Research Document

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 About

This project will involve creating a mobile phone application, which will allow a user to take pictures of vehicle registrations and check if that vehicle is permitted to be parked in the IT visitor's carpark. To do this, the application will analyse the image using various methods and algorithms, and extract a textual representation of the registration number. It will then query this text in an online database, and based on this result, will either give the all clear, send an automated email if the owner is a student or staff member asking them to move, or some other course of action if the owner is unknown to the IT. A web page will be available for staff to use to generate reports, such as:

- Monthly infringements
- Repeat offenders
- Average users

2 System Overview

The client will communicate with the database by using a Web Service. Any requests made by the clients are handled by this Web Service.

The web page will communicate with the database by using a Web Service. Any requests made by the web page are handled by this Web Service.

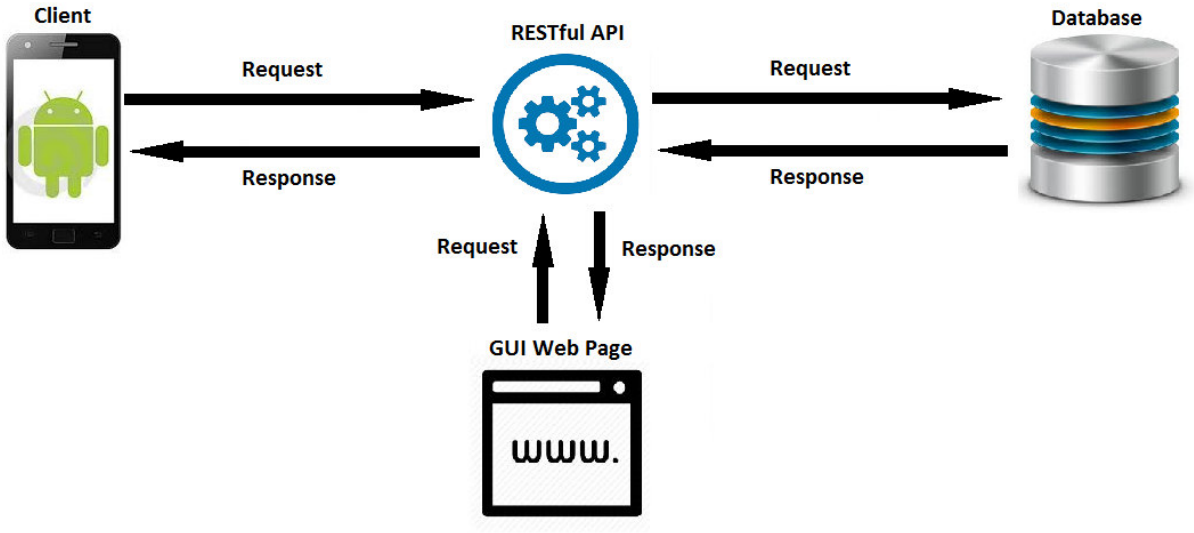


Figure 1 Architectural overview

A Web Service is used for security and scalability reasons. The mobile and wen clients shouldn't have direct access to the database as this requires them to have the database password, which if falls into the wrong hands could lead to malicious attacks. A de-compiler could be used on a phone to retrieve the password, for example.

3 Technologies

There will be a variety of technologies involved in making this app, and decisions on each need to be made:

- Platforms: Which device will it be developed for? Language used is device dependant.
- Image Processing: Which API will be used to process images?
- Web Service: What language will this be written in?
- Database: Which database will be used?
- Web Page: What will the web page be written in?
-

3.1 Platforms

3.1.1 Android

Android is a mobile operating system which is currently owned and developed by Google. As of 2015, it has the largest install base of all operating systems. Android apps are written using the Android Software Development Kit (SDK) and most often using the Java programming language. Java may be combined with other languages such as C/C++ and the Go programming language (developed by Google). Android is the cheaper OS to develop on as licences required to release apps onto their Play Store for purchase/download are relatively cheap. Developing apps with Android is relatively simple as there is an official development environment available called Android Studio. This IDE makes begging a new app extremely simple as a new project will contain the minimum amount of code necessary to have a simple Hello World app. [1]

3.1.2 iOS

iOS (iPhone Operating System) is the mobile operating system developed by Apple and distributed only for Apple hardware. In September 2015, it was the most commonly used mobile operating system in a few countries such as Canada, the United States, the United Kingdom, Japan, and Australia. iOS apps are developed using the Objective-C using programming language the Xcode IDE. Users can create and develop iOS applications using a free copy of Xcode. However, they cannot test or publish their applications on a physical iOS device without first paying the yearly $99.00 iPhone Developer or Mac Developer Program fee. [2]

3.1.3 Windows Phone

Windows Phone (WP) is a family of mobile operating systems created and developed by Microsoft for smartphones and is the successor to Windows Mobile and Zune. Apps can be published on their App Hub, which are then reviewed for quality etc. A yearly fee is required by developers who wish to submit an app. Windows Phone applications can be developed using C#/Visual Basic.NET (.NET), C++ (CX) or HTML5/JavaScript. Applications and games can be based on XNA, a Windows Phone-specific version of Silverlight. [3]

3.2 Image Processing Libraries

3.2.1 OpenCV

OpenCV (Open Source Computer Vision) is a library aimed at real-time computer vision. It was developed Intel originally, then supported by Willow Garage and is now currently maintained by Itseez. The library is cross-platform and free to use under the open-source BSD license. It is written in C++ and its main interface is C++ but there are versions available for Python, Java, and MATLAB.

“The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo

cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc." [4]

This library contains more than enough information needed for number ANPR and as such is a favourite candidate for use.

3.2.2 SimpleCV

SimpleCV is an open source framework written in Python. With it, you gain access to a few computer vision libraries, including OpenCV, but without having to worry about bit depths, file formats, colour spaces, buffer management, eigenvalues, or matrix versus bitmap storage, etc.

"It lets you work with the images or video streams that come from webcams, Kinects, FireWire and IP cameras, or mobile phones. It helps you build software to make your various technologies not only see the world, but understand it too. SimpleCV is free to use, and because it's open source, you can also modify the code if you choose to." [5]

While this library seems nice and simple, being written in Python could be a huge problem as not all platforms support it. A workaround could be found, but could simply be broken again in future with an update. It may be best to avoid this risk.

3.3 RESTful Web Service

3.3.1 What is REST?

"REST stands for **RE**presentational **S**tate **T**ransfer. REST is web standards based architecture and uses HTTP Protocol for data communication. It revolves around resource where every component is a resource and a resource is accessed by a common interface using HTTP standard methods.

In REST architecture, a REST Server simply provides access to resources and REST client accesses and presents the resources. Here each resource is identified by URIs/ global IDs. REST uses various representations to represent a resource like text, JSON and XML. Now a days JSON is the most popular format being used in web services?" [6]

3.3.2 What is RESTful Web Service?

"A web service is a collection of open protocols and standards used for exchanging data between applications or systems. Software applications written in various programming languages and running on various platforms can use web services to exchange data over computer networks like the Internet in a manner similar to inter-process communication on a single computer. This interoperability (e.g., between Java and Python, or Windows and Linux applications) is due to the use of open standards.

Web services based on REST Architecture are known as RESTful web services. These web services use HTTP methods to implement the concept of REST architecture. A RESTful web service usually defines a URI, Uniform Resource Identifier a service, provides resource representation such as JSON and set of HTTP Methods." [7]

3.3.3 Which Language?

A web service can be written in a variety of languages, such as JavaScript and Python. Python has a micro framework called Flask which in turn has an extension called Flask-RESTful, which supplies the functionalities required to create a RESTful web service.

"**Flask-RESTful** is an extension for Flask that adds support for quickly building REST APIs. It is a lightweight abstraction that works with your existing ORM/libraries. Flask-RESTful encourages best practices with minimal setup. If you are familiar with Flask, Flask-RESTful should be easy to pick up." [8]

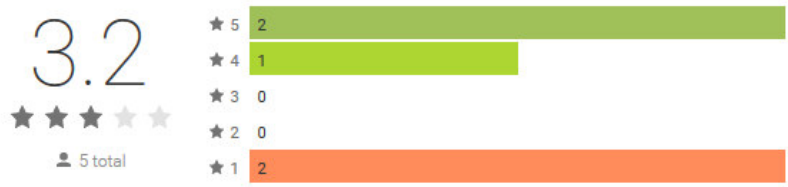
3.4 Database

3.4.1 MySQL

MySQL is an open-source relational database management system. It is used in conjunction with SQL to perform CRUD (Create, Update, Delete) operations. MySQL is a popular choice of database for use in web applications and since the data being stored is very tablature in nature, this is an ideal choice for this project. [9]

4 Existing Applications

4.1 ANPReader Lite



Cost: €1.17

ANPReader is a very basic number plate recognition Android application available for purchase on the Google Play Store. It allows the user to take a picture of car and then displays the text on screen, which the user can then select and copy/paste into another application if needed. The application all signals the reliability of the recognition by displaying the result in different colour texts:

- Green = high confidence
- Amber = medium confidence
- Red = low confidence

It can do recognition on number plates of different shapes, i.e., rectangle and square, as well number plates in bad lighting.

Figure 2 Various results

The application supplies a result in seconds, so there aren't any long waiting times. All processing is done on the device itself, so there is no connection to an external server needed, which can save users

using up their data. When taking a picture, a guide box is provided to help localise the number plate for better recognition. All processing done by app on phone. No data etc. require to connect to server.

The overall look of the application is very basic, and unprofessional looking. There isn't much of a user interface, and the displayed text doesn't look very slick. User reviews are not very positive about the recognition success rate. Another drawback is that the Lite version is limited to 20 reads per day.

4.2 Parking Enforcer

Cost: €18.04

Parking Enforcer is a number plate recognition and logging Android application available for purchase on the Google Play Store. It is an extension of the ANPReader application above. The application allows the user to take a picture of car and then displays the text on screen, which can then be logged and compared against a database of previously logged number plates to quickly find out parking durations. Images can also be stored on an SD card, along with the number plate code and time stamp to be used as evidence if needed. It also allows for a particular set of number plates to be whitelisted, for vehicles where no action should be taken. The application can maintain a database of up to 500 vehicles.

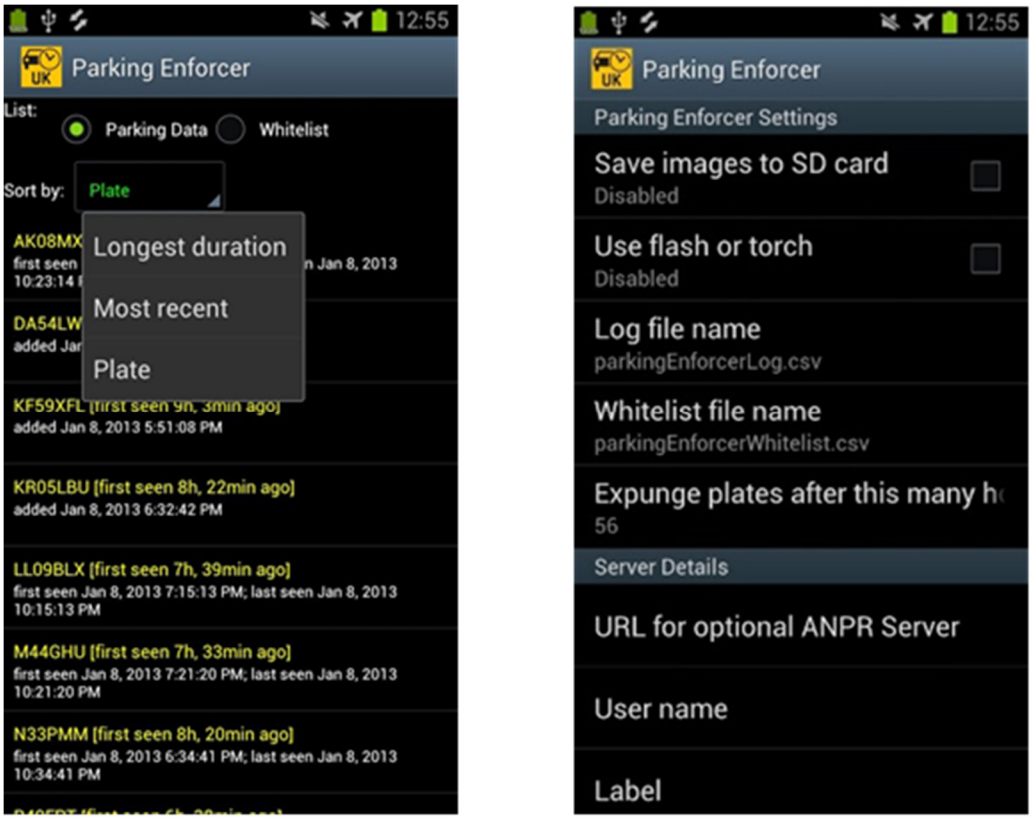


Figure 3 Displaying logged vehicles

The overall look of the application is again very basic, and unprofessional looking. The list of logged vehicles looks ok, but could be highly improved upon. No update was done to the general look and feel of the application when extending from the simple ANPReader application, which gives the impression that this extra logging functionality was tacked on to have another product to release. As with the ANPReader application, the recognition success rates are not great, and application provides more expensive versions of the application which supposedly provide better success rates.

4.3 License Plate Reader (ANPR)

Cost: Free

This application simply displays a text version of a number plate. There is no other functionality provided with this application.

Figure 4 Results screen

Again, the user interface is very poor and doesn't look very professional. The application is constantly processing the camera input in real-time, so a picture doesn't need to be taken. While this makes sense in certain applications, it is completely unnecessary here as the application doesn't do anything with this feature, and is using up processing power.

5 Basic Number Plate Recognition Steps

5.1 License Plate Localisation

Localising is an algorithmic function that determines what aspect of the vehicle's image is the license plate. For example, the algorithm must rule out a vehicle's mirror, grill, headlight, bumper, sticker, etc. In general, algorithms look for geometric shapes of rectangular proportion. However, since a vehicle can have many rectangular objects on it, further algorithms are needed to validate that the identified object is indeed a license plate. To accomplish this, key components of the algorithm look for characteristics that would indicate that the object is a license plate. The algorithm searches for a similar background colour of unified proportion and contrast as a means to differentiate objects on a vehicle.

5.2 License Plate Sizing and Orientation

Components of algorithms that adjust for the angular skew of the license plate image to accurately sample, correct, and proportionally recalculate to an optimal size.

5.3 Normalisation

Algorithm for adjusting the contrast and brightness of the captured license plate image.

5.4 Character Segmentation

Algorithm that locates the separate alpha numeric characters on a license plate. Algorithms also look for characters of equal colour and equidistance, with similar font structures to break apart each individual character. This sequential congruency of the characters embodies a characteristic set that is typically uniform, regardless of the type of license plate. Character Segmentation separates each letter or number where it is subsequently processed by optical character recognition (OCR) algorithms.

5.5 Optical Character Recognition (OCR)

Algorithm for translating the captured image into alpha numeric text.

5.6 Syntactical/Geometrical Analysis

Algorithm to verify alpha numeric information and arrangement with a specific rule set. The algorithms operate sequentially with instructions being executed in milliseconds. The successful completion of each algorithm is required before subsequent algorithms can be operational.

6 Image Pre-Processing

6.1 Image Blurring/Smoothing

When dealing with number plates, we want to ignore any background detail, such as the rest of the car, people, or scenery. To achieve this, we can blur the image, which will cause the background detail (noise) to be reduced. Most edge-detection algorithms are sensitive to noise, so applying the correct amount of blur will increase the correctness of any further image processing done, such as edge detection. The more an image is blurred, the less edges are detected. It is also important not to blur the image too much, otherwise the details of the number plate itself may become too faint to process.

6.1.1 Mean Blur

This is also known as a Box Blur, or Average Blur. Each pixel in the resulting image has a value equal to the average value in the neighbouring pixel in the original image, including itself. [10]

It has the following properties:

- Odd ordered: masks are 3x3, 5x5, 7x7, 9x9, etc.
- The sum of all elements = 1
- All elements are the same

A mask is used to apply the blur. The mask is centred on a pixel and the amount of neighbours in each direction

selected to average depends upon the size of the mask.

While the mean blur is not only easy to understand, it is extremely to apply. Below is a sample image, with the mean blur applied.

Figure 5 8 Standard OpenCV blur with kernel size = Size (5, 5)

As can be observed, the image is blurred rather thoroughly. The blue tag on the right with the letters IRL in white has become less prominent, as well the LAOIS text above the actual plate number. One issue that is noticeable is that the regions of interest for an OCR app are quite blurred. In the above case, this wouldn't be an issue as the plate is relatively clean. In the case where a lot of dirt or shadows are present, the characters could be potentially blurred together.

6.1.2 Gaussian Blur

Gaussian Blur, also known as Gaussian smoothing, is a blur achieved by using a Gaussian function.

"A Gaussian function of one variable with spread σ is of the following form, where c is some scale factor:

$f(x) = ce\left(-\frac{x^2}{2\sigma^2}\right)$" [11]

And the Gaussian Blur in 2D uses the function:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Here, σ is the standard deviation of the distribution and we assume the distribution has a mean of 0. [12]

The Gaussian blur is more mathematical in nature and offers a more realistic result. Below is a sample

Figure 6 9 OpenCV Gaussian blur with kernel size = Size (5, 5) and standard deviation Θ = 0

The result as seen above, is quite a natural looking blur, given the same kernel size was used as the mean blur above. A lot of subtle nose has been removed, such as the surface the plate itself is lying on, while preserving the shape and structure of each character. However, as with the mean blur, it runs into the issue of blurring unwanted detail into the areas of interest thereby distorting them. Overall however, this is definitely the most useful of the various blurs available.

6.2 Greyscale

To achieve a grey colour, the RGB values of a pixel must all be the same. In order to convert an image to greyscale, each pixel in the image must be examined, and a value of grey must be determined for that pixel. Using the RGB values of the pixel, a simple formula is applied to calculate the value of grey for that pixel. Once this value has been calculated, the RGB values of the pixel are set to this grey value. There are three popular methods of calculating this value. [13]

6.2.1 Lightness

The lightness method takes the average of the most prominent and least prominent colours of a pixel. To calculate the grey value, the following formula is applied:

$$Grey = \frac{\max(R,G,B) + \min(R,G,B)}{2}$$

Figure 7 Original Image (left) and Lightness applied (right)

The resulting image preserves all the detail of the original image, however the image appears dull and does not properly represent the original colours correctly.

6.2.2 Average

The average method simply takes the average of the pixel RGB values. To calculate the grey value, the following formula is applied:

$$Grey = \frac{R + G + B}{3}$$

Figure 8 Original Image (left) and Average applied (right)

The resulting image once again preserves all the detail of the original image, however this time it better represents the colours in the image. The image is not as dull as the lightness version, but it still doesn't feel right. It shows the difference in colours quite well, however it still does convey the brightness of the image.

6.2.3 Luminosity

The luminosity method is a more sophisticated method of applying greyscale in that it accounts for the human perception of colour. To achieve this, it will apply a weight to the RGB values. We're more sensitive to green than other colours, so green is weighted most heavily. To calculate the grey value, the following formula is applied:

$$Grey = (0.21 * R) + (0.72 * G) + (0.07 * B)$$

Figure 9 Original Image (left) and Luminocity applied (right)

This time the resulting image has a truer representation of the original image than the previous two methods. The brightness of the original image is evident in the greyscale version, and it more in line with how the human eye perceives colours.

OpenCV uses a variation of this method by weighting the RGB values differently. The formula it uses is as follows:

$$Grey = (0.299 * R) + (0.587 * G) + (0.114 * B)$$

This formula is based on CCIR 601 which is a standard for encoding interlaced analogue video signals in digital video form. The overall effect is very similar.

Below is an example of this method applied.

6.2.4 Histogram Equalisation

Image histogram equalisation involves adjusting the contrast of an image. An images histogram is a graphical representation of the intensity distribution of an image. It is a measure of the number of pixels for each intensity values considered. [14]

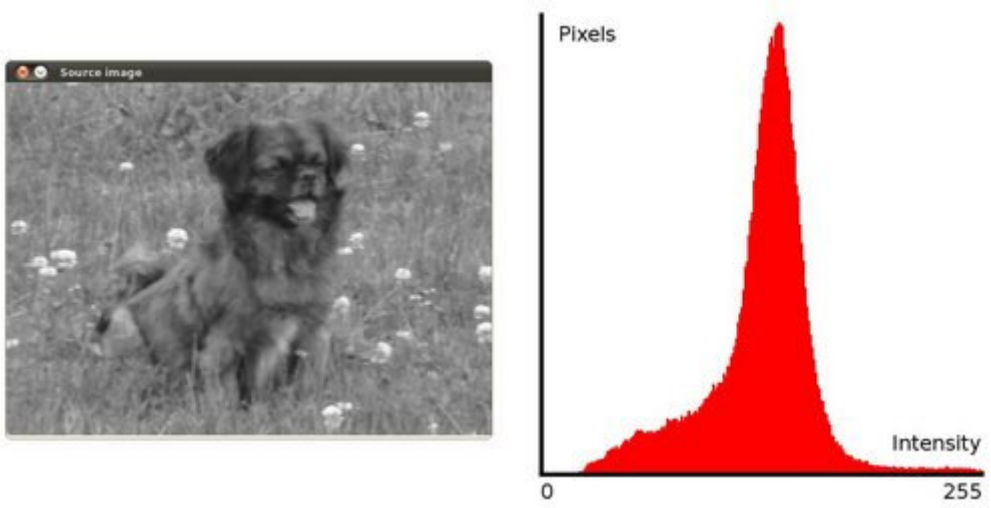


Figure 10 Histogram of an image

Histogram equalisation involves stretching out the intensity range in order to have a more spread out and even histogram.

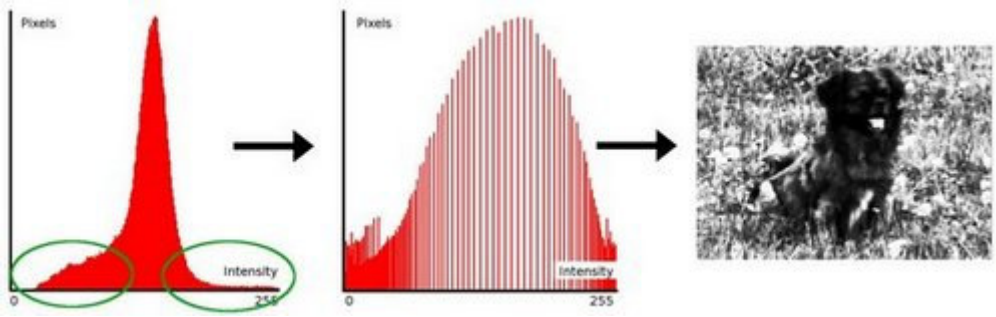


Figure 11 Applying Histogram Equalisation to an image

As can be observed, the intensities are more evenly spread out yet the resulting image is still noticeable. It has resulted in the main detail becoming slightly more obvious.

Below is an example of this being applied.

Figure 12 Original image and greyscale version before equalisation

Figure 13 Original image and greyscale version after equalisation

We can observe above that although the pixel intensities are more evenly spread, the characters themselves have actually become less obvious, and a lot of noise has been added. It is reasonable to expect an even worse result should the plate contain any dirt or shadows. Since the detail we are interested in is rather obvious to begin with, applying this step would appear to be counterproductive.

6.3 Binarisation

Binarisation involves converting an image to a binary black and white image. For any pixel in the image, the RGB values are either all 0 or all 255. [15]

6.3.1 Threshold

In its simplest form, some value *n* can be used as the threshold. Each pixel of the image is checked, and any of the RGB channel values is checked (a pixel in a greyscale image will have the same RGB values). If this value is less than the threshold, set the RGB values of the pixel to 255, otherwise set them to 0.

E.g.

A pixel *p* has the RGB values (50, 50, 50) and a threshold *t* is given as 127. We check any of the RGB values, since they are all the same, against the threshold value *t*. The grey value of *p* is less than *t*, so we set the value of *p* to be white: *p* = (255, 255, 255).

A pixel *q* has the RGB values (150, 150, 150) and a threshold *t* is given as 127. We check any of the RGB values, since they are all the same, against the threshold value *t*. The grey value of *q* is greater than *t*, so we set the value of *q* to be black: *q*= (0, 0, 0).

This can be taken a step further by adding a 2nd threshold value, which in conjunction with the 1st value can be used as a range. All values within the range become black, and all others become white, or vice versa.

E.g.

A pixel *p* has the RGB values (50, 50, 50), a threshold *t1* is given as 127 and *t2* is given as 210. We check any of the RGB values, since they are all the same, against the threshold values *t1* and *t2*. The grey value of *p* is outside of the range *t1-t2,* so we set the value of *p* to be white: *p* = (255, 255, 255).

A pixel *q* has the RGB values (150, 150, 150), a threshold *t1* is given as 127 and *t2* is given as 210. We check any of the RGB values, since they are all the same, against the threshold values *t1* and *t2*. The grey value of *q* is inside of the range *t1-t2,* so we set the value of *p* to be black: *p* = (0, 0, 0).

Below is an example of this step.

Figure 14 Thresholding using grey image, lower bound of 127, and upper bound of 255

As can be seen, the resulting image contains a clear view of the characters on the plate. It is also noticeable that the colours appear to be inverted, i.e. the black characters are now shown in white. This will be useful for later stages of the image processing, where OpenCV will consider white pixels as pixels of interest. In general, the resulting image contains a lot less noise and areas that are not of interest.

The effects of performing histogram equalisation become very apparent during this operation, as can be seen below.

Figure 15 Thresholding an equalised image with a lower bound of 127 and upper bound of 255

The resulting image has lost a lot of detail, due to image having a more even spread of grey values. This requires more precise values for upper and lower bounds when thresholding, which puts huge limits on the variety of images it can be used for. A set of values which work perfectly on one image, maybe completely useless on another. In general, this may be enough reason to discount histogram equalisation.

6.3.2 Otsu Method

This method is used to automatically perform clustering-based image thresholding. The algorithm assumes that the image contains two classes of pixels, foreground and background. It then calculates the optimum threshold separating the two classes so that their combined spread (intra-class variance) is minimal, or equivalently (because the sum of pairwise squared distances is constant), so that their inter-class variance is maximal. [16]

Figure 16 Image before and after Otsu thresholding

The nice feature of this method, is that it will calculate the threshold bounds to use on a per image basis. Considering that the detail we are interested on a number plate is black, we are almost guaranteed to preserve that after applying this method. Below is an example of this step.

Figure 17 Image before and after Otsu thresholding with a value of 118

6.4 Edge Detection

Edge detection is the name for a set of mathematical methods which aim at identifying points in a digital image at which the image brightness changes sharply or, more formally, has discontinuities. The points at which image brightness changes sharply are typically organized into a set of curved line segments termed edges.

6.5 Robert's Edge Detector

Robert's Edge Detection, also known as Roberts Cross, is an operator used in image processing and computer vision for edge detection. It was initially proposed by Lawrence Roberts in 1963. The idea behind it is to approximate a gradient of an image through discrete differentiation. This is achieved by computing the sum of the squares of the differences between diagonally adjacent pixels. [17]

Figure 18 Original image (left), Robert's Edge Detection (centre), Threshold of 80 applied (right)

Roberts proposed the following equations:

$$y_{i,j} = \sqrt{x_{i,j}}$$

$$z_{i,j} = \sqrt{(y_{i,j} - y_{i+1,j+1})^2 + (y_{i+1,j} - y_{i,j+1})^2}$$

Where x is the initial intensity value in the image, z is the computed derivative, and I and j represent the location in the image. The results of this operation will highlight changes in intensity in a diagonal

direction. One of the most appealing aspects of this operation is its simplicity; the kernel is small and contains only integers. However, with the speed of computers today this advantage is negligible and the Roberts cross suffers greatly from sensitivity to noise.

In order to perform edge detection with this method, the image must first be convolved using the following kernels:

$$\begin{bmatrix} +1 & 0 \\ 0 & +1 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix}$$

Let $I(x, y)$ be a point in the original image.
Let $G_x(x, y)$ be a point in an image by convolving with the first kernel.
Let $G_y(x, y)$ be a point in an image by convolving with the second kernel.

The gradient can be defined as:

$$\nabla I(x, y) = G(x, y) = \sqrt{G_x^2 + G_y^2}$$

The direction of the gradient can also be defined as:

$$\theta(x, y) = \arctan\left(\frac{G_y(x,y)}{G_x(x,y)}\right) \text{ [8]}$$

6.6 Sobel Edge Detection

The Sobel operator performs a 2-D spatial gradient measurement on an image and so emphasizes regions of high spatial frequency that correspond to edges. [18]

Figure 19 Original image (left) and Sobel applied (right)

The operator uses two 3×3 kernels which are convolved with the original image to calculate approximations of the derivatives, one for horizontal changes, and one for vertical changes. If **A** is defined as a source image, $\boldsymbol{G}_x$ and $\boldsymbol{G}_y$are defined as two images which at each point contain the horizontal and vertical derivative approximations, the computations are:

$$\boldsymbol{G}_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * \boldsymbol{A}$$

$$\boldsymbol{G}_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +0 & +1 \end{bmatrix} * \boldsymbol{A}$$

These can be combined together to find the absolute magnitude of the gradient at each point and the orientation of that gradient. The gradient magnitude is given by:

$$|G_x| = \sqrt{G_x^2 + G_y^2}$$

An approximate magnitude is computed using:

$$|G_x| = |G_x| + |G_y|$$

This is much faster to compute.

The angle of orientation of the edge relative to the pixel grid giving rise to the spatial gradient is given by:

$$\theta = \arctan(G_x/G_y) \text{ [10]}$$

Below is an example of this method being applied.

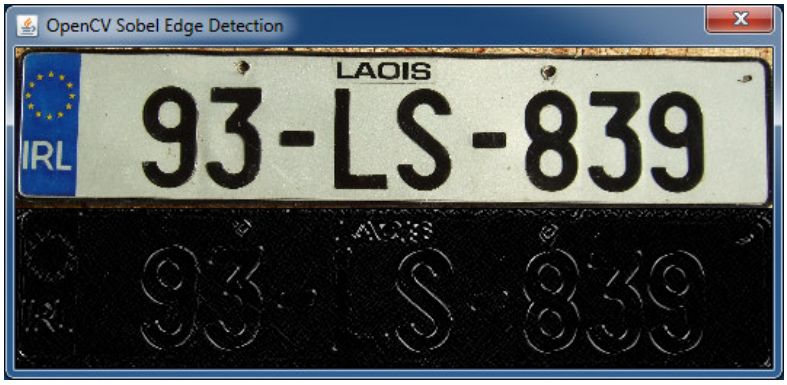


Figure 20 Sobel Edge detection

6.7 Canny Edge Detection

Canny Edge Detection is an operator used in image processing and computer vision for edge detection. The Canny edge detector was designed to be an optimal edge detector. It takes as input a greyscale image and produces and image showing detected edges.

Figure 21 Canny edge detection

It is a multistage operator:

The image is smoothed by Gaussian convolution.

A simple 2D first derivative operator is applied to the smoothed image to highlight regions of the image with high first spatial derivatives. Edges give rise to ridges in the gradient magnitude image.

The algorithm then tracks along the top of these ridges and sets to zero all pixels that are not actually on the ridge top so as to give a thin line in the output, a process known as *non-maximal suppression*.

The tracking process exhibits hysteresis controlled by two thresholds: *T1* and *T2*, with *T1* > *T2*. Tracking can only begin at a point on a ridge higher than *T1*. Tracking then continues in both directions out from that point until the height of the ridge falls below *T2*. This hysteresis helps to ensure that noisy edges are not broken up into multiple edge fragments. [19]

Steps:

1. Filter out any noise, using a Gaussian filter. An example is shown below.

$$K = \frac{1}{159}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix}$$

2. Find the intensity gradient of the image.
    a. Apply a pair of convolution masks (in x and y directions)

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}$$

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix}$$

    b. Find the gradient strength and direction with:

$$G = \sqrt{G_x^2 + G_y^2}$$

$$\theta = \arctan\left(\frac{G_y}{G_x}\right)$$

3. *Non-maximum* suppression is applied. This removes pixels that are not considered to be part of an edge. Hence, only thin lines (candidate edges) will remain.
4. *Hysteresis*: The final step. Canny does use two thresholds (upper and lower):
    a. If a pixel gradient is higher than the *upper* threshold, the pixel is accepted as an edge
    b. If a pixel gradient value is below the *lower* threshold, then it is rejected.
    c. If the pixel gradient is between the two thresholds, then it will be accepted only if it is connected to a pixel that is above the *upper* threshold.

    Canny recommended an *upper* to *lower* ratio of between 2:1 and 3:1. [20]

Below is an example of this method being applied.

Figure 22 Canny edge detection with lower bounds = 127, upper bounds = 255, aperture size = 3, and L2 gradient = false

7 Character Segmentation

7.1 Vertical and Horizontal Projection

Horizontal and Vertical project involve counting the number of black or white pixels in a row or column of the image. From this, a histogram can be created which shows the areas where the most amount of desired pixels lies. [21]

7.1.1 Horizontal Projection

A vertical projection involves counting pixels in each row. As can be seen, the areas of interest become quite apparent in the horizontal projection. A horizontal projection can be used to locate the text in an image, and then combined with a vertical projection to locate each individual character.

Figure 23 Example of a horizontal projection

The following pseudocode can be applied:

```
height = image.height
width = image.width
pixels = image.getPixels()
vProjection = new int[height]
for y = 0 < height
    currentCount = 0
    for x = 0 < width
        pix = pixels[x + y * width];
        if(pix == WHITE)
        currentCount++
     vProjection[y] = currentCount
return vProjection
```

7.1.2 Vertical Projection

A vertical projection involves counting pixels in each column. As can be seen, the areas of interest become quite apparent in the vertical projection. These groups of clusters show the locations of each character.

Figure 24 Example of a vertical projection

The following pseudocode can be applied:

```
height = image.height
width = image.width
pixels = image.getPixels()
vProjection = new int[width]
for x = 0 < width
    currentCount = 0
    for y = 0 < height
        pix = pixels[x + y * width];
        if(pix == WHITE)
        currentCount++
     vProjection[x] = currentCount
```

7.1.3 Masking Horizontal and Vertical Projections

Once the regions are extracted from the two projection methods, they can be overlaid and their intersections are noted. These areas of intersection would map to the individual characters.

Below, a bounding box is created for the area containing all the white pixels in the image.

Figure 25 The area of interest in the horizontal projection

Below, a bounding box is created for each of the areas containing all the white pixels in the image. This step is a bit more complex than the previous, as the number of characters appearing is unknown.

Figure 26 The areas of interest in a vertical projection

Combing the two images bounding boxes, we can create a mask, where their overlapping should contain the characters.

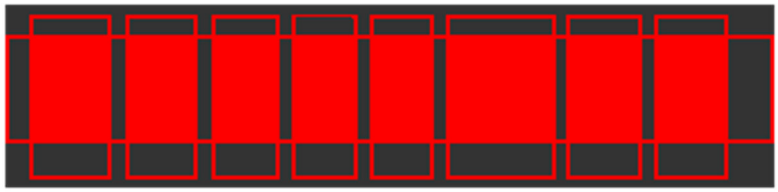


Figure 27 Overlaying the two projections

By applying the masks in the shaded areas to the original image, the following result is produced.

Figure 28 Masking with original image

It should be noted that in both cases, the example image used contains no noise. All the white pixels in the image belong to a character. In a real-life example, the image may be contaminated with noise, which makes finding the detail a lot harder. This can be observed in the images below.

Figure 29 Horizontal Projection

Figure 30 Vertical Projection

As can be seen, the horizontal projection picks up the boarders of the plate and the projections in the centre are very close together making locating the areas of interest extremely difficult. With the vertical projection, the result is not as bad, however it is picking up the blue tag with *IRL* writing on it. Looking at the projection of the letters *L* and *S*, it is difficult to see where the *L* finished and the *S* starts.

7.2 Contours and Bounding Boxes

Another approach to character segmentation is to locate all the shapes in an image, and determine which of them hold various characteristics of a potential character. Once each shape has been identified, a bounding box could be drawn around it, and Its properties such as aspect ratio and size could be used to determine if it holds the various characteristics of a potential character.

7.2.1 Contours

"Contours can be explained simply as a curve joining all the continuous points (along the boundary), having same color[sic] or intensity. The contours are a useful tool for shape analysis and object detection and recognition." [22]

The idea is similar to edge detection, however contours are often obtained from edges and they are aimed at being *object contours*. Thus, they need to be closed curves. Below is an example of this method.

Figure 31 Finding contours in an image

As can be seen, all the shapes have been located. Each contour is stored as a separate list containing all the points of the contour. From these points, a rectangle can be constructed around each shape, by using its leftmost, topmost, rightmost, and bottommost points.

7.2.2 Bounding Boxes

A bounding box is simply a box drawn around an object which contains the entire object. It can be represented as a rectangle with an x and y coordinate, as well as a width and height. A common occurrence of this is in 2D video games where it is used for collision detection, as shown below.

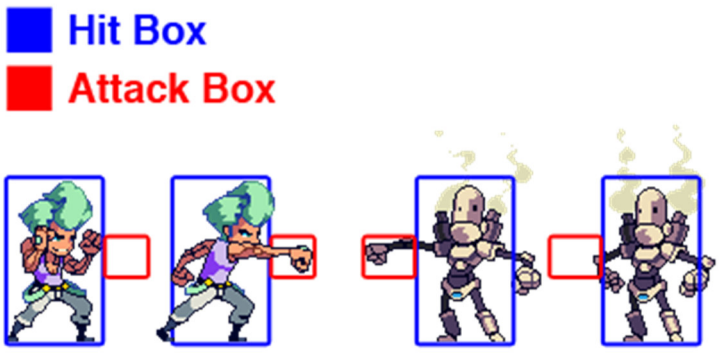


Figure 32 Bounding boxes in a 2D game [23]

Taking this and applying the technique to the previous contour image, produces the following result.

Figure 33 Bounding boxes of various objects

As can be seen, each shape in the image has been detected and a bounding box drawn around it. Now if we apply this step to the licence plate, we produce the following result.

Figure 34 Binding boxes of a licence plate

An interesting point to note is that some characters, such as the *9* and *8* which have holes in them, contain a binding box within a binding box, which is undesirable as we are only be interested in the outer binding box. By checking for this case, a much nicer result is produce.

Figure 35 Refined binding boxes of a licence plate

This results can be refined one step further. If we consider that the bounding box of a character is taller than it is wide, we can discount any which don't meet this property. If we also consider than the character will have a minimum height of roughly 50% of the image, we can discount those further. By taking these two points into account, the final more refined result is produced.

Figure 36 Filtering the binding boxes

It should be noted, that since the bounding boxes are filtered out using their dimension, any noise that meets those requirements will be considered a valid character. An issue to overcome will be deciding dimensional requirements which work among a large variety of images.

8 Optical Character Recognition

8.1 Tess Two

A fork of Tesseract Tools for Android (tesseract-android-tools) that adds some additional functions. Tesseract Tools for Android is a set of Android APIs and build files for the Tesseract OCR and Leptonica image processing libraries.

This project works with Tesseract v3.04.00 and Leptonica v1.72. The required source code for Tesseract and Leptonica is included within the tess-two/jni folder.

The tess-two subdirectory contains tools for compiling the Tesseract and Leptonica libraries for use on the Android platform. It contains an Android library project that provides a Java API for accessing natively-compiled Tesseract and Leptonica APIs.

The eyes-two subdirectory contains a second, separate library project with additional image processing code copied from the eyes-free project. It includes native functions for text detection, blur detection, optical flow detection, and thresholding. Building eyes-two is not necessary for using the Tesseract or Leptonica APIs. [24][25]

8.1.1 Issues

TessTwo requires a training file, which it uses when processing an image into text. There is a training file available for free, supplied by google. The main issue here is that it expects an image in perfect quality. No shadows, blurs, the image isn't skewed, and that the font used is neat, and of a similar size.

A possible issue is that TessTwo expects fonts with serifs:

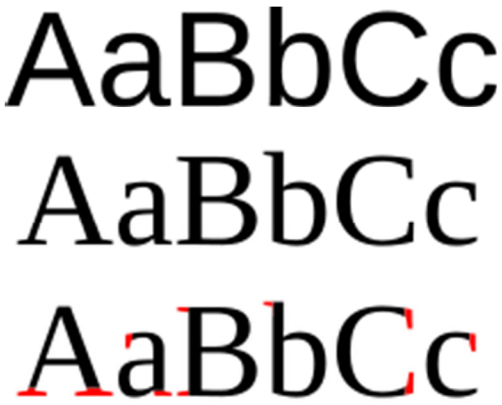


Figure 37 Sans Serif Vs Serif

The litany of issues and potential time wastage suggests that this is not the way forward, and as such other methods of character recognition will be investigated.

8.2 OpenCV Template Matching

Template matching is a technique for finding areas of an image that match (are similar) to a template image (patch).

To achieve this, two components are needed.

1. The source image
2. The template image.

The theory is rather simple; go through each image pixel by pixel and calculate how well they match.

OpenCV provides a variety of algorithms to perform template matching with. Each of these may be trialled and tested to see which provides the best results. They are listed below. [26]

1. CV_TM_SQDIFF

$$R(x,y) = \sum_{x',y'} (T(x',y') - I(x+x',y+y'))^2$$

2. CV_TM_SQDIFF_NORMED

$$R(x,y) = \frac{\sum_{x',y'} (T(x',y') - I(x+x',y+y'))^2}{\sqrt{\sum_{x',y'} T(x',y')^2 \cdot \sum_{x',y'} I(x+x',y+y')^2}}$$

3. CV_TM_CCORR

$$R(x,y) = \sum_{x',y'} (T(x',y') \cdot I(x+x',y+y'))$$

4. CV_TM_CCORR_NORMED

$$R(x,y) = \frac{\sum_{x',y'} (T(x',y') \cdot I(x+x',y+y'))}{\sqrt{\sum_{x',y'} T(x',y')^2 \cdot \sum_{x',y'} I(x+x',y+y')^2}}$$

5. CV_TM_CCOEFF

$$R(x,y) = \sum_{x',y'} (T'(x',y') \cdot I(x+x',y+y'))$$

where

$$T'(x',y') = T(x',y') - 1/(w \cdot h) \cdot \sum_{x'',y''} T(x'',y'')$$
$$I'(x+x',y+y') = I(x+x',y+y') - 1/(w \cdot h) \cdot \sum_{x'',y''} I(x+x'',y+y'')$$

6. CV_TM_CCOEFF_NORMED

$$R(x,y) = \frac{\sum_{x',y'} (T'(x',y') \cdot I'(x+x',y+y'))}{\sqrt{\sum_{x',y'} T'(x',y')^2 \cdot \sum_{x',y'} I'(x+x',y+y')^2}}$$

Typically, in OpenCV template matching, we would want to find where the template image is located in the source image. However, we are only interested in how well the two images match. Luckily, OpenCV provides a way to check this, using global minimums and global maximums. The process here would involve having a number of template images for each character, and matching them all against the character and deterring which image matches the best. This task may be more suited to running on the cloud, as a huge bank of templates could be stored in order to improve accuracy and efficiency.

9 Android

Android applications consist of classes called an Activity. “An activity is a single, focused thing that the user can do.” [27] An activity follows a lifecycle, which defines the methods called automatically at various stages. Figure 15 below outlines this life cycle.

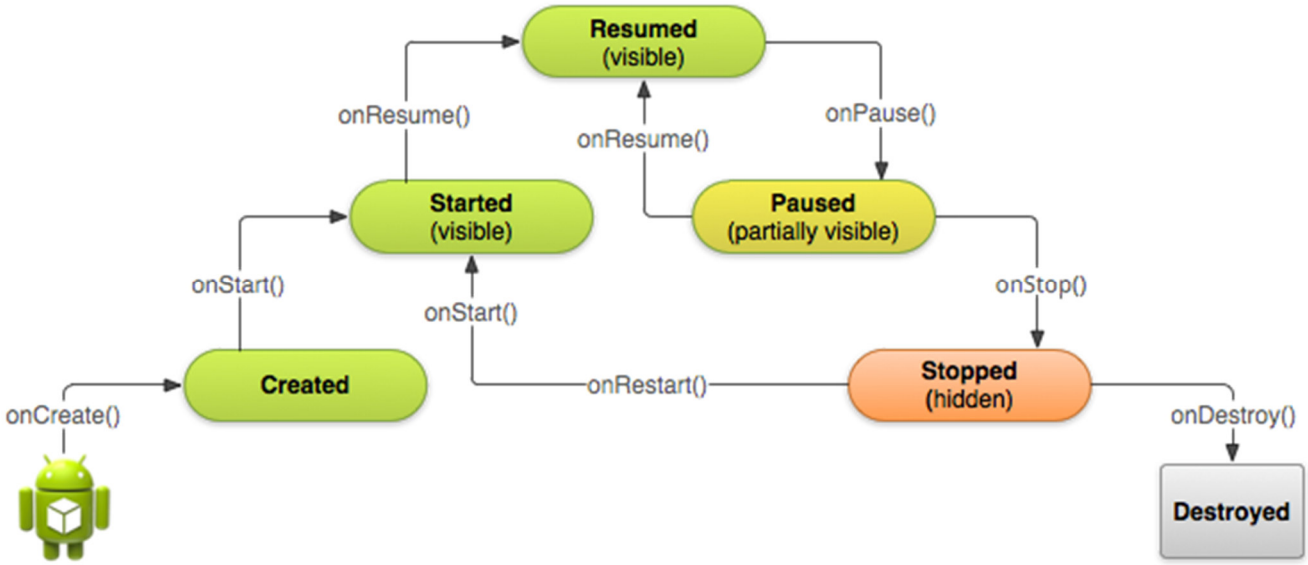


Figure 38 Activity Lifecycle [28]

Android manifest file defines all activities, the main activity, and any permissions / features the app requires.

All activities in an android application must be declared in the manifest file of the application. This manifest is also where you define any permissions the application requires, such as reading/writing to storage, and any features the application requires, such as the device camera, GPS, etc.

In order to use a devices camera, the application activity needs to open and release it as needed. It is important that an activity releases the camera when it no longer needs it, as another activity or application may need to access it. In order to do this correctly, it is important to understand the lifecycle of an Android activity. When an activity starts, the onResumed method will always be called. Likewise, when an activity ends, the onPause method will always be called. Both of these methods are also called when the activity loses focus, such as when another application starts without the main application ending. As the diagram in figure 15 shows, the onPause and onResume methods are the perfect places to access and release the camera. It is also important that to handle the case where the application has a lock on the device camera and then crashes. If the camera is not released, then the camera will be under the control of an app that has crashed and might no longer be open. Even if the app is restarted, the camera still believes it is opened by something else. This situation would mean that the device would have to be restarted to unlock the camera.

10 Android Camera

When an application wishes to take pictures to use within it, it has 2 option:

- Use native camera app
- Write your own camera app

A very common approach when an application wishes to take a picture is to launch the native camera application form within your application, allow it to handle taking and saving the image, and then returning the image back to your application. This approach works for most cases, but runs in to problems if the camera preview should contain some custom overlays, such as a guide box. The only way to achieve this, is write your own camera class, and customize the preview display to suit your applications needs.

As of Android API version 21, there are 2 camera packages available with android:

- android.hardware.camera
- android.hardware.camera2

The camera2 package was released in API version 21, and is a huge improvement on the previous android.hardware.camera package. With these added improvements comes added complexity

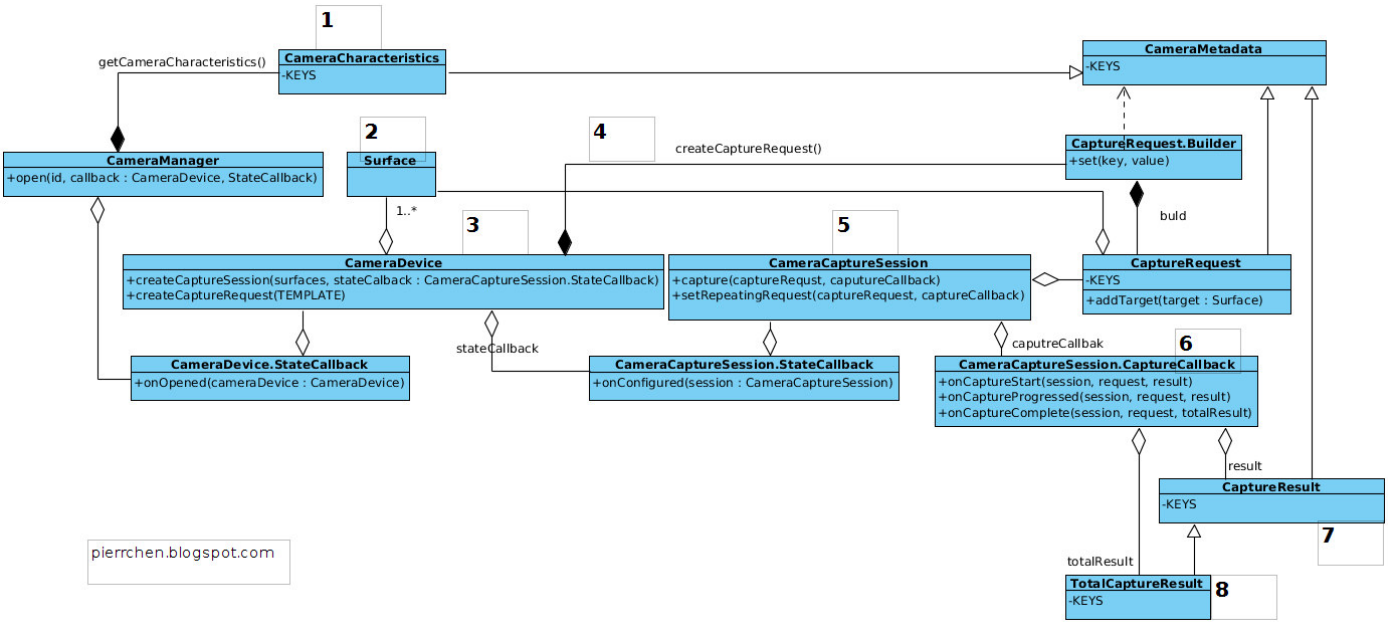


Figure 39 android.hardware.camera2 API [29]

10.1 Issues

10.1.1 Phone and Camera Orientations

Creating a camera application for Android is a complete and utter nightmare. The main source of this is when dealing with the device orientations, as phone manufacturers are not sticking to the set out standards. This is a device-specific issue that mostly affects Motorola devices. [30]

Another headache, which is actually intended, is that the natural device orientation is different to the natural camera orientation:

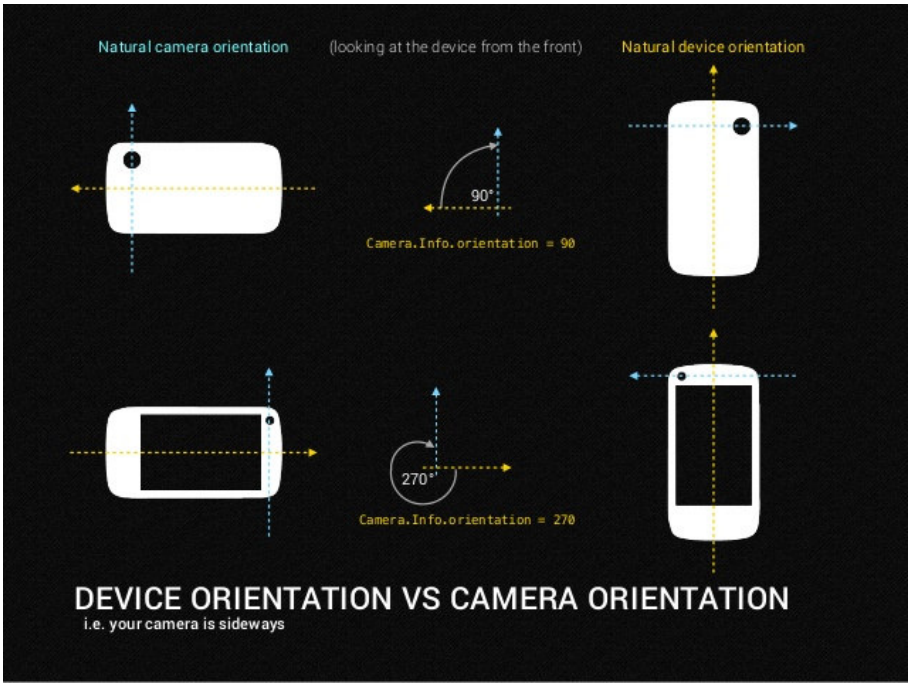


Figure 40 Device Vs Camera orientations

One thing to consider, when cropping the image from the guide box, is that the saved image size may be, and usually is, different to the screen size. This means that the coordinates of the guide box, won't map over to the saved image properly, as shown below.

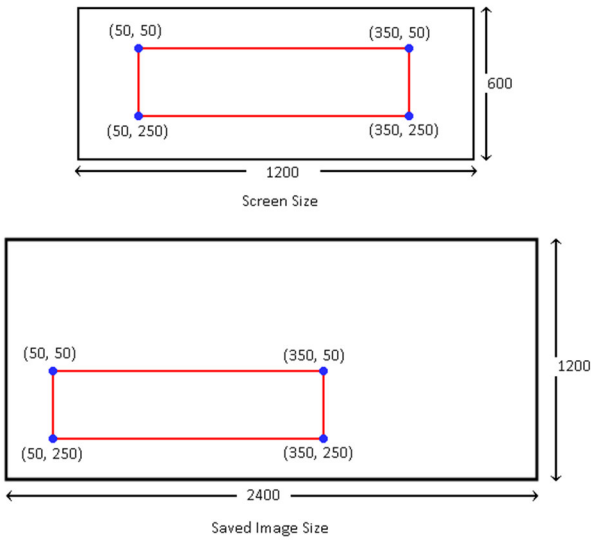


Figure 41 Incorrect mapping of guide box from screen to saved image

To compensate for this, the ratio of the screen and image size must be calculated, and the guide box scaled accordingly:

```
widthRatio = image.width / screen.width
heightRatio = image.height / screen.height
cropX = guidebox.x * widthRatio
cropY = guidebox.y * heightRatio
cropWidth = guidebox.width * widthRatio
cropHeight= guidebox.height * heightRatio
```

By applying this, ideally we end up with the following situation:

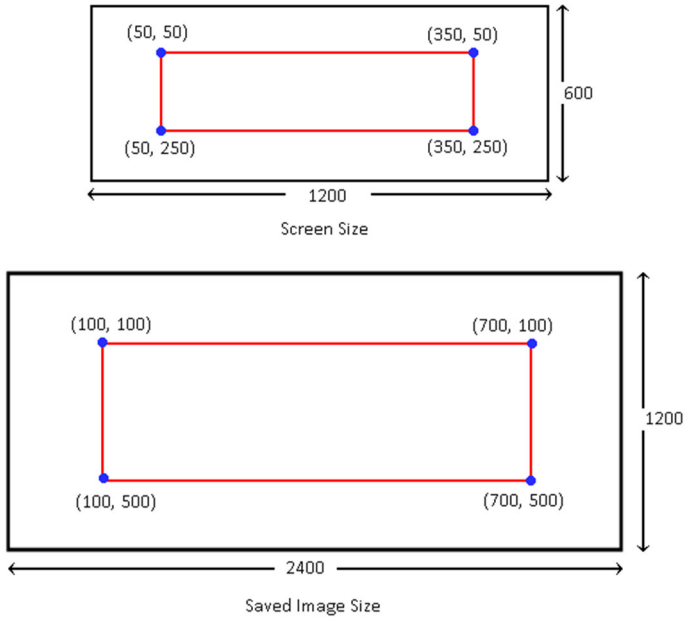


Figure 42 43 Incorrect mapping of guide box from screen to saved image

11 OpenCV Camera

OpenCV provides its own mechanisms to access the device camera. It is not compatible with the new android.hardware.camera2 package. This is due to the latest OpenCV for Android being a version or 2 behind the latest Android API. It is expected to become compatible, but for this project it is too risky to rely on it.

OpenCV for Android does not support the ability to take a picture. By design, OpenCV is intended more for real-time video processing. There are a couple of ways to get around this, but they involve workaround “hacks”.

Since the application will be processing photos, there is no real need to use the OpenCV components. A picture can be taken using the android.hardware.camera package, and then passed over to OpenCV to do some processing, and then handed back over if needed.

OpenCV uses a data type Mat to store images. These are stored as a matrix, hence the class name. Android uses a data type Bitmap to store images. In order to pass an image captured by Android to OpenCV, a quick conversion from one type to the other is needed. OpenCV supplies an Utils class, which amongst other things, will convert a Bitmap to Mat. The only prerequisite is that a Mat has been created with the same width and height as the Bitmap.

12 References

[1] https://www.android.com/

[2] http://www.apple.com/ie/ios/

[3] https://www.microsoft.com/en-us/windows/phones

[4] http://opencv.org/about.html

[5] http://tutorial.simplecv.org/en/latest/

[6] http://www.tutorialspoint.com/restful/restful_introduction.htm

[7] http://www.tutorialspoint.com/restful/restful_introduction.htm

[8] http://flask-restful-cn.readthedocs.org/en/0.3.4

[9] https://www.mysql.com/about/

[10] http://homepages.inf.ed.ac.uk/rbf/HIPR2/mean.htm

[11] Shapiro, L.G., & Stockman, G. C. (2001). In Computer Vision (pp. 137, 149). Prentice Hall

[12] Shapiro, L.G., & Stockman, G. C. (2001). In Computer Vision (pp. 149). Prentice Hall

[13] http://www.johndcook.com/blog/2009/08/24/algorithms-convert-color-grayscale/

[14] http://docs.opencv.org/2.4/doc/tutorials/imgproc/histograms/histogram_equalization/histogram_equalization.html

[15] http://felixniklas.com/imageprocessing/binarization

[16] http://www.labbookpages.co.uk/software/imgProc/otsuThreshold.htm

[17] http://homepages.inf.ed.ac.uk/rbf/HIPR2/roberts.htm

[18] http://homepages.inf.ed.ac.uk/rbf/HIPR2/sobel.htm

[19] http://homepages.inf.ed.ac.uk/rbf/HIPR2/canny.htm

[20] http://homepages.inf.ed.ac.uk/rbf/HIPR2/canny.htm

[21] http://www.cse.usf.edu/~r1k/MachineVisionBook/MachineVision.files/MachineVision_Chapter2.pdf pg. 36

[22] http://docs.opencv.org/3.1.0/d4/d73/tutorial_py_contours_begin.htm

[23] https://www.raywenderlich.com/24452/how-to-make-a-side-scrolling-beat-em-up-game-like-scott-pilgrim-with-cocos2d-part-2

[24] http://androidtesstwo.blogspot.ie/2014/03/making-android-ocr-app-using-tess-two.html

[25] https://groups.google.com/forum/#!forum/tesseract-ocr

[26] http://docs.opencv.org/2.4/doc/tutorials/imgproc/histograms/template_matching/template_matching.html

[27] http://developer.android.com/reference/android/app/Activity.html

[28] http://developer.android.com/training/basics/activity-lifecycle/starting.html

[29] http://2.bp.blogspot.com/-ybsvE3klLEY/VMDMtcnOpzI/AAAAAAAAAfc/AhxyayvJRuE/s1600/Selection_056.png

[30] http://stackoverflow.com/questions/8346955/android-camera-resulted-image-should-be-rotated-after-the-capture/8347574#8347574

[31] http://www.slideshare.net/queencodemonkey/droidcon-nyc-2014buildingcustomcameraapplications-39429059