Unmanned Aerial Vehicle-based Automobile License Plate Recognition System for Institutional Parking Lots

Julian Dasilva
Ricardo Jimenez
Roland Schiller
and
Sanja Zivanovic Gonzalez
Department of Mathematics and Computer Science, Barry University,
Miami Shores, FL 33161, USA

ABSTRACT

Unmanned aerial vehicles (UAVs), also known as drones have many applications and they are a current trend across many industries. They can be used for delivery, sports, surveillance, professional photography, cinematography, military combat, natural disaster assistance, security, and the list grows every day. Programming opens an avenue to automate many processes of daily life and with the drone as aerial programmable eyes, security and surveillance can become more efficient and cost effective. At Barry University, parking is becoming an issue as the number of people visiting the school greatly outnumbers the convenient parking locations. This has caused a multitude of hazards in parking lots due to people illegally parking, as well as unregistered vehicles parking in reserved areas. In this paper, we explain how automated drone surveillance is utilized to detect unauthorized parking at Barry University. The automated process is incorporated into Java application and completed in three steps: collecting visual data, processing data automatically, and sending automated responses and queues to the operator of the system.

Keywords: UAV, surveillance, image processing, ALPR, java application.

1. INTRODUCTION

Technology continues to grow at an exponential pace. Vehicle automation (both land and aerial) is now a reality, as automated aircraft technology has shrunk down to a civilian level, and artificial intelligence is making leaps and bounds in multiple disciplines. Research involving unmanned aerial vehicles (UAVs), also known as drones has vastly spread across sciences both within academics and industries. From automatic fire forest monitoring [11], power line detection [13], to road segment surveillance [5], the usage is expanding.

At Barry University parking is a big issue as the amount of people visiting the school greatly outnumbers the available parking locations that one would consider convenient. This has caused a multitude of hazards in parking lots due to people illegally parking, as well as unregistered vehicles parking in reserved parking areas. The public safety department struggles to cover all of the campuses numerous parking lots effectively and thus many people get away with illegally parking, and worse unregistered vehicles are going unnoticed due to the inefficiency of an officer having to manually check all decals. Beyond this, malicious individuals can falsify a decal to make it appear as if they are indeed registered and to gain access to parking areas without proper authorization. In this paper, we discuss how drone surveillance can be utilized for automatic parking enforcement. Providing such a tool will not just speed up the process of enforcement but also allow for better use of public safety personnel.

Development of Automatic License Plate Recognition (ALPR) algorithms in the last decade or so, (see [1, 3, 9, 10, 12]) has aided law enforcement on the road, at campus security, at airports, casinos, etc. University of Kansas is one of the places that uses ALPR for parking enforcement. Parking lots are monitored by a public safety person driving around the campus in a vehicle that has ALPR camera mounted on the roof. The camera takes pictures of the cars, images are processed by ALPR, and tickets are issued if necessary. PlateSmart (see [6], [8]), is developing software using ALPR for parking management among other things. The software works with almost any camera and provides alerts of parking violations by email and text messages. However, neither one of these examples and none that we found is employing drones for parking enforcement.

The paper is organized as follows. Section 2 gives an overview of different technology and services that have been utilized. Section 3 talks about automation of drone's flight path. Data obtained and results are presented in Section 4 while Section 5 presents difficulties encountered and future work.

2. TECHNOLOGY

We used DJI Phantom 3 Professional drone and utilized several existing technology/services in order to create a system that automatically returns license plates of illegally parked cars. Technology/services used are Litchi application, openALPR, FFmpeg and a driving record search.

Litchi, [4], is an Android/IOS application for DJI drones that includes numerous improvements over the official DJI Drone application such as in-depth waypoint mission planning and object tracking. The Litchi flight app allows us to easily program more aspects of the flights because it can be run from a computer. The DJI app is only available on mobile devices. From a computer we can then collect all of the data we have gathered about a particular parking lot and build the routes on a large screen as well as program all the commands we need to perfectly capture data in a parking lot. Litchi works like an extension of the DJI app as it still requires the DJI app to pull its user interface, and commands. Litchi also facilitates the saving and naming of the various waypoint missions we create that we are then able to load into the phone and fly through the app. Litchi stores these waypoint missions via our Litchi account.

OpenALPR, [7], is an open source Automatic License Plate

Recognition library written in C++ with bindings in C#, Java, Node.js, Go, and Python. In OpenAlpr, an input is an image that is processed in stages and the output is a list of possible license plate strings. OpenALPR operates as a pipeline since the stages are occurring in the following order: Detection, Binarization, Character analysis, Finding Plate Edges, Deskew, Character Segmentation, OCR and Post Processing.

The Detection phase only happens once for each input image and this is the most processing-intensive phase. It uses the LBP (Local Binary Patterns) algorithm to find possible license plate regions in the picture. These regions consist of x and y coordinates along with width and height information that is sent to the later pipeline phases for further processing. The next phases are occurring multiple times based on how many license plate regions were identified on the image. The Binarization phase creates multiple binarized images to give us the best possible chance of finding all characters on the region because a single binarized image may miss characters on the image that are too light or too dark. One of the methods that are used in this phase is called the Wolf-Jolien method fed with multiple parameters for better results and these images are processed in the next phase called Character Analysis. This phase attempts to find character sized areas in a single license plate region. This analysis is done multiple times, starting by finding smaller characters and gradually looking for bigger ones. It is being done by finding all connected blobs in the license plate region that are roughly the width and height of a license plate character and have tops/bottoms that are in a straight line with other blobs of similar width/height. If nothing is found in the license plate region, the region will get discarded and no further processing will take place. The next phase is Finding the Plate Edges. The Detection phase is only responsible for identifying a region where a possible license plate could reside and this phase will try to find the exact edges (Top/Bottom/Left/Right) of the license plate. The algorithm will start by finding the hough lines of the plate and computes the horizontal and vertical lines. Information from the previous stages will be used to determine the likeliest plate line edges and a number of configurable weights are used to determine which edge makes the most sense. The Deskew stage requires plate edges information in order to remap the plate region to a standard size and orientation. This phase helps to give a correctly oriented plate image without rotation or skew. The next step is to isolate all the characters that make up the plate image with a use of a vertical histogram to find gaps between the plate characters, called Character Segmentation. This also cleans up disqualifying character regions that are not good enough to be recognized as characters, then the OCR phase analyzes all newly calculated character regions separately. For each character region, it computes all possible characters and their confidence ratings. The final phase is the Post Processing phase where a possible list of OCR characters and confidences are given and organized in a topN fashion. This phase also disqualifies characters below a particular threshold given. Our work only modified certain parameters to an extent to fine tune the algorithm to our needs and full credit to the algorithm is given to the developers of OpenALPR.

There are numerous fine tunings for the algorithm which we can manipulate in a config file such as:

1. Calibrating your camera improves detection accuracy in cases where vehicle plates are captured at a steep angle
2. Detection will ignore plates that are too large. This is a good efficiency technique to use if the plates are going to be a fixed distance away from the camera
3. The detection doesn't necessarily need an extremely high resolution image in order to detect plates. Using a smaller input image should still find the plates and will do it faster. The algorithm resizes the images to a fixed resolution. We used 1280 by 720, and got good results.
4. Whether we want to use masks. This can be expanded in the future.
5. OpenALPR can scan the same image multiple times with different randomization. Setting this to a value larger than may increase accuracy, but will increase processing time linearly (e.g., analysis_count = 3 is 3x slower)
6. Detection strength/strictness of license plate before signaling if license plate region exits.

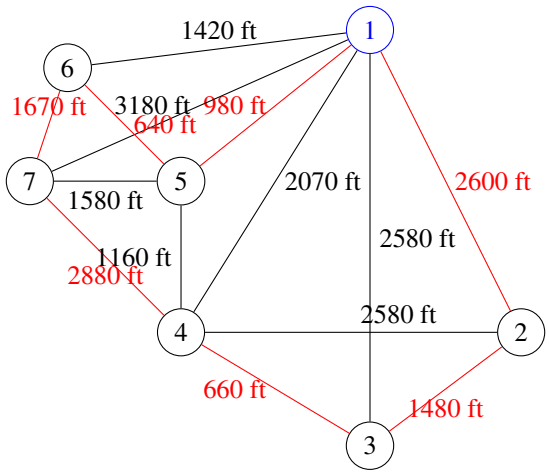

Figure 1: Undirected weighted graph. Nodes are the points of release of a drone. The red route represents the shortest path.

FFmpeg, [2], is a cross-platform solution to record, convert and stream audio and video. It is capable of splitting video output into standalone pictures.

3. DRONE AUTOMATION

The first step in successfully automating drones routes was to extensively map out all the parking regions of the school. Since it is legally required by Federal Aviation Administration (FAA) that if the drone is deployed, we have to stay within the line of sight of it, we designate a place (position), called a node, within each parking lot as a release point of the drone. It is a place that is easily accessible and covers one or possibly even two parking lots. Seven nodes were sufficient to cover all the parking lots at Barry University. Walking distance between them represent weights of the undirected graph. Edges are then formed based of nodes' positions on the real map and their connectivity. Starting from node 1, which is closest to public safety office, and returning to the same one, we brute-force to find the shortest path. The shortest path calculated is 1-2-3-4-7-6-5-1 for a total distance of 10910 ft, see Figure 3.

Routes are created by using our previously mapped out parking lots and then programming waypoints set at GPS coordinates. The methodology for creating a route is that the drone must effectively capture all the parking spaces as well as potentially illegal parking areas while avoiding obstructions. We take into consideration that cars may be parked either facing forward or in reverse. When the drone reaches the end of a particular line of cars within a route, it is then given a command to either rotate and retrace

the same route to cover the opposite row of cars or move onto the next section of the parking lot. Routes between rows of cars must be extremely precise as there are many hazards the drone can run into, such as wind, trees, cars, people, and random hazards that could change on a day to day basis. To handle these variables, extensive field testing was required. A projected path is created and then tested in varying conditions. Whenever a potential danger is noticed the path is modified accordingly.

To ensure that the visual data quality being captured was optimal a lot of tweaking of the flight height and gimbal angle occurred. The height setting and gimbal angle change on a per parking lot basis. For example, one of the lots had a height of 6 meters with a gimbal angle of 42.5 degrees. However, the height is not the only factor in determining gimbal angle. Changes in parking lot elevation from one end to the other can also affect the desired gimbal angle settings. Modification is needed to center the license plate in the image so that the image processing algorithm works efficiently. Further, we automate the drones flight by mapping out each parking lot on foot holding the drone in our hands. Using the DJI flight app, we take snapshots of GPS coordinates. Recorded GPS coordinates are used as waypoints which the drone follows when collecting data. Once a route has been mapped out, we do manual piloting tests using the specifications determined. Usually in manual pilot tests slight modifications to the GPS coordinates are made and the actions generated at particular waypoints are decided (actions such as turn x amount of degrees, tilt the camera down or up, raise or lower flight altitude). During manual testing we gradually raise the speed of the drone's flight as tweaks are made to the final path. Once all testing has finished we give the drone the fastest possible speed that we deem safe for that particular lot. Some lots being more open with less obstructions allow the drone to fly faster, while others require a slower velocity for safety purposes. If tweaks are necessary, then the process would begin again and another manual test would be conducted. Once testing is finalized, we automate drone's flight through the Litchi app. Namely, we plug the acquired GPS coordinates to build the necessary paths. Along the path Litchi app automatically facilitates the application of various commands to the drone such as, changing the gimbal angle, flight height, velocity, and drone orientation. These parameters depend on the parking lot and generally vary from parking lot to parking lot. The route is programmed on the computer through the Litchi app. This process is repeated for all the parking lots that will be captured by the drone. See Figure 2 for visualization of a programmed path on a parking lot.

Figure 2: Flight automation of a parking lot.

4. DATA AND RESULTS

A custom java application was created to connect several of above mentioned solutions into a combined one. The application first connects to the drone via USB connection and captures 4K footage. The 4K footage from the drone camera (max image size 4000x3000, ISO range, 100-3200 video, 100-1600 photo, lens: FOV 94 degrees 20mm (35mm format equivalent) f/2,8 focus at infinity, electronic shutter speed: 8-1/8000s, Sensor: 1/2.3" CMOS effective pixels: 12.4M, total pixels: 12.76M) with a third party application, called FFmpeg, is loaded in our java software which directly splits the video footage from the drone into several pictures. There are certain parameters that user can modify such as number of pictures to be extracted every second and type of image files that would be created. We ran multiple tests changing the type of image files between PNG, BMP, and JPEG, as well as the number of pictures to be extracted which varied between 1 and 3. We found out that three images per second and JPEG encoding of images were giving us adequate results. In Table 1 we present a statistical representation of some of the different parameters given by the extracting utility on a 2:57 minutes long, 3840*2160 video resolution sample footage.

	3 Frames per second	2 Frames per second	1 Frames per second
PNG	166 seconds, 5.58 GB	135 seconds, 3.73 GB	118 seconds, 1.87 GB
BMP	112 seconds, 12.3 GB	107 seconds, 8.25 GB	95 seconds, 4.14 GB
JPEG	128 seconds, 801 MB	118 seconds, 533 MB	116 seconds, 270 MB
	533 files	356 files	179 files

Table 1: Comparison between image types and image extraction statistics

Whenever the image extraction is done, all the newly created images are analyzed one-by-one by the OpenALPR algorithm. For every image processed, the algorithm creates a list of possible results, up to 10, with different confidence ratings, usually ranging from 70% to 95%. For one image, we get a list of license plate objects in the program. Each object contains a list of possible OCR reading for the license plate. We can get up to 10 possible results with a confidence rating in percentages, see Figure 3 and Table 2.

Figure 3: Sample Image from parking lot trial

Even though the results are around 85% to 90%, these are often invalid, having some of the very similar characters misread such as 8 and B, 0 and O. To overcome this issue, a map data structure is utilized and the license plate strings are used as keys to the map and the values are the frequency of the same keys. The value is incremented by one every time the same license plate string was read. At the end, the most frequent results will be shown to the user. The more image the program analyzes, the more accurate the results will be.

Even tough we are getting 95% confidence results, the algorithm is far from perfect and often gives unusable data, so we do the

plate0	confidence
DNYM26	91.1562
DNYH26	90.7403
DNY26	85.8861
DNYN26	85.4879
DNYB26	85.0558
ONYM26	84.8133
0NYM26	84.5391
QNYM26	84.4453
ONYH26	84.3974
ONYH26	84.1232

Table 2: Top 10 results for plate0 with respect to confidence interval. Processing Time is 48.2849ms.

following. The pattern matching feature runs the topN (parameter a priori set - the max number of results) results against a Regular expression (regex) matcher (another parameter inside openALPR) to find results that match common license plate patterns. The regex patterns are customizable and can be found in runtime_data/postprocess/*.patterns For example, using a pattern against this Czech plate results in only one possible match (which is the correct one), see Figure 4 and Table 3.

Figure 4: Sample image of a czech license plate

To visualize the output that the App produces see Figure 5. After the evaluation is finished, a third party service is utilized to check validity of the results and eliminate the invalid ones which resulted in 100% accuracy.

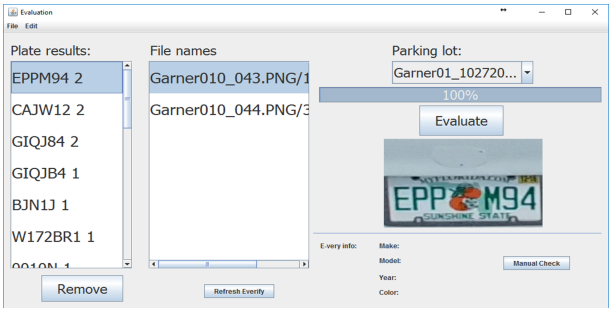

Figure 5: Screenshot of the App

5. DISCUSSION AND FUTURE WORK

Using a drone to visually scout parking lots is much faster and more effective across large areas, whilst obtaining accurate data. The drone can be connected to most devices (phones, tablets, computers) and have the visual data extracted quickly. Data is loaded into a program which efficiently analyzes license plates to detect unregistered vehicles, as well as illegally parked vehicles while automatically reporting violations, giving real time image captures of the 4K quality footage showing individual cars, and running simultaneously with the drone's surveillance routes.

plate0	confidence	pattern match
4S5O233	90.947	0
4S5O23	87.8683	0
4S5O2S3	85.8861	0
4S5O23G	85.4879	0
4S5O23	85.1644	0
4S5O23S	84.5445	0
4S5O23B	83.7395	0
4S50233	83.0457	1
4S5O2B3	82.5635	0
4S5O2	82.0857	0
4S5O2G3	81.5684	0
4S5O2J3	81.0409	0
4S5O2S	80.2911	0

Table 3: Some of the 40 results for this plate, notice the pattern matches 4S50233. The cz patterns are: cz #@##### and cz #@@####.

Initially, we wanted the drone to take several pictures per second using either the official DJI App or Litchi which extends capabilities for drone automation. The most we could have achieved was two images per second, which is not the best solution if we go at faster speed. It turned out that Litchi was unable to take some of the images and write it to the SD card. Error message was saying "failed to take photos for unknown reasons". Intuitively, the more detailed the images were, the bigger the file sizes were and that could have been the reason for failure of writing two images per second to the card. Even purchasing higher performance SD card, U3 UHS-II, did not help. After all, the solution was to record 4K videos and extract images from it using a batch file on a computer. Since Litchi is using GPS coordinates, based on how accurate the host devices' (android phone or tablet) GPS is, we have to "warm up" the GPS of the phone or tablet by enabling it via different methods. One such method is going into Google maps, or we can download an App that gives constant communication with GPS satellites for accurate GPS calculations.

The more accurate the results we want when we read the images, the more processing power is needed for the computer where results are evaluated. First, we wanted to create this project on an android device and get the images from the drone during flight and quickly evaluate the results and send the citation. It turned out that the algorithm to be run at acceptable speeds requires a powerful CPU, so low end devices are obsolete for this project. Also, we were not able to avoid connecting to the drone directly with a cable, since there was no method to get the images on the devices. We could have taken screenshots with the android devices, but the resolution would have not been adequate, and taking multiple screenshots per second on the device is difficult and inefficient. Maybe in the future as more powerful mobile devices, and better and more affordable drone technology is available, we could implement this software to android itself, skipping another device to carry.

Going further we can make improvements both in efficiency of the application and services it can provide. More precisely, we

can use image masks to make the areas of the images black that we don't want to get searched for. Since 90% of the time spent on one image is to find license plates, and 10% to guess the characters, we need to minimize the field to be searched. A good approach and easy approach is to make several footage of the same parking spot and get the average of coordinates of the license plates. Based on that we can create the black masks to skip parts of the image.

Furthermore, the application will employ the university's database to cross reference the license plates and receive information about the vehicles eligibility to park in the parking lot. In addition, we can create a form that automatically populates a citation for illegal parking based on school records or third party services. We can attach a proof image that the car is parked illegally or without permit. A system can be set up to send text messages to the owner of the car before finalizing the citation, if the mobile number is available along with sending an email to the owner. After some designated amount of time, a citation would be finalized and put into a database, or simply could be printed out from the software itself. Moreover, a weather forecasting system could also be implemented to check for any rain, humidity, and wind.

6. REFERENCES

[1] Chang, J., Ryoo, S., and Lim, H. (2011). "Real-time vehicle tracking mechanism with license plate recognition from road images." **The Journal of Supercomputing**, 65(1), 353-364. doi:10.1007/s11227-011-0580-x

[2] FFMPEG. "A complete cross-platform solution to record, stream, and convert audio and video." https://ffmpeg.org/

[3] Liang, Z; Zhang, S; Ye, X. "License plate state recognition based on logo matching and state name string classification." **Applied Mathematics and Information Sciences.** 2, 907, 2015.

[4] Litchi App. "Unlock the full potential of your DJI aircraft with Litchi." https://flyLitchi.com

[5] Liu, X., Guan, Z., Song, Y., and Chen, D. "An optimization model of UAV route planning for road segment surveillance." **Journal of Central South University: Science** & **Technology of Mining and Metallurgy**, no. 6 (2014).

[6] PlateSmart Parking Management. "Delivers the most accurate, flexible, and cost-effective turn-key Automatic License Plate Recognition (ALPR) solutions." https://www.platesmart.com/parking-management/

[7] OpenALPR - "Open Automatic License Plate Recognition system." https://github.com/openalpr

[8] "PlateSmart to Showcase Business Intelligence Tool Focusing on ALPR at ISCWest 2017." **ICT Monitor Worldwide.** 2017.

[9] Slapsinskas, P; Zilys, M. Efficiency of the electronic license plate recognition systems. Elektronika ir Elektrotechnika. 8, 59, 2013.

[10] Wanniarachchi, W. K., Sonnadara, D. U., and Jayananda, M. K. (2007). "License plate identification based on image processing techniques." **International Conference on Industrial and Information Systems.** 2007 doi:10.1109/iciinfs.2007.4579205

[11] Yuan, C., Zhang, Y., and Liu, Z. "A survey on technologies for automatic forest fire monitoring, detection, and fighting using unmanned aerial vehicles and remote sensing techniques." **Canadian Journal of Forest Research** , vol. 45, no 7 (2015) p. 783 - 792.

[12] Zhenghao, D; Fengxin. "Research on license plate recognition algorithm based on support vector machine." **Journal of Multimedia.** 2, 253, 2014

[13] Zhengrong, L., Yuee, L., Rodney, W., Hayward, R. and Jinglan, Z. "Towards automatic power line detection for a UAV surveillance system using pulse coupled neural filter and an improved Hough transform." **Machine Vision** & **Applications**, vol. 21, no. 5 (2010) p. 677 - 686.