**Parking Lot Enforcement with Drone**   
Sezer Kartal1, Dr. Tommy Dang2  
1Okan University, Computer Engineering, Istanbul, Turkey   
2Computer Science Department, Texas Tech University, USA

**Abstract**

This article describes a system that acts as a basis for parking lot enforcement by using a DJI Mavic 2 drone. This system collects data about license plates, their location and the detection time and stores them in a database for future use, however this system does not write tickets or any other type of enforcement.

Drone controlling part of the system is built for drones that support DJI Windows SDK, however the rest of the system could still work, if they were able to mimic the output of the main program, which is developed by using DJI Windows SDK.

1. **Introduction**

**Parking Lot Enforcement**

Parking lot enforcement is a well-defined problem, in this project we wanted to see if a drone could accomplish a this task and if it can, would it be a viable option to do so.

One advantage of using the drone is we can cover larger areas in a short amount of time and with less equipment, minimum requirements of this system is, a drone that capable of taking pictures and a computer.

**Automatic License Plate Recognition**

ALPR is a technology that allow us to recognize characters using optical character recognition in a given image, we can use this to detect license plate in a given image, we mainly used OpenALPR solution in our project since it can run locally.

1. **Problem Description**

Our main goal is to create a scalable system that has the main job of detecting license plates using a drone, we are not able to write tickets or any other actions since we don’t access to a database that allow us to do actions like this, instead we just collected data about license plates using this system.

**-Methodology**

For this system, 3 main parts is used, a software that controls a drone, a server that process the pictures that is taken by the drone, and a client which monitors all the data that is procced by the server. One main advantage of the system is, we can replace drone controlling software and the drone with different ones, and the system would still work.

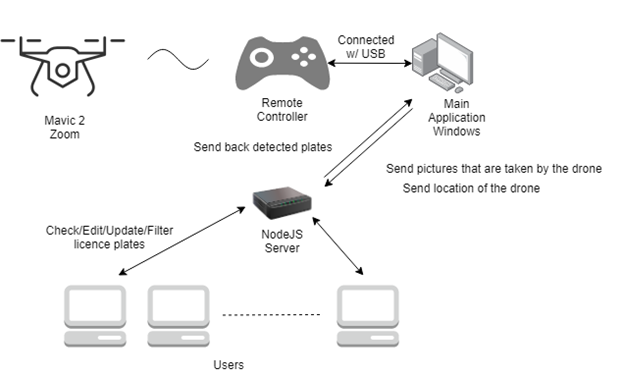


Figure Basic System Architecture

**Drone and Controlling Software**

DJI Mavic 2 drone is controlled by the remote controller, to control the drone via software we need to use DJI Windows SDK(C#), this allows us to control various specs of the drone, for example camera, media file download, Gimbal, lights and more. RC must be connected to the pc running the DJI Windows SDK application.

DJI Windows SDK allows us to control the drone in a waypoint system, basically tell the drone where to go and that to do after going there. The basic waypoint mission of a parking lot is shown in figure 2, Drone will safely go to the beginning of the waypoint mission and starts to strafe towards the next waypoint while looking at the cars and starts to take picture in 2 seconds intervals, usually drone flies 3m altitude from the ground level, -45 degree of the gimbal angle and 0.5 m/s speed, these are enough to identify the license plates, but it could depend on the place.

A picture containing screenshot

Description automatically generated

Figure Drone fly path of a parking lot

Whole purpose of the drone is to move around and take pictures, the better the fly path is we get better quality from the pictures, the drone is not aware of the cars or any other stuff, it just flies and take pictures, so a lot of work goes into to the planning of this mission path, to make sure drone gets the best results. These pictures send to the server which is the main processing part.

Controlling software of the drone also shares the location of the drone and live feed from the camera with the server, although server expects this information it is not mandatory for server to work properly.

DJI Windows SDK can also allow us to simulate the drone flight without actually flying, so this makes it really easy to test the software.

After drone starts to take pictures, these pictures are stored on the drone, so a download process is required to access these pictures, this process could take more than 1 second to do, depending on the distance and visual line of sight between the drone and the RC. Figure 3 shows how process is handled in a nutshell.

A screenshot of a cell phone

Description automatically generated

Figure Media download process and plate detection cycle

The Plate Detection action in figure 3 is the part where drone does a POST request to the server and parses the output of the server. So, there is no actual plate detection is done, in the controlling software of the drone, which defeats the purpose of scalability.

**Server** (NodeJS)

Server is the part of the system where pictures gets processed. Server receives images though POST requests, and the server expects the image data either jpg or png format, also latitude and longitude of the image that is taken. First server runs OpenALPR on the image that is received, if image contains a license plates server picks the most confident license plate and saves into database with latitude, longitude of the image and the time stamp, also sends back result in a JSON format to the caller.

Server has two optional connections with the controller software of the drone, server can receive location of the drone in real time, also server can receive live camera feed from the drone, these connections are optional, and it's not required for the system to work. Controller software will connect to these ports; port 8081 for a live location with latitude and longitude and port 1337 for live camera feed from the drone. Then server will show this extra information on the web interface

Also, server can receive data from multiple controller software, since server does not actually now the data is coming from an actual drone or something else.

Furthermore, server does not need to be in a different computer, server and the controlling software of the drone can be deployed in a single computer, this allows this system to be flexible and scalable.

**A close up of a map

Description automatically generated**

Figure Server license plate detection process

**Client** (NodeJS)

Client is the part of the system where a user (maybe an authorized person) can check all the data collected by the drone, as I mention before, all that is collected by the drone is saved in the server, so there is direct connection between the client and the server

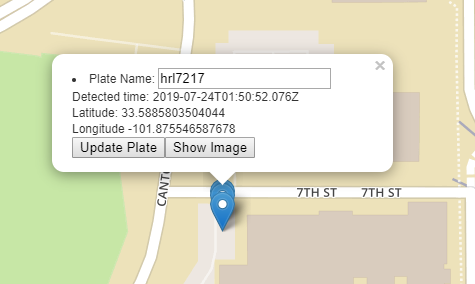


Figure Results Shown in Client-Side, North Side of parking lot R16

Figure 3 shows a piece of client UI, from here only interactions that user can do is to check the image and compare with the detect license plate to see if its correct and filter the results by time interval, plate name or wrong detected plates ( this is currently not possible since we don’t have access the database of real plates)

Multiple clients can connect and communicate with the server, the changes done in one client will affect the other ones, currently there is no user validation, anyone that has the address of the server can connect and experiment with UI.

**Summary**

In figure 5 you can see the data flow diagram of the resulting system, most complicated part of the system is controlling the drone properly, and it will change based on what kind of software and drone is used.

**A picture containing text, map

Description automatically generated**

Figure Data Flow Diagram of the Resulting System

1. **Results**

After testing the system on R16, R17, R25 with drone fly path similar the Figure 2, 3m altitude, 0.5 m/s speed and -45 degree of camera angle we were able to correctly detect all the license plates on the parking lot row. Figure 3 shows how the detected plates looks like, since the drone flies 0.5m/s speed sometimes we detect the same plate more than once, so this makes the UI really cluttered, a quick fix is to compare the detected plate with previous one and if the time difference too small we could discard or decide not to show the plate.

For the license plates we did test using 2 different solutions, table 1 shows all the results of the license plate recognition solutions, which is clearly OpenALPR is the winner here.

Table Results of two different ALPR solutions

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | TP | FP | Accuracy | Avg Time | Not Found |
| PlateRecognizer | 23 | 3 | 0.88 | 1.87 | 36 |
| OpenALPR | 47 | 4 | 0.921 | 1.15 | 11 |

Figure 6 shows, the drone point of view while it scans the parking lot, this pictures is taken by the drone and then send to the server by controller software, height and the angle of the camera as well as time of the makes a difference when detecting the license plates

A car parked in a parking lot

Description automatically generated

Figure Drone POV while scans the parking lot

Figure 7 shows the controller UI where a user can give commands to the drone and watch drones process, in this UI user can start the drone, land or give waypoint missions to the drone.

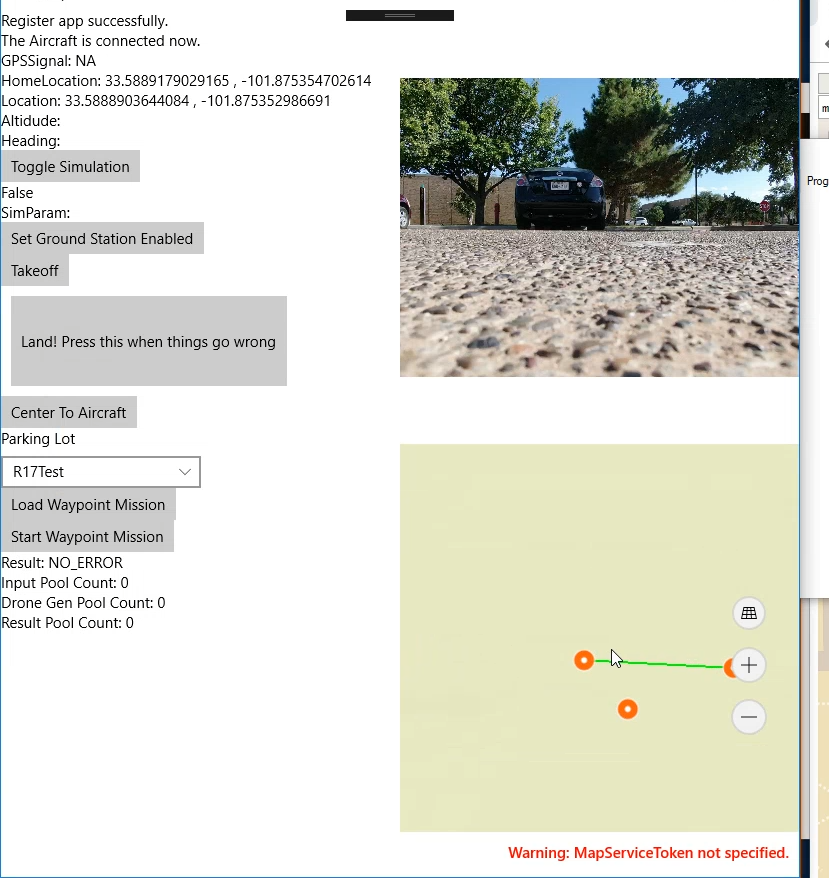


Figure Controller Software UI

1. **Conclusion**

These results show that, parking lot enforcement using a drone is possible, but we need more smart systems to accomplish more effective parking lot enforcement.

Drone has only 30 minutes of battery this makes it very ineffective to search all the parking lot with single go, a single drone will approximate take 7-8 hours to complete searching all the parking lot in the TTU campus,

Also, drone could save so much time if it was aware where the cars are, current system is not aware of that, so this makes it slow.

**Further Research**

Biggest problem of this system is, DJI drone has 30 minutes of flight time so large scale operations its not possible, so to make our flight most effective as possible, knowing the cars locations before hand could save us a lot of flight time, one possible solution is to taking a top down picture of the parking lot, by using that picture; we could try to identify the geographic location of the cars, and with this we can skip checking all the empty parking spots.

One other research opportunity is obstacle avoidance, trees and street lambs are really issue when doing autonomous flights.

1. **Acknowledgements**

I would like to thank Dr. Tommy Dang for his guidance and assistance in this project. also, I would like to thank Linh Ho for his assistance on field test.

Finally, I would like to thank TTU for this great opportunity.

1. **References**

DJI Windows SDK. Documentation Page. Retrieved July 29, 2019 from <https://developer.dji.com/windows-sdk/documentation/introduction/index.html>.

OpenALPR. Documentation Page. Retrieved July 29, 2019 from < http://doc.openalpr.com/ >.

NodeJS. Documentation Page. Retrieved July 29, 2019 from < https://nodejs.org/en/docs/>.