Representation of Contours by Connected Quadratic Curves and its Implementation and Simulation

XU WU LI

A MAJOR REPORT

IN

THE DEPARTMENT

OF

COMPUTER SCIENCE

PRESENTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREES OF MASTER OF COMPUTER SCIENCE

CONCORDIA UNIVERSITY

MONTREAL, QUEBEC, CANADA

SEPTEMBER 2002

0-612-72939-7

Abstract

Representation of Contours by Connected Quadratic Curves and its Implementation and Simulation System

Xu Wu Li

Recognition of characters is a problem in image recognition. Contours are often used. The contour of a character can be expressed with much less data than the original character. There are a suit of algorithms that represent contours by connected quadratic curves. With this representation, corners of characters can be detected by locating those corner points.

This project will emphasize on the implementation of those algorithms: including the contour tracking, segmentation and how quadratic curves are obtained from the contour. It is a java application and has a GUI interface. You can select an image, run the algorithm and see the simulation steps.

Acknowledgements

I would like to thank Dr. T.D Bui, my supervisor, for his continuous guidance, prompt support and valuable suggestions.

I am also grateful to Guang Yi Chen, who gave me some useful materials and the image database.

Finally, I would like to thank my family for their support.

Table of Contents

Chapter 1 Introduction

This document outlines the design and implementation of the algorithm of using quadratic curves to represent the contour of a character and detect those corner points of the contour. This project also provides a simulation of the algorithm.

1.1 Pattern Recognition

Many pattern recognition systems can be partitioned into the following components: input, sensing, segmentation, feature extraction, classification, post-processing and decision. A sensor converts images or sounds or other physical inputs into signal data. The segmentation isolates sensed objects from the background or from other objects. A feature extractor measures object properties that are useful for classification. The classifier uses these features to assign the sensed object to a category. Finally, a post processor can take account of other considerations, such as the effects of context and the costs of errors, to decide on the appropriate action.
Although there are seven components in many pattern recognition system; some are more critical than others. The feature extraction component is more important and more difficult to realize: An ideal feature extractor would yield a representation that makes the job of a classifier trivial; conversely an omnipotent classifier would not need the help of a sophisticated feature extractor.

The goal of extractor is to characterize an object to be recognized by measurements whose values are very familiar for objects in the same category, and very different for objects in different category. That is to say to find the distinguishing features. The feature-choosing step depends on the characteristics of the problem domain. Having access to example data will certainly be valuable for choosing a feature. However, prior knowledge also plays a major role. How do you know which feature is more pertinent and more important.

In selecting or designing features, we obviously would like to find features that are simple to extract, invariant to irrelevant transformations, insensitive to noises, and useful for discriminating patterns for different categories. Computational complexity and cost are also factors when we consider feature choosing. We need algorithms that scale well when the number of feature dimensions, or the number patterns or the number categories increase. We need a feature extractor to function within some constraints. We need to combine the prior knowledge and empirical data to find relevant and effective feature.

1.2 Problem domain

Much information of a character is included in its contour. Contours are often used for pattern recognition. Human beings can recognize a character by its contour, which shows that the contour contains all the information needed for this recognition. The contour of a character can be obtained usually by two steps: (1) Extracting the borders of the character with edge detection algorithms [2, 10]; (2) Representing the edge points as a discrete sequence, in which tracing algorithm are often used and some proper processing are taken [3,5,9]. To improve the contour, proper modification can be employed. Then the contour, which is a (or a few) 1D periodic signal(s), can be transformed to other forms so that features such as the curvature, inflexion points, corners, perimeters and moments can be computed. There are numerous researches on this subject such as chain code representation, curving fitting and so on. Curve smoothing is an extensively used method for contour analysis. It can denoise the digital contour and some mathematical differential parameters can be calculated robustly. A defect of such a representation is that it also blurs the corners of contours that locate the key features of the character. Therefore, multi-scale filters can be used as a compromise. In this report, the connected quadratic curves are used to represent the contour of characters. Based on representation, we will introduce a scheme to detect corners of characters.

1.2.1 Characteristics of characters and quadratic curves

From mathematics theory we know that quadratic curves, sometimes called spline curves are C2 smooth. We are familiar with them, lines are degenerated form of quadratic curves, parabolic curves, and trajectory curves are also quadratic curves. Quadratic curves have many important characters that make them very useful. First, the complexity and cost to use them are low, there are many software packages that deal with quadratic curves. Quadratic curves have a very important character that we will use through our algorithm: given three points (not on a line), we can only build a quadratic curve that passes two points and uses another point as vertex.

Human beings can detect characters easily by recognize their contours. The contour of a character can be divided into some segments. Each single segment is smooth; but the whole contour is not smooth.

Those points that connect those smooth curve segments are very important features of the contour of the character, those points make the character not smooth enough, because the contour may change direction at these points or these points are corner points which means that the angle at the corner point is too small - singular points.

For segments of a character contour, some are lines or approximate to lines, these can be approximated by lines (degenerate quadratic curves). Others are arcs that can also be approximated by quadratic curves. So we can imagine that the effects of approximation will be good.

Our quadratic curve algorithm provides a way to find those corner points.

1.2.2 The input of the system

We use two types of images to test the system. One is the GIF, JPG or JEPG format, the other is the character image. You can see these two kinds of format from Fig 1 and 2. One is a low-resolution image with 40*38 resolution; another is a black and white 256*256 image.

1.3 System Context

Character recognition has been the topic of pattern recognition for a long time; recently three professors L.H.Yang, C.Y.Suen and T.D.Bui have written an article on “Representation of Contours by Connected Quadratic Curves and its application to 2D image recognition.” (see [12]). This system is built to implement their ideas.

```
40  38
00000000000000000000000000000000000000
00000000000000000000000111111110000000
00000000000000000000111111111111111000
00000000000000000001111111111111111110
00000000000000000011111111111111111110
00000000000000000111111111101111111111
00000000000000001111110000000000001111
00000000000000001111100000000000000011
00000000000000001111100000000000000000
00000000000000011111110000000000000000
00000000000000011111000000000000000000
00000000000000011111000000000000000000
00000000000000011111100000000000000000
00000000000000001111100000000000000000
00000000000000001111110000000000000000
00000000000000000111110000000000000000
00000000000000000011111100000000000000
00000000000000000011111100000000000000
00000000000000000001111100000000000000
00000000000000000000111110000000000000
00000000000000000000011111000000000000
00000000000000000000001111100000000000
00000000000000000000001111100000000000
00000000000000000000000111110000000000
00000000000000000000000011110000000000
00000000000000000000000011111000000000
00000000000000000000000001111000000000
00000000000000000000000001111100000000
00000000000000000000000000111100000000
00111000000000000000000000111110000000
01111110000000000000000000111110000000
01111111000000000000000000111110000000
00111111110000000000000000111110000000
00011111111100000000000001111110000000
00000111111111110000000011111110000000
00000001111111111111111111111100000000
00000000011111111111111111111100000000
00000000001111111111111111110000000000
00000000000000111111111111000000000000
00000000000000000011110000000000000000
```

Figure 1. An image of five with resolution 41*38

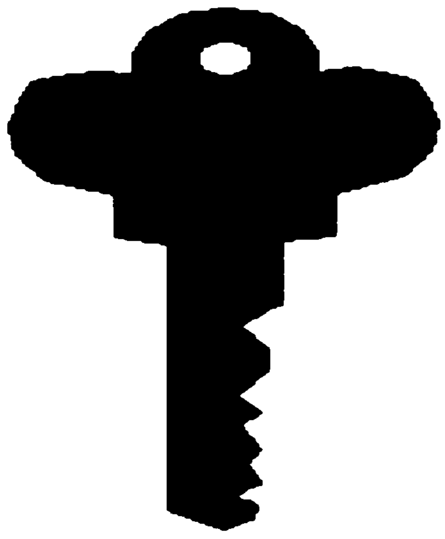


Figure 2. An image of JPEG format with resolution 256*256

Specifically, the main issues addressed through the system include several aspects:

- The system should provide a GUI interface.
- User should be able to select an image through GUI.
- User should be able to run the algorithm through GUI.
- User should be able to see animation of the algorithm from GUI.
- User should be able to set system parameters through GUI.
- The system should provide a help system for help and trouble shooting

1.4 Description of each function:

1.4.1 The system should provide a GUI interface

The system should use GUI components: such as windows, buttons, text fields, menus and Labels to visualize the system and give the user intuition of functions the system provides.

1.4.2 User should be able to select an image through GUI.

The system should provide a button or a menu for user to select an image. After the user has pressed the button or the menu, the file chooser dialog should appear to let the user select an image. After the user selects an image, that image should be displayed.

Exception:

If the user does not select an image or the image the user selects is not a valid one, the system should ask the user to re select a valid one.

1.4.3 User should be able to run the algorithm through GUI

The system should provide a button or a menu for user to run the algorithm to the image the user has just selected.

1.4.4 User should be able to see animation of the algorithm from GUI

The system should provide a button or a menu for user to see animation of the algorithm. Each frame should be marked which step it is in. The animation is only enabled after those algorithms have been run.

1.4.5 User should be able to set system parameters through GUI.

The system should provide a button or a menu for user to set up system parameters. When a user presses that menu or the button, a dialog will appear with text fields or combo boxes, a user can input or select the parameters.

Exception:

If a parameter the user set is not valid, the system should ask the user to re input that parameter.

1.4.6 The system should provide help system for help and troubleshooting.

The system should provide a button or a menu for user to trigger help system. When a user presses that menu or the button, help frame will appear displaying the information about the system and the FAQS when a user uses the system.

1.5 Use case view of the system

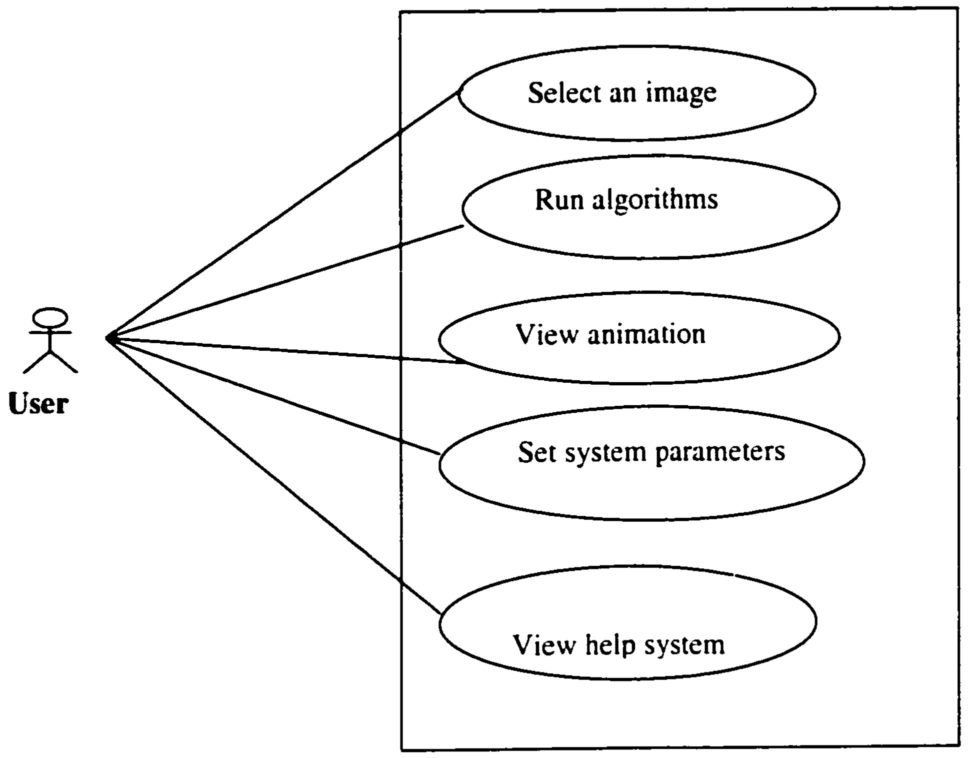


Figure 3. Use case view of the system

Chapter 2 Select a development environment and a programming language

2.1 Develop environments and Operating systems

There are several major develop environments: Windows, Linux and Solaris. Windows operating system has the largest market share in PC computers. PC's users have accepted Linux. On some aspects, Windows does better (ie GUI) and on other aspects Linux does better.

In the case of Linux, since it is a UNIX-based kernel, it has reliability and security that comes with UNIX. It is open source so that bugs are fixed quickly. One huge advantage of Linux has is that it is free.

Since I have to develop the system at home, windows is my preference.

As for the operating system, I use Windows 2000 Server, but the system can be run under any Windows even Linux and Solaris, as we will explain later.

2.2 Programming language

There are many programming languages such as C++, C, C# and java. They are both powerful.

After comparing these programming languages, we choose Java as our programming language because it is more competitive than others on the following aspects:

1. High performance: With the JIT, java is quite fast now.
2. It has enriched built in data structures for use: Collection Framework provides a uniform interface for many data structures. You can access those data structures through Iterators.
3. It has powerful graph process functions: it can display not only integer points but also double and float points. Java provides 2D, 3D API for drawing various graphs and images; see [1, 9].
4. Easy to build GUI: Java provides AWT and SWING package for building interfaces, you can build GUI easily with widgets: such as buttons, JButtons, labels, JLabels. Java even provides components such as: Tree, Table and TabbedPane to help you build complex GUI.

5. It is platform independent: You just write java code, compile it once and it can be run under all platforms.
6. Java is pure OO, so you can make full use of OO methodology.
7. Java is easy to learn, its syntaxes are similar to those of C and C++.
8. Java has garbage collection and exception handling, which make program easy to debug. Java gets rid of the C and C++ pointer to make software program safer.

From all above, I choose Java as the programming language. The most important reason is that: I am good at Java and Java is one of my favorite languages.

2.3 How to deploy and execute this system.

There are many java development tools such as JBuilder, Visual age for java and JDevelop, they are powerful and easy to use; but they consume too much system resources and on some computer they are quite slow. I code the system by using UltraEdit, compile and run the system under JDK1.3. Before compile and run the java code you must set up environment variables. Under different system the setting procedure is different. Under windows 2000, the procedures are:

1. Double Click the "My computer" menu.
2. Double click the "Control panel" menu.
3. Double click the "System" icon.
4. Double click the "Advanced" tab window button.
5. Click the "Environment variables" button.
6. If the classpath and path variable already exist just edit them, else create two variables with these two names.
7. classpath=the directory you want to put your java program;

path=the directory where jdk bin library is in. After that you can run the system.

After you have set the environment variables, you can compile and run this program, there are two methods:

Method 1:

1. Press the "Start" menu in your computer.
2. Press the "Run" menu.
3. In the open field input "cmd" and press "Ok".
4. The DOS window appears.
5. Go to your java file directory.
6. Type javac *.java to compile all java files and click enter.
7. Type java MainFrame or NewMainFrame and click enter.

Method 2:

1. Press "Start-Program-Accessory-DOS prompt" menus.
2. The DOS window appears.

The rests are the same as method 1.

Chapter 3 Design of the system

3.1 System GUI design

From those functions of the system, we know that the system should have the following interfaces:

1. Interface for select an image.
2. Interface for run the algorithm.
3. Interface for trigger the animation.
4. Interface for trigger the help system.
5. Interface for set system parameters
6. Interface the help system.

7. Interface for the original image
8. Interface for those animation graphs.

Since interface 1,2,3,4,5 are too simple, we can combine them into one GUI. That is our MainFrame, we still need interface 6,7 and 8. We name them: HelpFrame, ImageFrame and AnimationFrame.

A MainFrame should contain a label for a title of the system, buttons for system functions, text fields or combo box for parameters setting and labels.

A HelpFrame should contain a text pane that displays help information.

An ImageFrame is just a Frame with a title.

An AnimationFrame is just a Frame with a title

3.2 Other classes design

1. Since we will use a contour-tracking algorithm, we need a class to implement this algorithm.
2. Since we will conduct a lot of mathematics computations, we need a class to implement those computational algorithms.
3. Since we need to do animation, we need one or more classes to deal with it.
4. We need several exception classes to process exceptions.

3.3 Class diagram, sequence diagram and scenario

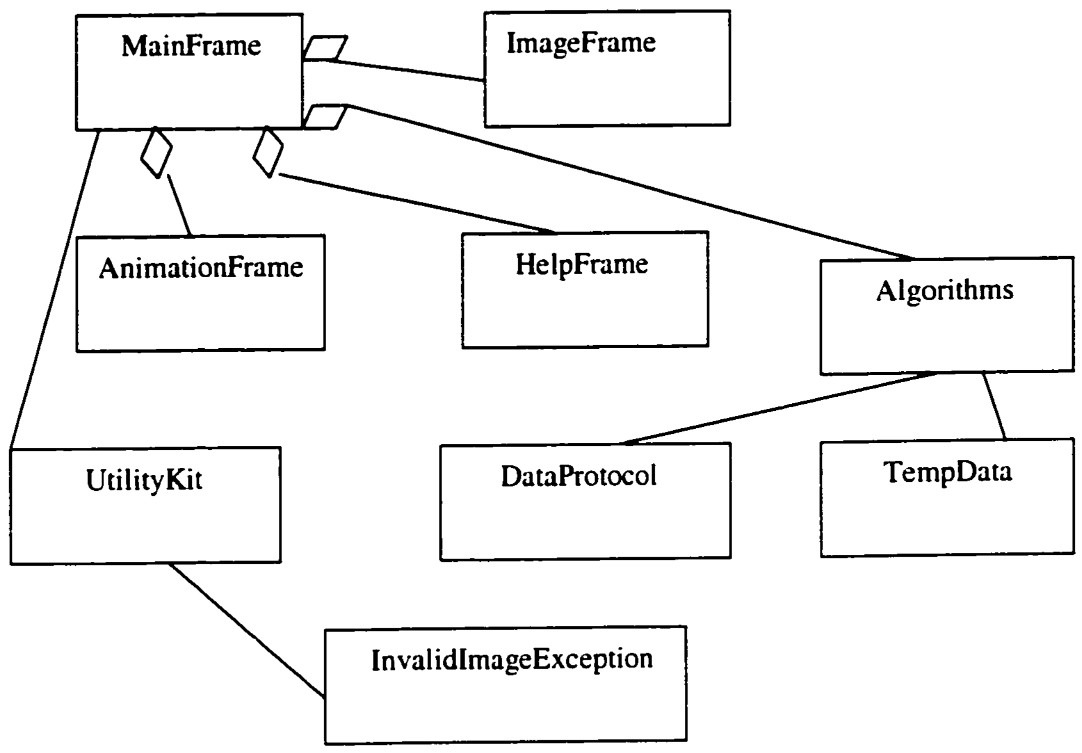


Figure 4. Class diagram of the system

The system main function scenario

1. The user presses the “Select image” button.
2. The file chooser dialog appears.
3. The user selects one image and presses the “Ok” button.
4. The user presses the “Run” button.
5. The user then press the “Animation” button
6. Those animation frames appear.

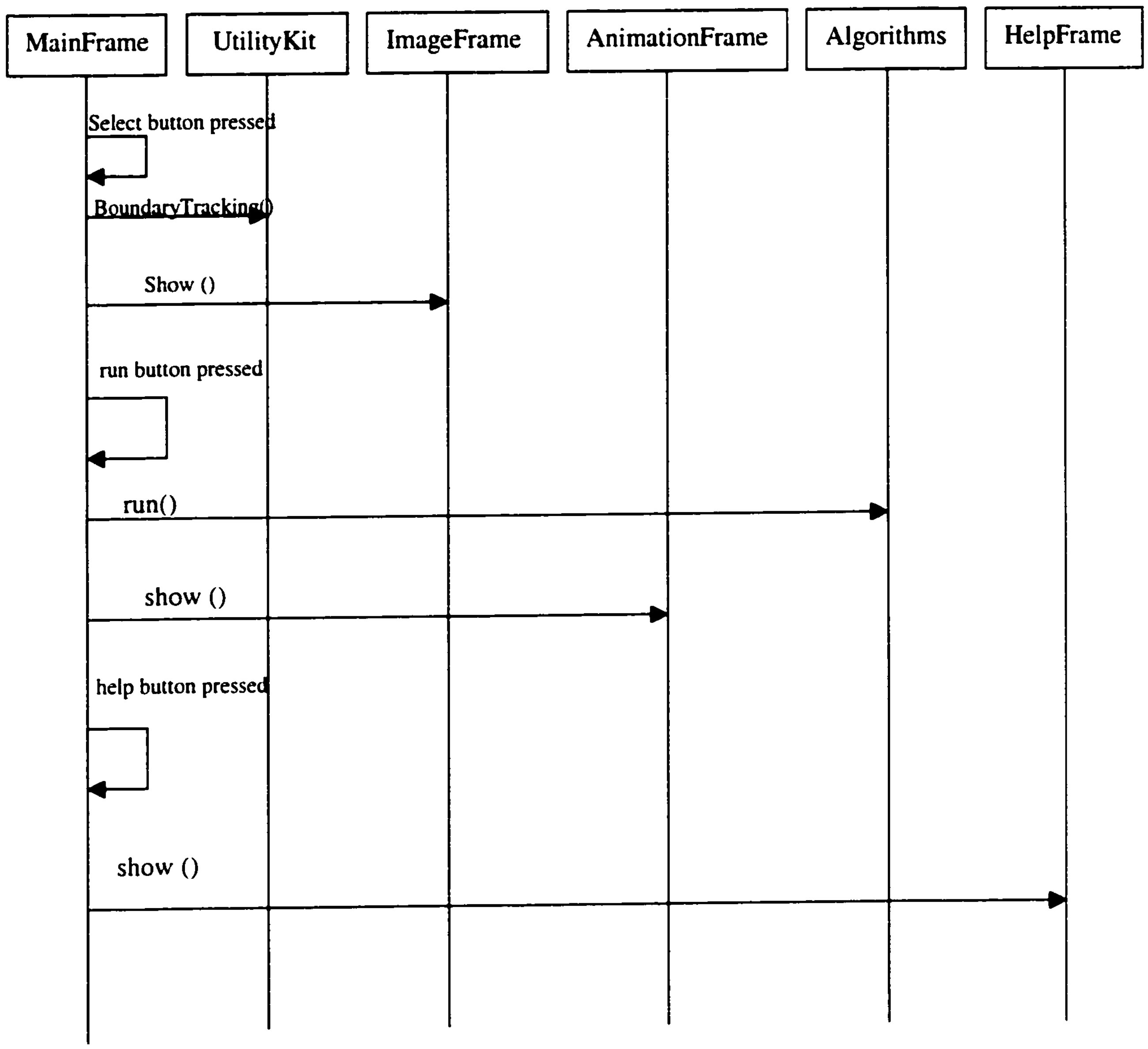


Figure 5. Main system sequence diagram

3.4 Classes and those functions they should perform

Class `AnimationFrame extends JFrame`

It performs one function:

1. Draws graphs when called by the run time environment.

Public class Algorithms

It has the following functions:

1. Runs main algorithm.
2. Finds the index of a point in some points, which has the maximal distance to a line.
3. Calculates the value of a function given three points and another parameter.
4. Calculates the only root of a equation in (0,1), given precision e.
5. Runs recursively the algorithm and animation function, it undertakes all major algorithms by call other functions.
6. Calculates the distance between the point to a curve segment.
7. Tests whether all the points have been processed.
8. Tests whether an ArrayList contains an object.
9. Calculates the coordination of the control point of the quadratic curve. It is the intersection point of the two tangent line at start point and end point.
10. Judges whether a curve with some parameters is out of the triangle determined by some parameters.

Public class DataProtocol implements Serializable

It performs one function:

Tests whether two DataProtocols are equal

Public class HelpFrame extends JFrame

It perform one function

1. Shows the help frame

Class ImageFrame extends JFrame

It has only one function:

1. Draws graphs when called by the run time environment.

Public class InvalidImageException extends Exception

It has the following functions:

1. Generates exceptions when needed.

Public class NewMainFrame extends Jframe

It has the following functions:

1. It is the main GUI of the system.
2. Builds the interface and contains event process.

Public class TempData

It has only one function:

1. Stores the temporary data.

class UtilityKit

It performs the following function:

1. Tracks the contour of an image.
2. Tests whether a point is within a range.
3. Tests whether a point is within a triangle.
4. Computes the inner product of two points.
5. Optimizes the two dimension array before runs the contour-tracking algorithm.

Public class ShowCornerFrame extends Jframe

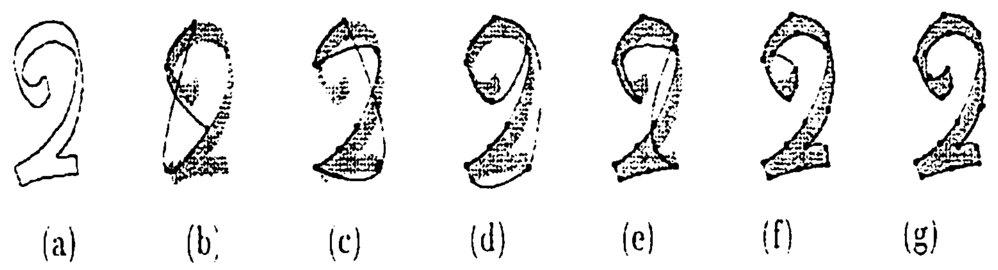
It perform following functions:

1. Draws graphs when called by the run time environment.
2. Finds the DataProtocol that connects with the segment with the end connecting to the begin.
3. Finds the DataProtocol that connect with the segment with the begin connecting to the end.

3.5 Algorithms and their design

Description of General and specific algorithm:

The whole algorithm is a divide and conquer process, we divide the contour equally into half by the line from the begin point to the central point; for each half of the two contours we pick up one point (details see below) and construct a quadratic curve that passes the point (as vertex), the begin point and the central point; so we have four curve segments. For each segment, if it satisfies our precision we keep it, otherwise we discard the quadratic curve segment and construct a new quadratic curve segment, this process repeats until all segments meet our precision, finally we can get a series of connect quadratic curve segments and some candidate corner points (see following image and those steps).



3.5.1 General algorithm:

Let T>0 be an error threshold. For a character image.

Step1: Get the contour sequence {C (I) | I=0,1,..........n-1} by using a contour-tracing algorithm.

Step 2: Let k=n/2; for C1={C (I) | I=0,1,..........k} and C2={C (I) | I=k, k+1,..........n-1} run the following specific algorithm.

3.5.2 Specific algorithm:

For a contour sequence C1={C (I) | I=n, n+1,..........k}

Step1: Find a point C (m) on the contour that has the maximal distance to line from C (n) to C (k).

Step2: If m=n then the segment is a line, mark all points from n to k. if all points are marked then return.

Step3: If m<>n construct a quadratic curve starting at C (0), passing C (m) (as vertex) and ending at C (k).

Step4: For C1={C (I) | I=n, n+1,..........m} and C2={C (I) | I=m, m+1,..........k} do the following (we use C1 as example):

1. Compute each C1 point's distance to the quadratic curve segment between C (n) and C (m).

2. Get the maximal distance of all the distances.

3. If the maximal distance < T, mark all points from n to m. If all points are marked then return.

4. Else recursively run specific algorithm for C1.

3.5.3 Root computing algorithm:

Description:

We want to compute the root of a continuous function. From mathematics we know that if f (a)<0 and f (b)>0; there is a value between a and b such that f(c)=0. This algorithm keeps cutting search interval to half of its previous one.

Let f(x) be a function f(a)<0, f(b)>0 and e be a precision double.

Step1: Initialize two end value a and b.

Step2: Compute the middle value of a and b.

Step3: If for that middle value f () gets value satisfy our precision return.

Step4: If f () gets positive value, we think the root is between a and that value.

Step5: Change ends to a and that middle value.

Step6: Go to step 2.

Step7: If f () is negative, we think the root is between that value and b.

Step8: Change ends to b and that middle value.

Step9: Go to step 2.

Comment: we can use many methods to compute the root of a function. The Newton iteration can be used. But the above method is also ok! And it is quite fast.

3.5.4 Distance to curve algorithm:

Description:

We want to compute the distance from a point to a curve segment. We divide the curve into small segments and find the approximate distance. We can adjust this division number.

Step1: Divide the curve segment into small enough segments.

Step2: Compute the distance from this point to those division points.

Step3: Find the smallest of the all those distances.

Comment: compute the distance from a point to a curve segment strictly speaking is not an easy job. The candidate points are those points at which tangent line is perpendicular to the lines connecting those points to the given point.

3.5.5 Within triangle algorithm:

Description:

We want to test whether a point is within a triangle. We know that given three points are not on a line, any planar point can be express as a linear expression of those three points; and the sum of the three coefficient is 1.

A Point P is with in Point a, Point b, Point c

Stepl: compute k1, k2.k3; k1+k2+k3=1 such that P=k1 * a+ k2 * b+k3 * c;

Step2: P is within triangle a-b-c if and only if all three coefficients are bigger than zero.

3.5.6 Contour tracking algorithm:

Description:

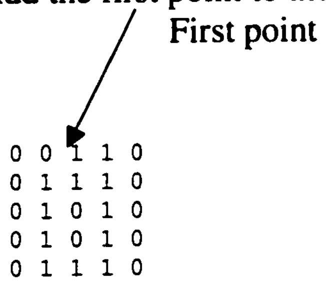
We want to track the boundary of an image clockwise(we use a small 0 image as example to illustrate it).

Stepl. Find the first point which is also the begin point.

Find first element in the image array with value 1.

Build the first point from the two indexes of the element.

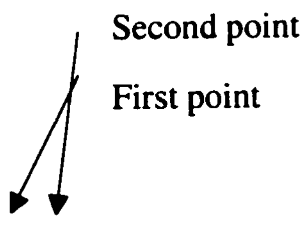
Add the first point to the boundary list.



```
0 0 1 1 0
0 1 1 1 0
0 1 0 1 0
0 1 0 1 0
0 1 1 1 0
```

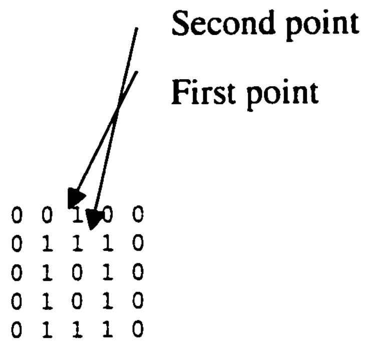
Step2. Find the second point.

Casel: If the point to the right of the first element is within the boundary and that point is an image point, then that right point is the second point.

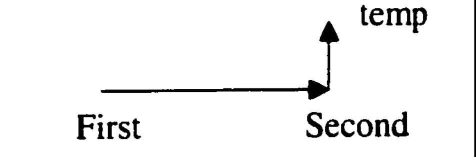
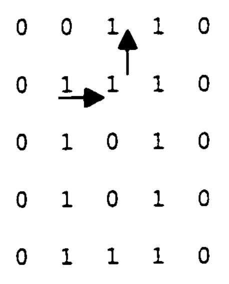
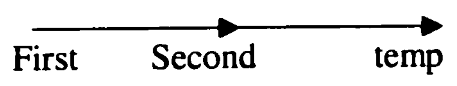
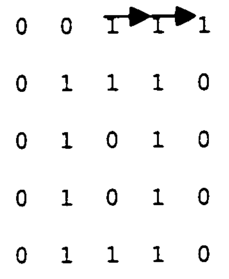
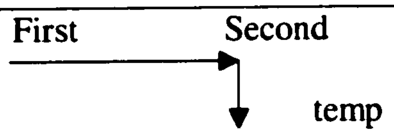
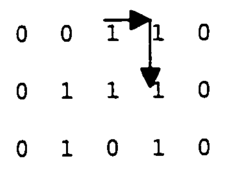


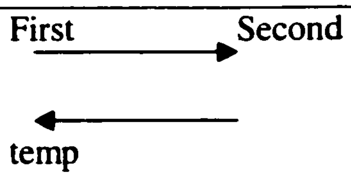
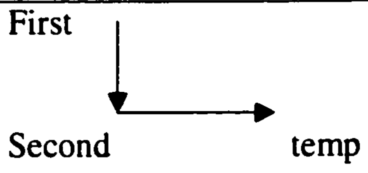
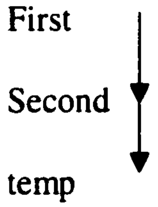
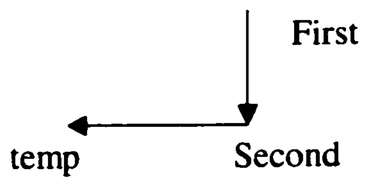
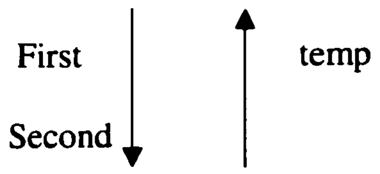
```
0 0 1 1 0
0 1 1 1 0
0 1 0 1 0
0 1 0 1 0
0 1 1 1 0
```

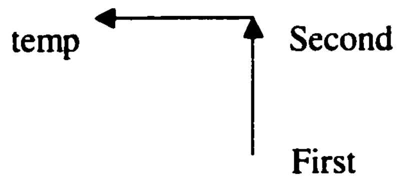
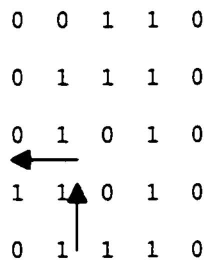
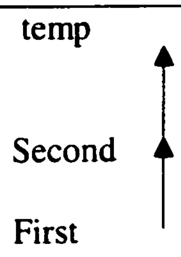
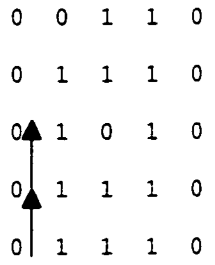
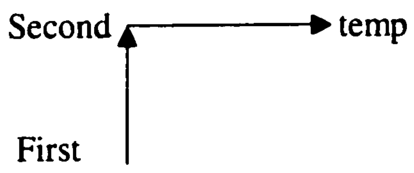
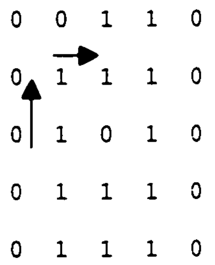
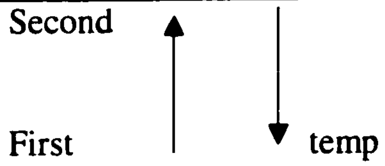
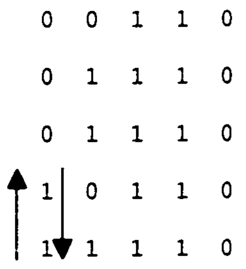
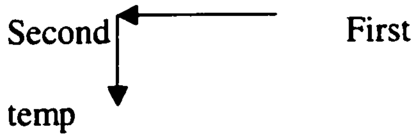
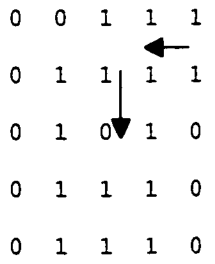
Case2: If the point to the beneath of the first element is within the Boundary and that point is an image point, then that beneath point is the second point.

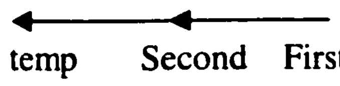
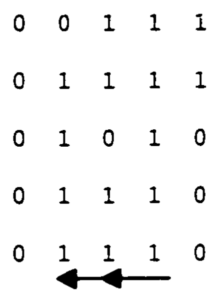
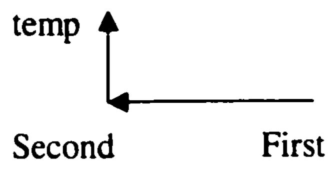
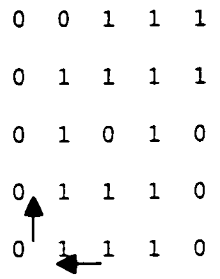
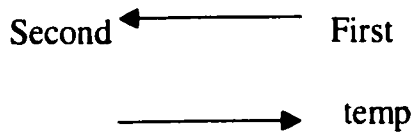
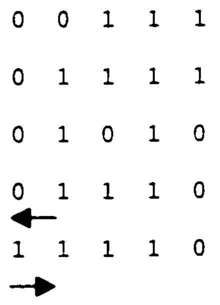


Step3.

Case1		 0 0 1 1 0 0 1 1 1 0 0 1 0 1 0 0 1 0 1 0 0 1 1 1 0	
Case 2		 0 0 1 1 1 0 1 1 1 0 0 1 0 1 0 0 1 0 1 0 0 1 1 1 0	
Case3			

		0 1 0 1 0 0 1 1 1 0	
Case 4		0 0 1 1 1 0 1 1 1 0 0 1 0 1 0 0 1 0 1 0 0 1 1 1 0	
Case 5		0 0 1 1 0 0 1 1 1 0 0 1 0 1 0 0 1 0 1 0 0 1 1 1 1	
Case 6		0 0 1 1 0 0 1 1 1 0 0 1 0 1 0 0 1 0 1 0 0 1 1 1 0	
Case 7		0 0 1 1 0 0 1 1 1 0 0 1 0 1 0 0 1 0 1 0 0 1 1 1 0	
Case 8		0 0 1 1 0 0 1 1 1 0 0 1 0 1 0 0 1 1 1 0 0 1 0 1 0	

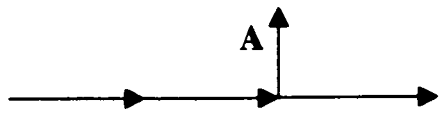
Case 9		$\begin{array}{ccccc} 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \end{array}$	
Case 10		$\begin{array}{ccccc} 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \end{array}$	
Case 11		$\begin{array}{ccccc} 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \end{array}$	
Case 12		$\begin{array}{ccccc} 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 \end{array}$	
Case 13		$\begin{array}{ccccc} 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \end{array}$	

Case 14		0 0 1 1 1 0 1 1 1 1 0 1 0 1 0 0 1 1 1 0 0 1 1 1 0	
Case 15		0 0 1 1 1 0 1 1 1 1 0 1 0 1 0 0 1 1 1 0 0 1 1 1 0	
Case 16		0 0 1 1 1 0 1 1 1 1 0 1 0 1 0 0 1 1 1 0 1 1 1 1 0	

3.5.7 Optimize algorithm

Before we run the contour-tracking algorithm, we can first optimize the array of points:

We can eliminate those points that have only one point in its neighbors.



As the above graph displays, point A is such a point. Other points are not.

3.5.8 Refined contour-tracking algorithm

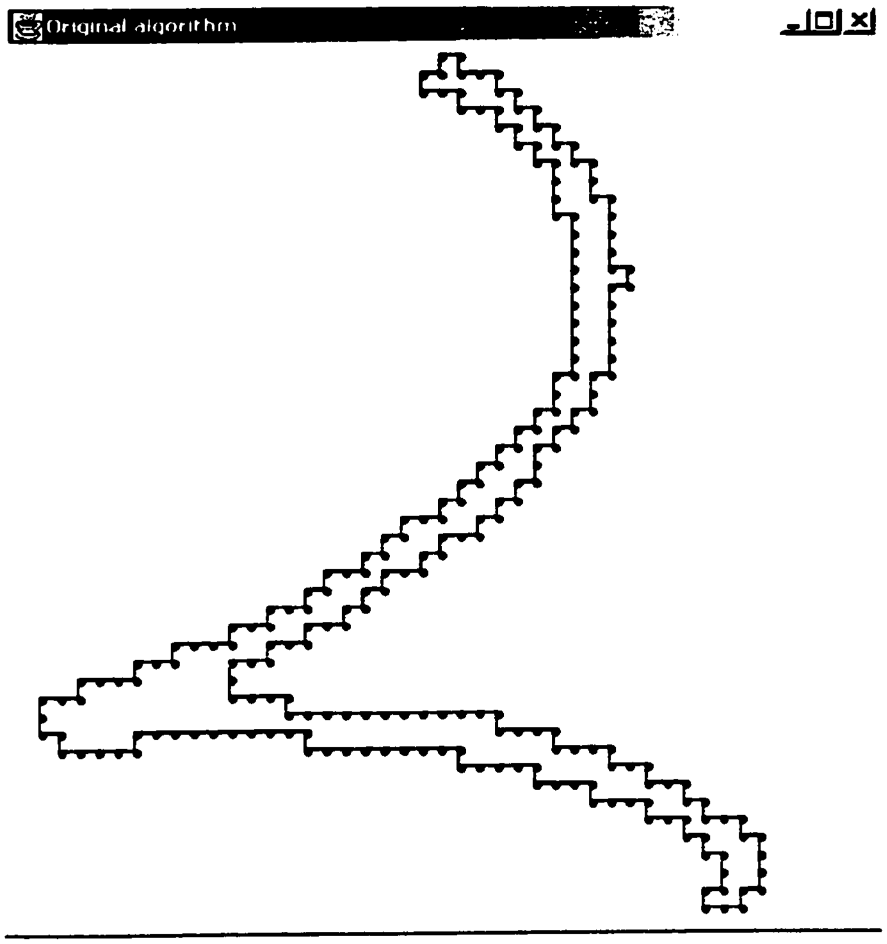


Figure 6. An image that the refined algorithm has not been run.

After we have run the contour-tracking algorithm, the image contains a lot of zigzag shapes, in order to erase these shapes we have to run the refined algorithm. The figure 6 and 7 show the effects of the refined contour-tracking algorithm.

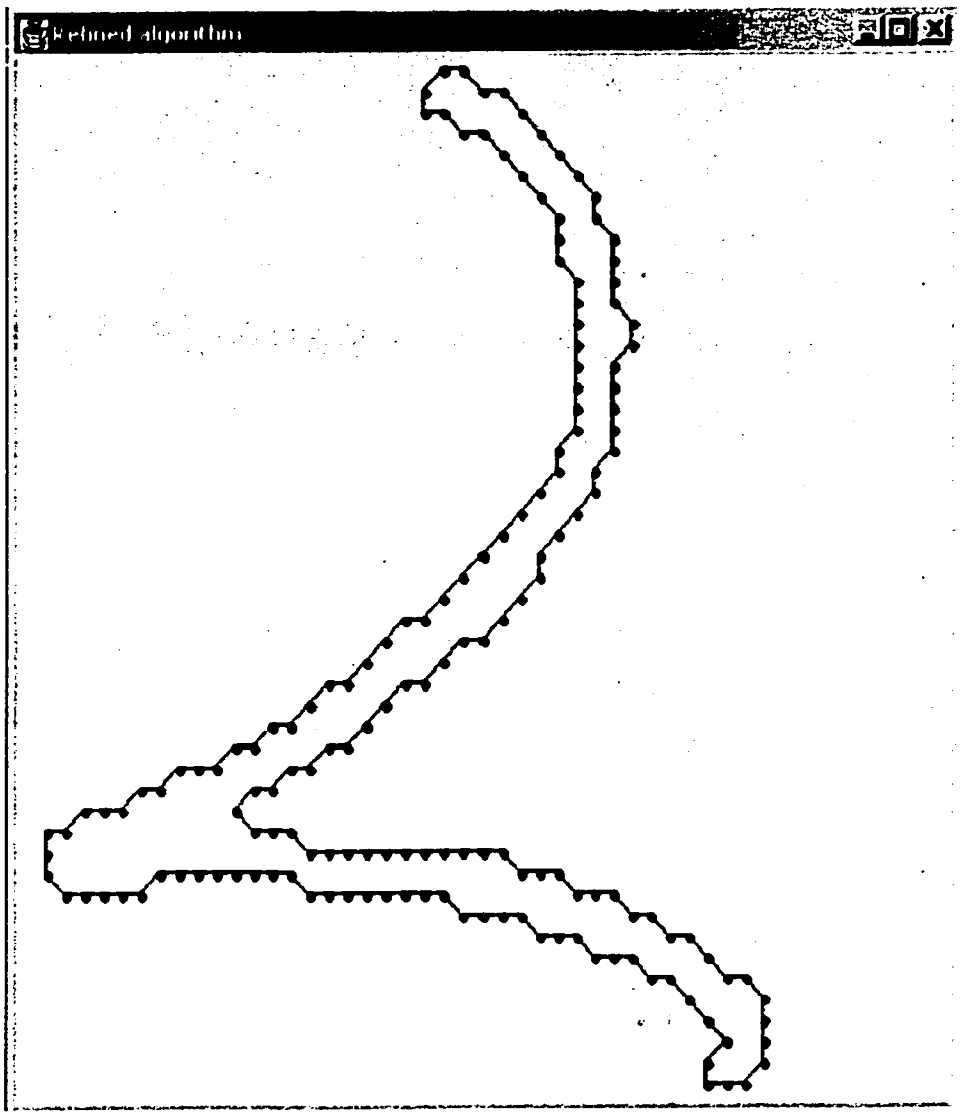
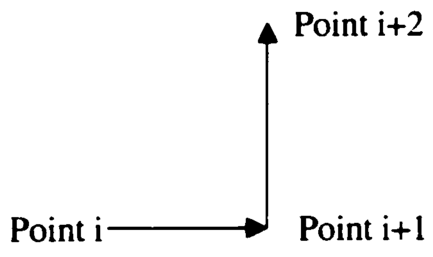
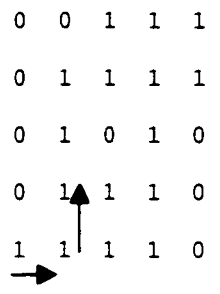
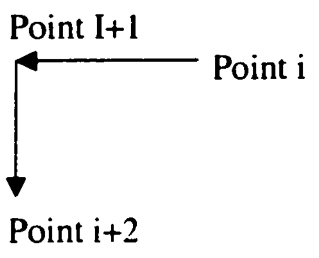
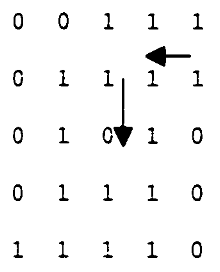
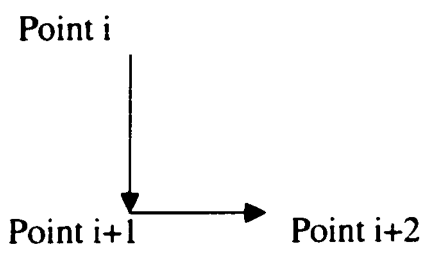
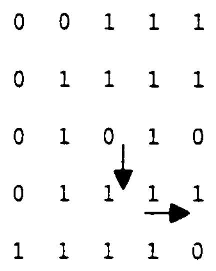
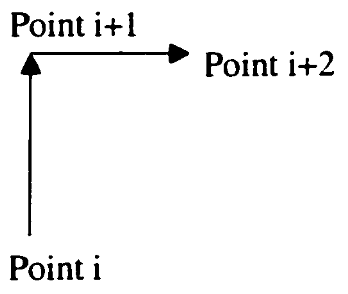
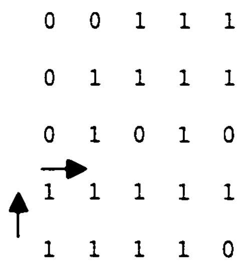


Figure 7. The same image after the refined algorithm has been run

You can see that the boundary is smoother, and there are fewer points on the boundary.

The algorithm works as follow:

Case 1		 0 0 1 1 1 0 1 1 1 1 0 1 0 1 0 0 1 1 1 0 1 1 1 1 0	**Remove point i+1**
Case 2		 0 0 1 1 1 0 1 1 1 1 0 1 0 1 0 0 1 1 1 0 1 1 1 1 0	**Remove point i+1**
Case 3		 0 0 1 1 1 0 1 1 1 1 0 1 0 1 0 0 1 1 1 1 1 1 1 1 0	**Remove point i+1**
Case 4		 0 0 1 1 1 0 1 1 1 1 0 1 0 1 0 1 1 1 1 1 1 1 1 1 0	**Remove point i+1**
Case 5	Mve forward to the point ahead	Move forward to the point ahead	

3.5.9 Animation algorithm

3.5.9.1 General algorithm:

Let T>0 be an error threshold. For a character image.

Let Array1 stores the quadratic curve segments that need to be drawn but are still not known whether they meet the precision.

Let Array2 stores the quadratic curve segments that meet the precision.

Step1: Get the contour sequence {C (I) | I=0,1,..........n-1} by using the Contour-tracking algorithm.

Step 2: let k=n/2; for C1={C (I) | I=0,1,..........k} and C2={C (I) | I=k, k+1,..........n-1}

Run the following specific algorithm.

For each element E in array 1, find all elements in Array2 whose steps are smaller than step of E and add those elements to E.

Serialize E as object.

3.5.9.2 Specific algorithm: For a contour sequence C1={C (I) | I=n, n+1,..........k}

Step1: Add that segment to Array1.

Step2: Let C (m) be the point that has the maximal distance to the line from C (n) to C (k) among all points from C (n) to C (k).

Step3: if m equals n then the segment is a line, mark all points from n to k. If all points are marked then return.

Step4: if m not equals n construct a quadratic curve starting at C (0), passing C (m) as vertex and ending at C (k).

Step5: for C1={C (I) | I=n, n+1,..........m} and C2={C (I) | I=m, m+1,..........k} do the following steps (use C1 as example):

1. Compute each C1 point's distance to the quadratic curve between C (n) to C (m).

2. Get the maximal distance of all the distances, if the maximal value < T, mark all points from n to m, add that segment to Array2. If all points are marked then return.

3. Else recursively run specific algorithm for C1.

3.5.10 Algorithm to detect corner points

First we need an algorithm to judge whether a quadratic curve segment is out of a triangle, the curve pass through two points of the triangle.

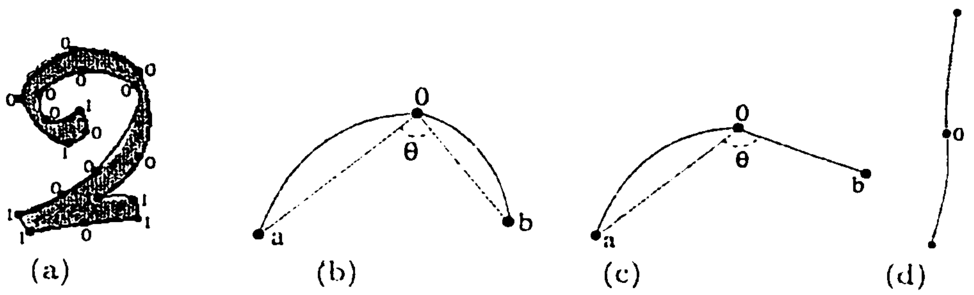
Step1. Divide the curve into many segments according to the parameter given.

Step 2. If one of the points is within the triangle, the curve segment is not out of the triangle, otherwise the curve is out of the triangle.

Corner detection algorithm

Let p (i) be all the candidate points (see 3.5 descriptions of the general and the specific algorithm). Those points satisfying the following conditions are not corner points and can be removed, those left points are corner points. If $\angle$ p (i-1) p (i) p (i+1) is bigger than some threshold alpha and one of the following conditions holds (see (a) points marked with "0" are not corner points; points marked with "1 are not corner points;

1. Both curve p (i-1) p (i) and p (i) p (i+1) are out of the triangle p (i-1) p (i) p (i+1), see (b).
2. One of the quadratic curves p (i-1) p (i) and p (i) p (i+1) is degenerated to a straight line and the other is outside the triangle p (i-1) p (i) p (i+1), see (c).
3. The quadratic curves between p (i-1) and p (i+1) are very close to the straight line from p (i-1) to p (i+1), see (d).



3.6 Data Structure selections

The system implementation adopts many algorithms; all these algorithms need many data structures to support them. In the contour-tracking algorithm, we need to read in image information from an image file and then pick out those boundary points and store them in a data structure. Because we cannot tell before hand how many boundary points will be picked out; we must use a data structure that can increase its size dynamically, Java Collection provides ArrayList and Vector; both can be used. I select the ArrayList, because it tallies the OO methodology.

3.7 System Data format and their storing method

Since we need to do simulation, quadratic curves of each step must be stored and retrieved when doing animation, the data format our system use are as follows:

TempData

It stores the following information

int step ------- the step that the data is in

DataProtocol dataProtocol -----------the data format

DataProtocol implements Serializable

It stores the following information

int startPosition ---------- the start position of a segment

int endPosition ---------------- the end position of a segment

int computeEnd -------------- the end position for computing

int indexPosition -----------------the index position of a segment

int drawType ---------------- the draw type 0-Quadratic 1-Line

int segmentType ------------------the segment type 0-whole 1- start to index 2- index to end

double root -------------------------the coordination of the control point

double controlX ----------------------the coordination of the control point

double controlY -----------------------the coordination of the control point

We can see in order to do simulation we need store many data. But how can they be stored. In C and C++ we can only store them in file; but later when you want to retrieve them the method is quite complicate. In java, we can directly write an object to a file and read in an object from a file; this makes the data storing and accessing procedure quite simple.

Chapter 4 Results, implementation and testing of the system

4.1 Related test methods with this project

4.1.1 Unit Testing

For each class and its main method we must perform a unit testing.

4.1.1.1 Static Unit Testing

Objective: Identify coding errors

Techniques:

Code inspection, walk through and buddy test.

All inspection questions must be checked.

Completion criteria:

Every line of code has been inspected.

For example: in class UtilityKit many lines of code are edited by copy and paste, this may create a lot of mistakes and they are not easy to find. There is only one method: print out the source code and examine line by line.

4.1.1.2 Dynamic Unit Testing

Objective: Ensure that internal functions work correctly.

Techniques:

White box testing.

For each class write a main method so that the class and its functions can be test independently.

Completion criteria:

All test cases must be executed.

For example: the class UtilityKit has the contour-tracking function, there are 13 branches and we can use white box and branches testing.

4.1.1.2.1 Testing of system algorithms

Each algorithm is a function of a class, and at lease need one test case.

1. Contour tracking algorithm

   The algorithm contains a multiple if-else branch

   Test method: Test as many as possible images so that all branches are tested. For each image print out the contour tracking results and compare with the original image.

2. Root computing algorithm

   Test method: Use some polynomials to test the algorithm, for each step print out the result to see how it approaches the final result and the iteration times.

4.1.2 Integration Testing

The integration testing should ensure the correctness and effectiveness of functional interactions and compatibility between interfaces. The test will focus on interactions among components, modules and subsystems. The integration testing should begin upon completion of the unit testing.

4.1.2.1 Testing of system algorithms

The main algorithm can only be test at this stage, because it needs many other algorithms.

Test method: Write a main function to call the run function, at each key code line print out key value to see the result.

4.1.3 Functional Testing

Functional testing should focus on the functions. The goal of these tests is to verify that all functions meet the requirement completely.

4.2 Running results of this system

4.2.1 Running results of the first format image

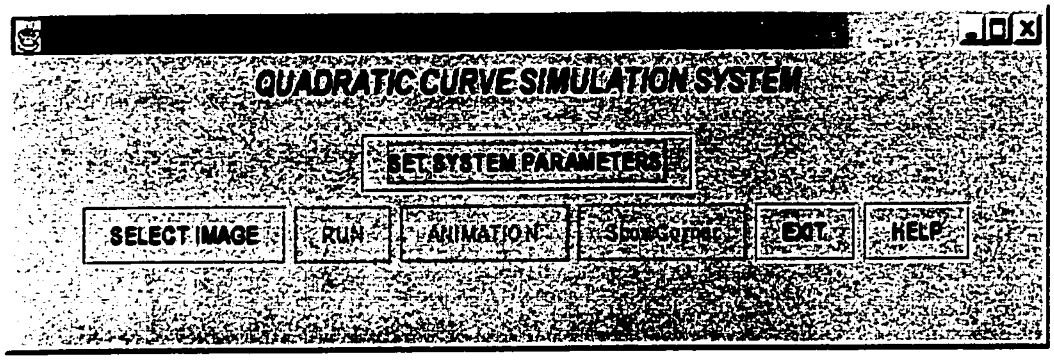
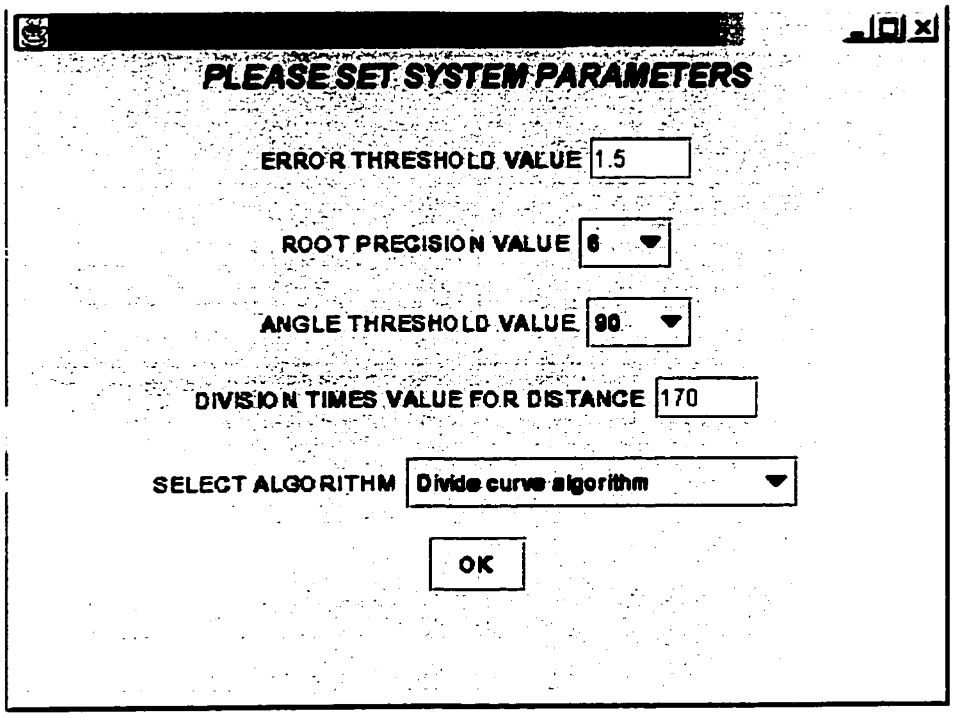


Figure 8. System Main Frame

As you can see from the above figure, the Main Frame contains three panels, one for title, one for setting system parameters and one for six buttons. If a user presses the "SET SYSTEM PARAMETERS" button, a frame will appear as the following figure. You can see at first the "Run" and "Animatiom" button are all disabled, this is because you have not selected an image. You can input into text fields or select from the Combo boxes, the system will check whether the input data is valid.


PLEASE SET SYSTEM PARAMETERS

ERROR THRESHOLD VALUE 1.5

ROOT PRECISION VALUE 6

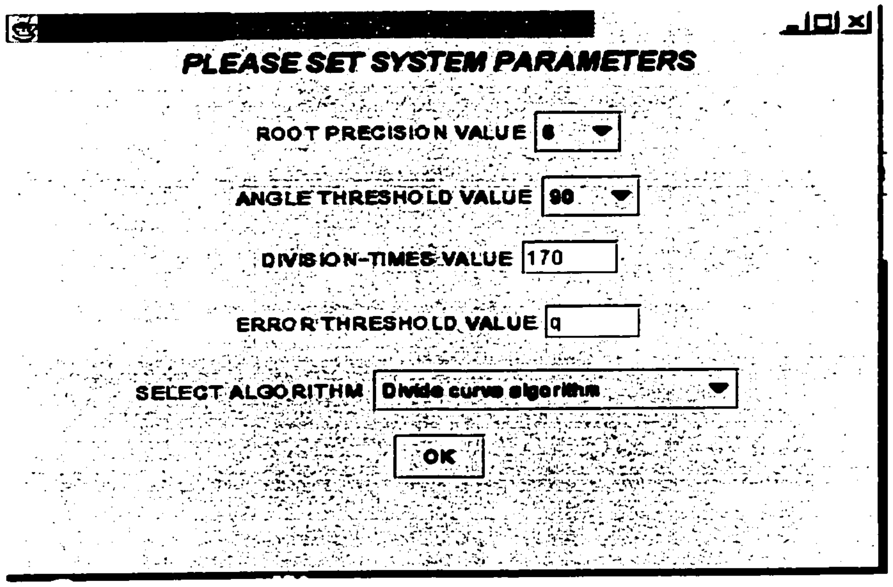
ANGLE THRESHOLD VALUE 90

DIVISION TIMES VALUE FOR DISTANCE 170

SELECT ALGORITHM Divide curve algorithm

Figure 9. Frame for system parameters setting

You can see if the input data is not valid, a message dialog will appear and the user will be asked to re input the number.


PLEASE SET SYSTEM PARAMETERS

ROOT PRECISION VALUE 6

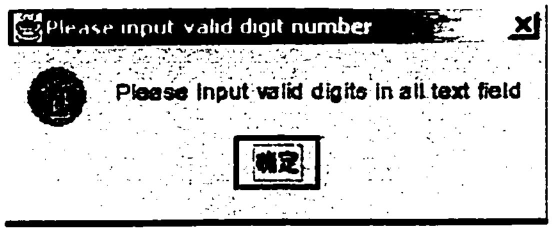
ANGLE THRESHOLD VALUE 90

DIVISION-TIMES-VALUE 170

ERROR THRESHOLD VALUE q

SELECT ALGORITHM Divide curve algorithm

Figure 10. When you input character q in one text field.



Please input valid digits in all text field

Figure 11. You will see a message box asking you to re input valid digits. After a user has pressed the "Select image" button, a file chooser dialog will appear.

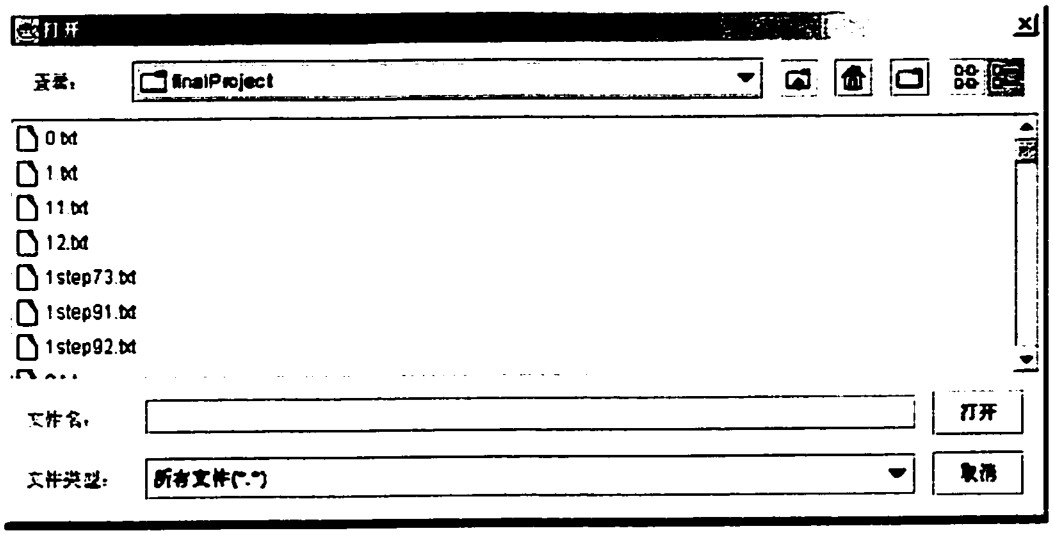
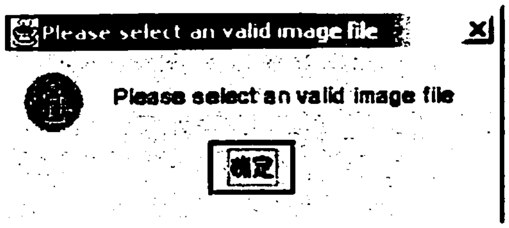


Figure 12. The file chooser

If the user does not select a valid file, the system will ask the user to re select one.



Please select an valid image file

Figure 13. A dialog box appears when you select an invalid image.

If the user has selected an image file, the system first closes all previous frames and the "Run" button is enabled.

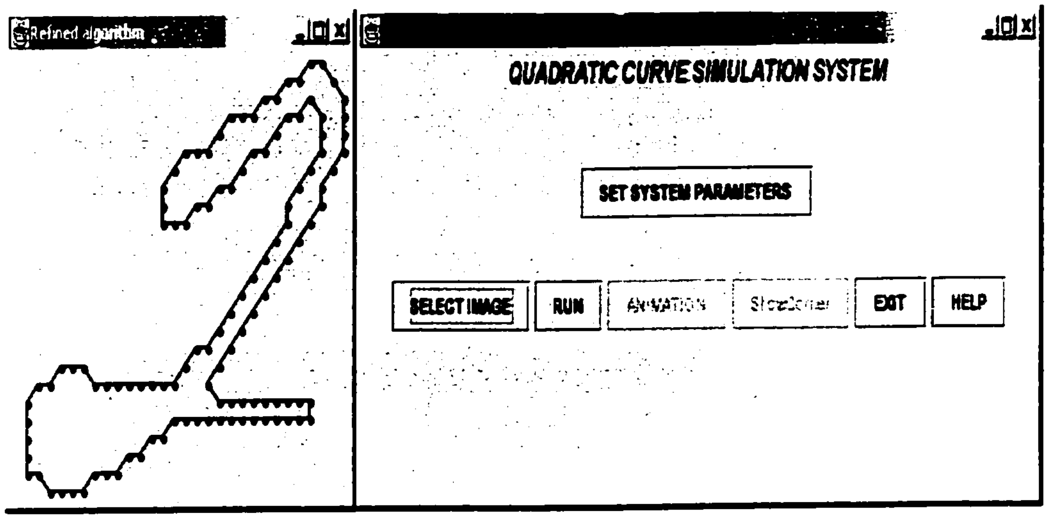


Figure 14. The image is displayed

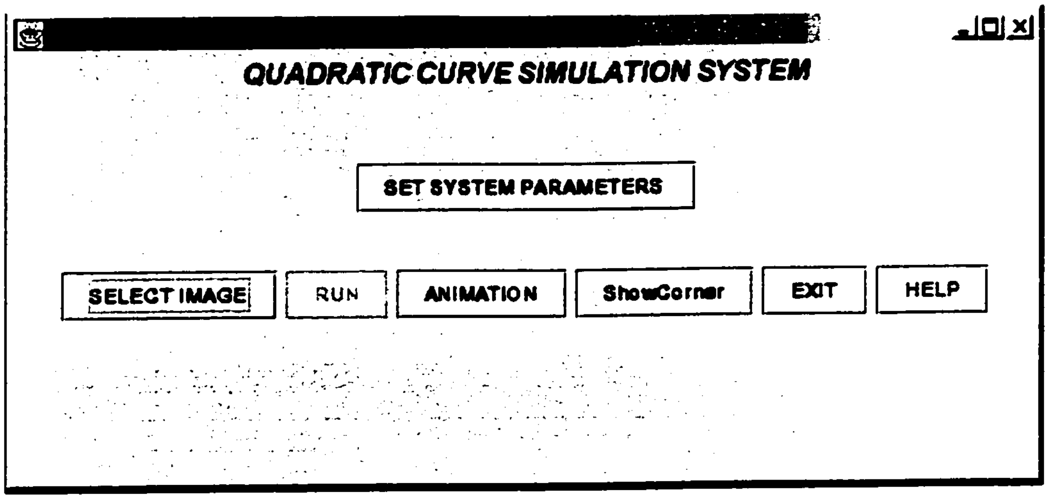
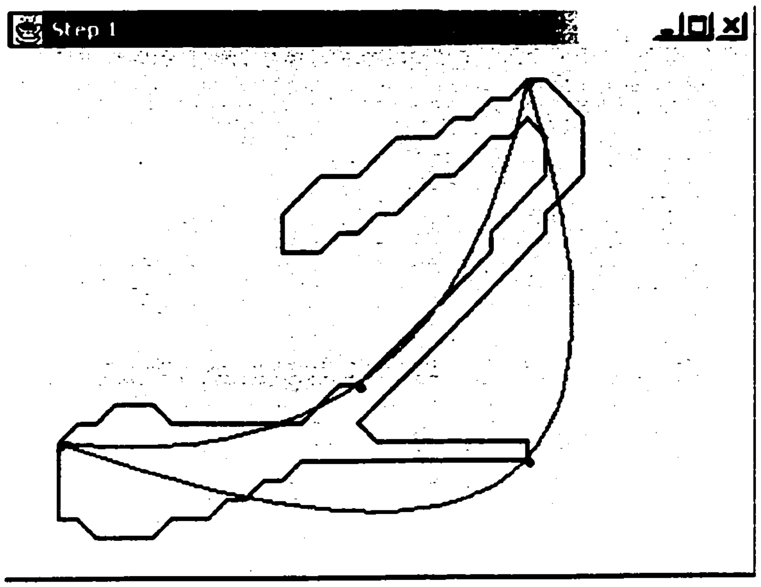
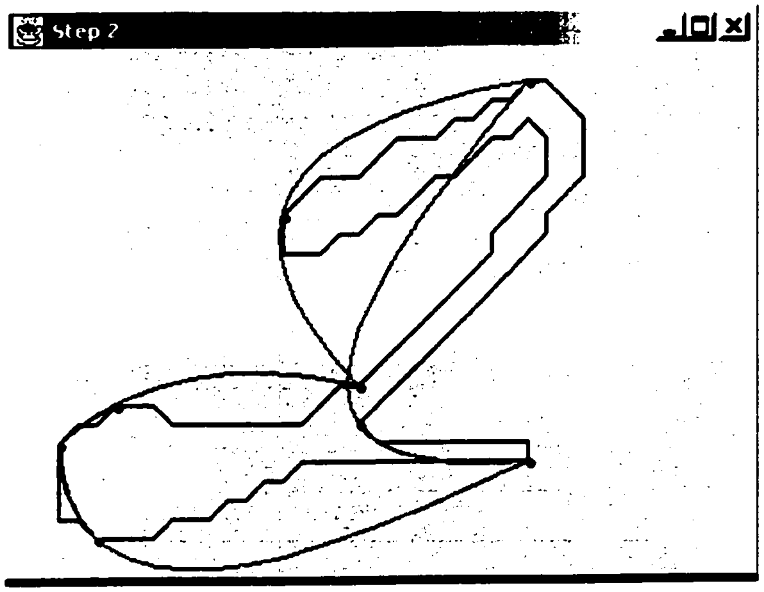


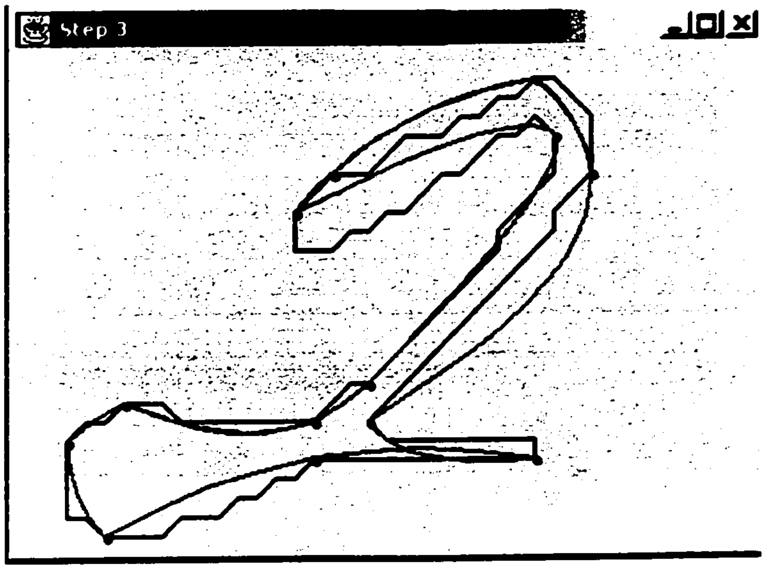
Figure 15. The “Animation” button is enabled

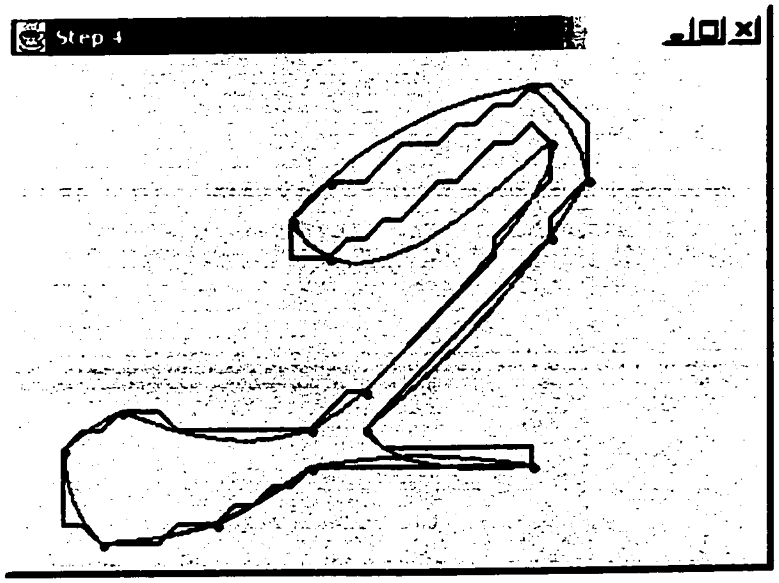
When a user press the “Run” button, the “Animation” button is enabled, but the “Run” button is disabled.

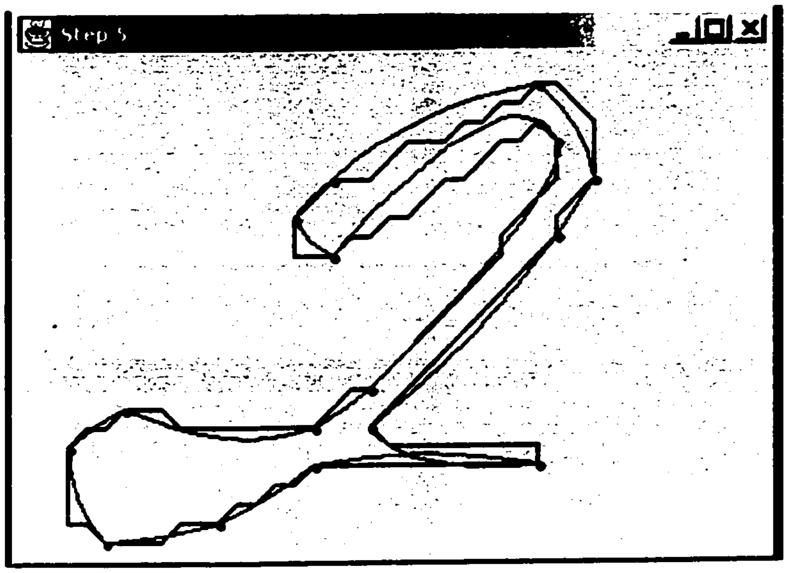
When the user presses the “Animation” button, the button will be disabled and the animation frames are created and displayed.


Step 1


Step 2


Step 3


Step 4



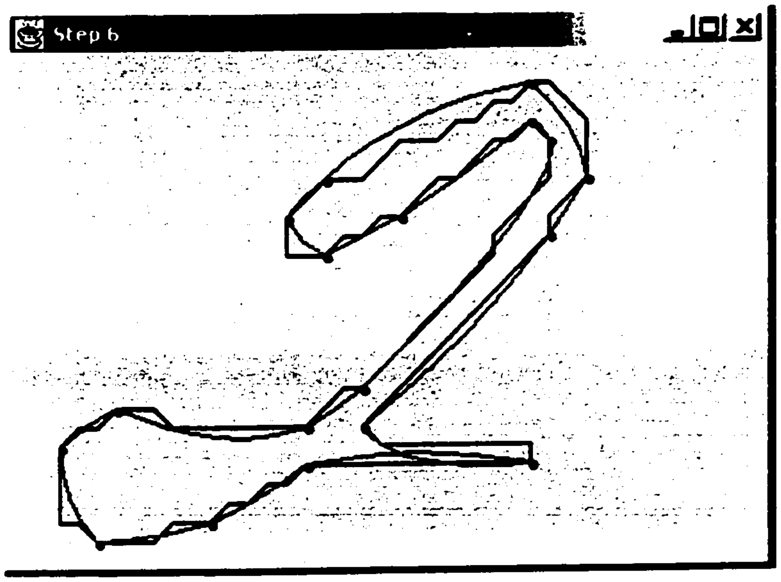


Figure 16. Animation frames

When you press the “Show corner” button, the show corner frame appears.

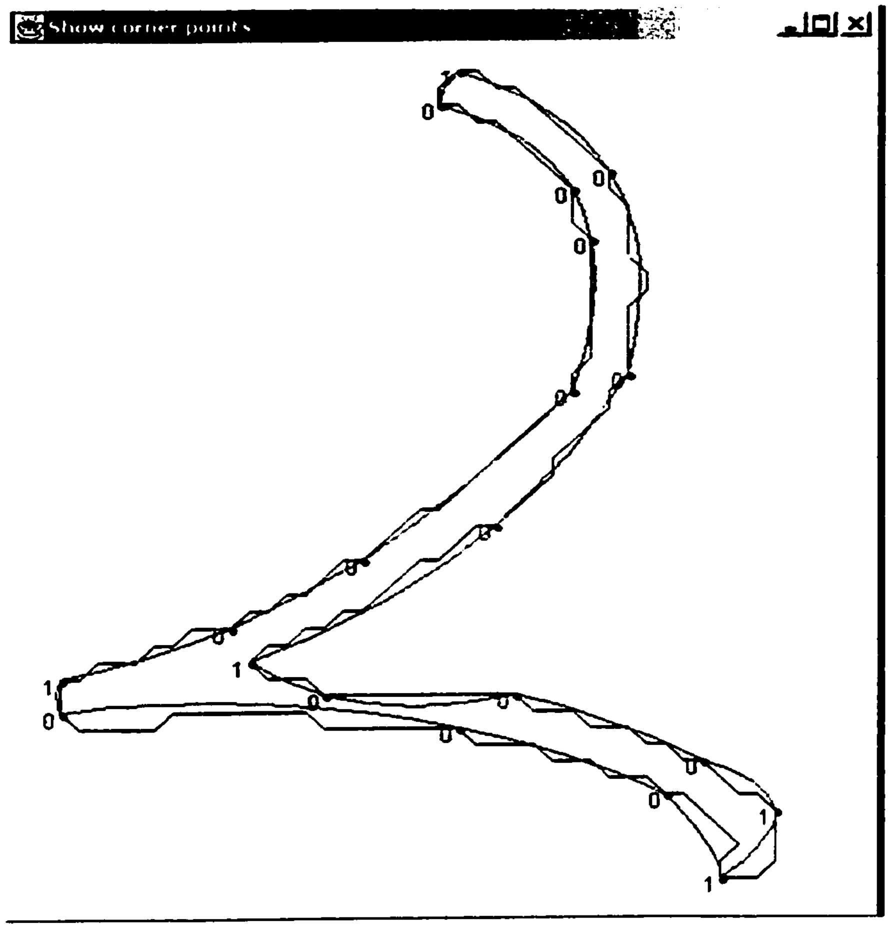
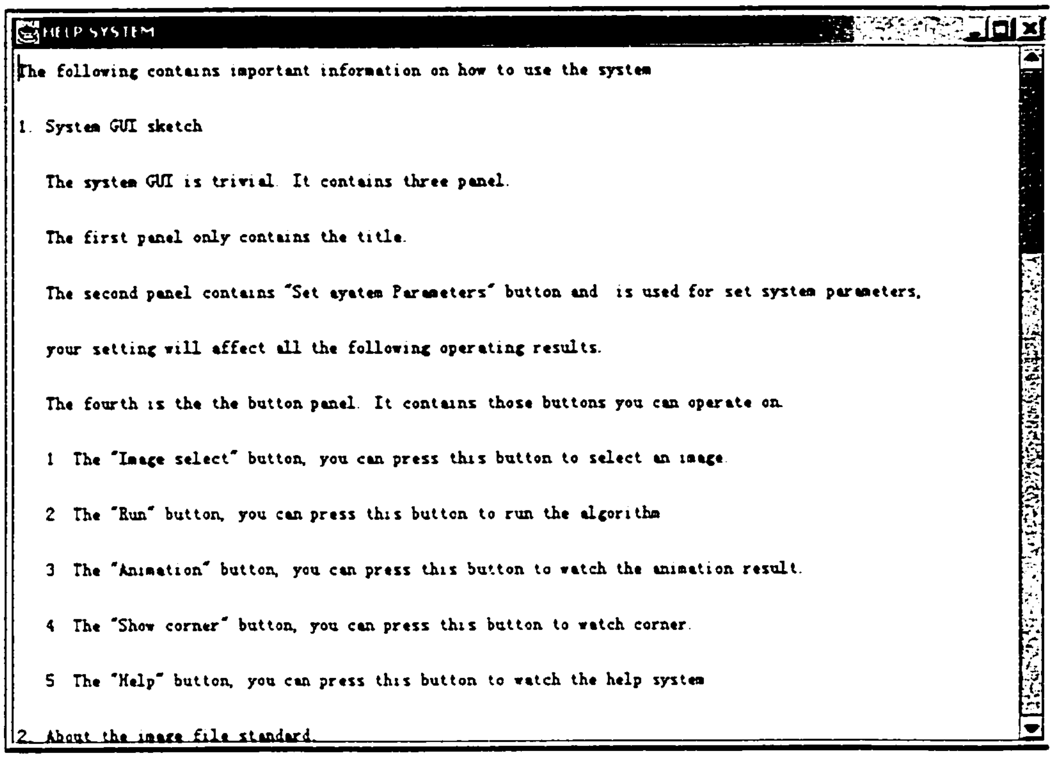


Figure 17. The Show corner frame

When you press the "Help" button, the help frame appears.


HELP SYSTEM

The following contains important information on how to use the system

1. System GUI sketch

The system GUI is trivial. It contains three panel.

The first panel only contains the title.

The second panel contains "Set system Parameters" button and is used for set system parameters,

your setting will affect all the following operating results.

The fourth is the the button panel. It contains those buttons you can operate on.

1 The "Image select" button, you can press this button to select an image.

2 The "Run" button, you can press this button to run the algorithm

3 The "Animation" button, you can press this button to watch the animation result.

4 The "Show corner" button, you can press this button to watch corner.

5 The "Help" button, you can press this button to watch the help system

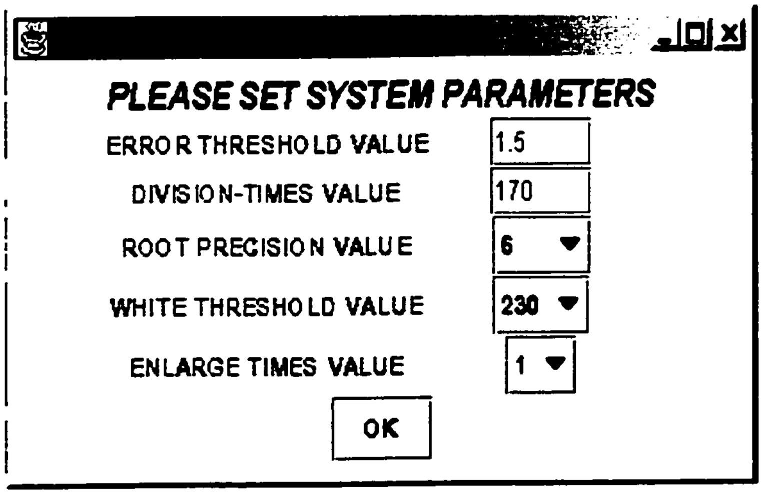
2. About the image file standard

Figure 18. The help frame

4.2.2 Running result of the second format image

The main frame is the same as the Figure 8.

When you press the "SET SYSTEM PARAMETERS" button, a frame will appear as figure 19. We have add another function so that we can watch the enlarged image of format JPG, GIF or JPEG. This is because sometimes it is not obvious to see animation effects.


PLEASE SET SYSTEM PARAMETERS

ERROR THRESHOLD VALUE 1.5

DIVISION-TIMES VALUE 170

ROOT PRECISION VALUE 6

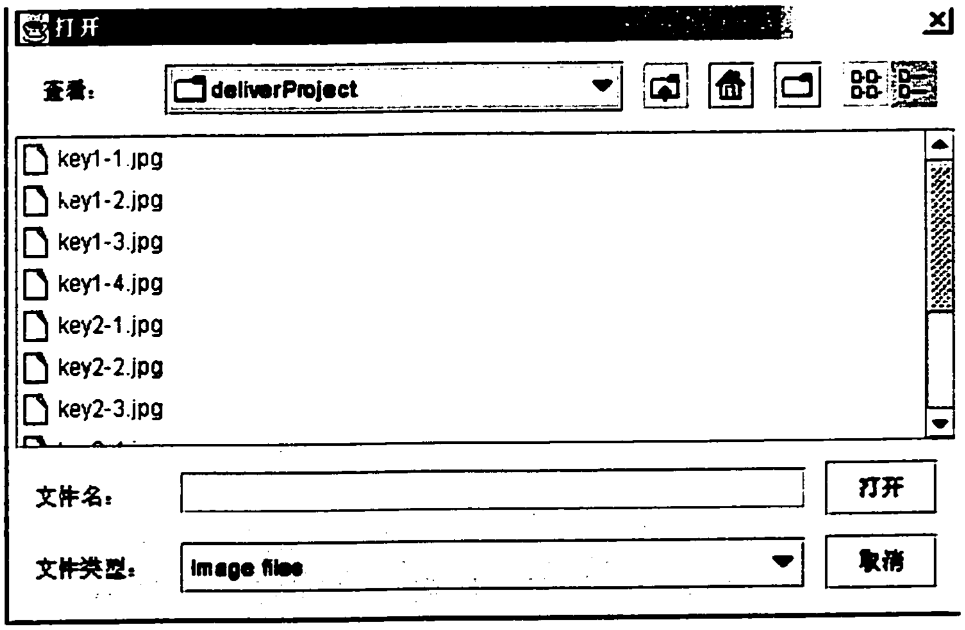
WHITE THRESHOLD VALUE 230

ENLARGE TIMES VALUE 1

OK

Figure 19. The set parameter frame

After a user has pressed the "Select image" button, a file chooser dialog will appear. You can see we have filter out non-image files.


打开

查看: deliverProject

key1-1.jpg
key1-2.jpg
key1-3.jpg
key1-4.jpg
key2-1.jpg
key2-2.jpg
key2-3.jpg

文件名:

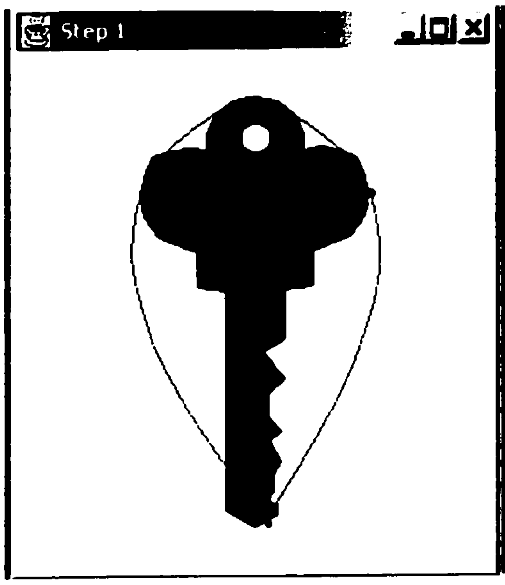
文件类型: Image files

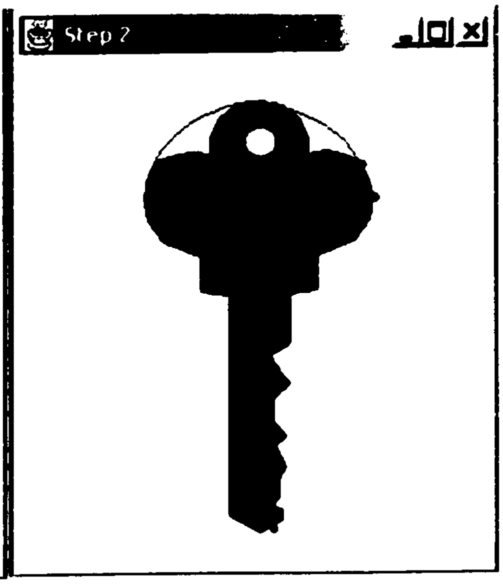
打开

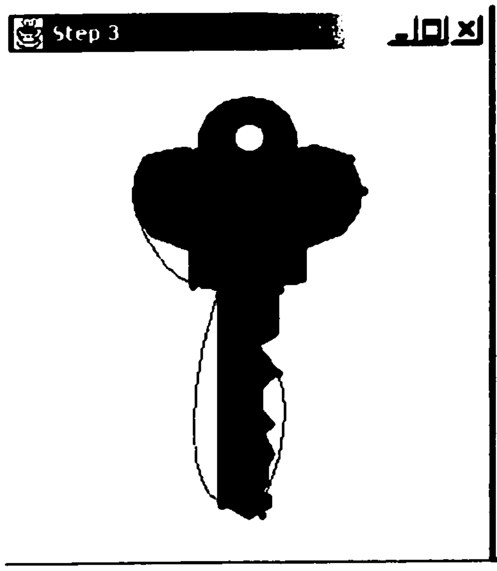
取消

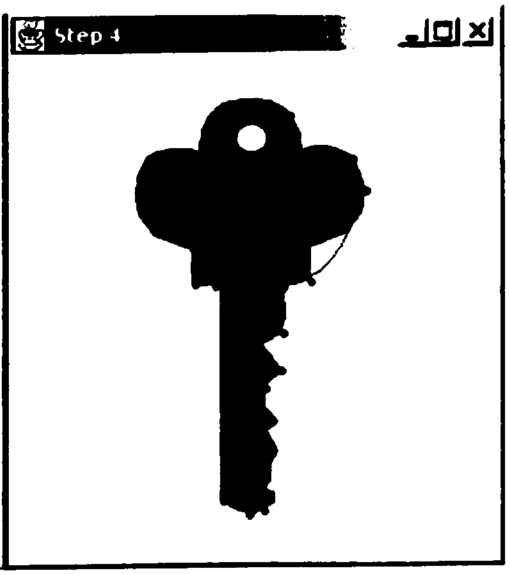
Figure 20. The file chooser dialog

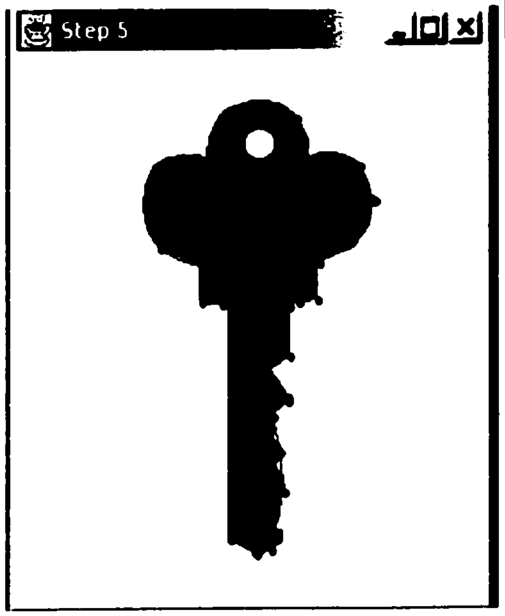
After a user has selected one image and pressed the "Run" button and then the "Animation" button, the animation frames will appear. The following nine frames are animation steps of an image under default system settings.

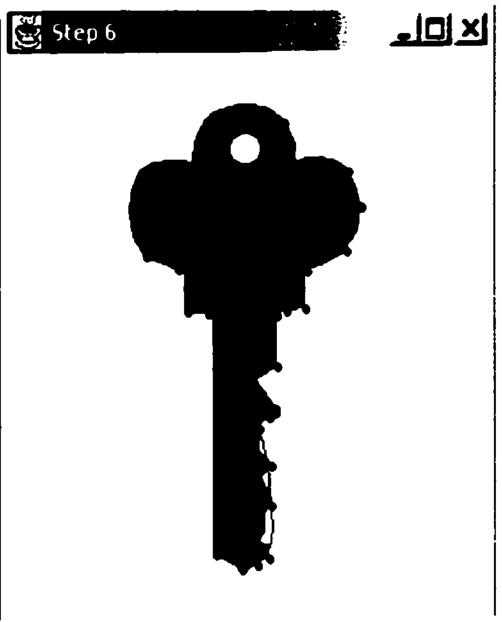


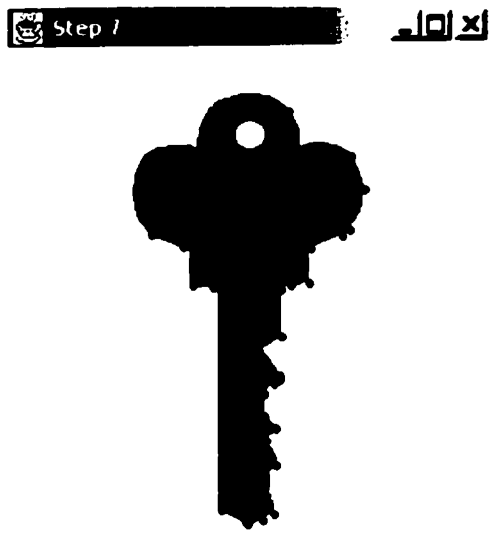


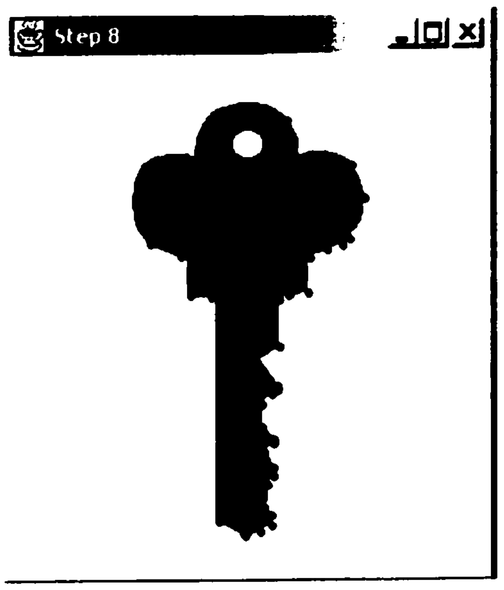





Step 5


Step 6


Step 7


Step 8

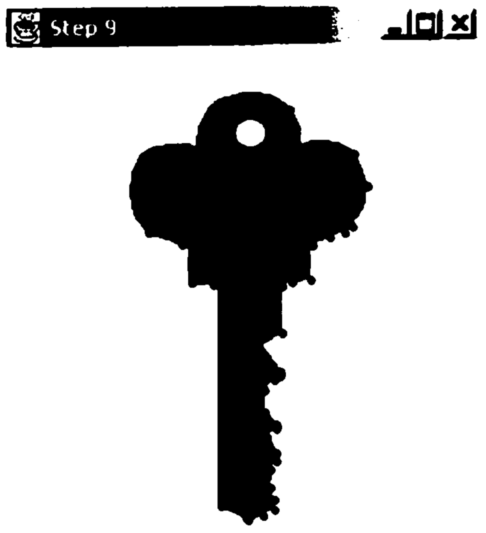


Figure 21. The animation frames of the a key image

We can see some points are congested, if we enlarge the image the effects can be seen more clearly.

The following one figure shows the last step frame of one image animation, you can see the effects are more obvious. The image is enlarged two times in both directions.

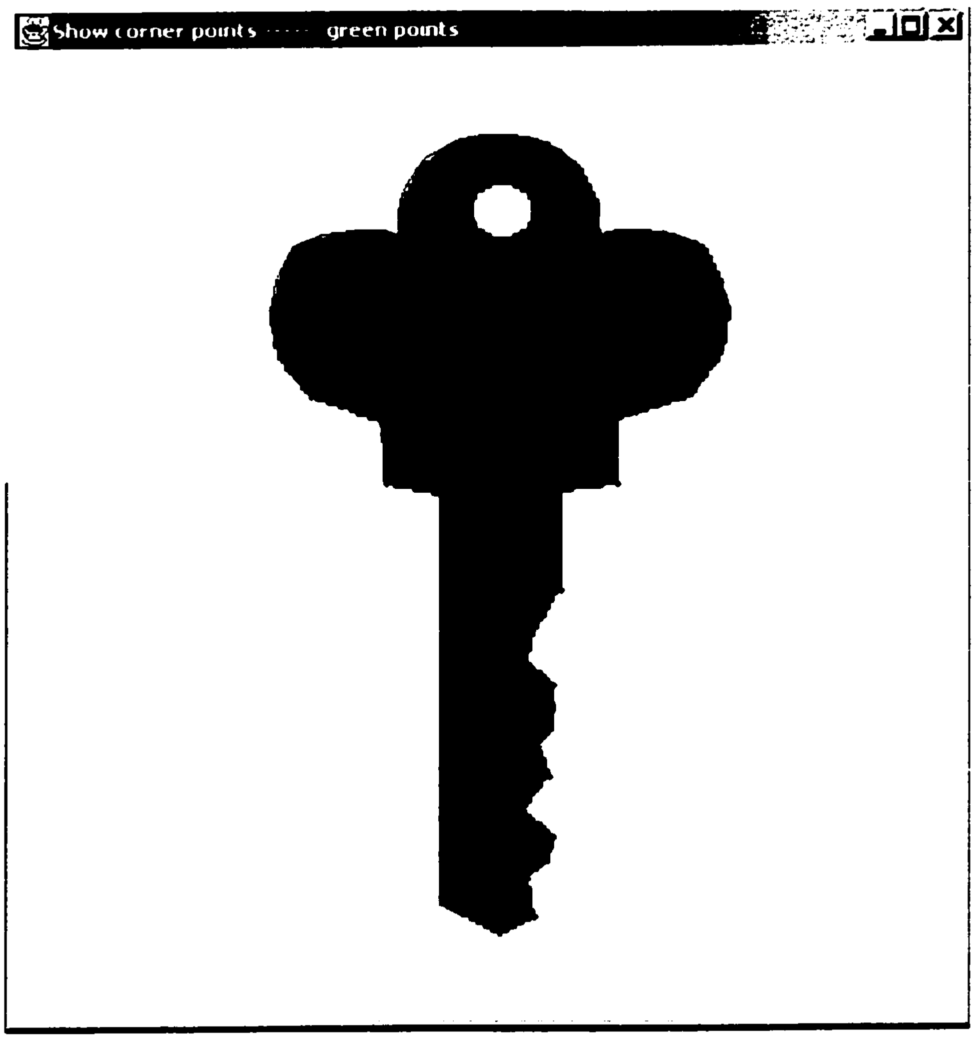


Figure 22. Those corner points of a key image when it is enlarged 2 times

4.3 Discussion of the result.

The performance of the system for an image is the steps required to finish the algorithm. There are many parameters that you can set, the error-threshold is the parameter that affects the performance most: the smaller the error-threshold is, the more are the steps. Other factors can also affect performance: the resolution, the smoothness of an image. We give some statistics data about the system performance.

Resolution	Character	Error-Threshold	Steps
256*256		1.5 pixels	9
256*256		3.5 pixels	8
256*256		2.5 pixels	8
256*256		1.0 pixels	10
256*256		5.0 pixels	6
27*26	0	2.0 pixels	3
23*17	1	1.0 pixels	3
51*40	2	3.5 pixels	3
27*30	2	4.0 pixels	3
35*38	0	2.0 pixels	3
26*16	9	2.0 pixels	4

Resolution	Character	Error-Threshold	Steps
30*29	4	1.5 pixels	4
35*32	5	1.5 pixels	6
26*24	6	1.5 pixels	4
23*20	7	1.0 pixels	6
32*23	8	1.0 pixels	6
35*24	9	1.0 pixels	5
40*26	9	2.0 pixels	4
51*40	2	2.0 pixels	4
51*40	2	4.0 pixels	3
35*38	7	2.5 pixels	3
26*16	9	4.0 pixels	3

From the last two rows in table 1 we can see that given the same error-threshold, smaller resolution does not mean fewer steps.

4.4 Implementation

We will not talk too much about the implementation. You can see the source code for details. I just want to point out that:

Our implementation follows the Java convention standard:

1. Class names start with capital letters.
2. Variable names start with lower case letters.

3. Use long variable names.
4. Use the java document standards.

Chapter 5 Conclusion and future work

5.1 Conclusion

We here build a system that can run the algorithm that uses quadratic curve segments to represent contours of character images; the system can also detect corner points. Our result is a stand-alone application coded with Java. Our system can run the algorithm to high and low-resolution images; finally we compare the performance of different images under different system settings. We can draw the conclusion that quadratic curves can be used to find corner points on the contours of character images or 2D black and white images.

5.2 Future work.

There are many algorithms that are implemented in this system; some are quite well known and easy to implemented; more algorithms need to be figured out by myself. So many algorithms can still be improved, for example although I have implemented two knots detection algorithms, they are not quite well and a lot of work can still be done.

Chapter 6 Reference

1. Herbert Schildt, The complete reference for Java 2 fourth Edition, OSBORNE 2001.
2. J. Canny. "A computational approach to edge detection". IEEE Trans. Patt. Anal. Machine Intell., PAMI-8:679-698, 1986.
3. B. D. Dessimoz. . "Curve Smoothing for Improved Feature Extraction From Digitized Pictures". Signal Processing, 1:205-210, 1979.
4. D. P. Dobklin, S. V. F. Levy, W. P. Thurston, and A. R. Wilks. "Contour tracking by piecewise Linear Approximation". ACM Trans. Graphics, 9 (4) :389-423, 1990.
5. M. J. Eccles, M. P. C. McQueen, and D. Rosen. "Analysis of the Digitized Boundaries of Planar Objects". Pattern Recognition, 9:31-42, 1997
6. H. Freeman. "On the Encoding of Arbitrary Geometric Configurations". IRE Trans Electron. Comput., 10:260-286, 1961.
7. Lee, J.S., Sun, Y.N., Chen, C.H., 1995. Multiscale corner detection by wavelet transform. IEEE Trans. Image Process. 4, 100-104.
8. Mallat, S., Zhong, S. 1992. Characterization of signals from multiscale edges. IEEE Trans. Pattern Anal. Mach. Intell. PAMI-14 (7), 710-732.
9. David M. Geary, Graphic Java 1.2 mastering the JFC 3rd edition, Sun micro system.
10. Abber George Ghuneim, Contour tracing, http://www.cs.mcgill.ca/~aghnei/ alg.html 2001.
11. Theo Pavlidis, Algorithms for Graphics and Image Processing, chapter 7 Computer Science Press, Rockville, Maryland, 1982.
12. L.H.Yang, C.Y.Suen and T.D.Bui, Representation of Contours by Connected Quadratic Curves and its application to 2D image recognition. To be appear.