Institute of Technology, Carlow

B.Sc. Hons. in Software Engineering

CW228

Project Report

Project Title: **Number Plate Recognition**

Name: Dongfan Kuang

Login ID: C00131031

Supervisor: Nigel Whyte

Date: 16 April 2010

Table of Contents

1. Problems Encountered and How They Were Resolved

1.1 Programming Language

I haven't use JAVA before, it is totally new for me, so the first problem I met was develop software by using an unfamiliar language. To solve this problem, I first learned the basic JAVA programming knowledge. In order to reduce the difficulty during codding, I chose NetBeans as my IDE, because it is one of the most easy to use IDE which suitable for beginners and it is also have very powerful GUI develop tools.

1.2 Algorithm

Number plate recognition is an image-processing based technology, which involves a lot of complex algorithms. I met two big problems in the algorithm.

1.2.1 Character Segmentation

After found the number plate area and isolated it from source image, I need to thresholding this area to segment the characters. At first I tried fixed-threshold algorithm, it use a fixed value as threshold to convert image into black and white format. It didn't work very well, because the contrast of every image is different, so a fixed threshold value can't adapt to all images. Figure 1 is a screenshot when using fixed-threshold algorithm.

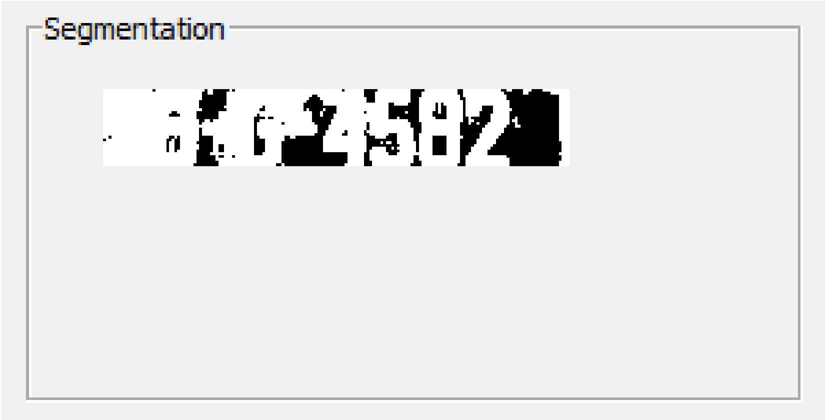


Figure 1 fixed-threshold algorithm

To solve this problem, I changed fixed-threshold algorithm into adaptive thresholding algorithm. The adaptive thresholding algorithm calculates a threshold value based on the pixels information of the target image, so it can work well with most of the image regardless their contrast. Figure 2 is a screenshot after using adaptive thresholding algorithm onto the same image in Figure 1.

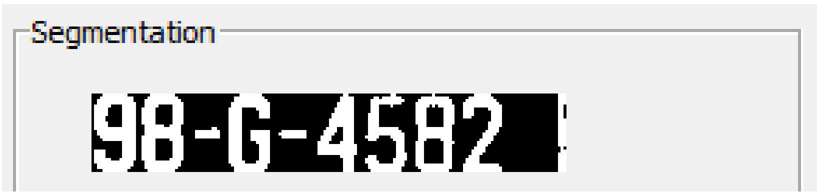


Figure 2 adaptive thresholding

1.2.2 Character Recognition

I mentioned two character recognition methods in my research manual: projection method and grid feature method. Projection method recognizes characters by their vertical and horizontal projection information. Grid feature method divides the target image into small grids, analyses the pixels information in each grid and then recognizes characters by this information. But in practice, both of these two algorithms can't work very well, because the result can be easily impact by noises in the target image and the adaptability of the two algorithms is also not strong. To solve the problem I first analysis the image after thresholding to find out its feature point and apply thinning algorithm to the image to find the end point information, and then recognize the character by using both the feature point and end point information. This algorithm has a relatively strong applicability and higher recognition rate. For example as it shown in Figure 3, when recognize a number, if it doesn't have end point, it only could be '0' or '8', but the center point of '0' is empty and '8' is not, so it is very easy to distinguish '0' and '8' with high recognition rate.

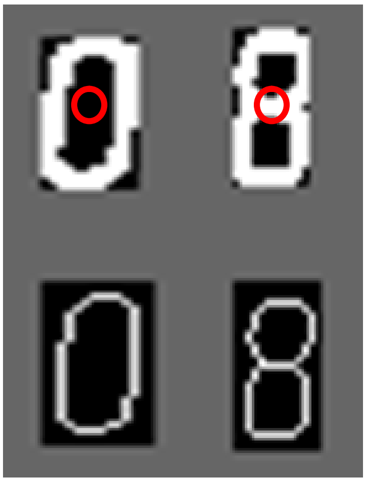


Figure 3

2. What I Achieved

I have achieved most of the functionalities that mentioned in specification and design manual:

Image Import: import a source image displays it on original image panel, support JPG and BMP image type (Figure 4).

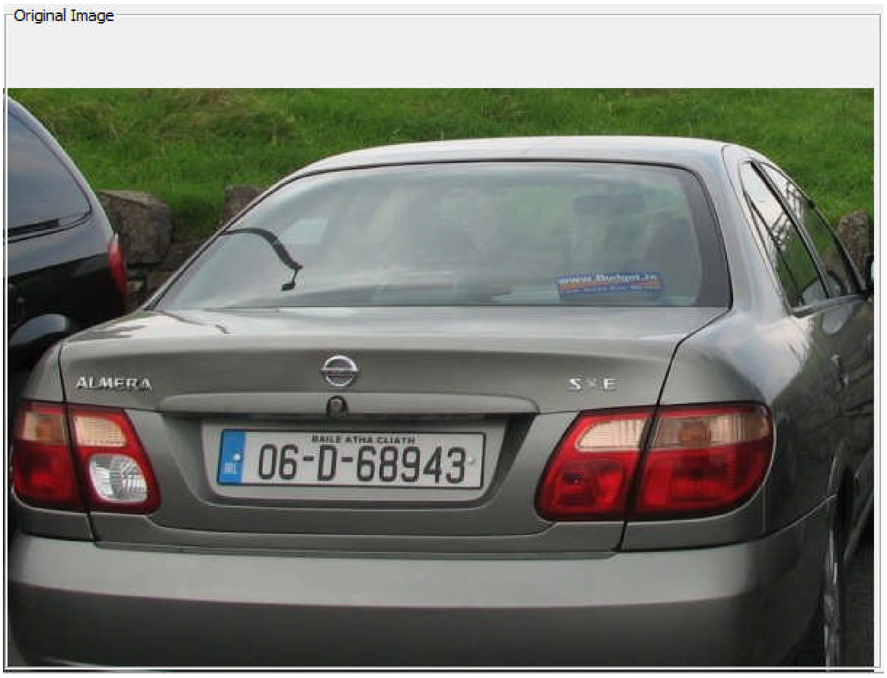


Figure 4 image import

Convert to Grayscale: convert the imported source image into grayscale, display the result on original image panel (Figure 5).

Figure 5 convert to grayscale

Canny Edge Detection: apply Canny edge detection algorithm to the grayscale image, display the result to original image panel (Figure 6).

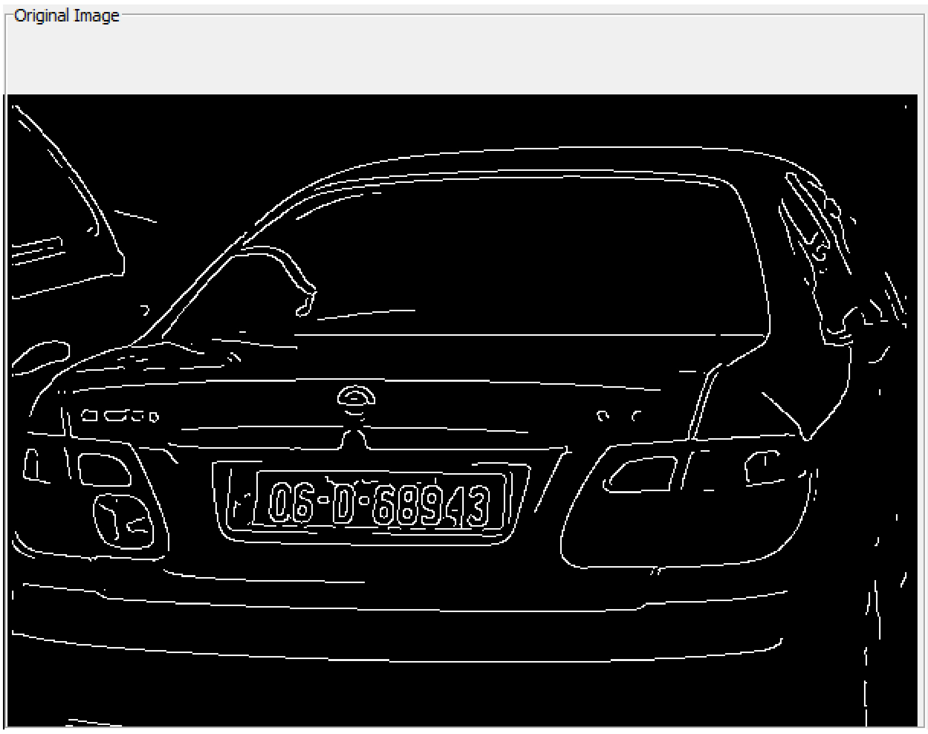


Figure 6 Canny edge detection

Number Plate Isolation: find the number plate area from the edge detected image and display the result to localization image panel (Figure 7).

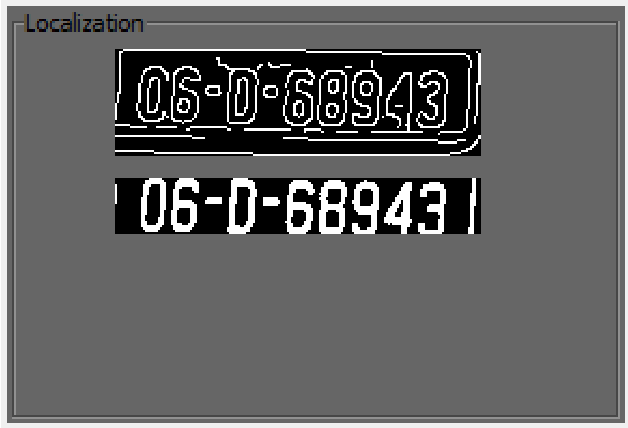


Figure 7 number plate isolation

Character Segmentation: thresholding the isolated number plate area and segment the characters, display the result to segmentation panel (Figure 8).

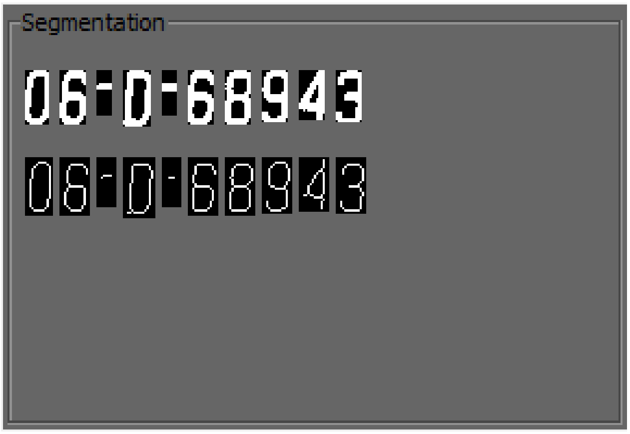


Figure 8 character segmentation

Character Recognition: apply character recognition algorithm to the segmented character, and output the result (Figure 9).

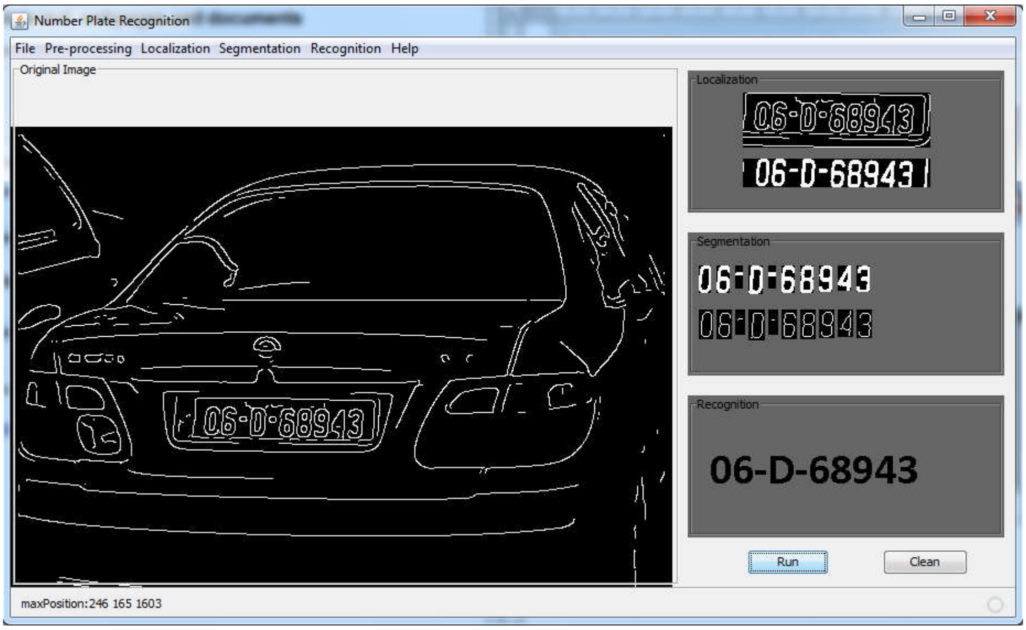


Figure 9 character recognition

3. What I Did Not Achieved

Function I did not achieved compared to my original function specification document is Gaussian Smoothing, because after further experiment I found it does not have much effect. And also the plate orientation function is not working very well, because tilt correction algorithm will reduce the quality of the image and will reduce the correctness of character recognition.

4. What I Learned

Java is a very good object-oriented programming language. Its class libraries are very standard and extremely strong. After finished this program, I am now familiar with JAVA now, and be able to use JAVA to develop other kinds of program especially which relate to image processing technology.

The another skill I learned from this project is about image processing technology, like how to open and read information from an image file, how to deal with pixels, and a lot of image processing algorithm including edge detect algorithm, binarization algorithm, tilt correction algorithm, as well as character recognition algorithm.

5. What I Would Do Differently If Starting Again

If I start this project again, I will choose using C# as my programming language, because my last year project is E-mail Merge which developed by using C#. E-mail Merge allows user to design and send personalized e-mail to a group of recipients by using a single template and a structured data source. If I developed Number Plate Recognition with C#, I can actually combine Number Plate Recognition and E-mail Merge together. The new software will have the function to record the recognized number plate, and compare it with the data source to find out the information of the car owner, and then send an e-mail to the owner with certain content by using the function in E-mail Merge.

6. Updates on Original Design Document

Deleted the Gaussian Smoothing function. Now in the pre-processing step, the image only needs to convert to grascale (Figure 10).

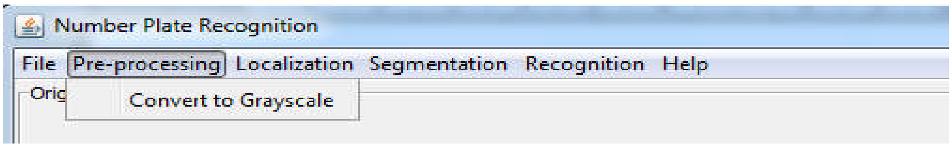


Figure 10 new Pre-processing menu

7. Updates on Original Research Document

7.1 Thinning Algorithm Based on Index Table

Thinning is used to remove selected foreground pixels from binary images, and convert them into 1-pixel wide lines. To determine whether a point should be removed, we need to analysis its 8 neighbor points.

Rules of delete a foreground pixel:

1) Internal point can not be deleted.
2) Isolated point can not be deleted.
3) Endpoint of line can not be deleted.
4) If P is an edge point, and after remove P won't produce new lines, P can be deleted.

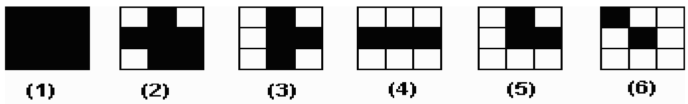


Figure 11 rules of delete a foreground pixel

In Figure 10, by using the rules of delete a foreground pixel, we can know (1), (2) can not be deleted, because of rule 1, (4) can not be deleted because of rule 4, (6) can not be deleted because of rule 3, and (3), (5) can be deleted.

Based on the above rules we can work out an index table which has 256 elements, and their value are either 1 or 0. We check the index table by using current point's 8 neighbor points' information. If the value is 1, that means current point can be deleted, otherwise keep current point (Figure 11).

P1	P2	P3
P4		P5
P6	P7	P8

Index = P1+P2*2+P3*4+P4*8+P5*16+P6*32+P7*64+P8*128

Figure 12 compute index

7.2 Adaptive Thresholding Algorithm

One of the easiest and fastest adaptive thresholding algorithms is called Otsu Thresholding Algorithm which named after its inventor Nobuyuki Otsu.

Otsu's thresholding algorithm involves iterating through all the possible threshold values and calculating a value which can decide the pixels that either falls in foreground or background. The aim is to find the threshold value where the sum of foreground and background spreads is at its minimum.

The basic idea of this algorithm is, for an image, set T as the threshold value, W_0 as the proportion of the number of foreground pixels in the whole image, U_0 as the average gray value of foreground pixels; Set W_1 as the proportion of the number of background pixels in the whole image, U_1 as the average gray value of background pixels. Then the average gray value for the whole image U is:

$$U = W_0 * U_0 + W_1 * U_1$$

$$G = W_0 * (U_0 - U)^2 + W_1 * (U^1 - U)^2$$

Iterate T from 0 to 255 to find the T which let G have the maximum value, and this T is the best threshold value.

8. Testing

Test consists of three parts: grayscale, edge detection and plate Localization part, character segmentation part and character recognition part. Each part contains screenshots of the outputs in different situation and the corresponding explanation.

8.1 Grayscale, Edge Detection and Plate Localization

After user import an image, it will be first converted into grayscale and then apply edge detection function to find the edge information in the grayscale image. Edges will be displayed as white lines in the edge detected image. Plate localization function is to find the area with maximum amount of edges, and this area is most likely to be number plate area.

Table 1

Condition	Shadow	Overexposure
Original Image		
Grayscale		

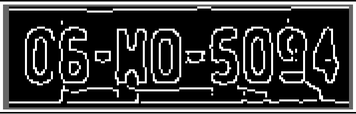
Edge Detection		
Plate Localization		
explanation	Shaw dosen't affect the result of edge detection.	Edge detection algorithm can works well in different lighting conditions, as long as the imported image do not overexposed too much

Table 2

Condition	Complicated Background	Complicated Background
Original Image		
Grayscale		

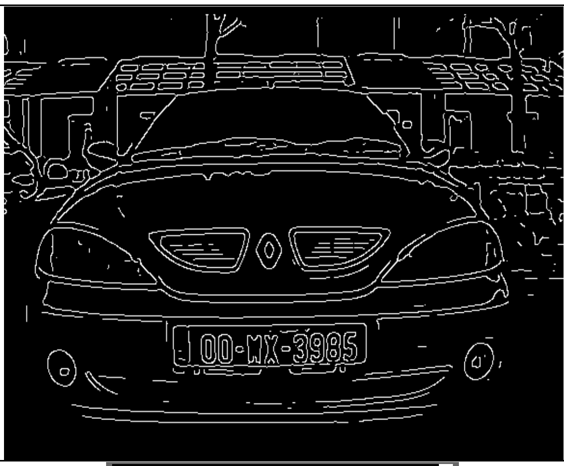
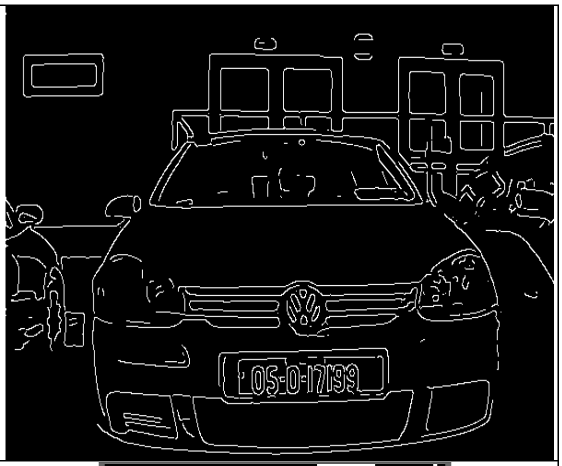
Edge Detection		
Plate Localization		
Explanation	Plate localization function can work well with complex background. The thought of plate localization algorithm is to find the area with maxmun amount of edge information and in most of the situations, number plate area is this kind of area.	

Table 3

Condition	Complicated Background	Complicated Background
Original Image		
Grayscale		

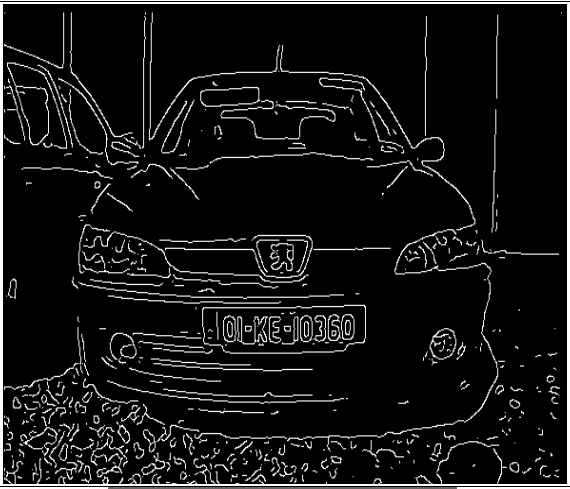
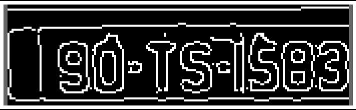
Edge Detection		
Plate Localization		
Explanation	Localization function works well with defferetn kinds of background and foreground.	

Table 4

Condition	Complicated Foreground	Complicated foreground
Original Image		
Grayscale		

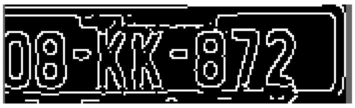
Edge Detection		
Plate Localization		
Explanation	Objects in front of the car also won't affect the result of the plate localization function in most of the situations.	

Table 5

Condition	Reflection	Dirty
Original Image		
Grayscale		

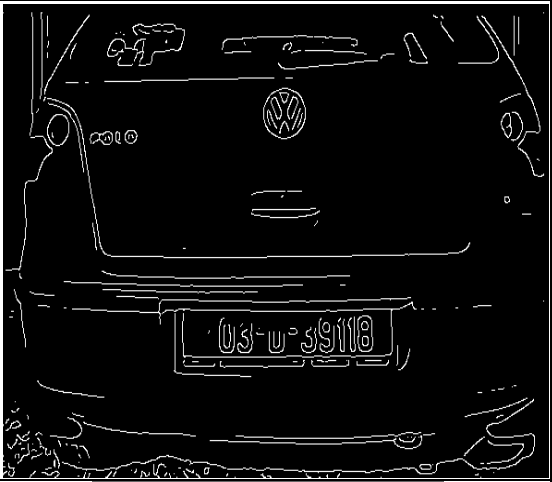
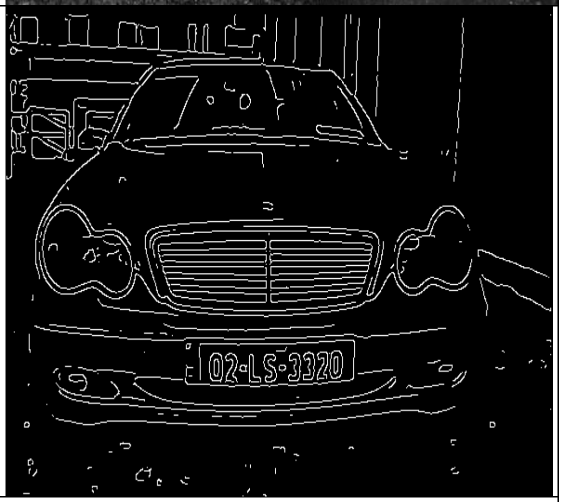
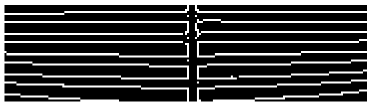
Edge Detection		
Plate Localization		
Explanation	Reflection on the car body won't affect the reslut of localization, and certain degree of tilt is acceptable.	Soil and dust on the plate will reduce the effectiveness of edge detection and may lead the result of localization wrong. In this example, the background is not complex, so plate area can be localized successfully.

Table 6

Condition	Complicated Background	Complicated Background
Original Image		

Grayscale		
Edge Detection		
Plate Localization		
Explanation	Big size of fence shape object will lead the localization result wrong.	

8.2 Character Segmentation

After found the number plate area, it will be resized. After resize, the top and bottom border of the plate will be removed, and then binarization function will be applied onto the corresponding area in the original image. Segmentation function works with the binary plate area, and separates the single character.

Table 7

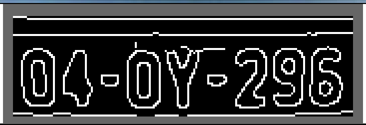
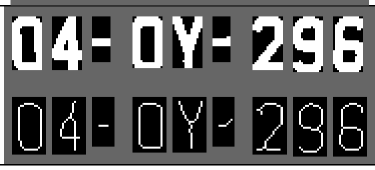
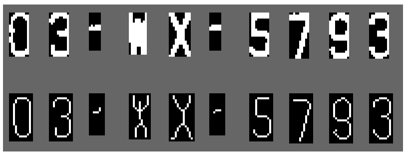
Condition	Insufficient light	Dirty
Original Image		
Isolated Plate		
Binary Plate		
Segmentation result		
Explanation	Image binarization function works well with dark light.	Certain amount of soil and dust won't affect the result of binarization function.

Table 8

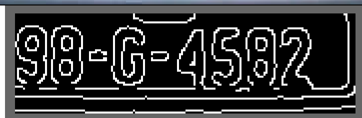
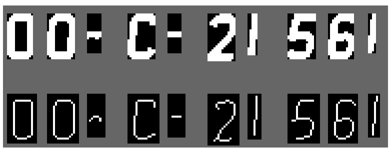
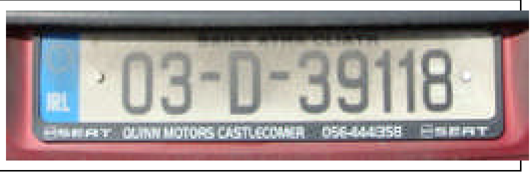
Condition	Compact arrangement of the font	Fuzzy and reflection
Original Image		
Isolated Plate		
Binary Plate		
Segmentation result		
Explanation	Segmentation function works well with certain degree of tilt and with compact arrangement of the font	Certain degree of fuzay and reflections on the character won't affect the result of binarization

Table 9

Condition	Shadow	Dirty
Original Image		

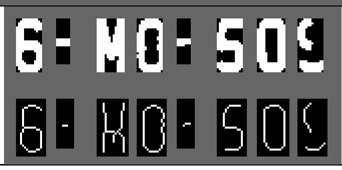
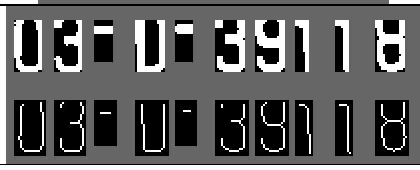
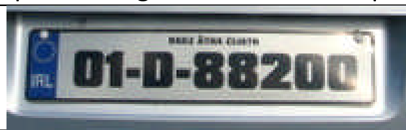
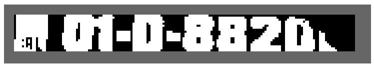
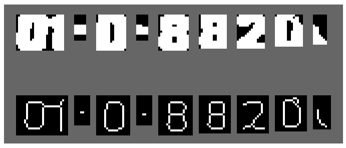
Isolated Plate		
Binary Plate		
Segmentation result		
Explanation	Shadow doesn't affect the result of edge detection, but it affects the binarization function. In this expample, character '0' and '4' is removed during the binarization function.	Soil and dust affects the result of edge detection and then leads the result of binarization function and segmentaion function imcomplete.

Table 10

Condition	Nail between characters	Compact arrangement and Overexposure
Original Image		
Isolated Plate		
Binary Plate		
Segmentation result		
Explanation	In this example the nail bwtween these two '9' connect then together, then lead the result of segmentation wrong.	In this example the space between each character is too less, after binarization the are linked together and then lead the result of segmentation wrong.

8.3 Character Recognition

After segmentation, thinning algorithm is applied onto each single character image, and the recognition result is based on analysis the binary image and thinning image of each single character.

Table 11

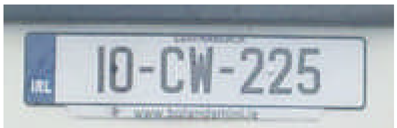
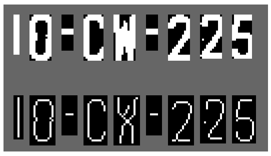
Condition	Tilt	Thin font
Original Image		
Segmentation Result		
Recognition Result	06-D-68943	10-CW-225
Explanation	Recognition function allows characters have a certain degree of tilt.	It work with several types of font

Table 12

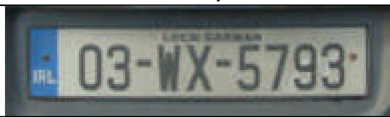
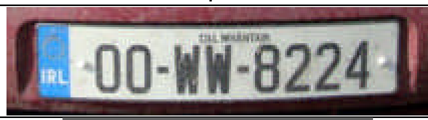
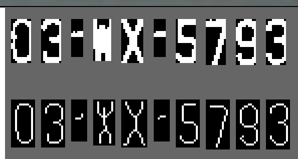
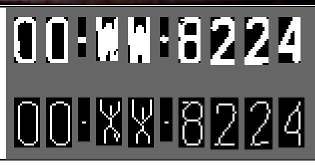
Condition	Dirty	Spot
Original Image		
Segmentation Result		
Recognition Result	03-WX-5793	00-WW-8224
Explanation	Certain amount of soil and dust doesn't affect the result of recognition.	Certain amount of spots doesn't affect the result of recognition.

Table 13

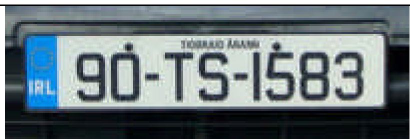
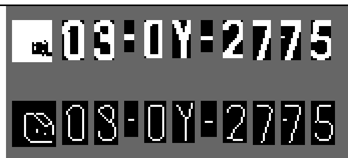
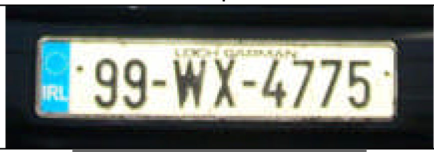
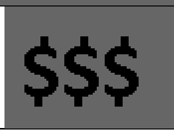
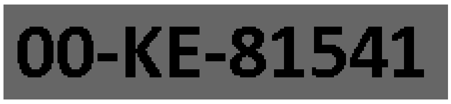
Condition	Different Font	Nails in Characters
Original Image		
Segmentation Result		
Recognition Result	90-TS-1583	03-OY-2775
Explanation	Different kinds of fonts are acceptable.	Nails in characters usually doesn't affect the result of recognition.

Table 14

Condition	Overexposure	Plate Border
Original Image		
Segmentation Result		
Recognition Result		
Explanation	The image in this example is too overexposure, and then the binary image of the single character is too thin, this leads the result of segmentation wrong.	In this example, the left border of the plate is too near the last character and the recogniton function treates it as a character.

8.4 Conclusion

After testing the program, it is works well with pictures take under different light condition and with complex background and foreground. Recognition function can recognize characters in several kinds of font style and with certain amount of soil, dust and nails. Certain degree of tilt is also acceptable. The problem which often caused trouble for localization module is fence shape object in the image, this is because fence shape object often lest large amount of edge after edge detection. Another problem is unusual kinds of font style of the plate characters and too much soil or dust in the number plate may case character recognition error.

Appendix A - Development Diary

Date: Thu 03/12/09

Start to design GUI, menu bar contains five menus: file, pre-processing, plate localization, segmentation and recognition. Four image panels can display result of each step during a recognition process: original image panel, localization image panel, segmentation image panel and recognition image panel. Two button on the bottom right: run and clear.

Date: Tue 15/12/09

Finished GUI design, start to write image import function, decided to support two kinds of image file: JPG and BMP.

Date: Thu 17/12/09

Convert imported image into grayscale form.

read the pixel information from source image:

```
int b = pixels[offset++] & 0xff;

int g = pixels[offset++] & 0xff;

int r = pixels[offset++] & 0xff;
```

change the pixel into grayscale:

```
int gb = ((b*7472)>>16)&0xff;

int gg = ((g*38469)>>16)&0xff;

int gr = ((r*19595)>>16)&0xff;
```

Date: Thu 24/12/09

Decide to skip the Gaussian smoothing step, and directly go to edge detection part, because after tried some similar software and also tried the Gaussian smoothing function in Photoshop, I find Gaussian smoothing doesn't have much effect on the final recognition result.

Date: Mon 04/01/10

Finished the Canny edge detection algorithm, the result is not bad. Plan to do some testing work next work to find the best threshold values for this algorithm.

Date: Mon 11/01/10

lowThreshold = 7 and highThreshold = 8, these two value are the best threshold value. The number plate area is very clear and a lot of backgrounds are removed.

Date: Wed 20/01/10

Finished the number plate isolation and resizing algorithm, now the plate area can be identified from the edge detected image and after resizing the top and bottom borders of the plate are removed, it will increase the success rate of character segmentation.

Date: Mon 25/01/10

Tested the isolation and resizing function, they can works well in most of the situation. Soil, dust and nails on the number plate may affect the isolation and resizing result.

Date: Fri 29/01/10

Convert the isolated number plate area into binary form from the original image, next step is develop a vertical projection algorithm and apply it to the binary plate area to segment characters.

Date: Mon 08/02/10

Finished segmentation function, but in a lot of situations the left or right borders of the plate are segmented as single character, next step is to analysis the result of the segmentation and remove the fake character.

Date: Mon 15/02/10

Characters now can be separated successfully, begin to develop character recognition function

Date: Fri 19/02/10

Decide to use feature point character recognition method instead of projection method and grid feature method I mentioned in the research document. Feature point method is an easy and high efficient way to recognize character. It has more flexibility than the other two methods.

Date: 26/02/10

Finished the number recognition part and next step is to find the position of letters in the plate. Letters always in between two '-', so first I need to identify the '-' in plate.

Date: Mon 15/03/10

Finished the letter recognition part, and analysis the result to see if it is a valid vehicle registration number.

Date: Mon 22/03/10

After testing, found that the binarization function couldn't work very well. So I decided to change the algorithm for binarization, the new algorithm called OTSU thresholding algorithm. It is an adaptive algorithm which can find the best threshold value for an image.

Date: Fri 26/03/10

Plan to add thinning algorithm into character recognition function which can increase the success rate of recognition. The main thought is apply thinning algorithm to every segmented character image to find the end points in these image, and then combine the end point position information and the feature point information together to identify a character.

Date: Mon 05/04/10

Almost finished the project, plan to add the tilt correction function to allow picture be took with a certain degree of tilt and the program can still recognize the character.

Date: Mon 12/04/10

Finished the project, the last task is to do some testing and fix bugs.