Imperial College London

Department of Electrical and Electronic Engineering

Final Year Project Interim Report 2017



Project Title: **Distributed Road Traffic Speed Monitoring**

Student: **Jeremy Chan**

CID: **00818433**

Course:  **4T**

Project Supervisor: **Dr. Ed Stott**

Second Marker:

Table of Contents

[I. Introduction 4](#_Toc473149248)

[II. Project Specification 4](#_Toc473149249)

[A. Project goals 4](#_Toc473149250)

[B. Advanced project goals 4](#_Toc473149251)

[C. Clarifications 4](#_Toc473149252)

[III. Related Work 5](#_Toc473149253)

[A. License plate recognition 5](#_Toc473149254)

[1) Hardware 5](#_Toc473149255)

[2) Software 5](#_Toc473149256)

[B. Peer to peer network 6](#_Toc473149257)

[C. Photo evidence publication 6](#_Toc473149258)

[D. Overall Feasibility 7](#_Toc473149259)

[IV. Estimation 7](#_Toc473149260)

[V. Background Reading and Literature Review 8](#_Toc473149261)

[A. License plate recognition 9](#_Toc473149262)

[B. Peer to peer network 11](#_Toc473149263)

[C. Photo evidence publication 11](#_Toc473149264)

[VI. Design 11](#_Toc473149265)

[A. License plate recognition 12](#_Toc473149266)

[1) Pre-Processing 12](#_Toc473149267)

[2) OpenALPR 13](#_Toc473149268)

[3) Post-Processing 13](#_Toc473149269)

[B. Peer to peer network 13](#_Toc473149270)

[C. Photo evidence publication 13](#_Toc473149271)

[D. Hardware 13](#_Toc473149272)

[VII. Build 14](#_Toc473149273)

[A. License plate recognition 14](#_Toc473149274)

[B. Peer to peer network 14](#_Toc473149275)

[C. Photo evidence publication 14](#_Toc473149276)

[D. Hardware 15](#_Toc473149277)

[VIII. Testing 15](#_Toc473149278)

[A. License plate recognition 15](#_Toc473149279)

[B. Peer to peer network 16](#_Toc473149280)

[C. Photo evidence publication 16](#_Toc473149281)

[D. Hardware 16](#_Toc473149282)

[IX. Risks and Fallbacks 16](#_Toc473149283)

[A. License plate recognition 16](#_Toc473149284)

[B. Peer to peer network 16](#_Toc473149285)

[C. Photo evidence publication 16](#_Toc473149286)

[D. Hardware 16](#_Toc473149287)

[X. Deployment and Maintainance 17](#_Toc473149288)

[XI. References 17](#_Toc473149289)

[XII. Appendix 18](#_Toc473149290)

[A. Table of Figures 18](#_Toc473149291)

[B. Table of Tables 19](#_Toc473149292)

[C. Gantt Chart 20](#_Toc473149293)

# Introduction

* Project motivation
  + Lots of traffic speeding (ref the freedom of info report)
  + Find some info on lethality of speeding
  + Estimate on non motorway speeding?
* Project aim
* Accenture delivery method?

# Project Specification

From the project description, the goals of the project can be separated into the following:

## Project goals

* Implement a number plate recognition system using existing computer vision algorithms on a low-cost, readily available hardware platform.
* Set up a peer-to-peer network to share vehicle passing times and detect violations without the need for a central server.
* Publish photo evidence of any violations

## Advanced project goals

* Use the changes in the number plate geometry as the vehicle passes to detect the instantaneous speed of a vehicle. This provides a stand-alone mode that will aid adoption in areas where there isn't already an established network.
* Implement automatic peer discovery so that each device can find its neighbours and calculate the minimum legal transit time between them using a public mapping database.
* Add an encryption layer so that a hacker or rogue peer cannot use the network to track the movements of law-abiding vehicles.
* Package the system so that it can be easily installed in a home by an inexperienced user.

## Clarifications

After discussions with Dr. Stott, the following clarifications were made:

1. The system should target license plates and deployment in the United Kingdom.
2. The number plate recognition system should be targeted at an off the shelf package, so there should be minimal setup and calibration done. This also means anyone, with the right equipment, should be able to download and compile the system if they have existing hardware.
3. The low-cost, readily available hardware platform will be a Raspberry Pi (RPi), with a camera attached to it. Using an RPi combines the best of cost (~£40 at time of writing), power (quad core CPU [1]), flexibility (camera can be any USB webcam or RPi’s official cameras), and support (development work on the RPi is extensive and there are ample tutorials/information online).
4. The peer to peer network should ideally be fully decentralised, so the system should be able to find peers without the help of a central server.
5. Publishing photo evidence will most likely be done onto a social network.
6. The public mapping database will be one accessible to most people – Google Maps API.
   1. https://developers.google.com/maps/documentation/roads/speed-limits
7. A hacker or rogue peer should not be able to extract license plates from the system remotely, nor should they be able to spoof detect license plates as another user to avoid framing an innocent driver.
8. If possible, there should be a whitelist of emergency vehicles in the case of there being an emergency vehicle over the speed limit.
9. The final package should be a software package uploaded onto GitHub, with clear instructions on how to compile and run the program with examples. As this project involves testing on hardware as well, a list of recommended hardware should also be provided (after successful testing), so the project can be easily replicated in the future.

# Related Work

Research into already existing products was done on the three main goals of the project, to judge the market feasibility of the project, and to prevent overlap with existing work where possible.

## License plate recognition

### Hardware

There are many license plate recognition systems available on the market, with many of them being used in speed cameras around the country [2]. Traffic cameras can generally be separated into radar based and optical based systems, with radar based cameras checking the instantaneous speed of the vehicle as it moves past, and optical based systems checking the average speed of the vehicle as it passes between two points. There has been a steady increase in average speed check cameras over the years [3], mostly along motorways but also by local councils [4]. These average speed check cameras often include multiple enticing features like 24/7 operation and 4G connectivity, as such in Jenoptik’s VECTOR cameras [5].

### Software

Most of the license plate recognition software available is proprietary and closed source. Some area only available for commercial use. These systems generally have features such as video stream processing, ability to detect plates from multiple points of view, and cloud services to take in a video stream and output detected license plates along with their timestamps [6]–[8]. However, the price of the commercial systems (if available for purchase) are definitely out of budget for this project, as shown in Table 1.

As the project specification specifies that ‘existing computer vision algorithms’ should be used, the choice was made to use the consumer version of OpenALPR. The main reasons for this was that it was open source, and very feature rich. Additionally, a choice was made to instead replicate the commercial features of OpenALPR using computer vision processing libraries instead of ignoring the extra features or paying for them as they were standard vision processing techniques, e.g. background subtraction, motion detection.

Table : Prices of different license plate detection systems

|  |  |  |
| --- | --- | --- |
| System | Price (one-off) | Price (per month) |
| ARES |  |  |
| DTK LDR SDK | 190 – 1430 EUR | N/A |
| OpenALPR | 1000 USD | 50 USD |

## Peer to peer network

Peer to peer networks (P2P) have existed for a long time and there are numerous implementations available for use on the Internet. The main uses for peer to peer networks are web, messaging, and file sharing.

Bitmessage (<https://bitmessage.org>) is used for encrypted messaging, and is decentralised. It also uses strong authentication to prevent spoofing of the sender of a message [9]; however, it removes information about the sender and receiver. It is therefore not suitable for this project as this system needs to know information about the sender to accurately work out an average speed.

Telehash (<https://github.com/telehash/telehash.github.io>) is open source, fully end-to-end encrypted, enforces strict privacy rules, and cross platform. It also supports JSON message sending with unique sender and receiver ID’s. Unfortunately, the development is focused around using Node.JS and not Python or C++ (the Python binding repository is still empty at time of writing). Therefore, it is not suitable as it would take too much time to implement.

## Photo evidence publication

There are official API’s for most social networks on uploading images to their respective social networks so there is no need to reinvent the wheel, no is there a need to use a third party service to upload images.

## Overall Feasibility

Overall, the project is very feasible as there has been a lot of work in all three areas, proving that the ideas in the project are not a dead end. Moreover, there is a free and open source implementation for the license plate recognition system which is immensely useful in providing a solid head start in license plate detection, a primary goal of the project. As most advanced features are behind a commercial paywall, a novel aspect of the project is to implement new algorithms / research to better improve the detection rate of the license plate detection.

The network is also a very feasible part of the project. P2P networks have extensive research and Python has libraries which support development using web sockets and different communication protocols. However, since the license plate detection is done using existing libraries, there has to be extra emphasis placed on the networking side to make sure this is the primary novel aspect to the project. At the time of writing, there has not been any projects tying license plate detection to a decentralised network which seeks out peers in its immediate vicinity.

Lastly, there are extensive documentation on uploading images to social networks – no foreseeable problem there unless the API access is revoked.

# Estimation

As the project timeline spans the entire duration of the 4th year, a plan to map out actions and results with reasonable time estimates is needed. This plan will be used to track progress throughout the project. However, given the open-ended nature of this design and build project beyond the defined project goals, there is a lot of design, build, and testing to be done. Hence, the estimates for design, build, and testing will inevitably overlap by a fair margin, thereby inflating the number of days to complete the project. The following table (Table 1) shows an estimate for each stage of project delivery, in days. A buffer is also included for unforeseen project issues that may push the timeline back. The corresponding Gantt chart is shown in section XI.B.

Table : Estimation of time needed for each task in the project

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Time needed (days) | Buffer (days) | Actual time used (days) |
| Initial research about topic before first meeting | 2 | 0.5 | 2.5 |
| Clarify project aims and goals | 1 | 0.5 | 1 |
| Prioritise project goals | 1 | 0.5 | 1 |
| Define project requirements | 1 | 0.5 | 1.5 |
| High level design of license plate recognition | 5 | 2 |  |
| High level design of P2P network | 5 | 2 |  |
| High level design of security issues, encryption | 5 | 2 |  |
| Coding of license plate recognition | 10 | 5 |  |
| Coding of P2P network | 10 | 5 |  |
| Coding of privacy issues, encryption | 10 | 5 |  |
| Testing of license plate recognition | 5 | 2 |  |
| Testing of P2P network | 5 | 2 |  |
| Testing of privacy issues, encryption | 5 | 2 |  |
| Integration of all systems, along with possible hardware | 10 | 2 |  |
| Packaging and release on GitHub | 2 | 0.5 |  |
| Documentation of code | 2 | 1 |  |
| Report writing | 10 | 5 |  |
| Making of demo | 5 | 2 |  |
| TOTAL | 99 | 39.5 | 6 |

To track the progress of the project, a project management software was used. There available for free use. Asana (<https://asana.com/>) was previously used in the 3rd year project to great effect. However, as this project is not collaborative, the project pane of GitHub was used as a simple tracker (Figure 1). Four groups of tasks were made; the GitHub project tracker allows for easy moving of tasks from one group to another. Another advantage of using the GitHub project tracker is that everything regarding the project is in one place, accessible and modifiable from anywhere.

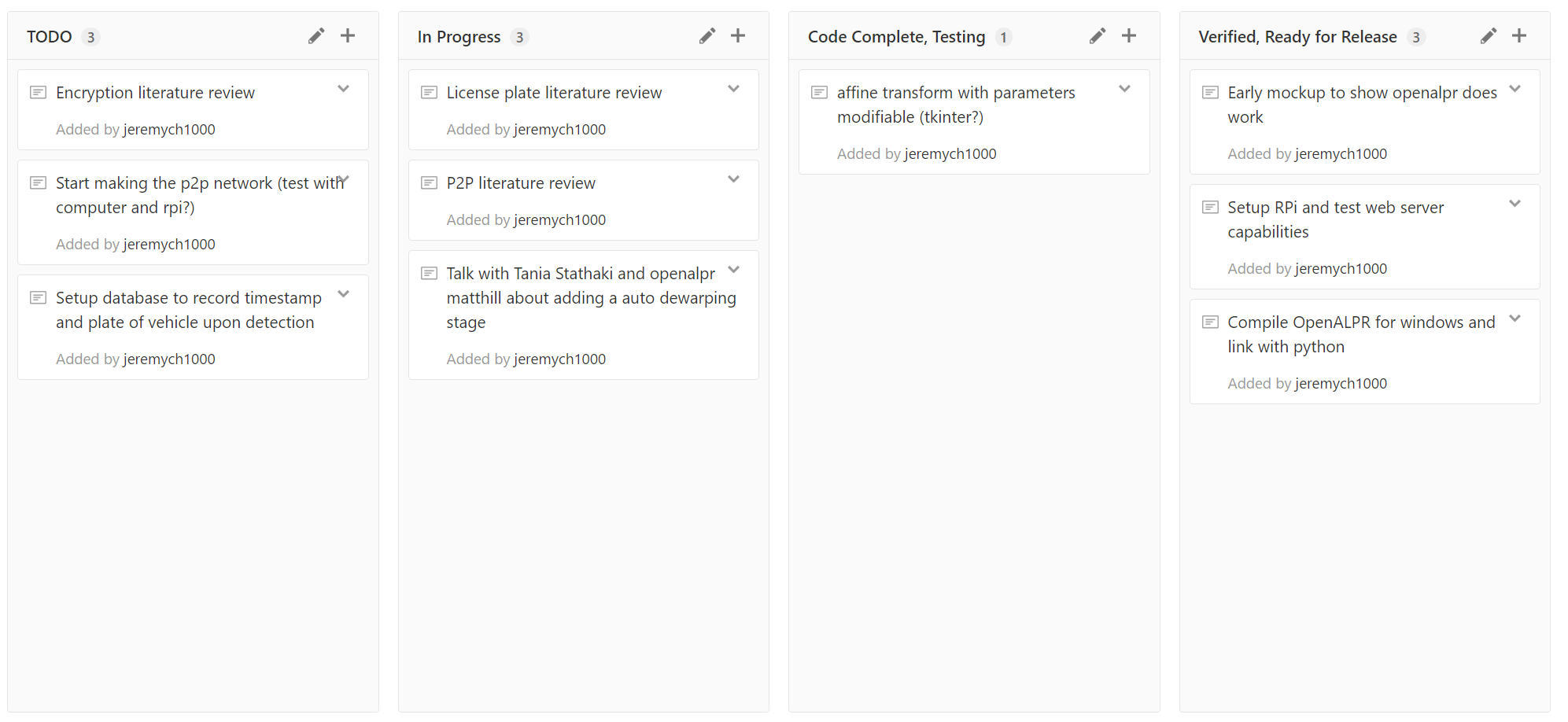


Figure : GitHub project tracker

# Background Reading and Literature Review

## License plate recognition

Most license plate recognition systems operate on a similar basis. Pre-processing, detection, and character recognition [10]. However, image processing and detection can used interchangeably out of order, as in the case of OpenALPR [11].

In the pre-processing stage, a combination of techniques can be used to make the image more recognisable to the following pipeline stages. Azad et al. [12] utilises the HSV colour space instead of the RGB colour space to better determine the location of the plate as the plate is assumed to be of a certain colour. Some implementations implement both the RGB and HSV colour space to get saturation and intensity maps [13]. Duan et Al. [14] first converts the image to greyscale, then normalises the histogram to perform histogram equalisation to get a picture with better contrast. This is especially important during the night when there are a lot of dark areas from shadows. An example is shown below in Figure 2.

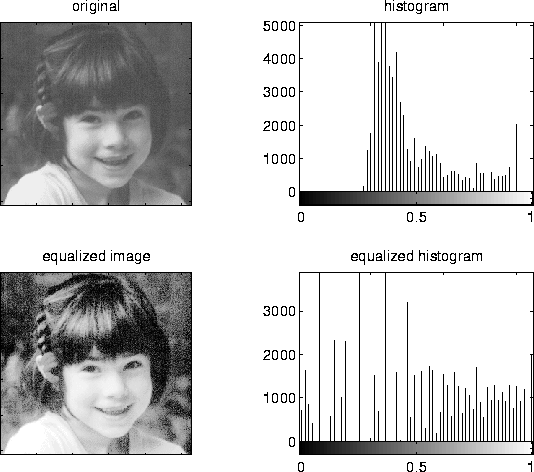


Figure : Histogram Equalisation (<http://electronicsinourhands.blogspot.co.uk/2012/10/histogram-equalization-in-image.html>)

In the detection phase, several algorithms are prominent. Edge detection is used extensively in Badr et al. [15], where the Sobel edge detection method is used in conjunction with thresholding techniques to determine the plate’s location in areas with high vertical lines. However, these methods often fail if the assumption that the license plate is captured from a fixed face-on angle is false as the plate can no longer be guaranteed to have perfectly vertical lines. Moreover, using edge detection as the first step often brings false positives, where buildings and road signs can also exhibit perfectly vertical lines – however they are not an area of interest for the license plate detection system.

Hence, other implementations such as OpenALPR’s and Liu et al. [16] use an algorithm called Local Binary Patterns (LBP), generally used in face recognition, to get areas of the image where it thinks there is a license plate. However, since LBP uses a sliding window approach, the performance of the detection is quite CPU dependent [11] (can be accelerated using a CUDA but the RPi does not support this). This method has been proven by Nguyen and Nguyen [17], who successfully used LBP with extra classifiers and algorithms in a real time license plate detector.

Edge detection is not unused, however. The most often used combination is a mixture of edge detection algorithms (Sobel, Laplacian, Canny), the Hough transform, and the contour algorithm. The Hough transform is used in implementations like [14] and OpenALPR [11] to find the actual edges of the plate as detection only gives a rough area for the location of the plate, and may end up with an area that does not have a plate in, or slightly bigger than the plate.

Other implementations such as Kwaśnicka and Wawrzyniak [18], and Chang et al. [13] use a technique called connected component analysis before applying the Hough transform. This technique is applied to a binary image of the license plate, and rejects any connected components whose aspect ratio is outside a prescribed range (a license plate is rectangular). The reliability of each implementation has yet to be tested against each other, however.

Lastly, the plate area needs to be warped to be a rectangle for the Optical Character Recognition (OCR) algorithms to work their best. A perspective transformation is often used to warp the image – the final image looks like it has been taken from another perspective (Figure 3). Moreover, straight lines remain straight [19] after the transformation so the next stage, character recognition, is not compromised. OCR libraries such as tesseract (<https://github.com/tesseract-ocr>) are then used to get the alphanumeric characters from the plate.

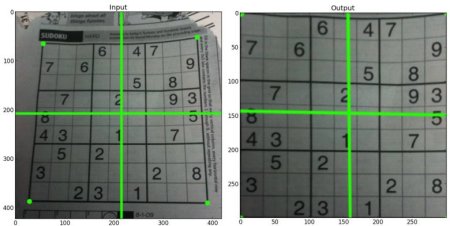


Figure : Using a perspective transformation [19]

## Peer to peer network

A P2P network is a network where many machines connect in an ad-hoc manner (they do not join and leave the network based on a schedule), and serve as a server for others. This means there is no need for files and messages or whatever the network is serving to pass through a central server, providing no one single point of failure.

P2P networks account for a significant portion of the web’s traffic [20], but its widespread nature means it comes with a few downsides as well. Making sure all copies of a file is free from corruption is one big challenge, as well as security. As all machines are servers of their own, a malicious hacker may attempt to distribute a virus or malware over the network to the many users using the network.

In this project, the main area of P2P networks that will be examined are decentralised networks. Centralised networks still have a central server that does routing, and peer finding, among other tasks [21], whereas decentralised networks do not.

Peer finding in a decentralised network is a challenge. As there is no central server, decentralised networks use techniques such as flooding the network with discover requests [21] and randomly pinging IP’s around the world to see if they are part of the network [22]. A well-known decentralised network, Gnutella, uses another technique where it finds new peers by expanding its search radius in its broadcasting phase. When a peer accepts this connection, it rebroadcasts the new peer’s address to its own peers, but decreases a counter called Time-to-Live (TTL) to ensure the message eventually dies out (when TTL=0) [23].

However, the broadcasting strategy in decentralised networks means as the search radius increases, there is an increasing amount of traffic in the network, and this may cause congestion in the network [23].

Other types of peer-finding techniques can be found in Mastroianni et al. [24].

## Photo evidence publication

Image upload API’s exist for many social networks, and the following table shows the documentation link for three popular image sharing sites.

|  |  |
| --- | --- |
| Network | Link |
| Facebook | <https://developers.facebook.com/docs/php/howto/uploadphoto> |
| Twitter | <https://dev.twitter.com/rest/reference/post/media/upload> |
| Imgur | <https://api.imgur.com/endpoints/image> |

# Design

## License plate recognition

As the project specification mentions using existing libraries, OpenALPR was chosen for its open source release and its usability in having Python links to the C code. Moreover, the developer was active on GitHub in responding in issues and queries, further cementing OpenALPR as the top choice for the project.

The license plate recognition system consists of the following stages: pre-processing, OpenALPR license plate detection, and post-processing.

### Pre-Processing

From the literature review, beneficial pre-processing steps include contrast normalisation and perspective transform. As using HSV thresholding and extracting license plate colours such as white and orange interfere with OpenALPR’s LBP algorithm (<https://github.com/openalpr/openalpr/issues/442>), HSV thresholding was not used. However, HSV thresholding remains a candidate for more pre-processing in the future as it works quite well, as demonstrated in Figure 4.

|  |  |
| --- | --- |
| F:\Python Work\Skew2\test_files\zipcar.jpg | D:\OneDrive\Pictures\Screenshots\2017-01-20 (1).png |

Figure : HSV Thresholding

OpenCV, the preferred computer vision library for this project, has a perspective transform. This transform takes in a set of coordinates of the input image, followed by where those coordinates map to in the output image. Ideally in the final product, the pre-processing system intelligently picks out the car using motion detection or library image recognition techniques, and comes up with these coordinates automatically, as opposed to having to define them by hand.

OpenALPR does not support automatic perspective transform – it has a utility to do pre-processing perspective transforms but again this is manual. This utility will be replicated in Python as well.

Last but not least is motion detection. Several simple motion detection techniques are used, like taking the absolute different between sum of squares of pixels for consecutive frames to see areas of motion, along with using dilation techniques to remove holes in the difference map from the sum of squares. However, currently this technique assumes that the first frame is the background, which is not a very sustainable assumption. There are more complicated motion detection techniques and these are all planned to be implemented.

### OpenALPR

The pre-processed image is then fed through OpenALPR, with suitable parameters defined such as country, region, max amount of results to return, etc. OpenALPR is also configured to return a JSON output, so Python can access the diferent members of the output array easily, for example, timestamp, confidence level, the actual plate, and the coordinates of such plates.

As OpenALPR is a complete library, there is not much to be modified in this section.

### Post-Processing

Post-processing includes replicating a commercially available feature of OpenALPR. Specifically, as OpenALPR processes video frames, a Python script will keep track of how many times a specific plate has occurred, and also take into account the confidence level to return a one plate for a period of video instead of multiple repeats of the same plate.

## Peer to peer network

Since the amount of users for this project’s P2P network is unknown, nor is it meant to be in the millions, bootstrapping the network using brute-force broadcast, or random walking IP’s is unlikely to result in finding a peer. Therefore, the proposed method of bootstrapping the network is to use a list of known hosts, along with a, or several, bootstrapping servers which hold knowledge of the IP addresses of current peers. Using several servers will provide multiple points of failure instead of one as that would be detrimental to a P2P network.

The actual sending of the plates through the network is still under consideration, but an initial direction would be to send them in JSON form, and store them on a local database on each device (RPi). This could be done using MySQL databases, which would provide a nice link to a web interface using Python and Django if so necessary.

## Photo evidence publication

Several scripts will be written to handle image upload. Pictures can be tagged with a location, and the plate data can be sent over HTTP or HTTPS using multipart/form-data.

## Hardware

The hardware package will target a Raspberry Pi running Raspian, a fork of Debian. The software package will contain directions on how to compile the system, as well as any dependencies (OpenALPR, OpenCV, Tesseract, Python, etc). A short wiki can also be written to put on GitHub pages or otherwise.

# Build

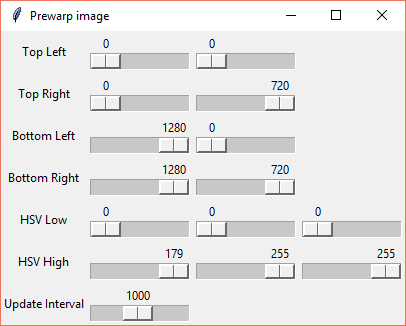
Development was done on a PC running Windows 10. The IDE is PyCharm from JetBrains. Python 3 is installed using Miniconda. OpenCV and OpenALPR are compiled from their GitHub repositories. OpenALPR’s dependencies, leptonica, and tessearact were compiled from their GitHub repositories as well.

Some web server work was done on the RPi using a LAMP stack. Similar packages to the PC are installed there to provide two testing environments.

GitHub is used to sync code across devices. SSH is used to access the Raspberry Pi.

## License plate recognition

TkInter, a Python GUI framework, is used to create simple GUI to change parameters of the pre-processing.



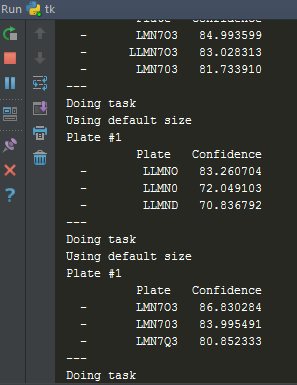
## Peer to peer network

## Photo evidence publication

## Hardware

# Testing

## License plate recognition





## Peer to peer network

## Photo evidence publication

## Hardware

# Risks and Fallbacks

## License plate recognition

## Peer to peer network

## Photo evidence publication

## Hardware

# Deployment and Maintainance

# References

[1] ‘Raspberry Pi 3 Model B’, *Raspberry Pi*. .

[2] ‘The UK’s Speed Camera Types | Fixed and Mobile speed cameras explained’. [Online]. Available: https://www.speedcamerasuk.com/speed-camera-types.htm. [Accessed: 23-Jan-2017].

[3] ‘Number of average speed cameras have doubled in three years’, *This is Money*, 31-May-2016. [Online]. Available: http://www.thisismoney.co.uk/money/cars/article-3617584/Number-average-speed-cameras-doubled-three-years.html. [Accessed: 23-Jan-2017].

[4] Y. D. Silva, ‘Average speed enforcement (ASE) camera’. [Online]. Available: https://www.birmingham.gov.uk/info/20163/road\_safety/364/average\_speed\_enforcement\_ase\_camera. [Accessed: 23-Jan-2017].

[5] ‘specs3\_vector\_v1.1\_final.pdf’. [Online]. Available: http://www.jenoptik.co.uk/sites/vysionics.vmdrupal04.lablateral.com/files/specs3\_vector\_v1.1\_final.pdf. [Accessed: 23-Jan-2017].

[6] ‘ARES | Fixed ALPR’, *PlateSmart*, 18-Jun-2015. .

[7] ‘LPR/ANPR License Plate Recognition SDK’. [Online]. Available: http://www.dtksoft.com/dtkanpr.php. [Accessed: 23-Jan-2017].

[8] ‘OpenALPR Features’. [Online]. Available: http://www.openalpr.com/features.html. [Accessed: 23-Jan-2017].

[9] J. Warren, ‘Bitmessage: A peer-to-peer message authentication and delivery system’, *White Pap. 27 Novemb. 2012 Httpsbitmessage Orgbitmessage Pdf*, 2012.

[10] V. Sharma, P. Mathpal, and A. Kaushik, ‘Automatic license plate recognition using optical character recognition and template matching on yellow color license plate’, *Int. J. Innov. Res. Sci. Eng. Technol.*, vol. 3, no. 5, 2014.

[11] ‘Accuracy Improvements — openalpr 2.2.0 documentation’. [Online]. Available: http://doc.openalpr.com/accuracy\_improvements.html#openalpr-design. [Accessed: 24-Jan-2017].

[12] R. Azad, F. Davami, and B. Azad, ‘A novel and robust method for automatic license plate recognition system based on pattern recognition’, *Adv. Comput. Sci. Int. J.*, vol. 2, no. 3, pp. 64–70, 2013.

[13] S.-L. Chang, L.-S. Chen, Y.-C. Chung, and S.-W. Chen, ‘Automatic License Plate Recognition’, *IEEE Trans. Intell. Transp. Syst.*, vol. 5, no. 1, pp. 42–53, Mar. 2004.

[14] T. D. Duan, T. H. Du, T. V. Phuoc, and N. V. Hoang, ‘Building an automatic vehicle license plate recognition system’, in *Proc. Int. Conf. Comput. Sci. RIVF*, 2005, pp. 59–63.

[15] A. Badr, M. M. Abdelwahab, A. M. Thabet, and A. M. Abdelsadek, ‘Automatic number plate recognition system’, *Ann. Univ. Craiova-Math. Comput. Sci. Ser.*, vol. 38, no. 1, pp. 62–71, 2011.

[16] L. Liu, H. Zhang, A. Feng, X. Wan, and J. Guo, ‘Simplified Local Binary Pattern Descriptor for Character Recognition of Vehicle License Plate’, in *Imaging and Visualization 2010 Seventh International Conference on Computer Graphics*, 2010, pp. 157–161.

[17] T.-T. Nguyen and T. T. Nguyen, ‘A real time license plate detection system based on boosting learning algorithm’, in *Image and Signal Processing (CISP), 2012 5th International Congress on*, 2012, pp. 819–823.

[18] H. Kwaśnicka and B. Wawrzyniak, ‘License plate localization and recognition in camera pictures’, in *3rd Symposium on Methods of Artificial Intelligence*, 2002, pp. 243–246.

[19] ‘OpenCV: Geometric Transformations of Images’. [Online]. Available: http://docs.opencv.org/3.1.0/da/d6e/tutorial\_py\_geometric\_transformations.html. [Accessed: 25-Jan-2017].

[20] The Government of the Hong Kong Special Administrative Region, ‘PEER-TO-PEER NETWORK’. [Online]. Available: http://www.infosec.gov.hk/english/technical/files/peer.pdf. [Accessed: 25-Jan-2017].

[21] H. Park, R. I. Ratzin, and M. van der Schaar, ‘Peer-to-peer networksprotocols, cooperation and competition’, *Streaming Media Archit. Tech. Appl. Recent Adv.*, pp. 262–294, 2010.

[22] C. Grothoff and C. GauthierDickey, ‘Bootstrapping Peer-to-Peer Networks’. [Online]. Available: http://grothoff.org/christian/dasp2p.pdf. [Accessed: 21-Jan-2017].

[23] Q. H. Vu, M. Lupu, and B. C. Ooi, *Peer-to-Peer Computing: Principles and Applications*. Springer Science & Business Media, 2009.

[24] C. Mastroianni, D. Talia, and O. Verta, ‘Designing an information system for Grids: Comparing hierarchical, decentralized P2P and super-peer models’, *Parallel Comput.*, vol. 34, no. 10, pp. 593–611, Oct. 2008.

# Appendix

## Table of Figures

[Figure 1: GitHub project tracker 8](#_Toc473034553)

## Table of Tables

[Table 1: Prices of different license plate detection systems 6](#_Toc473034551)

[Table 2: Estimation of time needed for each task in the project 7](#_Toc473034552)

## Gantt Chart

