Functional Specification

Automatic Number Plate Recognition

BSc (Hons) Software Development Year 4

Student name: Michael Reid
Student ID: C00112726
Project supervisor: Mr. Nigel Whyte

Table of Contents

1 Introduction

This Android app will allow a user to take pictures of car number plates and extract a textual representation which will then be sent to a cloud database. From here, the number plate value will be queried in a database containing all staff and students registered for on campus parking. If the value is found, then an infraction has occurred by the vehicle owner and an action will be taken.

2 Inspiration

One of the many tasks performed by IT Carlow campus staff is ensuring that the visitor's car park is not being used by students and staff. This is due to the congested nature of the on campus parking, and as a result, students and staff my fill up the visitor's carpark in an attempt to find easy parking. This however results in genuine visitors having to search for place to park, and in some cases causes them to park illegally themselves.

To monitor who is parked in the visitor's carpark, a member of staff will manually write down the registration plate numbers of all cars parked there, and then once complete will then enter them one by one into a computer to check if they belong to a student or staff member who are registered to park in the on campus car park. If any are found to be using the visitor's car park illegally, the member of staff then manually writes and sends an email to them asking to move their vehicle.

This task is very time consuming one, and that is highly suited to some degree of automation. With the rise of smartphones and the ever improving processing power of these devices, it is believed an app that can assist in this task is plausible.

3 Overview

The overview of the project is very simple. Create an app that can perform Optical Character Recognition on an image it takes, send that resulting text to an online cloud database to verify and log, and finally if needs be generate and automated email and send it to the vehicle owner. The project will consist of 5 core components:

1. The Android App which must be extremely simple to use.
2. Optical Character Recognition and related image processing
3. A Cloud database containing the registration number of all vehicles registered to park on campus
4. An administrator webpage to allow the creation of user accounts, report generating, and general database management.
5. A cloud web service to allow the app and webpage interact with the database.

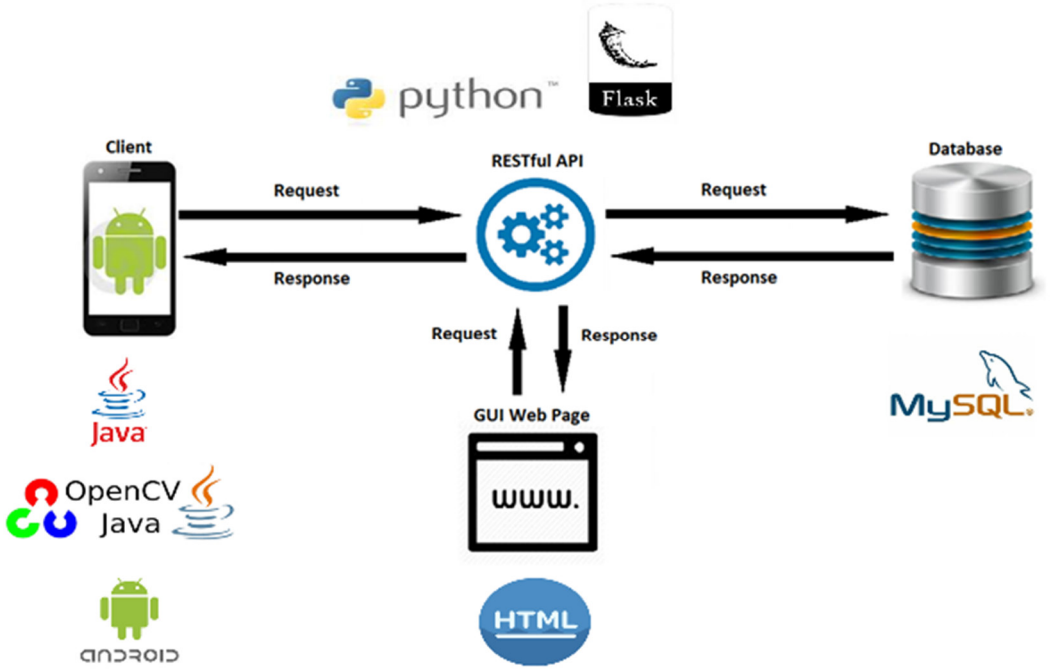


Figure 1 System Overview

4 Functional Requirements

4.1 Hardware Requirements

Android Device

4.2 Software Requirements

4.2.1 Java

The application will be written for Android devices and as such the Java programming language will be used. This app will support devices running at least Android API 19.

4.2.2 OpenCV

OpenCV is an open source image processing suite, originally written in C++ but available on other platforms, including Android. This suite of code will be used to carry out various image processing tasks.

4.2.3 Python

The cloud database will be accessed via a web service written in Python. This service will interact with the database directly and insert/return any information requested by an outside source, which in this case is either the Android device, or the admin web page.

4.2.4 Flask

Flask is a module available for Python which easily allows for the setting up web page routes. This will be used when creating the admin web page.

4.2.4.1 Flask Restful

Flask Restful is a module available for Python which allows a user to set up RESTful web service. This will be used to sending/receiving JSON data between the Android client and the cloud database. It takes care of extracting the JSON and also preventing the web service URLs being accessed via web browsers.

4.2.5 HTML

The admin webpage will be developed with HTML. Since this is not a major component of the overall project, it will be relatively basic in appearance. The webpage will be hosted on PythonAnywhere.

4.2.6 Database Requirements

A MySQL cloud database will be used. The data being stored is very tablature in nature and is the obvious choice. The database will be hosted on PythonAnywhere.

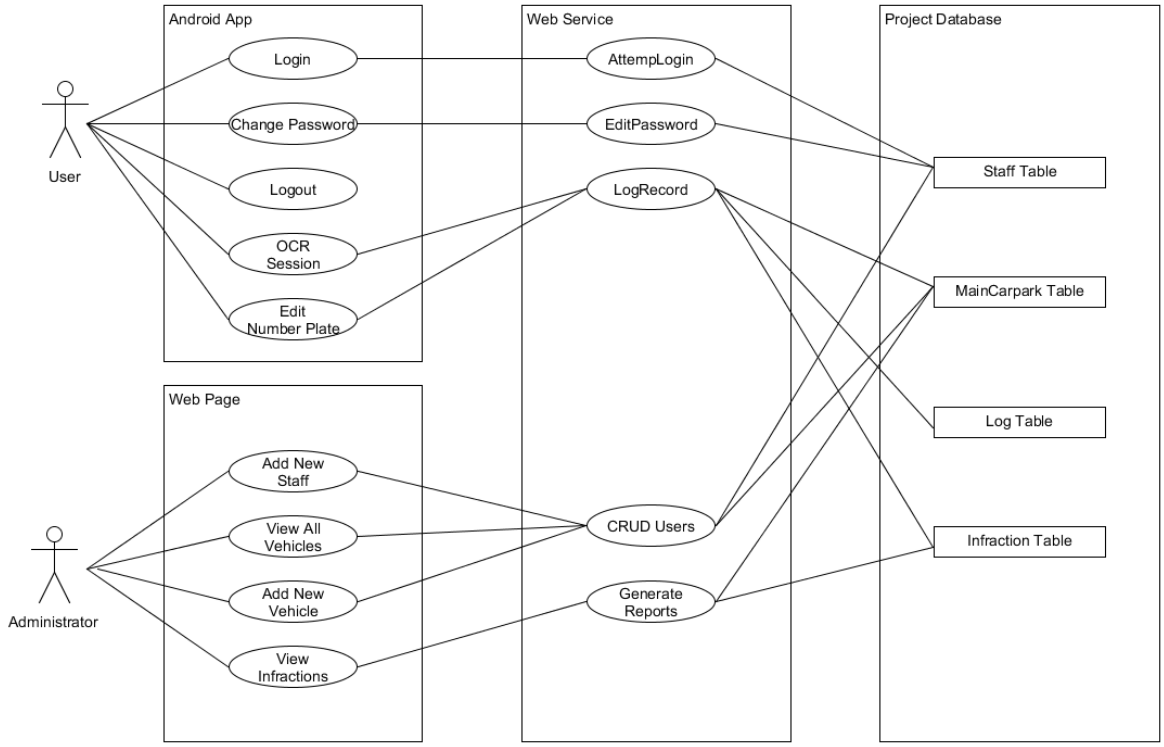
4.2.7 Performance and Reliability Requirements

The application should be able to process the image, and send a result to the database in a reasonable time. Anything over 10 seconds is considered unacceptable, and anything under 5 seconds is considered ideal. A key goal is to perform the recognition and transmit the result faster than the time it would take to upload the image and process it on the cloud.

To make the app viable, high accuracy is required. This project aims for an accuracy of over 90%. Anything below that should ideally be an error in recognising 1 or 2 characters at most. Low accuracy would result in the task being slower than the current manual process employed

5 Use Cases

5.1 Use Case Diagram



5.2 Brief Use Cases

5.2.1 Login

Use Case: Login

Actors: Staff, WebService

Description: This use case occurs when Staff wishes to log in to the Android app. The Staff will supply a username and password and press the log in button. The username and password supplied are validated and checked in the cloud database for a match. The Staff is informed whether or not their login was successful.

Main Success Scenario:

1. Staff enters their username and password
2. Username and password are checked for minimum lengths
3. A check is performed to ensure a network connection is available
4. The username and password are added to a JSONObject
5. The JSONObject is posted to the WebService
6. A new JSONObject is created by the WebService containing the result of the check, i.e. True
7. The JSONObject is posted back to the Staff
8. The JSONObject is checked for the result
9. The Staff is informed of log in success
10. The LogIn screen is closed and the MainMenu is opened

5.2.2 Logout

Use Case: Logout

Actors: Staff

Description: This use case occurs when Staff wishes to log out of the Android app. Staff will simply select the logout option, at which point the app will close.

Main Success Scenario:

1. Staff selects the Logout option
2. The app is closed

5.2.3 Change Password

Use Case: Change Password

Actors: Staff, WebService

Description: This use case will occur when Staff wishes to change their log in password. Staff will supply their current password and the new password. The current and new passwords are checked first to make sure they are not the same. The username for Staff, their current password, and their new password are sent to the cloud database for validation. Staff is informed whether or not the change was successful.

Main Success Scenario:

1. Staff selects the change password option
2. Staff enters their current password
3. Staff enters the new password and confirm it by entering again
4. The current password and new password are checked for minimum lengths
5. The current password and new password are checked to make sure they are not the same
6. The username, current password and new password are added to a JSONObject
7. The JSONObject is posted to the WebService
8. A new JSONObject is created by the WebService containing the result of the change, i.e. True
9. The JSONObject is posted back to the Staff
10. The JSONObject is checked for the result
11. The Staff is informed of password change success
12. The ChangePassword screen is closed and they are returned to the Settings screen

5.2.4 OCR Session

Use Case: OCR Session

Actors: Staff

Description: This use case occurs when the Staff wishes to start taking images of vehicle number plates. An image is taken, processed, and a textural representation of the number plate is extracted. Staff is given the option to edit the result if desired. The value is sent to the cloud database, where it is stored in the log database. Staff is provided with an update that the task was successful.

Main Success Scenario:

1. Staff selects Start Session
2. The camera is accessed and displayed on screen
3. Staff taps the screen to take a picture
4. The image is saved to the device
5. The image is pre-processed
6. The characters are segmented from the image
7. The stored templates are loaded from the device
8. Template matching is performed on the segmented characters and the templates
9. A String of text representing the number plate is created
10. Staff are presented with the result and asked to confirm

5.2.5 Edit Number Plate

Use Case: Edit Number Plate

Actors: Staff, WebService

Description: This use case occurs when Staff wishes to modify the result of an OCR Session. Staff is presented with fields to modify, such as year, county, and registration number. The edited value is sent to the cloud database, where it is stored in the log database. The carpark database is checked to see if it is registered to be there instead, and if it is found, an email is generated and sent to them. The user is provided with an update that the task was successful.

Main Success Scenario:

1. Staff selected the Edit option
2. The image is displayed, as well as the result of OCR Session
3. Staff may edit this to anything they wish
4. Staff selects Continue
5. The username, original OCR result, and the new OCR result are added to a JSONObject
6. The JSONObject is posted to the WebService
7. A new JSONObject is created by the WebService containing the result of the operation, i.e. true
8. The JSONObject is posted back to the Staff
9. The JSONObject is checked for the result
10. Staff is informed of the success

5.2.6 Add New Staff

Use Case: Add New Staff

Actors: Administrator, WebService

Description: This use case occurs when an Administrator wishes to add a new Staff member. They will supply a username and password. The username and password are posted to the WebService.

Main Success Scenario:

1. Administrator selected the Add New Staff option
2. Administrator enters the new Staff details
3. Administrator submits the new details
4. The WebService adds the new details to the database
5. Administrator is informed of the success

5.2.7 View All Vehicles

Use Case: View All Vehicles

Actors: Administrator, WebService

Description: This use case occurs when an Administrator wishes to add a new Staff member. They will supply a username and password. The username and password are posted to the WebService.

Main Success Scenario:

1. Administrator selects View All Vehicles
2. The request is made to the WebService for data on all vehicles
3. The details returned are parsed into a table

5.2.8 Add New Vehicle

Use Case: Add New Vehicle

Actors: Administrator, WebService

Description: This use case occurs when an Administrator wishes to add a new Staff member. They will supply a username and password. The username and password are posted to the WebService.

Main Success Scenario:

1. Administrator selected the Add New Vehicle option
2. Administrator enters the new vehicle details
3. Administrator submits the new details
4. The WebService adds the new details to the database
5. Administrator is informed of the success

5.2.9 View Infractions

Use Case: View Infractions

Actors: Administrator, WebService

Description: This use case occurs when an admin wishes to generate reports containing infractions imposed on all carpark users. It will include a list of all cars who have parked illegally, the number of times this occurred, and additional statistics such as repeat offenders.

Main Success Scenario:

1. Administrator selects the View Infractions option
2. The WebService is asked to return all recorded infractions
3. Administrator is informed of the success

5.2.10 Attempt Login

Use Case: Attempt Login

Actors: WebService, ProjectDatabase

Description: This use case occurs when the WebService is requested to log in a user. The supplied username and password are checked in the database for a match. First, the encrypted password which is linked to the supplied username is retrieved. Then it is decrypted and matched against the supplied password.

Main Success Scenario:

1. The WebService parses a supplied username and password from JSON
2. The database is queried for a password which is linked to the supplied username
3. If a result is found, the stored password is decrypted and matched against the supplied password
4. A JSONObject is created which contains the username and the success value, i.e. True
5. The JSONObject is returned to the client

5.2.11 Edit Password

Use Case: Edit Password

Actors: WebService, ProjectDatabase

Description: This use case occurs when the WebService is requested to update a Staff password. The supplied username, old password, and new password are checked in the database for a match.

Main Success Scenario:

1. The WebService parses a supplied username, old password, and new password from JSON
2. The database is queried for a password which is linked to the supplied username
3. Result found
    a. If a result is found, the stored password is decrypted and matched against the supplied old password. If they do not match, JSONObject is created which contains the username and the success value, i.e. false, and a reason (password mismatch). The JSONObject is returned to the client
    b. If a result is found, the stored password is decrypted and matched against the supplied new password. If they match, JSONObject is created which contains the username and the success value, i.e. false, and a reason (new password is same as old). The JSONObject is returned to the client
4. The new password is encrypted
5. The database is updated with the new password
6. A JSONObject is created which contains the username and the success value, i.e. true.
7. The JSONObject is returned to the client

5.2.12 Log Record

Use Case: Log Record

Actors: WebService, ProjectDatabase

Description: This use case begins when the WebService is requested to log a new number plate. The data supplied is parsed and stored in the database. The supplied data includes:

- guess – The apps original OCR result
- actual – The OCR result confirmed by the user
- username – The username of Staff using app.

An email is sent if an infraction occurs.

Main Success Scenario:

1. The WebService parses a supplied guess, actual, and username
2. The details are stored into a Log table, for debugging purposes
3. The actual value is queried in the database
4. If a result is found, a new infraction is logged, by supplying the vehicle registration and the username of the Staff. An email is generated and sent to the vehicle owner.
5. A JSONObject is created which contains the username and the infraction value, i.e. true
6. The JSONObject is returned to the client

5.2.13 Crud Users

Use Case: CRUD Users

Actors: WebService, ProjectDatabase

Description: This use case occurs when the WebService is request to add new Staff or vehicles to the database. The details are inserted into the relevant area of the database.

Main Success Scenario:

1. The WebService receives supplied details
2. The details are inserted into the database

5.2.14 Generate Reports

Use Case: Generate Reports

Actors: WebService, ProjectDatabase

Description: This use case occurs when the WebService is requested to generate a report of all infractions. They are grouped by All, Today, This Month, Repeat offenders.

Main Success Scenario:

1. The WebService receives a request to generate report
2. The database is queried for all infractions
3. The database is queried for all infraction with today's date
4. The database is queried for all infraction with this month's date
5. The database is queried for all vehicle registrations which appear 2 or more times.

6 Iteration 1

6.1 Overview

This iteration is if the first of three iterations, and will be primarily focus on research and available technologies. A clear and definite understanding of image processing techniques and how to apply them is vital if this application is to succeed. If the image processing is inadequate, then later steps such as OCR will be unreliable and the application will lose its purpose.

It will also be necessary to check that any additional API's are compatible and effective when used in an Android application.

6.2 Iteration 1 Goals

6.2.1 Image Processing with OpenCV

This application is highly reliant on the field of image processing, and as such the basics must be learned. It will be a major goal of this iteration to become familiar with various techniques available, and deciding which to apply in this project.

There are many standard steps and techniques which are recommended in ANPR so it will need to be decided which of these steps and techniques to use and how to apply them. As such this iteration will be heavily research based.

Topics to be considered include:

- Blurring
  - Box Blur
  - Gaussian Blur
- Greyscale
  - Average
  - MinMax
  - Luminosity
- Segmentation
  - Horizontal Project
  - Vertical Projection
- Thresholding
  - Basic
  - Otsu
- Edge Detection
  - Canny
  - Sobel
  - Prewwit

Most of the above steps are fully implemented in the OpenCV API, and can be configured very precisely to match a user's requirements. It will be important to understand the theory of these steps, and also how they are implemented in OpenCV. This will become important when deciding what parameters to use in the OpenCV function calls.

6.2.2 Android Development

Since this application will be developed on Android, some research into Android development will be required. This will involve getting a grasp of the basic setup of an Android application, the standards and best practices recommended, and just a general overview of what needs to be done to create a useable, working application.

6.2.3 Custom Camera Application

To reduce the amount of image processing the application needs to do, a guide box will be used in which a number plate is position, to remove the need to localise it from an image. This design decision means that the native Android app cannot be used, as you cannot apply an overlay of a guide box to it. As such, it will be necessary to create our own custom camera to be used. Since it

will only be used with this application, there is no need to over complicate the features of the custom camera. The basic requirements are:

- Guide box
- Take, crop, and save image from within the guide box
- Enable autofocus
- The simplicity of the requirements should mean that this is a trivial task in this iteration.
- Optical Character Recognition with TessTwo

In this project, the two most important areas are the image pre-processing and the optical character recognition. These two steps combined form the core of the project. TessTwo is an open source OCR engine maintained by Google, which has received praise for its usability and accuracy.

It will be checked if TessTwo can be run successfully on an Android device to ensure it can be used properly once later image processing features have been implemented.

6.3 Conclusion

Camera has basic functionality but plagued with bugs

OpenCV has been integrated and can be used effortlessly with the Android application. The image processing reduces image to a bare skeleton and removes most of the unwanted detail in the image. However, the values used in the various steps need to refined to work reliably on a larger range of image. A successful result of the image processing is shown below.

Figure 2 The original image (above) and the processed image (below)

The following steps were applied:

1. Gaussian Blur with kernel size = (5, 5) and standard deviation = 3
2. Greyscale, using a variation of Luminosity
3. Threshold using a lower bound of 127, an upper bound of 255, and inverting the black and white pixels.
4. Canny Edge Detector using a lower bound of 127, an upper bound of 255.

7 Iteration 2

7.1 Overview

This iteration is the second of three iterations, and will be primarily focus on additional research, cleaning up the image pre-processing, character segmentation, and character recognition. This iteration involves the two core parts of the OCR stage, that is, Character Segmentation and Character Recognition.

7.2 Iteration 1 issues

In general, iteration 1 was very successful. A lot of research was conducted, and the required API's were easily integrated into the Android project. The basic image pre-processing techniques were applied successfully, and can now be refined to make them more flexible on various other test images. There were 2 issues which could not be overcome in the time involving the Camera API and TessTwo, which are discussed below.

7.2.1 Custom Android Camera

The custom camera took a lot more time to get working stage than predicted, and is plagued by a litany of bugs. There are a number of factors which each contribute something to the list of problems.

7.2.1.1 Device Orientation Vs Camera Orientation

Android devices have a natural orientation of portrait mode. The camera however has a natural orientation of landscape mode. Since a picture can be taken in one of four orientations (portrait, landscape, reverse landscape, reverse portrait), it is required to know the camera orientation when saving the image, so as to save it with the correct orientation. An issue arises whereby some devices will automatically rotate the saved image for you to match the camera orientation, and some do not. The image needs to be saved correctly, otherwise later steps, such as segmentation and recognition will fail, as the image could be upside down, sideways, etc. Currently, the image is saved correctly, as the device the camera was tested on takes care of image rotation automatically. However, this will need to be addressed for device portability.

7.2.1.2 Aspect Ratio

This issue arises when portrait mode is enabled. As the camera has a natural landscape orientation, when the device is held in portrait mode, the image needs to be adjusted. Telling the camera to shoot in portrait mode is rather simple, however displaying the preview on the device is not.

The camera must be asked for all display sizes it supports, and then find the best match against the devices screen size. Although this is awkward to work with and seems to be a common issue with app developers, it does allow manufacturers to use a particular camera model on various other devices. To resolve the aspect ratio issue, and to potentially prevent issues in the image processing stage, the app will lock the camera into landscape or reverse landscape options only.

7.2.1.3 Camera API

A minor issue although still worth noting, the API used for this app is now deprecated and replaced with the new Camera2 API from Android version 6. This camera solves some issues with the older API and introduces a lot of new features. However, by using Camera2 the range of devices the app can run on is vastly decreased. As such, the older API will be used and is still supported by Android version 6, as many apps already released use it.

7.2.2 TessTwo

While TessTwo was integrated successfully, and shown to work easily with printed text, such as that in a book or magazine, it fails completely on all number plates tested with. The issue seems to be with the font that TessTwo has been trained to recognise. Further research is required to find out why, and to also discover possible fixes such as addition training files, retraining, or possible abandonment.

7.3 Iteration 2 Goals

7.3.1 Further Research on Image Processing

This iteration will require a lot of further image processing research, mainly involving getting code to work with a wider range of images. Additional techniques may need to be explored, and current ones may need to be modified slightly to be more robust.

7.3.2 Character Segmentation

Once a processed image is obtained, some way of extracting the actual characters needs to be found. A common technique used is horizontal and vertical projection. By creating a histogram of the number of white pixels both by row and column, it can be easily visible where the largest number of pixels lie, and these regions can be considered to contain the individual characters. An issue that will arise will be deciding how to ignore noise that still remains in the image. A mask can could be applied to the image using these two histograms, and the area where they overlap should be the individual characters.

7.3.3 Character Recognition

TessTwo is proving to be tricky to get working with the font type used on number plates. It may be necessary to find additional training files, or even creating brand new ones trained on the particular font required. This area will need more research and experimentation. Some other alternatives may need to researched and considered.

7.4 Conclusion

After much research and experimenting, the image processing steps have been altered and a slight change in direction was pursued. Some of the previous image processing steps were removed, while some new steps were added. Overall, the image processing is quite reliable and produces very few errors.

The Canny Edge Detection was removed completely. While the result form iteration 1 was pleasing visually, it provided no additional help. Edge detection would be more useful if the app were to perform plate localisation, i.e. locating the where the plate is in the image.

It was found that in most cases, applying a blur to the image was actually counterproductive. In the case of a clean number plate, it provided a slight assistance, however in the case of a dirty number plate the resulting blurred image would very often blur two characters together, or blur a character and the border together. This was making segmenting them a very difficult task. As such, this step has been removed completely. A new step was introduced called Image Opening. This is a morphological operation, which removes small objects from the foreground of the image, and placing them in the background. This step helps reducing the noise in the image without accidentally connecting any characters.

The horizontal and vertical project methods for segmenting the individual characters, while straight forward in theory, proved extremely challenging to implement. In the case where an image has a black background and the only white pixels are the characters, it is very easy to see where the

characters are. However, in an actual scenario this will never be the case. Small areas of dirt or shadows may appear and cause the histograms generated to be less precise and thus harder to decipher programmatically. What was decided was to use another approach which was assisted greatly by OpenCV. It involves searching for all contours in an image, creating a bonding box around each “shape”, and filtering them out by only keeping ones with a certain aspect ratio and size. Shown below is an example of this step.

Figure 3 The original image (above) and the processed image (below) applying the altered steps

One obvious issue is that some unwanted detail may appear on the image which has the same aspect ratio and size as what is considered a valid character, and will be one. This will result in bad OCR later and a solution will need to be found.

8 Iteration 3

8.1 Overview

This iteration is the final of three iterations and will involve wrapping up all the loose ends of the project. The app itself needs to be developed further to make it more releasable, such as adding in a login screen, a main menu, and others as required. The app also needs to have the functionality to connect with the cloud database and store the number plates there. Finally, the admin web page needs to be developed. This will be a busy iteration as some elements that were not completed in previous iterations will roll over into this one.

8.2 Iteration 2 issues

Iteration 2 was generally quite smooth, however a few changes in direction were made after hitting a few dead ends. The most notable was the decision to not use edge detection. This was something where a lot of time was invested researching and understanding, and has now been abandoned.

TessTwo has also been abandoned. It was determined that in order for TessTwo to work, a custom training file would need to be created. The time spent in setting up the training environment, sourcing all the test images, and finally training it, would be a huge time sink which may not pay off in the end. In order to avoid this situation, alternatives such as Neural Networks will be explored.

8.3 Iteration 3 Goals

8.3.1 Optical Character Recognition

This iteration involves getting the OCR to work. The method of template matching supplied by OpenCV will be investigated. This is an element which was hoped to be completed by now, and is really the core of the project. The process of creating templates will be time consuming, so a tool to aid in this might be required.

8.3.2 Cloud Database

The cloud database is set up, however the code for the web service needs to be written. This will be split up in two parts. One part will deal solely with the Android client. It will be responsible to receiving and sending JSON, as well as manipulating the database.

8.3.3 Admin Web Page

The web page will be created to allow for some generic reports to be created, as well as some primitive CRUD operations. The overall appearance and error handling will be left basic due to time constraints. This area could be considered a prototype feature for illustration purposes.

8.3.4 Testing

The app will need to undergo a lot of manual testing. This involves taking pictures of a large number of vehicles and checking the overall result.