OPTICAL CHARACTER RECOGNITION

Using FEZ Spider Kit and .NET Gadgeteer

Student name - Matrikel

Ngoc An Ha - 967813

Nam Hung Le – 967826

Duy Thuoc Tran – 967897

Xuan Tung Vu – 967907

S.M.A.MANNAF- 1065361

Date

May 12th , 2015

Contents

Introduction

Written by Ngoc An Ha-967813

Nowadays, the technique of Optical Character Recognition is being refined more and more due to its usefulness. Various people tried to use their own algorithm on many types of device. Optical Character Recognition results more in the training phase. By comparing the test image's patterns it receives with the record database, Optical Character Recognition can help computers recognize various type of different structures or different characters.

This project provides an attempt to develop Optical Character Recognition by using FEZ Spider circuit board and .NET Gadgeteer library. Due to the device's characteristic, we use the camera capabilities of the device and the image processing strength of the computer. In this project, we will learn how to capture an image of the test subject, transfer it to the computer and process Optical Character Recognition algorithm on the test subject, and provide an acceptable result.

I. Image capture and host on device server:

(S.M.A.MANNAF, Matrikel: 1065361)

Capture the image of character and then import it to a pc/laptop is the first task of our Character recognition project. To detect the character we use FEZ Spider .NET Gadgeteer Kit and we use C# as a programming language to make the kit work. We use FEZ Spider .NET Gadgeteer Kit with different module and use some other software, which are required by FEZ Spider .NET Gadgeteer Kit.

FEZ Spider .NET Gadgeteer Kit includes the following module for our project:

01. FEZ Spider Mainboard
02. Display TE35 Module
03. USB Client DP
04. Camera
05. Button
06. EthernetJ11D

List of Software we used for the project:

01. Visual Studio 2013 express for Desktop
02. Microsoft .NET Micro Framework 4.3
03. **Microsoft .NET Gadgeteer Core**

The entire process of the image capture and import the image to a pc/laptop can be divided in to three segments.

01. Setup and configuration
02. Deploying and Running the Program
03. Image capture and import

A. Setup and configuration:

At first, we connect the entire FEZ Spider .NET Gadgeteer Kit module to the main board properly which is described clearly in the instruction manual. Then we connect the Ethernet module to a network or pc with the help of network cable. Actually we use the Ethernet port as a communication medium with FEZ Spider .NET Gadgeteer Kit and pc/laptop.

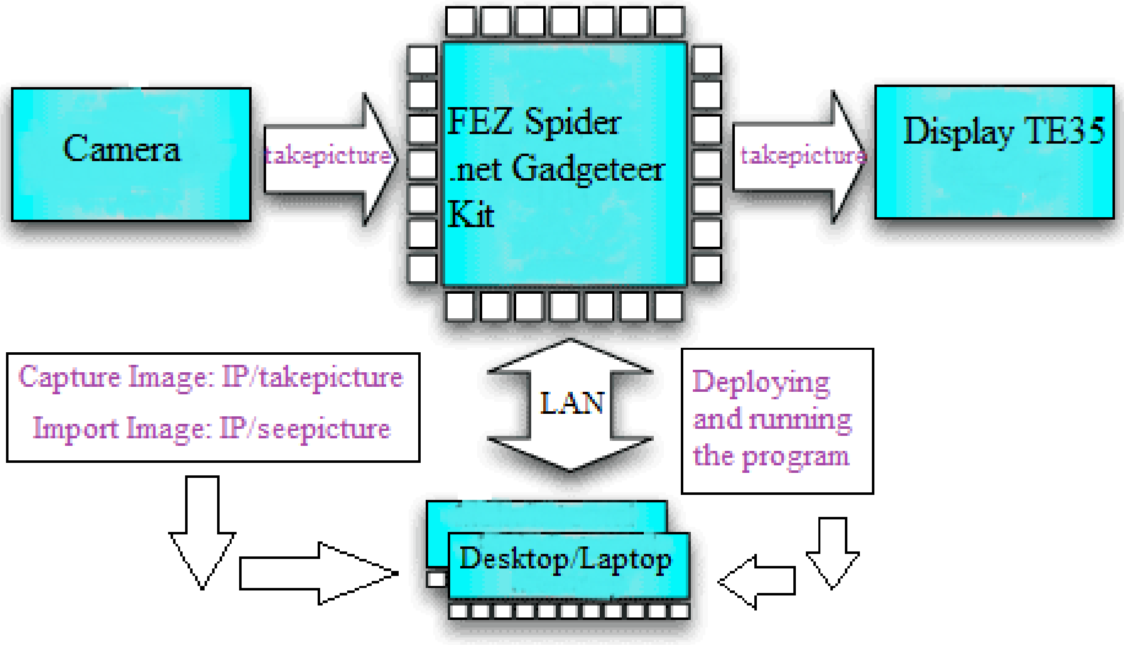

Camera
takepicture
FEZ Spider .net Gadgeteer Kit
takepicture
Display TE35
Capture Image: IP/takepicture
Import Image: IP/seepicture
LAN
Deploying and running the program
Desktop/Laptop

Figure 1 Block diagram of Image capture & Import

B. Deploying and Running the Program

After setup and configuration of FEZ Spider .NET Gadgeteer Kit we need to power it on from usb port of pc/laptop and run the C# program for deploy. The unique thing is in this project we used a webserver created with .NET Gadgeteer Kit. When making a web server, we need to connect our Gadgeteer to at the very least to our local network with switch/router. More importantly for this project we need Ethernet patch lead to connect Gadgeteer Kit with pc/laptop. While we are programming it, the Gadgeteer will also be connected to our computer, which will probably also be providing it power from the USB connection. Once the Gadgeteer is programmed we don't need the USB connection anymore.

After deploying the program successfully a status on Visual Studio 2013 indicates that FEZ Spider .NET Gadgeteer Kit is ready for operation.

C. Image capture and import

After successfully deploying and running the program we need to open a web browser. In address bar we have to write the device IP address (Static[1]/ Dynamic[2]) and takepicture command separated by a single back slash (IP Address/takepicture). This command will help to take a picture with camera module. It is important to mention that, we need to focus the camera properly in case we need a very clear picture. It takes few seconds to capture the image and at preliminary stage we can see this image on Display TE35 Module. To import this image to pc/laptop we need to write a different command seepicture with device ip address separated by back slash (IP Address/seepicture). Within few second we can see the picture on the webpage.

Further this image can be use for analysis and character recognition which are done by the other group members and also described in the other part of this report.

D. Scenarios:

Communication between pc/laptop and FEZ Spider .NET Gadgeteer Kit with code are described below.

1. Use case 1:

At first we use the dynamic IP address for communication between pc and FEZ Spider .NET Gadgeteer Kit. In this case we need to create a network with the help of a router/switch. It looks like it is not very simple and then we try to suppress some complexity.

```
Gadgeteer.NETworking.WebEvent hello;
Gadgeteer.NETworking.WebEvent takePicture;
Gadgeteer.NETworking.WebEvent seePicture;
```

The event hello, takePicture and seePicture is a WebEvent and to some extent, it is similar to the idea of a webpage. In our code, we always display the output in the browser, but in a more complex case, we can set up several WebEvents for different actions on the web server.

```
ethernetJ11D.UseDHCP();
ethernetJ11D.NETworkUp += new
GTM.Module.NETworkModule.NETworkEventHandler(ethernet_NetworkUp);
camera.PictureCaptured += new Camera.PictureCapturedEventHandler(camera_PictureCaptured);
```

[1] Static IP Address: A permanent IP address given by service provider. it does not change. The device always has the same IP address.

[2] Dynamic IP Address: Most devices use dynamic IP addresses, which are assigned by the network when they connect. These IP addresses are temporary, and can change over time.

Above mentioned code represents the ProgramStarted method first tells the ethernet module to use DHCP. It is a mechanism that automatically assigns an unique IP address to a device when it is connected to the network. In our project .NET Gadgeteer does not end up with an IP address that is already in use. From above mentioned code we see that ProgramStarted also attaches handlers to the events NetworkUp and PictureCaptured.

```
hello = Gadgeteer.NETworking.WebServer.SetupWebEvent("hello");
hello.WebEventReceived += new WebEvent.ReceivedWebEventHandler(hello_WebEventReceived);
takePicture = Gadgeteer.NETworking.WebServer.SetupWebEvent("takepicture");
takePicture.WebEventReceived += new
WebEvent.ReceivedWebEventHandler(takePicture_WebEventReceived);
seePicture = Gadgeteer.NETworking.WebServer.SetupWebEvent("seepicture");
seePicture.WebEventReceived += new
WebEvent.ReceivedWebEventHandler(seePicture_WebEventReceived);
```

The method Ethernet NetworkUp makes the web server aware of the hello, takePicture and seePicture .The string "hello", "takePicture" and "seePicture" associates the name "hello", "takePicture" and "seePicture" with this web event. That means that if we put "/hello", "/takePicture" and "/seePicture" on the end of our URL in the browser, it is this WebEvent that will be invoked. That is why we attach yet another handler, this time to the event WebEventReceived of hello, takePicture and seePicture.

```
string content = "<html><body><img src=" +'"'+ pic +'"'+ "alt='Mountain View'
style='width:304px;height:228px'></body></html>";
string content = "<html><body><h1>Hola Mundo</h1><img src= \"logo.jpg\"></body></html>";
byte[] bytes = new System.Text.UTF8Encoding().GetBytes(content);
```

The handler method needs to construct some HTML to be displayed on the browser. However, Respond does not take a string, but rather a byte array, so we need to do a little magic to convert our string into a byte array. The third argument to Respond tells the browser what kind of response to expect to be contained in the byte array, and "text/html" is the standard way of saying it is HTML.

2. Use case 2:

To reduce complexity we simply use static IP address and edit the program according to static IP Address. We fixed a IP address in the code and we have to change IP address of our pc/laptop also before deploying of program. In this case we don’t need any router for communication between pc and FEZ Spider .NET Gadgeteer Kit. We need to connect these two device simply with a network cable.

```
Gadgeteer.NETworking.WebEvent takePicture;
```

The event takePicture is a WebEvent and in this code we use only one WebEvent for making a simple and efficient code.

```
ethernetJ11D.UseStaticIP("192.168.1.2", "255.255.255.0", "192.168.1.1");
ethernetJ11D.NETworkUp += new
GTM.Module.NETworkModule.NETworkEventHandler(ethernet_NetworkUp);
camera.PictureCaptured += new Camera.PictureCapturedEventHandler(camera_PictureCaptured);
```

Above mentioned code represents the ProgramStarted method first tells the ethernet module to use StaticIP. It is a mechanism that we have to put manually assign an unique IP address to a device when it is connected to the network. In our code we mentioned some ip address such as IP address, Subnet mask and Default gateway which we have to assign manually into our pc/laptop. From above mentioned code we see that ProgramStarted also attaches handlers to the events NetworkUp and PictureCaptured.

```
takePicture = Gadgeteer.NETworking.WebServer.SetupWebEvent("takepicture");
takePicture.WebEventReceived += new
WebEvent.ReceivedWebEventHandler(takePicture_WebEventReceived);
```

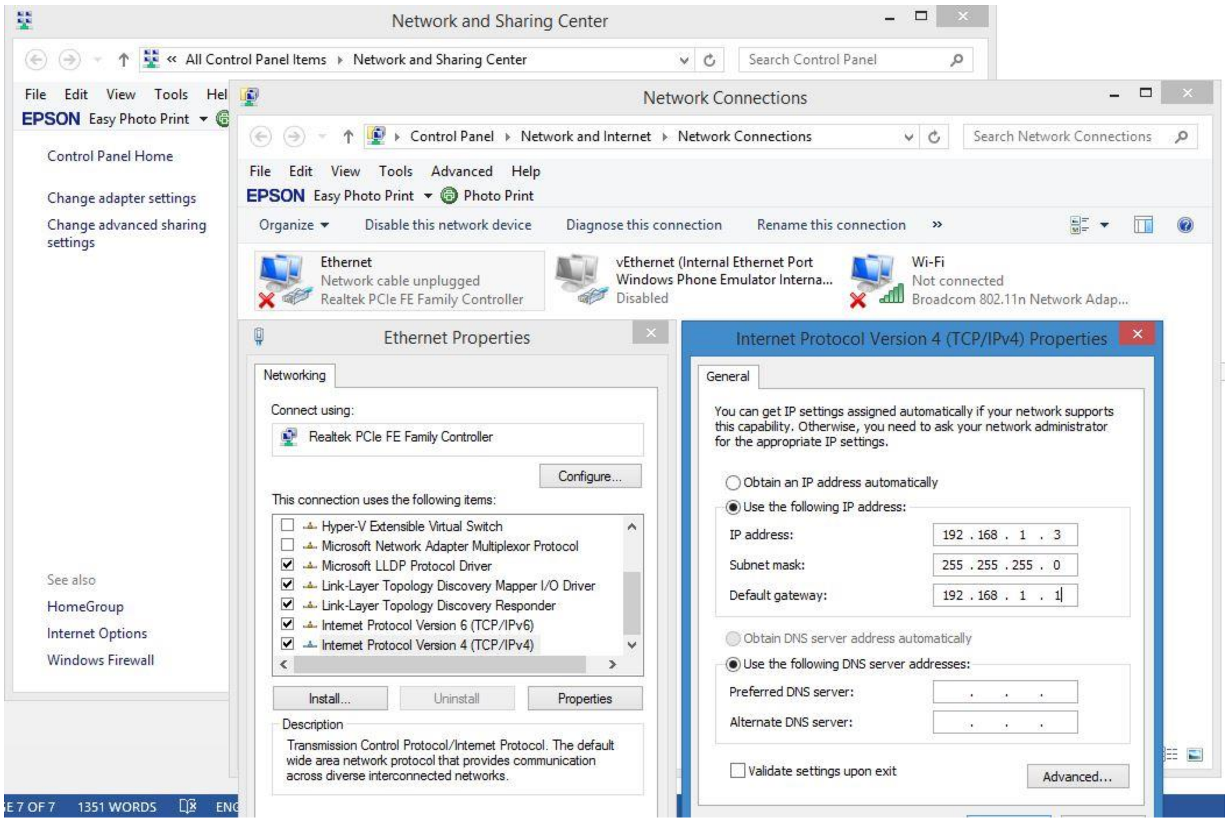
The method Ethernet NetworkUp makes the web server aware of the takePicture. The string “takePicture” associates the name “takePicturewith this web event. That means that if we put “/takePicture” on the end of our URL in the browser, it is this WebEvent that will be invoked. That is why we attach yet another handler, this is the event WebEventReceived of takePicture.

II. CONNECT FEZ SPIDER DEVICES TO THE MAIN APPLICATION

Written by Ngoc An Ha, Matrikel 967813

A. Connect the device to the computer

For the connection between the FEZ Spider Kit and the computer, our group chose to use the second scenario, which is setting the static IP for the device. For this device, we set the device IP is "192.168.1.2,", the subnet mask is "255.255.255.0", default gateway is "192.168.1.1" .In order to connect to the device using this method, we also have to change the computer static IP for the Ethernet port.



Network and Sharing Center
All Control Panel Items › Network and Sharing Center
Search Control Panel
File Edit View Tools Help
EPSON Easy Photo Print
Control Panel Home
Change adapter settings
Change advanced sharing settings
See also
HomeGroup
Internet Options
Windows Firewall
Network Connections
Control Panel › Network and Internet › Network Connections
Search Network Connections
File Edit View Tools Advanced Help
EPSON Easy Photo Print Photo Print
Organize Disable this network device Diagnose this connection Rename this connection
Ethernet Properties
Networking
Configure...
Install... Uninstall Properties
Description
Internet Protocol Version 4 (TCP/IPv4) Properties
General
Advanced...

Ethernet
Network cable unplugged
Realtek PCIe FE Family Controller

vEthernet (Internal Ethernet Port Windows Phone Emulator Interna...
Disabled

Wi-Fi
Not connected
Broadcom 802.11n Network Adap...

Connect using:
Realtek PCIe FE Family Controller

This connection uses the following items:
- [ ] Hyper-V Extensible Virtual Switch
- [ ] Microsoft Network Adapter Multiplexor Protocol
- [x] Microsoft LLDP Protocol Driver
- [x] Link-Layer Topology Discovery Mapper I/O Driver
- [x] Link-Layer Topology Discovery Responder
- [x] Internet Protocol Version 6 (TCP/IPv6)
- [x] Internet Protocol Version 4 (TCP/IPv4)

Transmission Control Protocol/Internet Protocol. The default wide area network protocol that provides communication across diverse interconnected networks.

You can get IP settings assigned automatically if your network supports this capability. Otherwise, you need to ask your network administrator for the appropriate IP settings.

Obtain an IP address automatically [✓ Use the following IP address:]

IP address:	192 . 168 . 1 . 3
Subnet mask:	255 . 255 . 255 . 0
Default gateway:	192 . 168 . 1 . 1

Obtain DNS server address automatically [✓ Use the following DNS server addresses:]

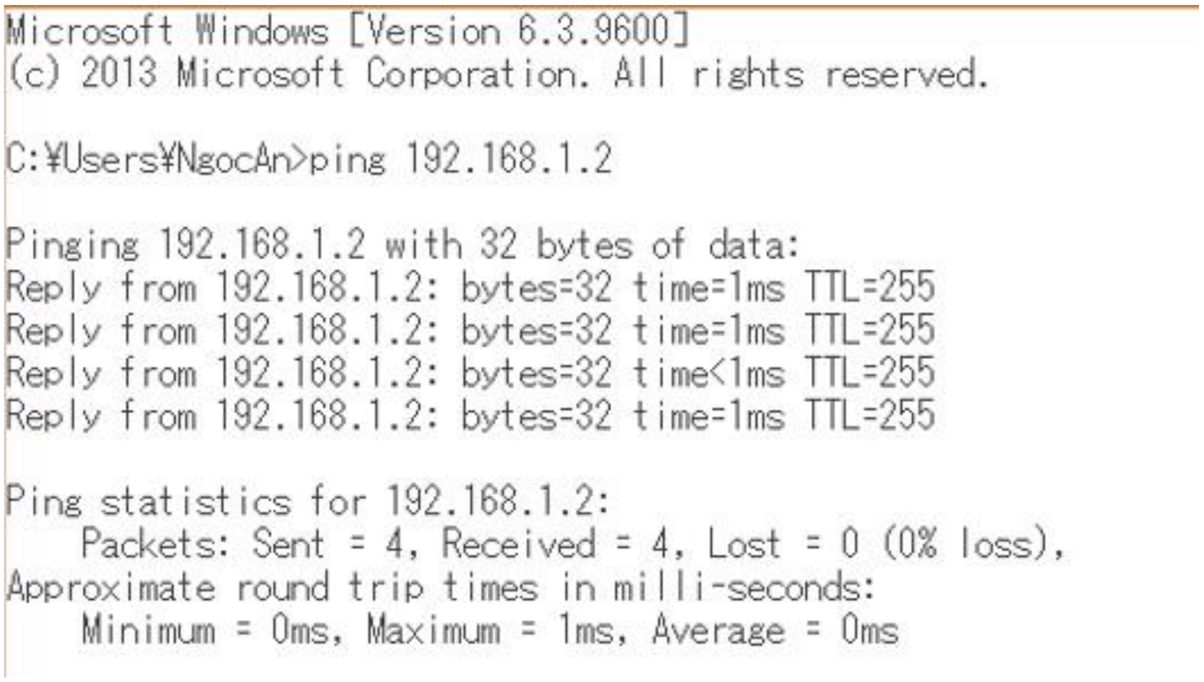
Preferred DNS server:	. . .
Alternate DNS server:	. . .

- [ ] Validate settings upon exit

Figure 2 How to change the static IPv4 of the computer

The reason why we had to resort to use Ethernet as a mean of connection between the device and computer is that it is not possible to use USB connection, since it is made only for debugging, not for connecting. Using Ethernet is by far the fastest and most stable way we can achieved. By changing the default gateway to be the same as the device, we made a small client-server like relation between the FEZ Spider and computer. Although this can be achieve by one computer for both deploying and control, it is more highly recommended to separate the deploying process to one computer, while another can connect the device, which also, the main computer which will process the algorithm. This way, it is safer and more stable.

After we connect the device to the computer by using the Ethernet cable and deploy the application for the FEZ Spider, we can check the device connection by pinging the address "192.168.1.2" of the FEZ.



```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:¥Users¥NgocAn>ping 192.168.1.2

Pinging 192.168.1.2 with 32 bytes of data:
Reply from 192.168.1.2: bytes=32 time=1ms TTL=255
Reply from 192.168.1.2: bytes=32 time=1ms TTL=255
Reply from 192.168.1.2: bytes=32 time<1ms TTL=255
Reply from 192.168.1.2: bytes=32 time=1ms TTL=255

Ping statistics for 192.168.1.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 1ms, Average = 0ms
```

Figure 3 Success connecting to the device by pinging the FEZ Spider

B. Programming the application

1. Function prepare the server request

Because we use the web event "takepicture" to trigger the camera function of the device, so the address of the client request will be a combination of the device IP address and the event request:

```
private string GetURL(string uri)
 {
  if (!(uri.Contains("http://") || uri.Contains("https://")))
   {
    uri = "http://" + uri + "/takepicture";
    return uri;
   }
  else return uri;
 }
```

2. Function download image from the device web server to the computer

First, we set the file name and directory path for the test image

```
string url = "192.168.1.2";
string filename = "test.bmp";
string path = AppDomain.CurrentDomain.BaseDirectory;
 string filepath = Path.Combine(path, filename);
```

Then we send a download request to the server to get the data streaming of the image

```
byte[] buffer = new byte[1024];
HttpWebRequest httpRequest = (HttpWebRequest)WebRequest.Create(GetURL(url));
httpRequest.Timeout = 30000;
httpRequest.Method = "GET";
httpRequest.UserAgent = "Mozilla/5.0 (Windows NT 6.1; rv:12.0) Gecko/20100101
Firefox/12.0";
httpRequest.Accept = "image/bmp,image/*;q=0.8,*/*;q=0.5";
```

Next, we create the file by writing all the streaming data bytes into one file and close the streaming, based on the predefine directory path

```
FileStream fileStream = new FileStream(filepath, FileMode.OpenOrCreate,
FileAccess.Write);
using (HttpWebResponse httpResponse =
(HttpWebResponse)httpRequest.GetResponse())
   { using (Stream responseStream = httpResponse.GetResponseStream())
     { int bytesRead;
      while ((bytesRead = responseStream.Read(buffer, 0,
buffer.Length)) > 0)
         {
             fileStream.Write(buffer, 0, bytesRead);
        }
            fileStream.Close();}
   }
```

Finally, we can call the image from the source folder and apply the algorithm

```
 try
      {  Bitmap image = new Bitmap(filepath);
         frame = new Image<Bgr, byte>(image);
      }
      catch (Exception ex)
      { MessageBox.Show(ex.Message);}
```

III. Contour Analysis for image recognition

Written by Xuan Tung Vu - 967907

This chapter includes the theoretical bases of the contour analysis and aspects of its practical application for image recognition. There are 2 main parts in this chapter, the first part is about the main definitions and theorems of the contour analysis, problems of optimization of algorithms of the contour analysis, and the second part contains the results of work of algorithms, problems and shortages of the given method.

A. Contour Analysis (CA) theories [1]

The CA allows to portray, store, analyze, and find the objects presented in the form of the exterior outlines – contours.

The contour includes the essential information on the object shape.

CA can solve the main problems of a pattern recognition - transposition, turn and a rescaling of the image of object. CA methods are fixed to these transformations.

B. Contour definition

The contour is the outline of a figure or body, the edge or line that bounds a shape or objects, separating object from a background.

In the Contour Analysis, the contour is encoded by the arrangement comprising of complex numbers. On a contour, the starting point is fixed. After that, the contour is scanned, and each vector of offset is noted by a complex number a+ib, where:

- a: point offset on x axis
- b: point offset on y axis

The contours are always closed and cannot have self-intersection. The last vector of a contour always leads to the starting point.

Figure 1[1] shows the given definitions

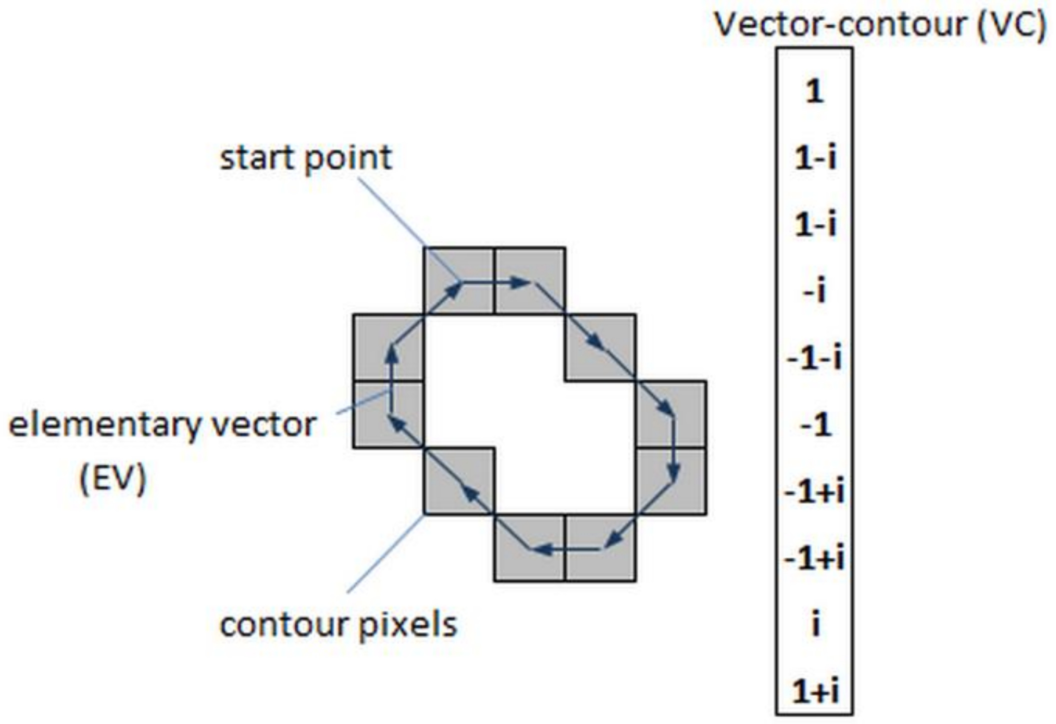


Figure 4: Contour and it elements [1]

Elementary vector (EV): a vector of a contour

Vector contour (VC): sequence of complex valued number

Vector contour ɼ of length k can be designated as:

ɼ= (γ_1, γ_2,…, γ_{k-1})

1. Scalar product of Contour [1]

η= (ɼ,N)== $\sum_{□=0}^{□-1}$(γn, υn) (1)

Where:

- ɼ and N are complex numbers
- k: dimensionality of a VC
- γn: n elementary vector of contour ɼ
- υn: n EV of contour N
- (γn, υn): the scalar product of complex number

The scalar product of complex number [1]:

$(a + ib, c + id) = (a + ib)(c - id) = ac + bd + i(bc - ad)$ (2)

If we multiplied an EV as a vector, their scalar product is [1]:

$$((□,□),(□,□)) = □□ + □□ \quad (3)$$

The normalized scalar product (NSP) [1]:

$$\eta = \frac{(\Gamma,N)}{|\Gamma||N|} \quad (4)$$

Where |Γ| and |N| - the length of contour [1]:

$$|\Gamma| = \sum_{□=0}^{□-1} |□□|^{2\frac{1}{2}} \quad (5)$$

So unity is greatest possible value of norm of NSP if

$$\Gamma = \mu □ \quad (6)$$

where μ is the arbitrary complex number.

At multiplication of complex numbers, their lengths are multiplied, and arguments (angles) - are added. The contour μN means it is the same contour N, but turned both scaled. The scale and turn is defined by a complex number μ.

So if to count a NSP of a vector most on itself, we receive NSP=1, if to turn a contour on 90 degrees, we receive NSP=0+i, turn on 180 degrees gives a NSP=-1. Thus, the real part a NSP will give us a cosine of the angle between contours, and the norm of NSP will be always equal to 1

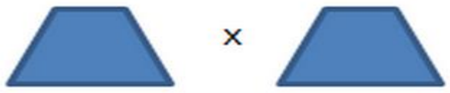
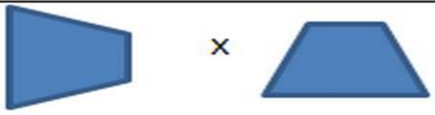
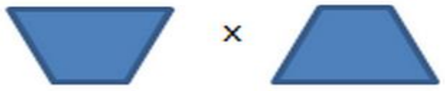
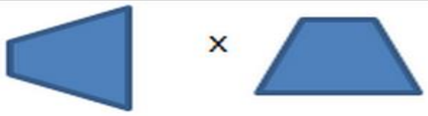
	NSP	Re(NSP)=cos(a)	|NSP|
	1	1	1
	i	0	1
	-1	-1	1
	-i	0	1

Figure 5Properties of the normalized scalar product of contour [1]

2. Correlation Functions of Contours

As we can see, NSP is very useful for finding of contours similar among themselves. However, it has a problem with a starting point choice.

The equation (6) is available when the starting points of contours are coincident. If not, the norm of the NSP will not be equal to a unity.

The intercorrelation function (ICF) of 2 contours [1]:

$$\tau(m) = (\text{Г}, \text{N}(m)) \quad (7)$$

Where:

- m= 0 ,..., k-1
- N(m): a contour received from N by cycle shift by its EV on m of elements.

For example:

If N=(n1,n2,n3) => N(1)=(n2,n3,n1)

The results of intercorrelation function show how much similar of contour Г and N when shifting m position of starting point N.

Maximum norm values [1]:

$$\tau max = \max\left(\frac{\tau(m)}{|\text{Г}||\text{N}|}\right) \quad (8)$$

We can conclude that τmax is the value of similarity of 2 contours, invariant to transposition, scaling, rotation, and starting point shift.

- Norm | τmax | reaches unity for identical contour.
- Arg(τmax) is an angle of rotation of one contour with another.

The autocorrelation function (ACF) is an ICF for the case N=Г [1]:

$$\upsilon(m) = (\text{Г}, \text{Г}(m)) \quad (9)$$

Where m= 0, ..., k-1.

Some properties of the ACF:

- The choice of starting point of a contour will not affect the result of ACF.
- The norm of ACF is symmetric concerning a central reference k/2.

For example: N= (n1, n2, n3, n4)

⇨ Norm of ACF (0) and ACF (4) are coincident, norm of ACF (1) and ACF (3) are coincident.

- If any contour is symmetric, its ACF is also symmetry.

Figure 3 will give some overview about this property.

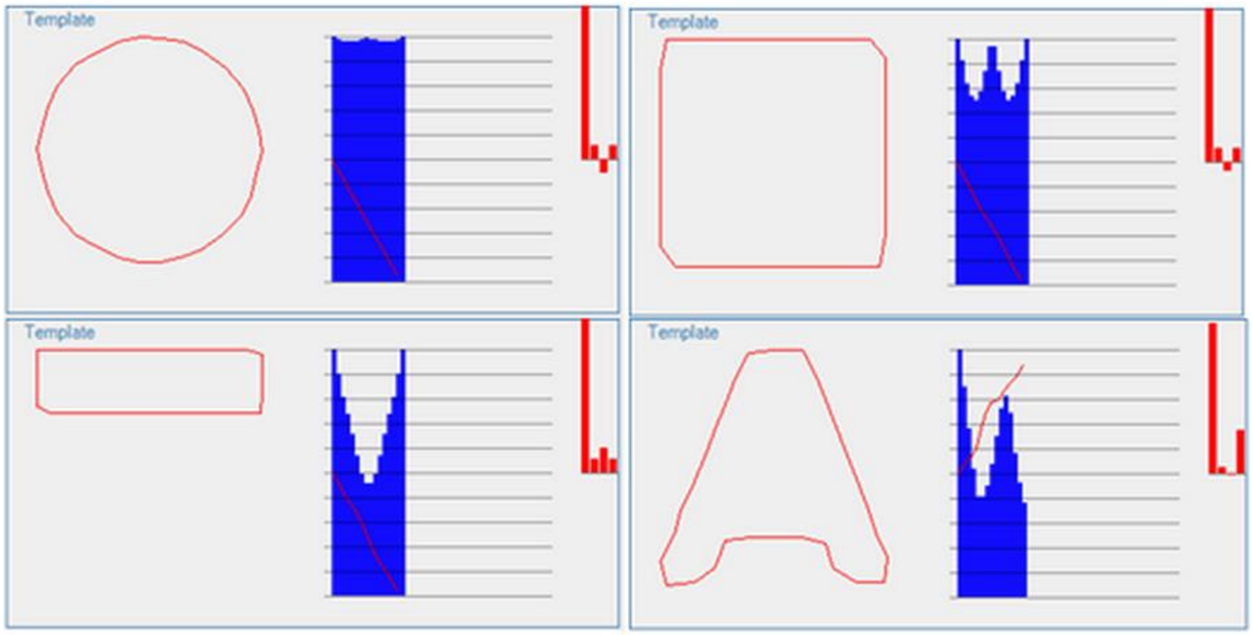


Figure 6Some examples about the ACF for contours [1]

In Figure 3, the norm of ACF is represented in blue color (interval from 0-k/2). The bottom right one is not symmetric, so the ACF is also not symmetric.

- Norm of the ACF follows from point of 1st and from properties.

C. Practical Application of the Contour Analysis

1. General algorithm of recognition

The algorithm is based on CA

- Searching and comparing contours with templates.
- Binarize the images.
- Coercion of contours to uniform length, smoothing.

2. Estimation of performance of algorithm in a CA

We only deal with points of contours, and we estimate their general amount of the image. To do this, we take the image with a size n*n pixels. Then breed its uniform gird with a step s. The total length of all grid lines is [1]:

$$\square = \frac{2\square^2}{\square} \quad (10)$$

One thing we have to concern is the complexity $\square(\square^2)$. The ICF demands evaluations of order $\square(\square^2)$, and to compare contours, we need to find the maximum value of ICF.

The total time of finding a template for a contour in a set of templates can be calculated as:

$$\square(\square^2\square) \quad (11)$$

Where:

- k: length of a contour
- t: number of template contour

So, the complexity of identification of all contours:

$$\square(\square^2\square^2\square) \quad (12)$$

3. Contour Descriptor

For finding templates, it is important to present the certain descriptor describing the shape of the contour. And the ACF can be selected as the descriptor of shape of a contour.

4. Equalization of Contours

For finding and comparing of contours, all of them should be led to uniform length .The process is called equalization.

Firstly, the length of a VC will be fixed and designated as k.

Then we create vector contour N with length k for each initial contour A.

The problem is how large the value k should be?

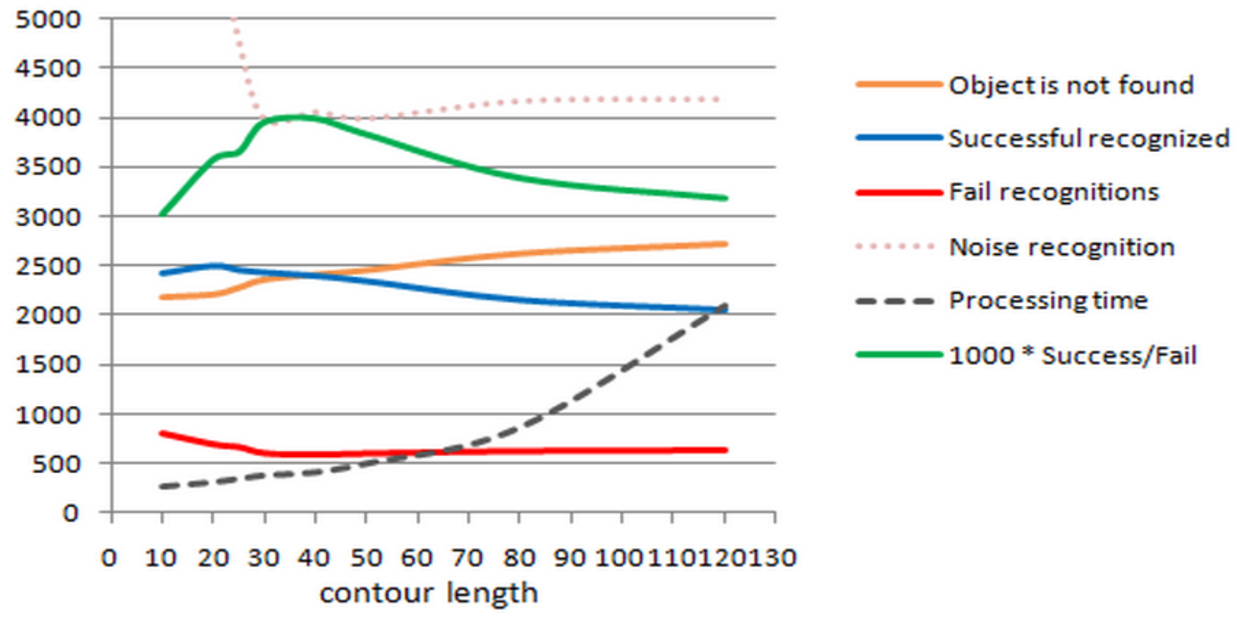


Figure 7Contour length for Latin letter recognition system [1]

Figure 4 is the diagram, which was built for Latin letter recognition system. From this picture, value k= 30 is optimal for the given recognition system.

5. Contour Analysis Limitation

- If the object's boundary is not clear for some reasons like it has the same color with the background, or it has noises, the contour cannot be recognized, or it is recognized incorrectly.
- If there are some intersections between the contour and other objects, the contour also cannot recognized.

D. References

1. http://www.codeproject.com/Articles/196168/Contour-Analysis-for-Image-Recognition-in-C?fid=1626785&fr=26#xx0xx
2. http://www.aforgenet.com/aforge/framework/docs/html/7ed6c207-a962-57c8-d430-8b5a782d8401.htm

IV. Filters

Written by Duy Thuoc Tran - 967897

A. Grayscale filter [1]:

This filter is used for image grayscaling. This filter takes arguments of RGB coefficients with a specific value to converse color image to grayscale image. Other filters have to inherit from this class.

The filter takes 24,32, 48 and 64 bpp color images and gives 8 (if source is 24 or 32 bpp image) or 16(if source is 48 or 64 bpp image) bpp grayscale image.

After applying the filter, all the RGB coefficients are changed to eliminate all the other color but black and white.

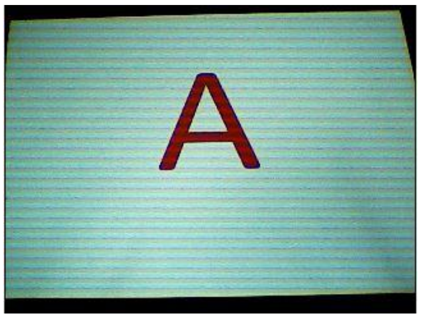


Figure 8: The input image

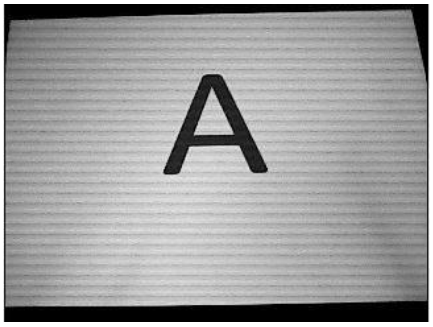


Figure 9: The output image after using grayscale filter

B. Closing filter [2]:

Closing morphology operator equals to dilatation followed by erosion.

Applied to binary image, the filter may be used connect or fill objects. Since dilatation is used first, it may connect/fill object areas. Then erosion restores objects. But since dilatation may connect something before, erosion may not remove after that because of the formed connection. After I apply closing filter, a lot of the background noises are eliminated.

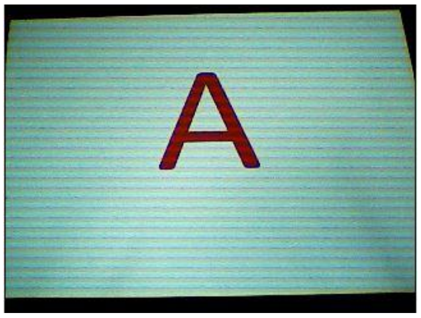


Figure 8: The input image

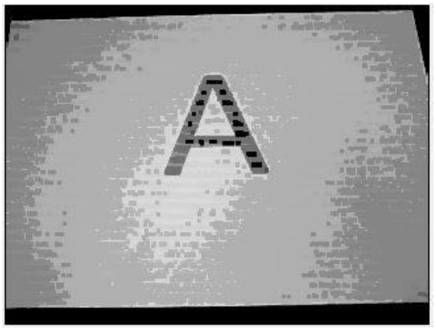


Figure 11 : The output image after using Closing Morphology

C. Dilation filter [3]:

The filter assigns maximum value of surrounding pixels to each pixel of the result image. Surrounding pixels, which should be processed, are specified by structuring element: 1 - to process the neighbor, -1 - to skip it.

The filter especially useful for binary image processing, where it allows to grow separate objects or join objects.

For processing image with 3x3 structuring element, there are different optimizations available, like Dilatation3x3 and BinaryDilatation3x3.

The filter accepts 8 and 16 bpp grayscale images and 24 and 48 bpp color images for processing. After applying this filter, the background noises are reduced to the maximum.

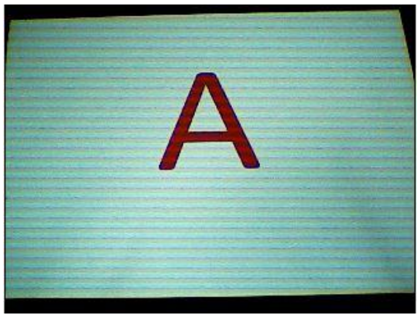


Figure 9: The input image

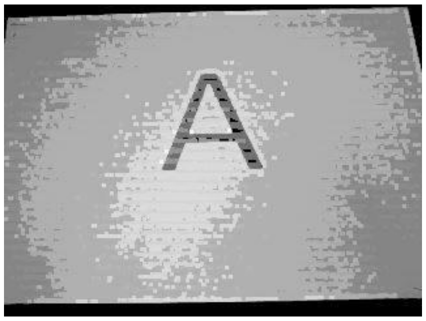


Figure 13: The output image by using Dilation

D. Sobel Edge Detector:

The Sobel edge detector is an example of the gradient method. It is the set of smallest difference filter with odd number of coefficients that smoothies the image perpendicular to the differentiation (4). The Sobel edge detector uses a pair of 3x3 convolution masks which are given as (5):

For x-Direction:

$$\Delta x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad (1)$$

For y-Direction:

$$\Delta y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{2}$$

The convolution mask is usually much smaller than the actual image [(6)]. Then, we apply the pair of the masks **Gx**, **Gy** (3) to the input image **A** to calculate the gradient of the image intensity at a time**.**

$$\mathbf{G}_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * \mathbf{A} \qquad \mathbf{G}_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * \mathbf{A} \tag{3}$$

At each point in the image, the resulting absolute magnitude of the gradient is the output edge (3).

$$|G| = \sqrt{Gx^2 + Gy^2} \tag{4}$$

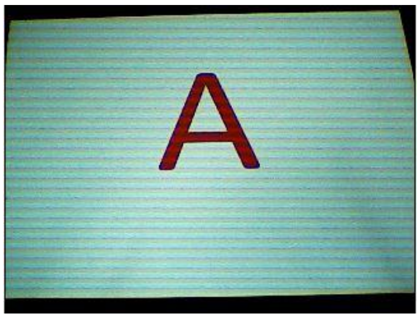


Figure 10: the input image

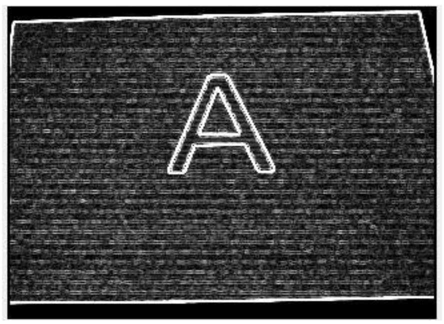


Figure 11: the output image after using Sobel edge detector

The advantages of the Sobel edge detector [(7)]:

- It is a simple approximation to the gradient magnitude.
- It is simple to detecting edges and their orientations.

The disadvantages of the Sobel edge detector [7]:

- It is sensitivity to the noise.
- It is inaccuracy, as the gradient magnitude of the edges decreases.

E. Canny edge detector

The Canny operator works in a multi-stage process:

1. The input image is smoothed by Gaussian convolution in order to remove the noise. Here is an example of a 5x5 Gaussian filter, with $\sigma = 1.3$ [8]

$$\mathbf{B} = \frac{1}{159}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix} * \mathbf{A} \tag{5}$$

   The larger the size of the Gaussian kernel is, the lower the detector's sensitivity to noise.

2. Then, we find the intensity gradient of the image to determine the edge gradient (equation 4) and the direction (equation below) [8].

$$\mathbf{\Theta} = \operatorname{atan2}(\mathbf{G}_y, \mathbf{G}_x) \tag{6}$$

   The edge direction angle is rounded to one of four angles representing vertical, horizontal and the two diagonals. An edge direction falling in each color region will be set to a specific angle values [8].

3. Apply non-maximum suppression to give a thin line in the output. The non-maximum suppression scans the image along the image gradient direction, and if pixels are not part of the local maximal, they are set to zero. The algorithm for each pixel in the gradient image is [8]:
   - Compare the edge strength of the current pixel with the edge strength of the pixel in the positive and negative gradient directions.
   - If the edge strength of the current pixel is the largest compared to the others in the mask with the same direction, the value will be preserved. Otherwise, the value will be 0

4. Apply double threshold to determine potential edges. Two threshold values, which are the high threshold value and the low threshold value, are set to find the different types of edge pixels. If the edge's gradient value is higher than the high threshold value, it is

called the strong edge pixel. If the edge's gradient value is higher than the low threshold value and lower than the high threshold value, it is called the weak edge pixel. If the pixel value is lower than the low threshold value, it is set to zero [8].

5. To achieve an accurate result, the weak edges cause from noise or color variations should be removed by looking at a weak edge pixel and its 8-connected neighborhood pixels. As long as there is one strong edge pixel is involved in the Binary Large Object- analysis (BLOB), that weak edge point can be defined as one that should be preserved [8]. .

The advantages of the Canny edge detector [7]:

- The finding errors are effective by using the probability.
- Improving the signal with respect to the noise ratio.
- Better direction of edges especially in noise state.

The disadvantages of the Canny edge detector [7]:

- The computation of Gradient calculation for generating the angle of suppression.
- Time consumption because of complex computation.

F. References

1. http://www.aforgenet.com/aforge/framework/docs/html/d7196dc6-8176-4344-a505-e7ade35c1741.htm
2. http://www.aforgenet.com/aforge/framework/docs/html/7ed6c207-a962-57c8-d430-8b5a782d8401.htm
3. http://www.aforgenet.com/aforge/framework/docs/html/88f713d4-a469-30d2-dc57-5ceb33210723.htm
4. Image Processing Lecture - University of Applied Sciences Frankfurt am Main, Department Computer Science & Engineering, Masters Program in Information Technology, Summer semester- Dr.Ing. Christian Goerick
5. A Descriptive Algorithm for Sobel Image Edge Detection – O.R. Vincent and O. Folorunso http://proceedings.informingscience.org/InSITE2009/InSITE09p097-107Vincent613.pdf
6. Sobel operator – Wikipedia http://en.wikipedia.org/wiki/Sobel_operator
7. A Comparison of various Edge Detection Techniques used in Image Processing – G.T.Shrivakshan http://ijcsi.org/papers/IJCSI-9-5-1-269-276.pdf
8. Canny edge detector – Wikipedia http://en.wikipedia.org/wiki/Canny_edge_detector

G. Flowchart diagram of the application

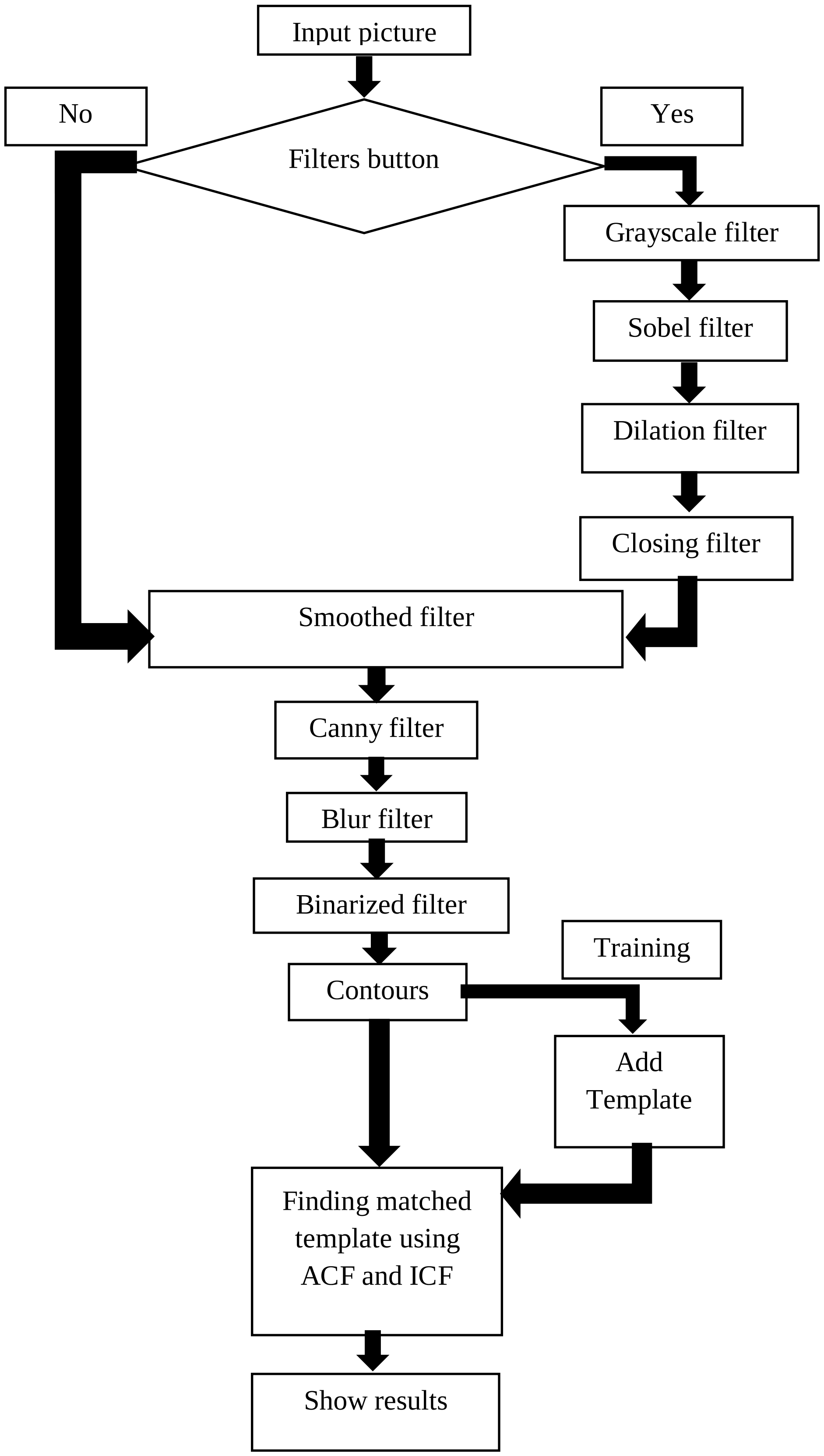
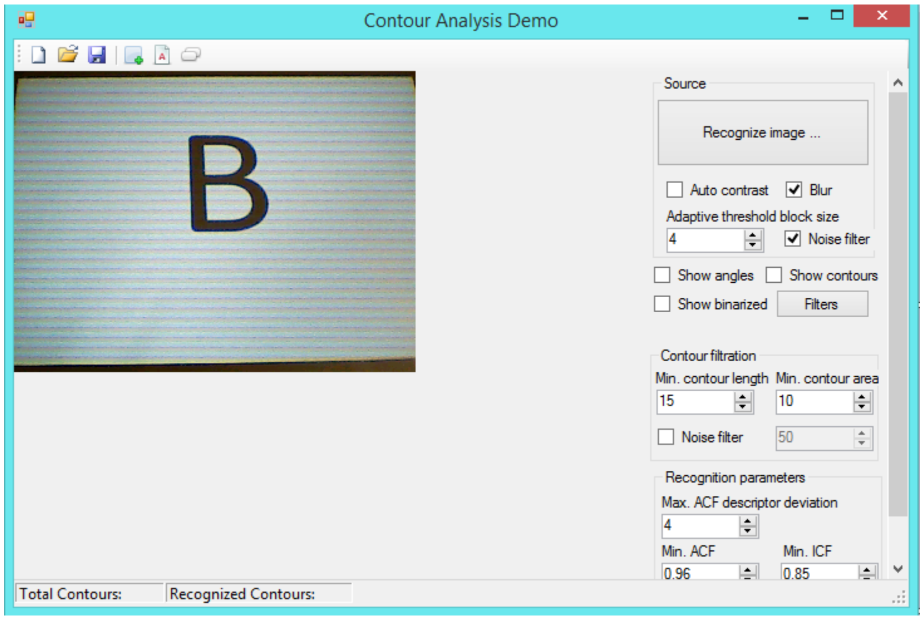


Figure 12 Application flowchart

V. Manual of the program

(Written by Nam Hung Le – 967826)

A. Main interface



Auto contrast [ ] Blur [✓]
Adaptive threshold block size: 4
Noise filter [✓]
Show angles [ ] Show contours [ ]
Show binarized [ ]

Contour filtration
Min. contour length: 15 Min. contour area: 10
Noise filter [ ] 50

Recognition parameters
Max. ACF descriptor deviation: 4
Min. ACF: 0.96 Min. ICF: 0.85

Figure 13 Interface of the program

1. To load an image, we click on the Recognize image or the Filters buttons. The Recognize image button is for general image processing and the Filters button is for pre-processing filters used for better image recognition.

2. To apply further filters in the program we can check on Blur, Auto contrast and Noise filter. Usually we checked on the Blur and Noise filter.

3. The Adaptive threshold block size is used for changing the feature of the binary image (black and white picture). Usually we keep the value at 4.

4. The Show angles, Show contours, and Show binarized check box are for showing the corresponding picture after applied these filters.

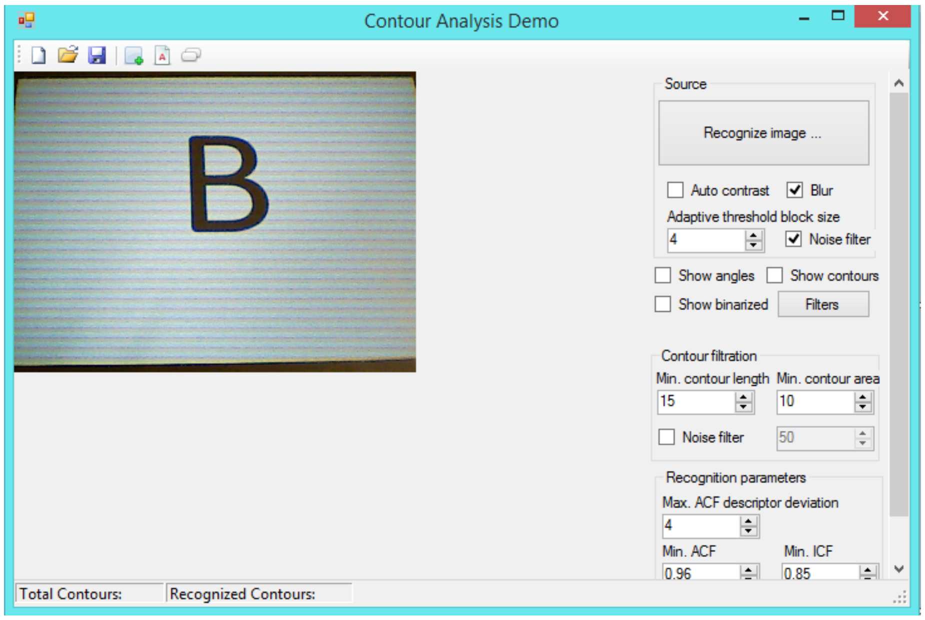
5. In the Contour filtration section, we can change the parameters for finding the contours in our picture. The Min.contour length and Min. countour area values at the beginning are set at 15 and 10. We also have the Noise filter check box and when checked we can also change the value of the filter. Usually when the program running we only need to change the Min. contour length value to 100 to filter out the noise and only get the character.

6. In the Recognition parameters, we can change the Max. ACF descriptor deviation, Min.ACF and Min.ICF values. Usually when the program running we change the Max.ACF descriptor deviation value to 10.

B. The training phase

In our program, we integrate the function to train it from the real picture for later recognition. The training procedure is in following steps:

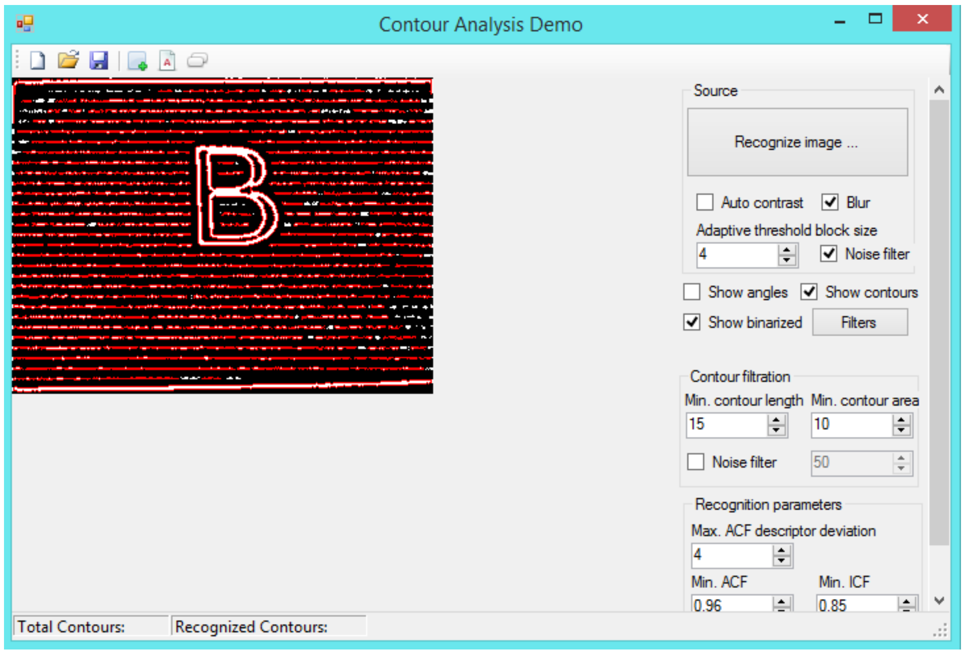
1. Load the image by clicking the Recognize image button or the Filters button. Usually when we start the program of the new input picture we clicked on the Recognize image button. All the parameters are set at default.



Auto contrast: ☐
Blur: ☑
Adaptive threshold block size: 4
Noise filter: ☑
Show angles: ☐
Show contours: ☐
Show binarized: ☐
Min. contour length: 15
Min. contour area: 10
Noise filter: ☐ 50
Max. ACF descriptor deviation: 4
Min. ACF: 0.96
Min. ICF: 0.85
Total Contours:
Recognized Contours:

Figure 18 The loaded picture

2. Now we can checked the Show binarized and Show contours to see the actual contour of the image after the process.


Contour Analysis Demo

Source

Recognize image ...

☐ Auto contrast ☑ Blur

Adaptive threshold block size: 4 ☑ Noise filter

☐ Show angles ☑ Show contours

☑ Show binarized Filters

Contour filtration

Min. contour length: 15 Min. contour area: 10

☐ Noise filter 50

Recognition parameters

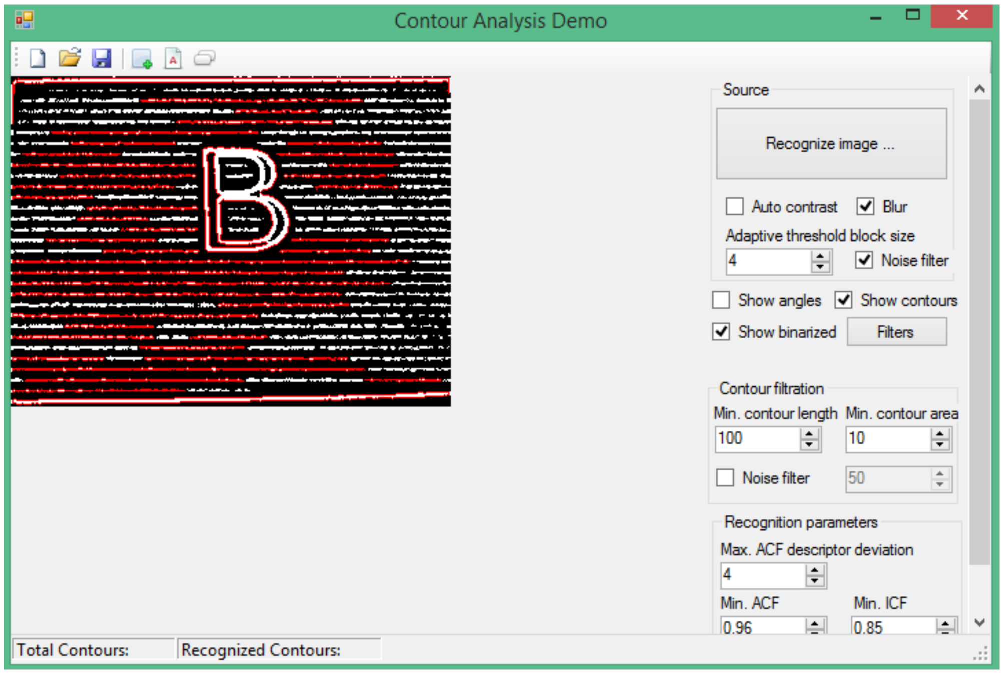
Max. ACF descriptor deviation: 4

Min. ACF: 0.96 Min. ICF: 0.85

Total Contours: Recognized Contours:

Figure 19 The binarized and finding contours picture

3. As you can see we get a lot of noise after finding the contours. To reduce the unwanted noise contours we raise the Min. contour length to the higher value, in our case, to 100.


Contour Analysis Demo

Source

Recognize image ...

☐ Auto contrast ☑ Blur

Adaptive threshold block size: 4 ☑ Noise filter

☐ Show angles ☑ Show contours

☑ Show binarized Filters

Contour filtration

Min. contour length: 100 Min. contour area: 10

☐ Noise filter 50

Recognition parameters

Max. ACF descriptor deviation: 4

Min. ACF: 0.96 Min. ICF: 0.85

Total Contours: Recognized Contours:

Figure 20 The contours with min length over 100

4. Now we can create template for our character. To do that we clicked on the Create template tab on the upper left of the program and chose the suitable contours then set a name for its template of our character. Click on Add selected contour as Template to confirm.

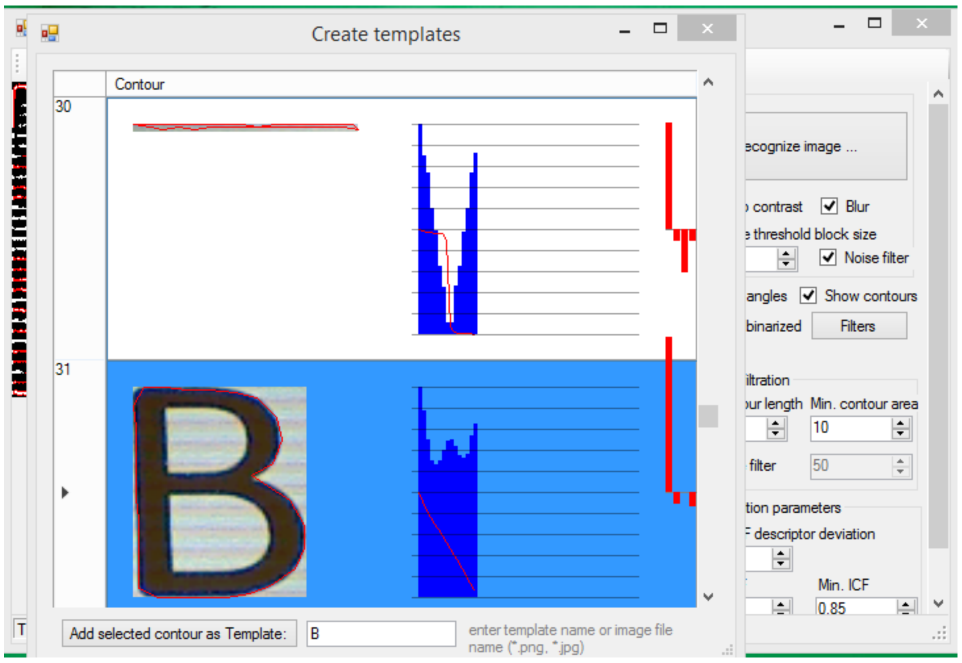


Figure 14Find the contour of the character and set the name

5. To save the template for later use, clicked on the Save templates tab at the upper left of the program and save to the chosen .bin file. In our program it is traning.bin.

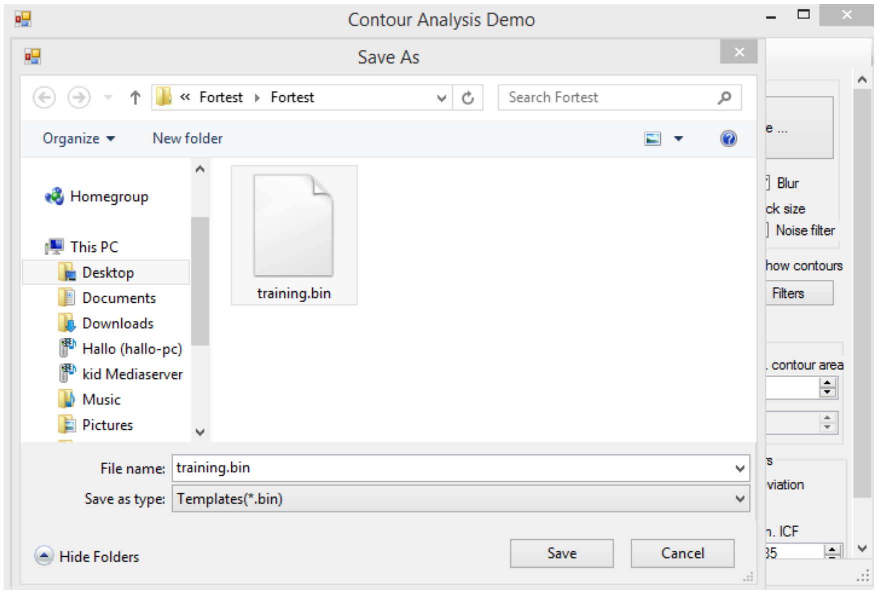
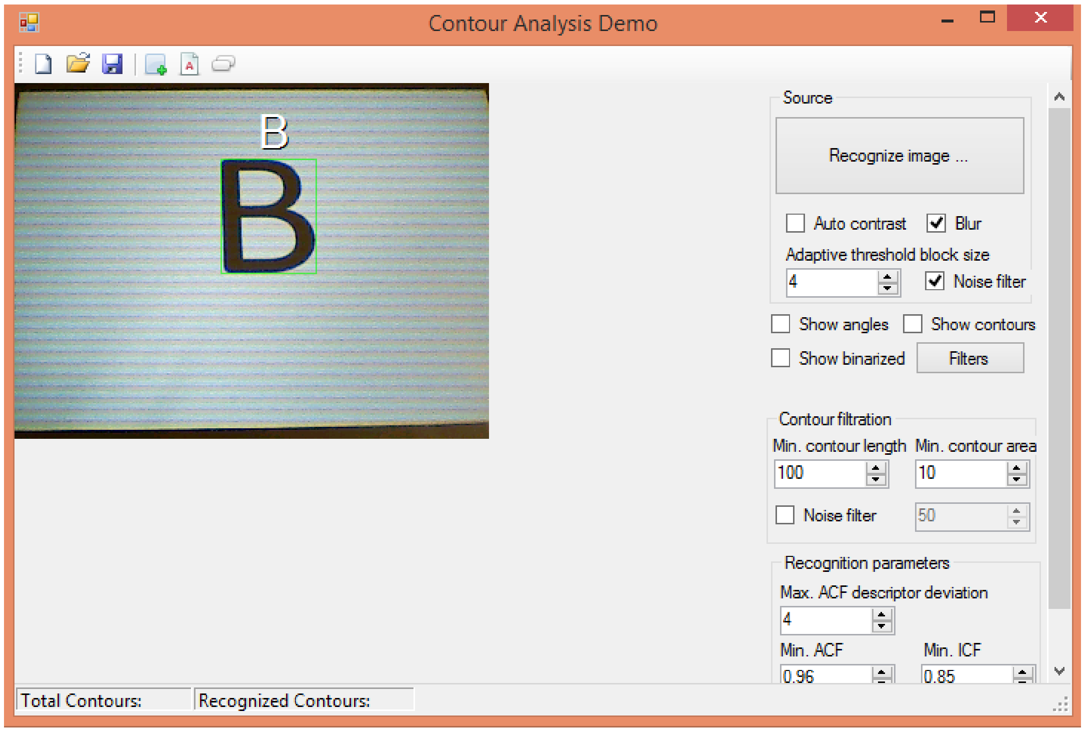


Figure 22: The saving of the trained templates

Now we can use the template for further character recognition. Below is the result



Auto contrast [ ] Blur [✓]
Adaptive threshold block size: 4
Noise filter [✓]
Show angles [ ] Show contours [ ]
Show binarized [ ]
Contour filtration
Min. contour length: 100
Min. contour area: 10
Noise filter [ ] 50
Recognition parameters
Max. ACF descriptor deviation: 4
Min. ACF: 0.96
Min. ICF: 0.85
Total Contours:
Recognized Contours:

Figure 23 The result is showed in our program

C. Ways to improve our result.

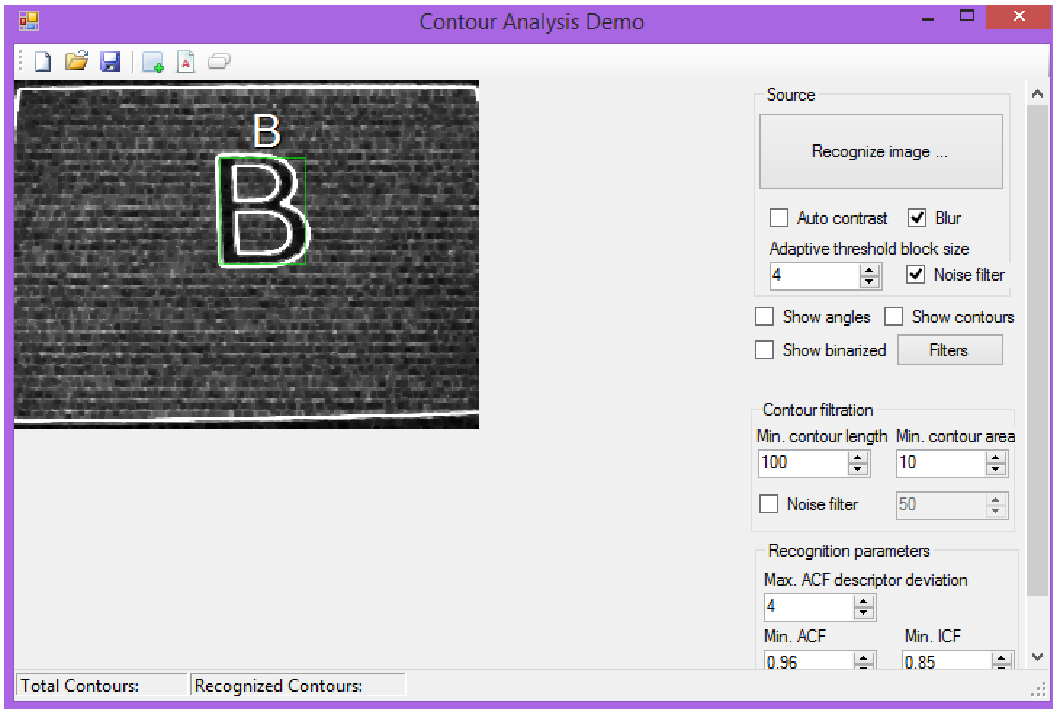
In some case the picture contain too much noise and it hard for our standard program to detect the character contours. In those cases we need to change some of the parameters values or applying the pre-processing filters to our picture by clicking the Filters button. Below are some common ways that we use to get the better result in finding contours of the characters for both training and detection.

1. Change the Max.ACF descriptor deviation: Usually for better recognition at the star of our program we set the value to 10, but in some case the characters contours can only be found at value around 4-7. After you find these contours for templates training, the later use of the program the value can set to 10 and it will still find the contours for detection.

2. Change the Min. contour length and Min. contour area: At the star of our program we set the value of Min. contour length to 100 and the Min. contour area to 10 because usually

the size of the characters in our picture is big with the aspects rations quite large for our picture size, but with the smaller character we must lower these two value. Be cautios as with lower value, the noise will affect the results.

3. Change the input from Recognize image to Filters: Instead of using direct picture capture from the camera, we apply the pre-process filters before our image to reduce the noise and better contours finding. Because with some characters the applying of pre-process filters can cause the wrong detection of the characters so it is not the good ideal to always use these filters.(Notes: The pre-process filters apply the grayscale filter for and in our image process also have another grayscale filter but it does not affect the overall result)



Auto contrast [ ] Blur [✓]
Adaptive threshold block size: 4 Noise filter [✓]
Show angles [ ] Show contours [ ]
Show binarized [ ]
Min. contour length: 100 Min. contour area: 10
Noise filter [ ] 50
Max. ACF descriptor deviation: 4
Min. ACF: 0.96 Min. ICF: 0.85
Total Contours: Recognized Contours:

Figure 23 15Using the pre-process filters

VI. Conclusion

Written by Ngoc An Ha -967813

The task of this project is to create an Optical Character Recognition program. Using the FEZ Spider camera to take the picture and process the test image on the computer. The application provided acceptable results with different characters along with different backgrounds. Although there are some cases that it was not possible to recognize, overall speaking, the project is quite successful.