                              Prepore data for processing, data() contains all the volues in chain(( without the starting coordinates, data(0) is made equal to the first direction code in the input chain code.
                                                                                                                                                                                                                                                                                               free{ chain );
free{ runlen );
free{ location );
                                                                                                                                                                                                                                                                        h.
                      -/
data{D{ = choin{2};
icopy{ chain+2, data+1, npoints-1 };
                                                                                                                                                                                                                                                                                                                                                                                             -----
                                                                                                                                                                                                                                                                                                                                          upper_tecmination
Find the upper terminoting position given the nose to
and the nose botton positions. A reference length is
computed from the distance between the two given
positions. The upper terminoting position is defined
as the pixel on the curve from which the distance to
treference length. Is proportional to the
reference length.
nose_bottom_pos nose top position.
pos____upper terminating position.
                       /*
•••
                                                                                                                                                                                                                                                                         Function:
Intent:
                                              Get the run-lengths and the locations of each pixel on the curve.
                       chain_info{ chain, npoints, runlen, location );
                       /*
                                              Apply a series of filters to locate the feature points on the curve.
                                                                                                                                                                                                                                                                                             Arguments:
                      */
incr{ doto, npoints );
sum{ data, npoints, 7 );
sum{ data, npoints, 11 );
median( data, npoints, 11 );
clipping{ data, npoints, 3 );
                                                                                                                                                                                                                                                                        void
uppet_termination( nose_top_pos, nose_bottom_pos, pos )
COORD nose_top_pos;
                                                                                                                                                                                                                                                                                              counstion{ nose_t
nose_top_pos;
nose_bottom_pos;
*pos;
                                                                                                                                                                                                                                                                         COORD
COORD
                        mark_feoture{ doto, npoints, fpoint, &nfp );
                                                                                                                                                                                                                                                                                              double dist;

lnt *choin;

COORD out_pos;

COORD end_pos;

int npoints;

double clen;
                                                                                                                                                                                                                                                                                                                                                                                 /* distance & nose top ond bottom */
/* choin code */
/* a position outside of the image */
/* last pixel position in the chain */
/* number of pixels in the chain */
/* curve length of the choin */
                       /*
**
                                               The first positive turn is considered the nose bottom.
                      out_pos.x = IMAGE_DX;
out_pos.y = IMAGE_DY;
                                             )
                        free{ choin );
free{ dato );
free{ runlen );
free{ location );
                                                                                                                                                                                                                                                                                                choin = {int *}molloc { sizeof {int} *256 };
                                                                                                                                                                                                                                                                                               )
                                                                                                                                                                                                                                                                                               _____
                                                                                                                                                                                                                                                                                               pos->x = end_pos.x;
pos->y = end_pos.y;
/*********
                                                                    nose top

Find the nose top position given the nose tip and the

nose bottom positions. The nose top is defined as

o pixel on the profile of the nose. When o line is

drawn between that position and the nose bottom, the

immediate extension of the line from that position

must be within the face.

nose tip pos nose bottom position.

pos nose top position.
                        Function:
Intent:
                                                                                                                                                                                                                                                                                               free{ chain );
                                                                                                                                                                                                                                                                        )
                                                                                                                                                                                                                                                                        /*-----
**
**
**
                                                                                                                                                                                                                                                                                                                                          chin position

Find the chin position given the nose top and the nose

bottoe positions. The chin position is 16 Jound by the

curvature analyzis of the curve below the nose bottoe.

The first feature point with a negotive curvature

and the distance from which to the nose bottom

is greater than the distance between the nose top

and the nose bottom is considered the chin position.

nose top pos nose tip position.

nose_bottom pos most by position.

pos chin position.
                      Arguments:
                                                                                                                                                                                                                                                                                                Function:
Intent:
vvid
nose_top(nose_tip_pos, nose_bottom_pos, pos)
COORD nose_tip_pos, nose_bottom_pos;
COORD *pos;
                                                                                                                                                                                                                                                                        ... curvature analysis of the cu
... The itrat feature point with
... and the distance from which
... is greater than the distance
... and the nose bottom is consi
... Arguments: nose_toppos nose bottom
... post
... post
... post
... post
... conspondent constant
... component for post
... post
                        COORD out_pos;
int *chain;
int npoints;
COORD end_pos;
double clen;
double dist;
double d;
int x, y;
int i;
int done;
                                                                                                                    /* o position outside of the image */
/* number of pixels in the choin */
/* last pixel position in the chain */
/* curve length of the chain */
/* distance from the nose bollow */
/* distance away from the nose */
                                                                                                                                                                                                                                                                                               COORD out_pos;
int *chain;
int npoints;
COORD end_pos;
double clen;
double dist;
double ref_len;
int 1;
                                                                                                                                                                                                                                                                                                                                                                                           /* o position outside of the image */
/* choin code */
/* number of pixels in the chain */
/* last pixel position in the chain */
/* curve length of the chain */
/* distance i nomes top and hostcom */
/* distance i nomes top and bottom */
                         double *runlen;
COORD *location;
                                                                                        /* run-length of the curve */
/* coordinates of each pixel */
                         out_pos.x = IMAGE_DX;
out_pos.y = IMAGE_DY;
                        chain = {int *)malloc{ sizeof(int)*512 );
                                                                                                                                                                                                                                                                                                                                                                                           /* chain code w/o the starting coord */
/* run-length of the curve */
/* coordinates of each pixel */
/* feature points indexes in data{{ */
/* number of feature points found */
                                                                                                                                                                                                                                                                                                Int *deta;
double *runlen;
COORD *location;
int fpoint{8{;
int nfp;
                        get_chain_code{ nose_tip_pos, SWEUT, CW, NOSE_WIN_TOP,
9999., out_pos, chain, 512, &npoints, &end_pos, &clen );
                         runlen = {double *}malloc { sizeof {double} *npoints {;
location = {COORD *}malloc { sizeof {COORD} *npoints };
                                                                                                                                                                                                                                                                                                out_pos.x = IMAGE_DX:
out_pos.y = IMAGE_DY;
                         /*
                                                Get the run-length and the location of each pixel on the curve.
                                                                                                                                                                                                                                                                                               chain_info{ chain, npoints, runlen, location );
                                                                                                                                                                                                                                                                                               chain = (int *)malloc( slzeof(int) *512 );
                         /*
**
                                                                                                                                                                                                                                                                                              get_chain_code( mose_bottom_pos, SWEST, CCW, IMAGE_DY-1, 2.*ref_len,
out_pos, chain, S12, inpoints, iend_pos, iclen ):
                                                 For each pixel on the curve, test to see if it reaches the nose top position.
                         */
for { i = 0: i < npoints; i++ ) {
    dist = 01STANCE{ nose_bottos_pos.x, nose_bottos_pos.y,
        location(i(.x, location(i(.y));
</pre>
                                                                                                                                                                                                                                                                                                 date = (int *)malloc( sizeol(int)*npoints );
runlen = (double *)malloc( sizeof(double)*npoints );
location = (COORD *)malloc( sizeof(COORD)*npoints );
                                                 /*
**
**
                                                                       Check the pixels away from the nose boundory.
If they are not white, thot means the nose top position
has been reached.
                                                                                                                                                                                                                                                                                                                        Prepare data for processing, doto{{ contains oll the values in chain{} without the starting coordinates, dota{D{ is }
```

```
..
                             made equal to the first direction code in the input chain code.
                                                                                                                                                                                                                                      sign == -1 is data[i] > max ; | /* -max */ fpoint[nfp] = i - max_occ/2 + 1; nfp++; found = 1;
                data[0] = chain[2];
icopy( chain+2, data+1, npoints-1 );
               /* Get the run-lengths and the locations of
** the curve.
*/
chain_info( chain, npoints, runlen, location );
                                                                                                                                                                                                                       4
                                                                                                                                                                                                         1
                                Get the run-lengths and the locations of each pixel on the curve.
                                                                                                                                                                                                     ::
                               Apply a series of filters to locate the feature points on the curve.
                 ...
                */
incr( data, npoints );
sum( data, npoints, 11 );
sum( data, npoints, 31 );
mediam( data, npoints, 11 );
clipping( data, npoints, 10 );
                                                                                                                                                                                                                                      max = data[i]: /* record ist max */
max_occ * 1;
found = 0;
                mark_feature( data, npoints, fpoint, infp );
                                                                                                                                                                                                                       1.
                               The first positive turn is considered the nose bottom.
                if ( data[fpoint[i]] < 0 & dist > ref_len ) |
    pos->x + location[fpoint[i]].x;
    pos->y + location[fpoint[i]].y;
    breat;
                                                                                                                                                                                                                      else if ( mul -= 0 )
sign = 0;
                                                                                                                                                                                                                                                                          /* exit region */
                                                                                                                                                                                                       ^{\circ}
                             160
                                                                                                                                                                                           *nfpoints + nfp;
                1
                free( chain ):
free( data ):
free( runlen ):
free( location );
                                                                                                                                                                                      File: feature.h
 ï
                                             lower_technation
Find The upper terminating position given the nose top
and the mose bottom positions. A reference length is
computed from the distance between the two given
satisfies. The upper terminating position is defined
as the pixel on the curve from which the distance to
the pixel on the curve from which the distance to
satisfies the distance of the distance to the
reference length.
nose top positions.
nose bottom_positions.
pos upper terminating position.
  feature.h
Feature points detection function declarations.
                Function:
Intent:
                                                                                                                                                                                          File:
Intent:
                                                                                                                                                                           #include "...\include\humanf.h"
                                                                                                                                                                           #ifndef _EXTRACT_H
#define _EXTRACT_H
                Arguments
                                                                                                                                                                          /*
** Function declarations.
*/
extern void nose_tip(COORD * 1;
extern void nose_top(COORD * 0);
extern void nose_top(COORD * 0);
extern void upper_termination(COORD * 00000 * 1;
extern void chin_position(COORD COORD * 0);
extern void lower_termination(COORD, COORD, COORD * 0;

 void
lower termination( nose_top_pos, nose_bottom_pos, chin_pos, pos )
COORD nose_top_pos;
COORD nose_bottom_pos;
                hose_top_pos;
hose_top_pos;
hose_bottom_pos;
chin_pos;
*pos;
  COORD
                double dist;
int *chain;
COORD out_pos;
COORD end_pos;
int npoints;
double clen;
                                                                  /* distance 1 nose top and bottom */
/* chain code */
/* a position outside of the image */
/* last pixel position in the chain */
/* number of pixels in the chain */
/* curve length of the chain */
                                                                                                                                                                            sendit
                                                                                                                                                                                      File: fft.c
                out_pos.x = IMAGE_DX;
out_pos.y = IMAGE_DY;
                                                                                                                                                                          fft.c

Fast Fourier transform.

void fft(COMPLEX *, int)

void ifft(COMPLEX *, int, int)

void lowpass(COMPLEX *, int, int)

void magnitude(COMPLEX *, double *, int)
                chain = (int *)malloc( sizeof(int)*256 );
                                                                                                                                                                                          File:
                                                                                                                                                                                          Intent:
Routines:
                #include *..\include\humanf.h*
#include "fft.h"
#include "complex.h*
                pos->x = end_pos.x:
pos->y = end_pos.y;
                free( chain );
                                                                                                                                                                           static void fft2(COMPLEX*, int, int);
static void bit_reverse(COMPLEX*, int);
extern double cos_table();
 -3
Mark feature
Mark all the feature points in the data based on the
locations of marina and minima.
data dista effert the filters.
n number of points in data.
fpoint indexes of the feature points,
nfpoints number of feature points extracted.
                Function:
Intent:
                                                                                                                                                                           /•.....
•• Function: fft
                                                                                                                                                                                         Function:
Intent:
Arguments:
                                                                                                                                                                                                                      fft
Forward FFT. N is the size of the array.
x input data in complex values.
N size of the input array.
                                                                                                                                                                           Arguments:
                                                                                                                                                                           void
fft( x. N )
COMPLEX *x;
int N;
*/
static void
mark_feature! data, n, fpoint, nfpoints )
int "data:
int "fpoint:
int "fpoint:
int "nfpoints:
                                                                                                                                                                                         fft2( x, N, FALSE ):
                                                                                                                                                                           ά.
                                                                                                                                                                          int
int
int
int
int
int
                              sign = 0;
                                                                        /* maximum magnitude in the region */
/* #occurrences of the max */
                              max:
max_occ;
mul;
i;
nfp = 0;
found = 1;
                                                                                                                                                                                         Function:
                                                                                                                                                                                                                      ifft
Inverse FFT. N is the size of the array.
x input data in complex values.
N size of the input array.
                                                                                                                                                                                                                        ifft
                                                                                                                                                                                         Intent:
Arguments:
                                                                        /* initially set to 1 */
                if (ifound ) {
    if ( dtaii) == max ) /* increment #occurrences */
        Bax_occ++;
    else if ( sign += 1 && data[i] > max ) | /* +ve reg */
        Sign == 1 && data[i] > max ) | /* -ve reg */
        Bax = data[i] /* reset #occurrences counter */
        Bax_occ = 1; /* reset #occurrences counter */
    }
}
                                                                                                                                                                                         fft2( x. N. TRUE );
                                                                                                                                                                           11
                                                                                                                                                                          lowpass
A lowpass filter that cuts out the high frequencies in
a transformed sequence. This procedure makes use of
                                                                                                                                                                                  Procedure:
Intent:
                                              )
else if ( sign == 1 44 data(i) < max || /* leave +max */
```

the identity Arguments: Return value: ess(x, N, k) ÆX \*x; N; k; int 11 if ( k < 0 (1 k > N/2 ) return 0; /\* process +ve frequencies \*/ for ( i = N/2+1; i < N - k; i++ )
 COMPLEX( x(i), 0., 0. );</pre> /\* process -ve frequencies \*/ return 1: 5 fft2 Compute either the FFT or the inverse FFT of the complex sequence x[0], ..., x[N-1] in-place, using the radix 2 declastion-in-time method (cf. Peled & Liu, 'Digital Signal Processing'). x input data in complex values. N size of the input array. inverse a flag indicates the direction of the transform: zero for forward transformation and non-zero for inverse transformation. ttt2 /\* d = 2\*(s-1) is the distance (in index) between \*/ /\* terms in butterflies at stage s; also equal to \*/ /\* the musher of distinct twiddle factors at stage s \*/ /\* stage of the computation \*/ /\* index of first term used with twiddle factor \*/ /\* difference in indexes of terms with twiddle \*/ /\* difference in powers of twiddle factors \*/ /\* opwer of twiddle factor \*/ /\* the number of distinct twiddle factors at si int si /\* stage of the computation \*/ int li /\* index of first term used with twiddle factor int line; /\* difference in indexes of terms with twiddle int pine; /\* difference in powers of twiddle factors \*/ int power; /\* power of twiddle factor \*/ int log2ofN: COMPLEX rl, r2, ctemp; /\* temporaries used in butterflies \*/ bit\_reverse( x, N ): /\* reorder the sequence using bit reversal \*/ ÷. pinc = N >> 1; : Proceed through log2ofN stages. \*/ for ( s = 0: s < log2ofN: s++ ) ( power = 0: k1 = 0; kinc <<= 1; for ( j = 0; ) < d; j++ ) {
 COMPLEX( w, COS(power, N), SIN(power, N) );
 if ( inverse )
 COMPLEX\_CONJ( w );</pre> for { k = k1; k < N; k += kinc } { COMPLEX MULT( c1, v, x(k+d) ); COMPLEX ASSIGN( c2, x(k) ); COMPLEX ASSIGN( c2, x(k) ; COMPLEX SUB( x(k), c2, r1); COMPLEX SUB( x(k+d), r2, r1); ÷. kl++;
power += pinc; 1 d <<= 1; pinc >>= 1; ï if (inverse )
for { k = 0; k < N: k++ } {
 COMPLEX( ctemp, 1.0/N, 0.0 )
 COMPLEX( MULT( x(k), x(k), ctemp );</pre> 1 bit reverse First step in the radix 2 decimation-in-time FTT. The complex sequence x[0], ..., x[N-1] is reordered in-place using bit reversal; e.g. if N = 0, x[4] = x[100b] will be interchanged with x[1] == x[001b]. X input data in complex values. N size of the input array. Function: Intent: Arguments static void bit\_ceverse( x, N ) COMPLEX "x; int N; i, j. k. n2; int

COMPLEX ctemp: j = 0; n2 = N >> 1; 1 3 ++ 8: 1 į. /\* Function: Intent: Arguments: \*\* void magnitude(x,y,N) COMPLEX \*x: double \*y: int N: I magnitude Compute the magnitude of each element in the input complex array. x input data in complex values. y magnitudes in double values. N size of the input array. int 22 for ( i = 0; i < N; i++ ) COMPLEX\_MAG( x(i), y(i) ); File: fft.h File: Intent: fft.h FFT function declarations. #include "complex.h" Mitndet \_FFT\_H Mdefine \_FFT\_H Witndef TRUE #define TRUE Wendit 1 #ifndef FALSE #define FALSE 0 #endif /• •• Macros to be with the array defined in costable.c. #define COS(x,n)
#define SIN(x,n) cos\_table[ ( (int) ((1024./n)\*x) )\1024 ]
cos\_table[ ( (int) ((1024./n)\*x+768.) )\1024 ] /\* Function declarations. \*/
\*/
extern void fft(COMPLEX \*, int);
extern void ifft(COMPLEX \*, int);
extern int lowpass(COMPLEX \*, int, int);
extern void magnitude(COMPLEX \*, double \*, int); sendit File: fgrabber.c /···· ·· ·· fgrabber.c Contains functions related to the frame grabber board. void init itexpc( void ) void freeze( void ) File: Intent: Routines: fitdet \_FRAME\_GRABBER #include "...\include\humanf.h" edetine THRESHOLD 230 /\* binarization threshold value \*/ ;; ;; ..... Function: init\_itexpc Intent: Initial the ITEX PCVISIONplus frame grabbet board. void init\_itexpc( void ) sethdw( 0x100, 0x00000L, DUAL ); setdim( 512, 512, 0 ); initialize(); select med( NDM A ); display\_med( NDM A ); display\_med( NDM A ); setlut( INPUT, LINEAR ); setlut( INPUT, LINEAR ); ï

```
**
*/
void
bsort
int
int
                                                                     .....
                                                                                                                                                                                                                                                                                       Bubble sort in ascending order.
data the input data array (modify-in-place)
n number of data points in data
                                                                                                                                                                                                                                                          Intent:
Arguments:
                     Function:
Intent:
                                                   freeze
To shap a binary image when the teyboard is hit.
  void
 freezel word h
                                                                                                                                                                                                                                                      ( data, n )
*data;
n;
                      ·····
                                     Set the input look-up table to 255 ( white ) for the input values equal or above the threshold and linear for the rest.
                                                                                                                                                                                                                                                          int
int
                                                                                                                                                                                                                                                                              1, ];
temp;
                      */
linlut( INPUT, 4 ):
contour( INPUT, 4, THRESHOLD, 256-THRESHOLD, 255 ):
                                                                                                                                                                                                                                                        ::
                                      Prepare the binary input look-up table for snapping.
                     */
threshold( INPUT, 5, 255, THRESHOLD ):
                     setlut( INPUT, 4 ): /* use the first table when grabbing */ grab( 0 ):
                                                                                                                                                                                                                                     1
                     getch();
                                                                                /* waiting for keyboard input */
                                                                                                                                                                                                                                     Function:
Intent:
Arguments:
                                                                                                                                                                                                                                                                                                icopy
Integer array copy function.
* the input integer array.
y the output integer array.
n number of data points in x.
                     setlut( INPUT, 5 ): /* use the second table when snapping */
snap( WAIT );
                     setlut( INPUT, LINEAR );
                                                                                                    /* reset back to the linear table */
 à.
                                                                                                                                                                                                                                       void
icopy( x, y, n )
int *x, *y;
int n;
 Wendit
                 File: fgrabber.h
                                                                                                                                                                                                                                                          int i;
                                                                                                                                                                                                                                                          for ( i = 0; i < n; i \leftrightarrow ) 
 y(1) = \kappa(1) :
                                                                                                                                                                                                                                       1
  /*---
                                                                                                                                                                                                                                      /* Function

* Intent:

* Argument

*/

void

inc ( deta, n )

int *data:

int n;

(
                                                              fgrabber.h
Function declarations related to the frame grabber
board.
                      File:
Intent:
                                                                                                                                                                                                                                                                                                 incr
Compute the first-order difference of the input data.
The input values are modulo-8. The output values are in
the range of -4 to +3,
data the input data array (mod)fy-in-place)
n number of data points in data
                                                                                                                                                                                                                                                          Function:
Intent:
  #ifndet_FGRABBER_H
#define_FGRABBER_H
                                                                                                                                                                                                                                                          Arguments:
 #ifdef FRAME GRABBER
                                                                   /* define only if using the frame grabber */
 extern void init_itexpc( void );
extern void freeze( void );
  Rendif
                                                                                                                                                                                                                                                                         temp1, temp2:
                                                                                                                                                                                                                                                           int
int
   Fendit
                                                                                                                                                                                                                                                           temp1 = data[0]:
data[0] = 0:
                                                                                                                                                                                                                                                          for ( i = 1: i < n: i++ ) (
    tesp2 = data[i]:
    tesp1 = tesp2 - tesp1:
    tesp1 = tesp1 > 3 7 tesp1 - 0 : tesp1;
    data[i] = tesp1 < -4 7 tesp1 + 0 : tesp1;
    tesp1 = tesp2;
    tesp1 = tesp1 = tesp2;
    te
                 File: filters.c
 /-----

File:

Intent:

Routines:

···
                                                             filters.c
filters.c
filters.c
filters.for one-dimensional integer arrays.
void baset(int *, int, int )
void baset(int *, int,
void gus(int *, int,
void gus(int *, int, int)
void clipping(int *, int, int )
                                                                                                                                                                                                                                                           £.
                                                                                                                                                                                                                                       \hat{U}_{i}
                                                                                                                                                                                                                                       Sum

Convolue the data with an all-1's ternel of site m.

The result of each value in the data array is

essentially the sum of all numbers within the window.

data the input data array (modify-in-place)

n mumber of data points in data

mumber of ternel (preferably odd number)
  #include <stdlib.h>
/-----

Punction:

Intent:

Arguments:

*/

void

median(data, n, m)

int "data;

int s;

(
                                                            median
Median fliter of Size B.
data the input data array (modify-in-place)
n number of data points in data
n filter size (preferrably odd number)
                                                                                                                                                                                                                                                            int
int
int
                                                                                                                                                                                                                                                                               •win;
sum;
nd2 = n/2;
p, i;
                                                                                                                                                                                                                                                           win = (int *)malloc( sizeof(int)*m );
                      void bsort( int *, int ):
void icopy( int *, int *, int );
                                                                                                                                                                                                                                                           int
int
int
                                         •win; •t_win;
md2 = m/2;
                                                                                               /* window size / 2 */
                                         p, i:
                      win = {int *}malloc( sizeof(int)*m );
t_win = {int *}malloc( sizeof(int)*m );
                                                                                                                                                                                                                                                            ...
                                                                                                                                                                                                                                                                         Convolution begins here.
                      dataimd21 = sum;
                                                                                                                                                                                                                                                          p = 0:
for ( i = m; i < n; i++) (
sum -= vin(p):
vin(p) = data(i):
sum += data(i):
data(i-md2) = sum:
p = (p+1)lm:
                      icopy(win, t_win, m);
bsort(t_win, m);
                      data[md2] = t_win[md2];
                     p = 0;
for [ i = m; i < n; i++ ) [
uin[p] = data[i];
icopy[ vin, t_vin, m ];
baort[ t_vin, m ];
data[i=d2] = t_vin[md2];
p = (p+1) % m;
                                                                                                                                                                                                                                                            for ( i = 0; i < md2; i++ ) /* clear the beginning and the end */ data[i] = 0;
                                                                                                                                                                                                                                                            for { i = n-md2; i < n; i++ }  /* .. */
    data[i] = 0;</pre>
                      tree( win );
tree( t_win );
                                                                                                                                                                                                                                                             free( win ):
                                                                                                                                                                                                                                        ï
                                     bsort
                                                                           .....
                      Function:
                                                                                                                                                                                                                                         /*-----
                                                                                                                                                                                                                                                           Function
                                                                                                                                                                                                                                                                                                    clipping
```

** Type defi */	hitions.	::	Function: Intent:	get_image ( _FRAME_GRABBER   To snap a binary image using the frame grabber board
<pre>#include "c:\itex #include "c:\itex /*</pre>	clincludelitexp1g.h" pclincludelitdtyp.h"	#ifdef	e BUF_SIZE _FRAME_GRABS	IER
#include <string. #include <sys\typ< th=""><th>s.h&gt;</th><th>#inclu</th><th>de "\fgrabbe</th><th>()uman: :r\fgrabber.h* 4096*4</th></sys\typ<></string. 	s.h>	#inclu	de "\fgrabbe	()uman: :r\fgrabber.h* 4096*4
<pre>#include <fontl.h: #include <io.h> #include <math.h></math.h></io.h></fontl.h: </pre>		•/	de "\include	
#include <stdio.h: #include <graph.h: #include <conio.h:< th=""><th></th><th>Ë</th><th></th><th><pre>int loadim_pc( char *, int *, int *) int loadim_pc2( char *, int, int, int ) void saveim_pc( char *, int, int )</pre></th></conio.h:<></graph.h: </stdio.h: 		Ë		<pre>int loadim_pc( char *, int *, int *) int loadim_pc2( char *, int, int, int ) void saveim_pc( char *, int, int )</pre>
include estdlib.	۰.	:	Routines:	<pre>void get_image( void ) int itex_image_header( char *, int, int, int, int, int, char * )</pre>
#ifndet_HUMANF_H #define_HUMANF_H			File: Intent:	image.c Image loading and saving routines.
File: Intent:	humanf.h Useful macros and definitions.			
			File:	image.c
File:	humanf.h			
		#define #endif	. INGAGE_DY	ann ', Accrear arte of rue made y
extern NEIGHBOR			IMAGE_DX	512 /* horizontal size of the image */ 480 /* vertical size of the image */
Mefine _GLOBAL_H	neighbor[6]:	#define #define		1
/ Witndet _GLOBAL_H			MIN(a,b)	( (a) < (b) 7 (a) . (b) )
File: Intent:	giobal.h Declarations of global variables.		Intent: MAX(a,b)	Maximum and minimum of two numbers, ( (a) > (b) 7 (a) : (b) )
··			Macro:	MXX, MIN
File:	global.h		FA85 (x)	((x) > 0.07 (x) : -(x))
			Macro: Intent:	FABS Floating point absolute function.
n (	-1, 0, 1. ) /* WEST */			
t i	1, 1, ROOT2 ), /* SEAST */ 0, 1, 1. ), /* SOUTH */ -1, 1, ROOT2 ), /* SWEST */	ødetine	MOD8 (x) MOD16 (x)	( (x)>7 7 (x)-8 : ((x)<0 7 (x)+8 : (x)) 1 ( (x)>15 7 (x)-16 : ((x)<0 7 (x)+16 : (x)) )
1	1, -1, ROOT2 ), /* NEAST */ 1, 0, 1, ), /* EAST */		Land Chr. 58	Range of operation is $-\theta < x < 16$ for MOD8, and $-16 < x < 32$ for MOD16.
	-1, -1, ROOT2 ), /* NWEST */ 0, -1, 1, 1, /* NORTH */	:	Macro: Intent:	MCD6, MOD16 Modulo of 8 and 16.
finclude "\curve ÆIGHBOR neighbor[				(double)((y1)-(y2))*(double)((y1)-(y2)))))
/ finclude *\inclu	de\humanf.h*		DISTANCE (x1,)	<pre>( sqrt( (double) ((x1) - (x2))*(double) ((x1) - (x2))* \</pre>
<ul> <li>File:</li> <li>Intent:</li> </ul>	global.c Global variables.	;	Macro: Intent:	DISTANCE Compute the distance between 2 points.
File:	global.c	·/ ·/ #define	Intent: ROUND(x)	Round a positive real number x to the closest integer $(x) = (double) ((int) (x)) < 0.5 7 (int) (x) : (int) (x)+1$
<b>m</b> '4		:	Macro:	ROUND
endif		#endif	a consection of the last of th	
xtern void su	cr(int *, int): *( int *, int, int); ipping( int *, int, int);	ødefine Øelse	WRITE_PIXEL (x	(,y) wpixel(x,y,255)
xtern void ic xtern void by	<pre>opy( int *, int *, int ); ort( int *, int );</pre>	#ifdet	FRAME GRABBE	æ
define _FILTERS_H xtern void me	dian( int ., int, int );	€else #define #endif	READ_PIKEL(x,	y)getpixel(x,y)
/ lindef _FILTERS_H		#ifdef #define	FRAME_GRABBE	р yı rpixel(x,y)
Intent:	One-dimensional integer array filter fun declarations.	ction ** •• •/	Intent:	Work with the frame grabber as well as the VGA memory. Operation depends on the flag at complilation time.
· File:	filters.h		Hacros	READ PIXEL, WRITE_PIXEL
i nç.	inters.fr		I DATAB	AGE:
File	filters.h		doub1 char int	e **vector: /* array of vectors */ **vector_id: /* array of vector id's */ *vector_ctr: /* array of vector sample counters */
			i int	<pre>nvectors: /* number of vectors in the database * vsize: /* vector size */</pre>
	<pre>data[i] &lt; 0 &amp; data[i] &gt; -t )     data[i] = 0;</pre>		struct databa	30
for ( i = ) if	: i < n: i++ ) ( data[i] > 0 && data[i] < t			e len: /* length of displacement */
int i;			int int	<pre>x: /* x incremental step */ y: /* y incremental step */</pre>
nt *data; ht n; ht t;		typedef	struct neighb	or /* neighbor pels and their distances *
/ bid Lipping( data, n,			int I COORD	y;
	data the input data array (modify-in-) n number of data points in data t threshold value	blace) typedef	struct coord (	<pre>/* integer (x,y) coordinates */ x;</pre>
Acquaents:				

```
whlle | |len = rcad| fd, bp, BUF_SIZE || | |
for | i ≠ 0: i < lcn; i++ | |
if | bp|i1 (≠ 0 |
@RITE_PIXEL| x, y |;
vold
get_imagc| vold |
               init_itexpc[[;
                                                                                                                                                                                                                                it | ++x == dx | |
x = 0;
y++;
               printf: *Press any hit to freaze the imaga * (; printf: n^{*});
               freeze II;
#else
/*----
**
**
**
                                                                                                                                                                                                  ι
                                                         -----
                                    get_image
To load a image from an ITEX image file to the VGA
memory.
                Function:
Intent:
                                                                                                                                                                                                  close: fd [;
frec! bp [;
                                                                                                                                                                                                   *width = dx;
*length = dy;
-/
void
get_imagell
l
                                                                                                                                                                                                  return 1;
               int
                           loadim_pc( char *, lnt *, int * (;
                                                                                                                                                                                   ı
               char string[BD];
int dx, dy;
                                                                                                                                                                                  /*-----
** Fr
** In
** A
** A
**
                                                                                                                                                                                                                                                    _____
                                                                                                                                                                                                                                saveia_pc
Save an uncospressed ITEX image from the VGA mcmory
at frame position [0,0] to a file.
file_name the output filename.
width width of image in pixels.
length length of image in pixels.
                                                                                                                                                                                                  Function:
Intent:
                printf( "Please enter input file name: " (;
scanf( "%s*, string (;
                                                                                                                                                                                                  Arguments:
                 if | 'loadim_pc| string, &dx, &dy | | |
    fprintf| stderr, "Unable to open input file.\n" |;
    exit| 1 |;
                                                                                                                                                                                   */
void
saveim_pc| file_name, width, length |
char *file_name;
lnt width, length;
.
                 ī.
 #endlf
                                                                                                                                                                                                   FILE
UCHAR
int
int
int
int
int
                                                                                                                                                                                                                 *fp;
*bp;
zero = 0;
x, y;
ctr;
done;
L;
 /*-
**
                                                itex_Lmage_header

This function returns information regarding the

specified TTX image tile.

file mame the image filemame.

x, y upper left-hand corner of frame memory

dx, dy size of image

format 0 FIGHT PIT: 1 COMPESSION

comment comment associated with the image file

operation failed
                 Function:
Intent:
 * * * * * * * * * * * *
                 Arguments:
                                                                                                                                                                                                   bp = |UCHAR *|malloc| BUF SIZE [;
                                                                                Comment associated with the image ,
operation failed.
operation successful. The value
represents the offset to the image
data for the beginning of the file.
                 Return value:
                                                0 > 0
                                                                                                                                                                                                   tp = topen: tile_name, "wb" 1;
                                                                                                                                                                                                  ty = topon( true mass, to f);

twrite( izero, sizeof(int), i, fp ); /* 0 comment size */

twrite( izero, sizeof(int), i, fp ); /* dx */

twrite( izero, sizeof(int), i, fp ); /* dy */

twrite( izero, sizeof(int), i, fp ); /* x origin */

twrite( izero, sizeof(int), i, fp ); /* x origin */

twrite( izero, sizeof(int), i, fp ); /* torsat - EIGMT_BIT */

for | i = 0; i < 50; i*+ | /* torsat - EIGMT_BIT */

twrite( *\0*, i, 1, fp );
  •/
 */
itex_
char
int
                 ege_headcr|file_name; x, y, dx, dy, format, comment |
*file_name:
*x, *y, *dx, *dy, *format:
*comment:
  char
                  int
char
lnt
                                                                                                                                                                                               Twin:
x = y = 0;
done = 0;
while | (done | |
bp|ctc++[ = REAQ PEXEL | x, y | == 0 7 0 : 255;
if | ++y == udth | |
x = 0;
lf | ++y == length | |
done = 1;
break;
                                 fd;
ftype[2];
csize;
                 read| fd, ftype, 2 |;
if | ftype[0| |= 'I' || ftype[1| != 'N' | | /* Image file signature */
                                 close: fd [;
return D;
                  read | fd, icmize, 2 |;
read | fd, dx, 2 |;
read | fd, dy, 2 |;
read | fd, dy, 2 |;
read | fd, y, 2 |;
read | fd, y, 2 |;
read | fd, format, 2 |;
read | fd, comment, SD |;
read | fd, comment, csize |;
                                                                              /* size fo comment area */
/* horizontal size of the image */
/* vertical size of the image */
/* upper lett-hand corner x */
/* upper lett-hand corner y */
/* upper lett-hand corner y */
/* o = EIGHT_BIT: l = COMPRESSION */
/* reserved by ITEX */
/* comment */
                                                                                                                                                                                                                   fwritel bp, 1, ctr. fp [;
                                                                                                                                                                                                   ı.
                                                                                                                                                                                                   tclose; tp (;
                                                                                                                                                                                    ı
                  closel to 1;
                                                                                                                                                                                    /*------
** Fr
** IT
**
**
**
**
**
**
**
**
**
**
                                                                                                                                                                                                                                 loadim_pc2
Load an uncompressed ITEX image into the VGA memory
at the specified frame position with a linear
reduction in size.
file_name the image filename.
x_org, y_org upper lett-hand corner of the image.
reduction reduction factor in one dimension.
0 operation failed.
1 operation successful.
                                                                                                                                                                                                                                                      _____
                                                                              /* offset to image data */
                                                                                                                                                                                                   Function:
Intent:
                  return| 64+csize |;
   ı
                                                                                                                                                                                                   Arguments:
            _____
 /*---
**
**
**
**
**
**

        Function:
        loadim_pc

        Intent:
        Load an uncompressed ITEX image into the VGA memory at frame position 10.01.

        Arguments:
        file name

        the image filename.

        width
        width

        length
        length of image in pixels.

        length
        length of image in pixels.

        Return values:
        0

        operation successful.

                                                                                                                                                                                                   Return value:
                                                                                                                                                                                     */
indim_pc2!file_name, x_org, y_org, reduction |
char *file_name;
int x_org, y_org;
int x_ourg, y_org;
   */
int
loadim_
char
int
                  pc| file_name, width, length |
*filc_name;
*width, *length;
                                                                                                                                                                                                    int
int
char
int
                                                                                                                                                                                                                 offset;
x, y, dx, dy, format;
comment[256];
fd;
*bp;
                  int offäet:
int x, y, dx, dy, format;
char comment[256];
int fd;
UCHAR *bp;
int len, temp, i;
                                                                                                                                                                                                     UCHAR
                                                                                                                                                                                                                  v_end, temp;
i, j, k, ctr;
th = reduction*reduction/2; /* threshold value */
                                                                                                                                                                                                     int
int
int
                                                                                                                                                                                                     td = open | tile_name, o_RDONLY | o_BIHARY |;
                  fd = open! file_name, o_RDONLY | o_BINARY |;
                                                                                                                                                                                                     if | format == 1 |
return D;
                                                                                                                                                                                                                                              /* unable to process compressed image */
                  if | format == 1 |
return 0; /* unable to process compressed image */
                                                                                                                                                                                                     read; fd, comment, offset [; /* move to the beginning of the data */
                    read| fd, comment, offset [; /* move to the beginning of the data */
                                                                                                                                                                                                     /-

** Now read the image data.

_setvideomode[_VRES16CoLoR [:

*/
                   /•
••
                                  Now read the image data.
                   bp = [UCHAR *[malloc] dx*reduction [;
                    bp = |UCHAR *|malloc| BUF_SIZE |;
                                                                                                                                                                                                      x_end = x_org + dx/reduction;
x = x org;
                                                                                                                                                                                                      y = y_org;
x = x_org;
                    x = y = 0;
```

```
break:
                                                                                                                                                               .
                                                                                                                                                               if ( ctr >= th ) (
    __setcolor( 255 );
    __setpixel( x, y );
                                                                                                                                                                             else
                                                                                                                                                                                       vector_ok = 1;
                                     else (
                                                                                                                                                 #ifndef _FRAME_GRABBER
    if ( lbatch )
        getch();
                                                                                                                                                                            vnum = classify( database, vector, ncompare, distance );
                                                  _setcolor(0);
_setpixel(x, y);
                                     .
                                     .....
                       1
                                                                                                                                                                            _setvideomode( _DEFAULTHODE );
                                                                                                                                                  .......
                        x = x_org;
y++;
                                                                                                                                                                            b_ctr++;
if (vector_ok_44_distance(vnum) < THRESHOLD16 ) (
    fill_page(database.vector_id(vnum));
    if (batch )
        fprintf(fp, *td/ttd\n*, b_ctr, vnum+1);</pre>
           1
           close( fd ):
free( bp ):
_setcolor( 255 );
                                                                                                                                                                            )
else (
if ( vector ok ) (
fill_page( "Unable to identify human subject" ):
if ( batch )
fprintf( fp, "t@ttD\n", b_ctr );
            return 1
                                                                                                                                                                                         i == {
    fill_page( *Unable to extract face profile* );
    if ( batch )
        fprintf( fp,
 *Id(tunable to extract face profile.\w*, b_ctr );
}
         File: image.h
1
            File:
Intent:
                                       image.h
Image loading and maving function declarations.
                                                                                                                                                                            if ( (batch )
getch();
Mitndet _IMAGE_H
                                                                                                                                                                             if (vector_ok)
show_distance(database, distance);
                                                                                                                                                                            print( "\h\nbo you wish to continue? (y/n) " ): scanf( "\ls", but ): if ( 'but ** 'n' ) breat:
extern void get_image( void );
...
            ITEX image file routines.
./
                                                                                                                                                                 3
                          itex_image_header( char *, int, int, int, int, int, char * );
loadim_pc( char *, int *, int * );
loadim_pc( char *, int, int, int );
savelm_pc( char *, int, int );
extern int
extern int
                                                                                                                                                                if ( batch )
fclose( fp );
            int
void
extern
extern
                                                                                                                                                                 print( "Goodbye.\n" ):
exit( 0 ):
sendi
         File: matching.c
                                                                                                                                                    ·····
                                                                                                                                                                Function:
Intent:
Argument:
                                                                                                                                                                                          fill page
Fill the screen with the given string.
string the input string.
.....
                                                                                                                                                    void
fill_page(string)
char *string:
                                       matching.c
This is the main program for the recognition system.
It compares the test vector of an input profile to
the vectors in the database and outputs the result.
             File:
Intent:
                                                                                                                                                    char
                                                                                                                                                                 char buf[80];
int len;
int i:
:;
Minclude "...\include\humanf.h"
Minclude "...\database\database.h
Minclude "...\classify\classify.h
Minclude "...\vector\vector.h"
                                                                                                                                                                 strcpy( buf, string );
strcat( buf, * *);
len = strlen( buf );
                                                                                                                                                                                                                  /* add some spaces */
                                                                                                                                                                 system( "cls" );
                                                                                                                                                                                                                  /* clear the screen for output */
 #define THRESHOLD16 0.000400
                                                           /* threshold for 16-point comparison */
                                                                                                                                                                                                                             /* 80 character * 20 lines */
                                                                                                                                                                  main( argc, argy )
                                                                                                                                                                 printf( "\n" );
 int
            argc;
**argv;
                                                                                                                                                     ¥
                                       fill_page( char * );
show_distance( DATABASE, double * );
             void
void
                                                                                                                                                  show distance
Display the distance from the test vector to each
vector in the database.
distance the database in use.
distance an array containing the distance
from the test vector to each vector
in the database.
              char
                                       buf[80];
*vector;
database;
ncompare;
                                                                /* test vector */
/* database being used */
/* database being used */
/* do coefs used in the cosparison */
/* vector # selected for the match */
/* distances from the test vector */
/* get vector() successing */
/* number of faces processed */
/* batch output file pointer */
             double
DATABASE
int
int
                                       ncompare.
vnum:
*distance:
vector_ok:
batch = 0:
b_ctr = 0:
*fp:
              double
int
int
              FILE
             batch = 1;
fp = fopen( "batch.out", "w" );
                                                                                                                                                                  int
char
int
                                                                                                                                                                              1:
but[80]:
space_left = 80;
              1
             printf( "Please enter the name of the database: " );
scanf( "\s", buf );
                                                                                                                                                                                                                    /* clear the screen for output */
                                                                                                                                                                  system( *cls* );
                                                                                                                                                                 for ( i = 0; i < databas.nvectors: i++ ) !
    sprint( buf, "ts 1-6.0f ",
    databas.vector idi), distance(i|*1000000 );
    space_lett -+ strlen( buf );</pre>
             if ( !load_database( idatabase, buf ) ) (
    fprintf( stderr, "Unable to open the database file.\n" );
    exit( -1 );
              1
                                                                                                                                                                              if ( space_left > 0 )
    printf( "ts", but );
else (
    printf( "ts", but );
    space_left = 80 - strien( but );

              printf( "Database loaded.\n" );
printf( "Vector size: Id\n", database.vsize |;
printf( "Number of vectors in the database: \d\n", database.nvectors );
              vector = (double *)malloc( sizeof(double)*database.vsize );
distance = (double *)malloc( sizeof(double)*database.nvectors );
                                                                                                                                                                              13
                                                                                                                                                                 3
                                                                                                                                                     1
              while (1) (
```

```
File: sampling.c
                                                          sempling.c
Open curve sampling function and display functions.
int csample[ int *, int, double, COMPLEX **]
void sdisplay(COMPLE*, int, int, int)
void cdisplay(int *, int, int)
void cdisplay2( int *, int, int, double)
File:
Intent:
Routines:
#include *..\include\humanf.h*
#include *..\curvelib\curvelib.h*
#include *..\fft\complex.h*
#include *..\global\global.h*
 #define CHAIN_TERMINATION
                                                                             255
                                                                                                /* chain code termination value */

        Function:
        csaple

        Purpose:
        To extract a specified number of interpolated
points from an open curve in from of chain code.
The curve saple array is terminated with a negative
coordinate value.

        Arguments:
        Curve_saple array is terminated with a negative
coordinate value.

        Arguments:
        Curve_saple array is terminate to be obtained.
clength

        the total length of the curve.
curve_sample the total length of the curve.
i operation failed.

        Return value:
        0

 ......
 int
 int
cample( curve, nsamples, clength, curve_sample )
int "samples:
double clength;
CMPLEX *=curve_sample;
                     COMPLEX "mample:

int curr x, curr y: /* current pixel position */

int prev x, prev y: /* previous pixel position */

double prev_pos, curr_pos: /* curve length since the Starting pt */

double desired_pos: /* desired curve len from starting pt */

double fraction: /* fraction in length between pixels */

int mat move: /* direction of next moveent */

double sien = clength/(double)(myamples-1): /* segment length */

int i = 1: /* index of curve_mample */
                      prev x = curre x = curve[0]: /* header contains starting position */
prev y = curr y = curve[1]: /* ditto */
sample[0]:x = (double)curr x: /* tirst sample is the first point */
sample[0]:y = (double)curr y: /* ditto */
desired_pos = slen: /* next sample position */
                     while { curve[m] != CHAIN_TERMINATION } (
    next_move = curve[m];
    curt_x *= neighbor[next_move].x;
    curt_y *= neighbor[next_move].y;
    curt_pos *= neighbor[next_move].len;
                                          if (curt_pos >* desired_pos ) {
    fraction * (double)(desired_pos - prev_pos) /
        (double)(curt_pos - prev_pos);
                                                               sample[i].x = prev_x + fraction*(curr_x - prev_x);
sample[i].y = prev_y + fraction*(curr_y - prev_y);
                                                               desired_pos +* slen: /* next desired position */
i++;
                                          1
                                          prev_x = curr_x;
prev_y = curr_y;
prev_pos = curr_pos;
a++;
                        ×.
                        "...
                                          Ensure the last point is included in the samples.
                       */
sample[nsamples-1].x = (double)curr_x: /* last sample is the last pt */
sample[nsamples-1].y = (double)curr_y: /* ditto */
                                                                                                     /* array termination */
/* ditto */
                       sample(nsamples).x = -1.;
sample(nsamples).y = -1.;
                                                                                                     /* inconsistent result */
                        if ( curr_pos != clength )
return 0;
                       else
                                                                                                      /* operation successful */
                                          return 1:
    1
                                                                                                             Function:
Purpose:
Arguments:
                                                               sdisplay
Output the sample points on screen.
curve sample curve sampl
nsamples number of s
                                                                                                               number of sample array.
number of samples.
coordinate of the upper-left
corner of the output frame.
                                                                frame_x, frame_y
      biou
    void
Sdisplay( curve_sample, nsamples, frame_x, frame_y)
COMPLEX *curve_sample;
int nsamples;
int frame_x, frame_y;
                        int
int
                                       curr_x, curr_y:
prev_x, prev_y;
                        curr_x = ROUND( curve_sample[0].x ):
curr_y = ROUND( curve_sample[0].y ):
wRITE_PIXEL( frame_x+Curr_x, frame_y+curr_y );
```

```
for ( i = 1: i < nsamples: i++ ) (
    curr_x = ROWND(curve_sample(i).x );
    curr_y = ROWND(curve_sample(i).y ))
    wRITD_PIXEL( frame_x+curr_x, frame_y+curr_y );</pre>
1
                                                             _____
cdisplay

Output the curve on screen,

curve chain code of the open curve

cruve chain code of the open curve

frame_x, frame_y coordinate of the upper-left

corner of the output frame.
               Function
               Purpose:
Arguments:
   hie
 cdisplay( curve, frame_x, frame_y )
int *curve:
 int
               frame_x, frame_y;
                int
int
                               curr_x, curr_y;
next_move;
m = 2;
                                                                               /* data start from curve[2] */
               curr_x = curve[0]:
curr_y = curve[1]:
                                                                               /* starting coordinate header */
/* ditto */
                WRITE PIXEL( frame x+curr x, frame y+curr y ))
               while ( curve[m] i= CHAIN_TEMMINATION ) (
    next_move = curve[m]:
    curr_x += neighbot[next_move].x:
    curr_y += neighbot[next_move].y:
    WRITE_PIXEL( frame_x+curr_x, frame_y+curr_y ):
    m++:
                Ý.
1
 edisplay2
output the curve on screen with linear reduction.
curve chain code of the open curve.
frame_X, frame_Y position of the upper-left
conter of the output frame.
reduction linear reduction factor.
                Function:
Purpose:
Arguments:
 void
cdisplay21 curve, frame_x, frame_y, reduction )
int 'curve:
int frame_x, frame_y;
double reduction;
                                                                              /* current position */
/* direction of next movement */
/* data start from curve[2] */
                 double curr_x, curr_y;
int next_move;
int m = 2;
                 ···
                                 The header contains the starting position.
                 curr_x = (double)curve(0)/reduction:
curr_y = (double)curve(1)/reduction;
                 frame_x = ROUND({double}frame_x/reduction)
frame_y = ROUND((double)frame_y/reduction)
                  WRITE_PIXEL( frame_x+ROUND(curr_x), frame_y+ROUND(curr_y) ):
                 while ( curve[m] != CHAIN_TERMINATION | |
    next_move = curve[m]:
    curry += (double)neighbor[next_move]:x/reduction:
    curry += (double)neighbor[next_move].y/reduction:
    wolfte_PICLL(!frame_xFROMD(curr_y).frameyPHOMD(curr_y) );
                 3
               File: sampling.h
                                                  sampling.h
Open curve sampling function and display function
declarations.
   File:
Intent:
  #include "..\fft\complex.h"
   #ifndef_SAMPLING_H
#define_SAMPLING_H
                                  csample( int *, int, double, COMPLEX ** );
sdisplay( COMPLEX *, int, int, int);
cdisplay( int *, int, int);
cdisplay2( int *, int, int, double);
   extern int
extern void
extern void
extern void
    sendit
               File: training.c
    training.C
This is the main program for entering new entries to
database or blending new vectors in the existing ones.
                   File:
Intent:
   #include *..\include\humanf.h*
#include *..\database\database
#include *..\vector\vector.h*
                                                               e. h*
    vold
main( argc, argv )
int argc;
char **argv;
```

```
í
        char
char
double
DATABASE
                         database_name[80];
buf[80], buf2[80];
*vector;
database;
vnum;
        int
int
int
                          vaize:
                                                /* extraction successful */
/* default value */
/* default value */
/* non-batch processing */
                         vector_ok;
new_database = 0;
same_person = 0;
batch = 0;
                                         0;
         int
int
        printf( "Please enter the name of the database: " );
scanf( "%s", database name );
       elpe i
                         printf( "Goodbye.\n" ):
exit( 0 );
                 1
         ï
        if ( inew_database )
printf( "Database loaded.\n" );
        eine
                 printf( "Database created. \n" 1:
        printf( "Vector size: ld\n", database.vsize );
printf( "Number of vectors in the database: ld\n", database.nvectors );
         vector = (double *)mailoc( sizeof(double)*database.vsize ):
        ::
                 Get face profile vector here.
        */
while (1) (
    if (get_vector(database.vsize, vector))
        vector_ot = 1;
vector_of = 0;
                 _setvideomode( _DEFAULTHODE ):
#endit
                 streat( buf, * * );
streat( buf, buf2 );
                 ï
                 else.
                                printf( "Vector discarded.\n" );
                 if ( vector_ok 44 *buf2 == 'y' ) |
blend_vector( 4database, vector, vnum ):
    save_database( database, database_name );
                          else
                                printf( "Vector discarded.\n" );
                 printf( "Do you wish to continue? (y/n) * 1;
scanf( *l15*, buf2 );
if ( *buf2 == 'n* )
breat;
                 else (
                         print("Will that be the same person? (y/n) " );
scant("N18", but2 );
if ("but2 == 'y')
same_person = 1;
else
______merson = 0;
                 ï
         1
         save_database( database, database_name );
printf( "Goodbye.\n" );
exit( 0 );
```

```
File: vector.c
                     File:
Intent:
                                                              vector.c
To obtain Fourier descriptors vectors from face
                                                              profile images.
int get_vector( int, double * )
                     Routines:
include '..\include\humanf.h'
include '..\include include '..\
include '..\currelib\currelib.n'
include '..\currelib\currelib.n'
include '..\sampling.hampling.h'
include '..\fetticsiplex.h*
include '..\ffticsiplex.h*
idefine DISPLAY_OFFSET 60 /* curve display offset */
idefine RBX_CHRIN_SIZE 1024 /* exkinum chain size */
idefine RBT_NRDNTS 20 /* founts in the reference dotted line */
idefine RATTO_A 0.75 /* ratio for the upper termination */
idefine RATTO_B 0.5 /* ratio for the lower termination */
get_vector

To get a fourier descriptors vector from a

face profile image.

value the desired vector size. It should be

FFT_SIZE/2.

vector the output vector.

0 operation failed.

1 operation successful.
                     Function:
Intent:
                 Arguments:
                   vector
Return value: 0
int
get_vector( vsize, vector )
int vsize:
 int vsize:
double 'vector:
                     static void draw_cross_marker( int, int );
static void draw_square_marker( int, int );
static void draw_dime( int, int, int, int, int );
static void trace_mark( COMPLEX *, int, COMPLEX **);
                    COORD nose_tip_pos:
COORD nose_top_pos:
COORD nose_top_pos:
COORD nose_top_pos:
COORD upret_term_pos:
COORD chin_pos:
COORD lower_term_pos:
COORD end_pos:
                                                                                                      /* nose tip */
/* nose bottom */
/* nose top */
/* upper terminating position */
/* chin position */
/* lower terminating position */
                                                                                                   /* size of the FFT operation */
/* number of pasples on the curve */
* number of pixels in the curve */
* horizontal offset for display */
/* length of the profile curve */
/* samples of the curve */
                     int
int
                                       fft_size = vsize*2;
nsamples = vsize*1;
                     int namples - vsize+l;
int npoints;
int offset;
double clen;
COMPLEX *curve; sample;
COMPLEX *curve;
double *mag;
int *chain;
int i;
                                                                                                      /* magnitudes of the transformed data */
/* chain code of the profile curve */
                     get image();
                                                                              /* get image from the camera or an input file */
                    mose_tip(inose_tip_pos ):
nose_bottos( nose_tip_pos, inose_bottos_pos );
nose_top( nose_tip_pos, nose_bottos_pos, inose_top_pos );
upper_tensination( nose_top_pos, nose_bottos_pos, isupper_tens_pos );
chin_position( nose_top_pos, nose_bottos_pos, iso_top_pos, inose_bottos_pos, chin_pos,
ilower_tensination( nose_top_pos, nose_bottos_pos, chin_pos,
ilower_tensination( nose_top_pos, nose_bottos_pos, chin_pos,
                     mag = (double *)malloc( sizeof(double)*fft_size );
chain = (int *)malloc( sizeof(int)*MAX_CMAIN_SIZE );
                    offset = DISPLAY OFFSET:
cdisplay( chain, offset, 0 );
                     draw_square_marker()offset=nose_top_pos.x, nose_top_pos.y ):
draw_square_marker()offset=nose_tip_pos.x, nose_tip_pos.y ):
draw_square_marker()offset=nose_botton_pos.x, nose_bottom_pos.y ):
draw_square_marker()offset=nose_bottom_pos.y, nose_bottom_pos.y ):
draw_square_marker()offset=nose_x, chan_pos.y ):
draw_square_marker()offset=nose_x, nose_y ):
                    ...
                                       Sample the curve and obtain the Fourier descriptors vector.
                      */
csample( chain, nsamples, clen, &curve_sample );
offset += DISPLAY_OFFSET*3/2;
sdisplay( curve_sample, nsamples, offset, 0 );
                      trace_back( curve_sample, nsamples, incurve );
fft( ncurve, fft_size );
magnitude( ncurve, mag, fft_size );
                      ...
                                          Normalize the vector such that vector[0] = 1.0.
                      ** Normalize the vector such that vector[0] = 1.5
*/
for ( i = 1: i < vsite: i++ )
vector[i] = msg[i+1] / msg[i];
vector[0] = 1.0: /* = msg[i]/msg[i] */</pre>
```

```
free( chain );
free( mag );
free( curve_sample );
free( curve_);
                return 1:
                                                                          /* operation successful */
а
Punction: trace back
Intent: Trace the samples from the beginning to the end and
trace back to the beginning to form a closed boundary.
Arguments: curve sample the coordinates of the samples.
closed_curve output closed curve samples.
*/ static void
trace back( curve sample, nsamples, closed_curve )
COMPLEX *curve samples;
in nsamples;
COMPLEX **closed_curve;
}
               COMPLEX *curve;
int i, j:
                *closed_curve = curve =
(COMPLEX)*(nsamples-1)*2 );
                :::>
                            Trace back. Note that the two ending points are not included in the retrace.
                // > = nsamples;
for ( i = 1; i < nsamples-1; i++ ) (
COMPLEX_ASSIGN( curve[]), curve_sample[nsamples-1-i] );
j++:
                 ÷.
 1
Function: draw_dline
Intent: Draw'a dotted line between two points.
Arguments: x1, y1, x2, y2 two points.
mpoints mmaker of points on the dotted line.
                double slope;
double dx, dy;
double x, y;
int i;
 #itndet _FRAME_GRABBER
_setcolor(255);
#endif
                tor ( i + 0; i < npoints; i++ ) {
    x = (double)xl + dx*(double)i;
    y = (double)yl - slope*((double)xl-x);
    wRITE_PIXEL(ROWND(X), ROWND(Y) );</pre>
                                х
                 else (
                                 slope = (double) (x1-x2) / (double) (y1-y2);
dy = (double) (y2-y1) / (double) (npoints-1);
                                 for { i = 0; i < npoints; i++ ] {
    y = (double)y1 + dy*(double)i;
    x = (double)x1 - slope*((double)y1+y1;
    wRITE_PIXEL( ROUND(x), ROUND(y) 1;
</pre>
                                 ÷.
                1
  1
  draw_cross_marker
To draw a cross-hair marker at the specified
position.
x, y the position of the marker.
                 Function:
Intent:
                 Arguments:
  */
static void
draw_cross_marker( x, y )
int x, y:
  f
#itndef _FRAME_GRABBER
__setcolor(255):
#endif
                 WRITE_PIXEL( x, y ):
                WRITE_PIXEL( x-3, y );
WRITE_PIXEL( x-2, y );
WRITE_PIXEL( x-2, y );
WRITE_PIXEL( x-1, y );
WRITE_PIXEL( x+1, y );
WRITE_PIXEL( x+2, y );
WRITE_PIXEL( x+3, y );
                  WRITE_PIXEL( x, y-3 );
WRITE_PIXEL( x, y-2 );
WRITE_PIXEL( x, y-1 );
WRITE_PIXEL( x, y+1 );
WRITE_PIXEL( x, y+2 );
WRITE_PIXEL( x, y+3 );
   X
   Function:
Intent:
Arguments:
                                         draw_square_marker
To draw a square marker at the specified position.
x, y the position of the marker.
   static void
```

#### File: vector.h

```
/•-----

• File:

• Intent:

• ·
```

vector.h Function declaration for a high-level function which obtains fourier descriptors vectors from face profile Langes.

#### #ifndef\_VECTOR\_H #define\_VECTOR\_H

#endif