

Control of Autonomous License Plate Recognition Drone in GPS Denied Parking Lot

¹Mohammed Shais Khan
¹BE, Osmania University, Hyderabad, Telangana, India

Date of Submission: 01-01-2023

Date of Acceptance: 08-01-2023

ABSTRACT

It is difficult to autonomously navigate an unmanned aerial vehicle (UAV) around obstacles in an indoor environment. Localization, mapping, and path planning for navigation to the destination while avoiding collisions and there is also no GPS data available. One of the current drone-based application areas is parking lot occupancy monitoring, which may be used to effectively manage parking spaces, eliminate line-ups, reduce the time it takes to find a parking spot, and issue tickets in cases of parking violations. To solve this problem First, proposing a usual object-based detection using LPR cameras coupled with a PD controller for waypoint data fusion and localization next, performing a simulation in MATLAB for trajectory visualization and optimization in a parking space of a parking lot. The photographs of each parking spot acquired by the drone are then used to extract the number of character present from automobile and non-car images. Finally, a perfect controller graphs, characters and optimized trajectory results were obtained from the simulation performed using the proposed method .

Keywords: Quadrotor, LPR camera, MATLAB, PD controller.

I. OVERVIEW

UAV indoor applications have become popular in recent years as computer vision technology is becoming more and more powerful and seeing the need for autonomous and non-autonomous navigation and process planning in indoor surveillance where drone must move Multi storied parking structures that needs efficient surveillance and parking services for multiple car spaces in order to support this problem first using three floor parking lot where drone enter from one floor takes the images and moves to next top floor and repeat the same process for the enhanced

monitoring and inspection the layout can be seen in the fig-1 below.

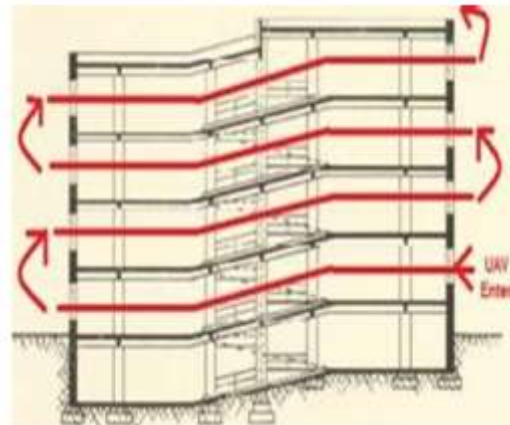


Fig-1: The 3-floor Parking lot demonstration

building UC Davis parking lot 2D floor and then created a 3D model. The proposed quadrotor working with coupled PD controller and LPR camera that enters the parking lot and provides necessary data to the inspector monitor the lot on a daily basis.

II. LITERATURE REVIEW

The localization of the obstacles and target are, many a times, performed by exploiting a scan matching approach based on a customized (controlled set) version of the iterative Closest Point algorithm, while mapping is done by extracting robust line features from LIDAR measurements thus the LIDAR makes the application of this concept costlier and time consuming [1]. In another research, processing LiDAR data gives real-time pose and altitude of the quadrotor and course surrounding geometric information. Simultaneously, using a camera to texture the LiDAR points for 3D reconstruction [2]. This inspired us to use fused sensor and camera systems to detect the vehicles in the parking structure and

conduct LPR to get the vehicle information for the objective. In the study [3] Considers the restrictions of system positioning precision, a unique trajectory planning technique for an unmanned aerial vehicle the UAV cannot precisely find itself due to the limitations of the system structure. The mission may fail if the positioning inaccuracy reaches a specific level. They then focused on rectifying errors that occur during a UAV's flight. In trajectory planning, the enhanced genetic algorithm (GA) and the A* algorithm were utilized but in real time this process is not as straightforward as depending on only one camera instead of costly sensors. Paper [4] used a mono-camera and laser range finder to study UAV navigation in GPS-denied interior situations, with all processing done on-board in real time. [5] worked on autonomous underwater vehicle navigation and localization, while [6] created a localization system for mobile robots in urban environments that are too large to use traditional indoor navigation techniques and too cluttered (with buildings and other structures) for GPS to work reliably.[7] used a laser range finder to navigate UAVs over forest situations without using GPS. [8] created an algorithm to fly a micro-RC helicopter autonomously in a small indoor setting using just an on-board lightweight camera as a sensor. Moire patterns in interior environments were employed by [9] to determine quadrotor posture. Infrared and ultrasonic sensors were utilized by [10] to fly a quadrotor in a big room.

Hokuyo Laser was utilized by [11] to fly a quadrotor in a GPS-denied environment. Visual SLAM has been used by certain authors, including [12] and [13] to create a map of the environment using pictures collected from a camera. He, Roy, and others employed 2D laser range finders for tiny quadrotors performing indoor navigation, localization [14]

A drone-assisted rapid and efficient monitoring method for real-time parking occupancy and automobile number plate identification was presented in the paper [15]. A drone- mounted camera was employed to gather photographs of the parking lot under consideration in this method. The number of occupied and unoccupied parking places in a parking lot is then determined using a deep neural network-based parking lot occupancy monitoring system. Each photograph was obtained by the drone. Parking spot are then tested with a pre- trained model based on car and non-car images. Then the automatic license plate recognition (ALPR) algorithm is used for parking rule enforcement. Finally, experimental results are verified using a web-based application that is connected with a cloud database but in real time

there is multiple errors in path planning and localization with online database with risk of loss of network connection and they did not present perfect trajectory planning and simulation of quad to support their problem statement and apart from that the LPR camera itself can be used for path planning and surveillance itself so the report studies the LPR camera along with the target drone control architecture and simulation for trajectory optimization and localization .

III. PROJECT DESCRIPTION

The figure-3 schematically shows the sensor data processing and control flow architecture of the UAV platform. Waypoint data is processed for target localization by the PD controller in the ground station computer. This provides the position of the UAV and the relative obstacle map. This information is used to plan the path towards the goal point while avoiding collision with obstacles. The path planning algorithm, implemented on the on-board computer, provides waypoints to be followed. This information is relayed to the ground station computer that uses on-board IMU information and implements higher-level proportional derivative (PD) control to provide pitch, roll and yaw commands to the APM controller on-board the UAV. The communication between APM and the ground station is established through.

a. Structure Modeling

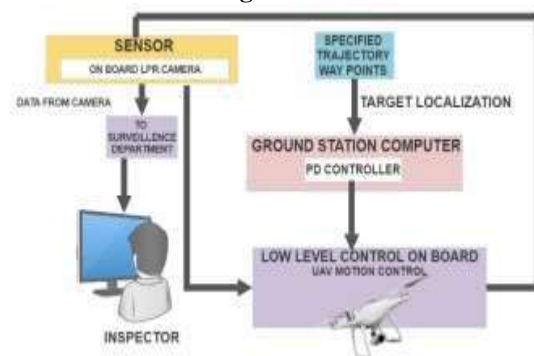


Fig-1:Project Schematic

STRUCTURE MODELLING

For the Successful implementation of the quad and the controller we have consulted the UC Davis facilities department to get the floor plan of the one of the largest parking structures in the Davis. Fig-2 is the 2-D floor plan of the structure and we have analyzed the 5 such individual 2-D plans and designed the 3-D parking structure in the solid works.

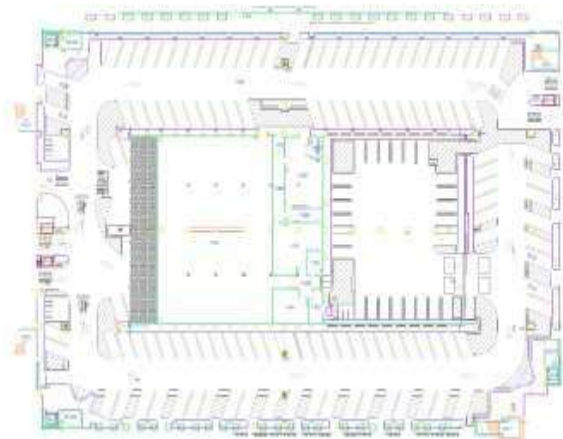


Fig-2: Constructed UC-Davis 2D floor plan along with 3D model

Fig-3 shows the 3-D model of the structure is designed using solid works. The structure is designed with the exact dimensions from the blueprint. The parking spaces are allotted as per in the draft and at respective places as shown in the fig-2. The vehicles along with the license plate are introduced in the parking structure design in order to reiterate the real environment for the quad and perform the simulation.



Fig-3: Designed parking lot 3D model

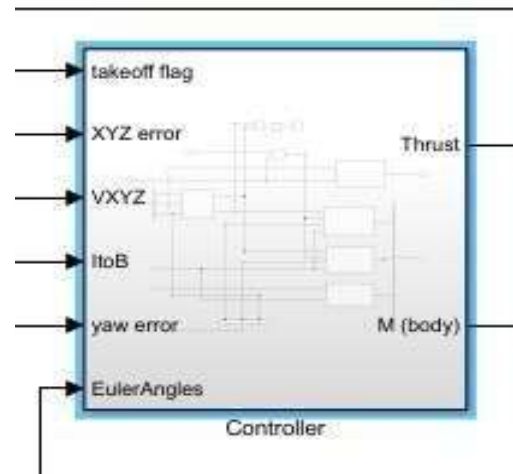
Controller description

For Quadrotor attitude and altitude control, a cascade PD controller is proposed. It is made up of an outer PD loop and an inner PD loop. The main controller is the outer loop. The set point for the inner loop, which is the secondary controller, is given by it. The position controller is in the outer loop. The required angle is the control output, which is useful. as the inner loop's setpoint the attitude controller is in the inner loop. The outer PD control loop generates the input reference signal for the inner angle PD control loop in a cascaded PD loop. Yaw Tracker: point the camera axis toward the target, control the yaw angle. Target Follow: keep the distance with the target, if the surrounding angular speed is set to zero, then the drone flies in 1st mode. Otherwise, it flies in 2nd

mode.

Assumptions

- Assumed that the motor is perfect. There are no fluctuations for the motor output.
- The Sensor is assumed to be perfect. There are no noises in the feedback.
- We are assuming the near perfect environment with minimal noise, wind disturbance or shaking.



Here we used 4 Here controllers (Z controller, x controller, y controller, yaw controller) to address the individual noises in the states & identify the errors effectively.

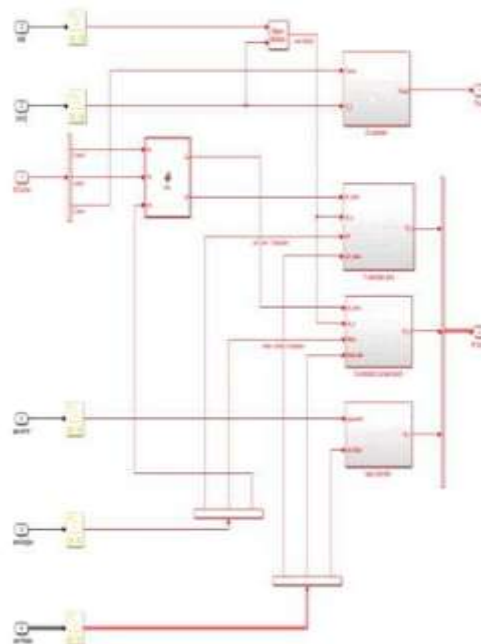


Fig-4: control structure

3.2.2 Z controller

In z controller PD controller is used for which z error acts as the gain for P controller and velocity for the d controllers. To make the drone hover the feed forward loop is introduced as a multiple of $m * g$. Within the thrust Simulator the Reasonable thrust limitation is given in Newton.

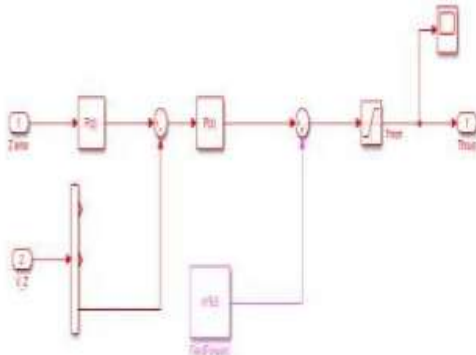
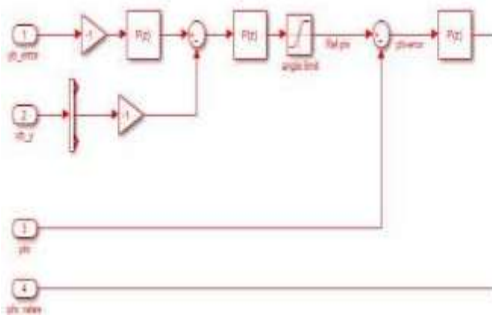


Fig-5: Z-control

3.2.3 X & Y Controller

The X & Y controller are in similar. X controller is associated with theta and y controller is associated with Phi. To control the drone moving in X-direction the drone needs to pitch(theta). Similarly in Y-direction the drone needs to roll (phi angle).



In these controllers 2 PD controllers are used by giving phi & theta rates as direct input to avoid the disturbances associate with oscillations. The outer PD controller will transform the x & y position error & velocity to a reference theta & phi angle respectively. The inner PD controller will convert the reference theta & phi along with theta & phi error to the exact torque.

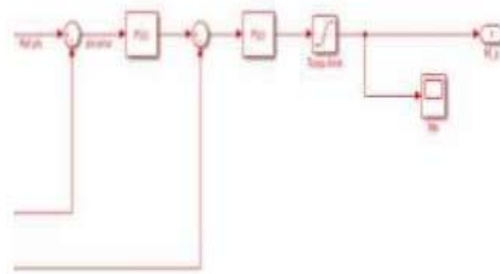


fig 6: X and Y-control

3.2.4 YAW controller

The yaw angle is directly fed into PD controller, here we only need single PD controller to transform from angle to torque.

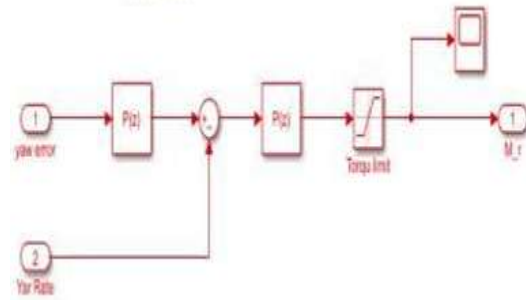


Fig-7: Yaw control

Dynamics model

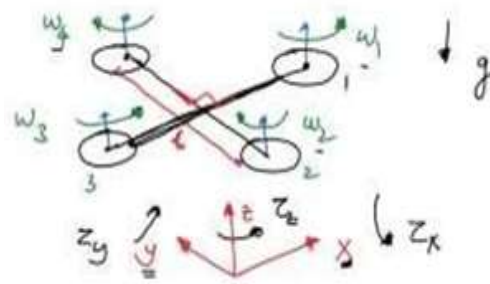


Fig-8: Coordinate system and free body Diagram

The dynamic model of a quadrotor UAV is shown above. The world frame is shown by axis, which is the fixed reference frame. The body frame is attached to the vehicle's center. The vertical forces on the rotors are provided by force 'F'. Each rotor is normal to the plane of rotation moment generated is τ along x, y and z axis τ_x , τ_y and τ_z

The rotational matrix to transform world frame to the body frame can be written as below [16]: -

$$R = \begin{bmatrix} c\psi c\theta - s\phi s\psi s\theta & -c\phi s\psi & c\psi s\theta + c\theta s\phi s\psi \\ c\theta s\psi + c\psi s\phi s\theta & c\phi c\psi & s\psi s\theta - c\phi c\theta s\phi \\ -c\phi s\theta & s\phi & c\phi c\theta \end{bmatrix} \quad (1)$$

Where c and s denote cosine and sine angles. The equations of motion for the quadrotor can be written as:

$$m \begin{bmatrix} \ddot{X} \\ \ddot{Y} \\ \ddot{Z} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + R \begin{bmatrix} 0 \\ 0 \\ \sum F_i \end{bmatrix} \quad (2)$$

where m is the quadrotor's mass and g are the acceleration caused by gravity.

3.3.1. Motor control

The vertical force F_n generated by each rotor is given by [16]:

$$F_n = k_N \omega_n^2 \quad (3)$$

where ω_n is the rotational speed of rotor n and $k = 6.11 \times 10^{-5} \text{ N/rpm}^2$

The Euler equations for the vehicle's angular accelerations are as follows:

$$I \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} L(F_2 - F_4) \\ L(F_1 - F_3) \\ M_1 - M_2 + M_3 - M_4 \end{bmatrix} - [q] \times I [q] \quad (4)$$

Here,

L : the distance between the center of gravity and each rotor.

I : the moment of inertia matrix along the x , y , z directions respectively:

$M_n (n = 1, 2, 3, 4)$: Moment produced due to rotor rotations given by:

$$M_n = k_M \omega_n^2 \quad (5)$$

ω_n = angular velocity $k_M = 1.5 \times 10^{-3} \text{ Nm/rpm}^2$

As a result, it is simple to deduce that when hovering:

$$F_{n,0} = \frac{mg}{4} \quad (6)$$

which transforms to:

$$\omega_{n,0} = \omega_H = \sqrt{\frac{mg}{4k_F}} \quad (7)$$

Attitude control

The relationships involving in equation (2) and (4) rotor speed inputs can be used to create a proportional derivative (PD) controller.

Assuming that the angular velocity component in the z direction can be neglected compared to other terms, we can substitute (3) into (5) into equation (6) to obtain the desired rotor velocities as [16]:

$$\begin{bmatrix} \omega_1^{des} \\ \omega_2^{des} \\ \omega_3^{des} \\ \omega_4^{des} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -1 & 1 \\ 1 & 1 & 0 & -1 \\ 1 & 0 & 1 & 1 \\ 1 & -1 & 0 & -1 \end{bmatrix} \begin{bmatrix} \omega + \Delta\omega_H \\ \Delta\omega_\phi \\ \Delta\omega_\theta \\ \Delta\omega_\psi \end{bmatrix} \quad (8)$$

$$\Delta\omega_\phi = k_{p,\phi}(\phi^{des} - \phi) + k_{d,\phi}(p^{des} - p)$$

$$\Delta\omega_\theta = k_{p,\theta}(\theta^{des} - \theta) + k_{d,\theta}(q^{des} - q)$$

$$\Delta\omega_\psi = k_{p,\psi}(\psi^{des} - \psi) + k_{d,\psi}(r^{des} - r)$$

$$\Delta\omega_F = \frac{m}{8k_F\omega_H} \ddot{z}^{des}$$

where ω_n^{des} = desired angular velocity

$n = 1, 2, 3, 4$ are the desired angular velocities of the rotors and

ω_H is the velocity at which drone is hovering. As a result, the deviations are provided by the PD controller. $\Delta\omega_\phi$, $\Delta\omega_\theta$, $\Delta\omega_\psi$ and $\Delta\omega_F$ these contribute to the forces that cause roll, pitch, and yaw.

Position Control

Let's take a look at the quadrotor UAV's desired trajectory, r , T . The PD controller calculates the command acceleration,

r_n^{des} , depending on the position error, which is as follows:

$$\ddot{r}_{i,T} - \ddot{r}_{i,T}^{des} + k_{d,i}(\dot{r}_{i,T} - \dot{r}_{i,T}^{des}) + k_{p,i}(r_{i,T} - r_{i,T}^{des}) = 0$$

here r_i and $r_{i,T}$ ($i = 1, 2, 3$) are the 3D position vector of the quadrotor and e desired trajectory respectively. It follows that $\dot{r}_{i,T} = \dot{r}_{i,T}^{des} = 0$ for hover.

To stabilize the vehicle in flight, we linearize the equations of motion responding to nominal hover states ($r = r_0$, $\phi = \theta = 0$, $\psi = \psi_T$, $\dot{r} = 0$, $\dot{\theta} = \dot{\psi} = \dot{\phi} = 0$). The rate of the pitch and roll angles is

desired to be all during flight. Thus, the desired roll and pitch angles to follow $r_{i,T}$ can derived by linearizing equation (2) about these nominal hover states: r_n and $r_{n,T}$ ($i = 1,2,3$) represent the quadrotor's 3D position vector and the intended trajectory, respectively. As a result, $r'_{n,T} = r''_{n,T} = 0$ for hover. To keep the vehicle stable in flight, we linearize the equations of motion that respond to nominal hover states ($r = r_0, \phi = \theta = 0, \psi = \psi_T, r' = 0, d\theta' = \psi' = \phi' = 0$) During flight, the rate of pitch and roll angles should remain constant. Thus, by linearizing equation (2) about these notional hover states, the appropriate roll and pitch angles to follow, $r_{i,T}$. Can be determined [16]:

$$\theta^{des} = \frac{1}{n} (X''^{des} \cos \psi_T + Y''^{des} \sin \psi_T)$$

$$\phi^{des} = \frac{1}{n} (X''^{des} \sin \psi_T - Y''^{des} \cos \psi_T)$$

LPR integration

With the rapid increase in the use of automobiles in daily life, license plate recognition has become a very important technical means. It has a wide range of applications in the fields of illegal photography, smart transportation, and vehicle management. At the same time, with OCR, deep learning, etc. With the rapid development of new technologies, license plate recognition algorithms are also being updated and iterated, and many accurate and efficient license plate recognition algorithms are emerging.

As a course report, this paper will introduce a non-deep learning method based on image segmentation and template matching in traditional vision. It realizes the recognition of the target license plate through morphological processing, pixel histogram segmentation and segmentation correction and other technologies. do not. Using this algorithm to recognize the provided sample license plate, the accuracy of recognition can be 100%, which demonstrates the accuracy and reliability of the algorithm. At the same time, we also screened some pre-positioned license plates. The recognition, accuracy is still very high, which proves the strong generalization of character segmentation and recognition algorithm. However, morphological accuracy is still very high, which proves the strong generalization of the character segmentation and recognition algorithm. However, due to morphological the parameters used in processing and perspective correction are related to the specific image environment, which makes

the license plate localization algorithm accurate and reliable with certain limitations. Finally, we use the MATLAB program to achieve all algorithm flows are encapsulated into some simple functions, which can be debugged in the MATLAB environment. In the license plate localization algorithm, the segmentation threshold of the color space and the selection of target points in perspective correction are related to the specific image. Reviewing the entire algorithm flow, we used a traditional vision method that is not deep learning, and completed all samples accurately. Recognition of license plates. The whole process better reflects the keen and efficient search, extraction, and utilization of image features in the field of computer vision. The parameters used in processing and perspective correction are related to the specific image environment, which makes the license plate localization algorithm accurate working of the LPR is given in the following points.

- Using morden OpenCV using available licence plate data as programming technique in MATLAB.
- Input for the network (CNN).
- Trained network is based on object detection receptive vision.
- The image conversion follows the sequence from RGB to Grayscale → Edge detection → Cropping and labelling → character recognition → Supervisor Inspection.

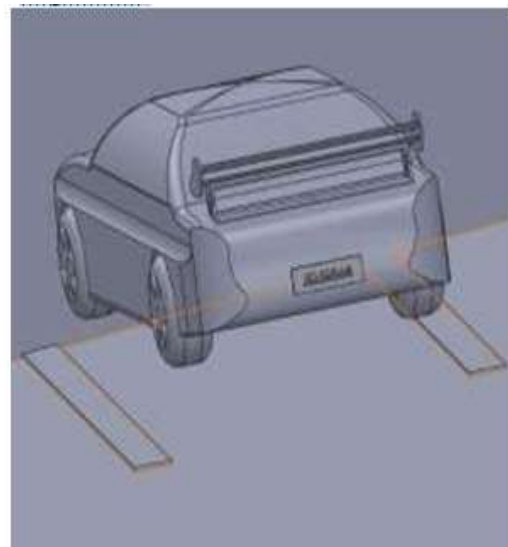


Fig-9: CAD model of a car with licence plate in parking lot

IV. RESULTS

Simulation setup

The code in MATLAB consists of a combination of three files to execute the path planned for the Quad-coptor. The three files being “setdronecontrol_copy.m”, “XYZsignal_copy.m”, and “animation_direct.m” and “generate”. The files individually perform various tasks important to the simulation of our drone in the target space which in our case was a parking structure which the drone must navigate. This Setdronecontrol.m file consists all the parameters that are used for drone control. Literature review of conventional quad-copter led us to using the following parameters. the quadrotor is essentially symmetric about all three axes, therefore $I_{xy} = I_{xz} = J_{yz} = 0$ which implies that:

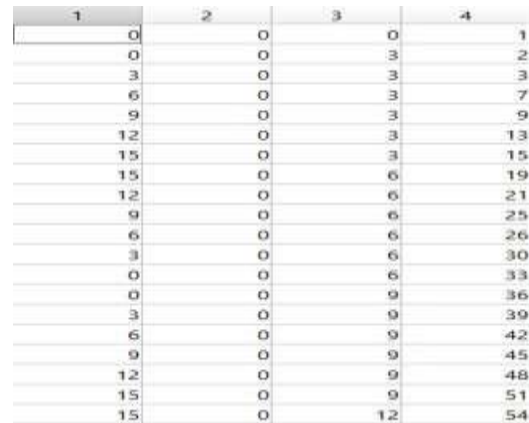
$$I = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix}$$

parameter	Value
Mass	0.2Kg
I_{xx}	$0.1kg - m^2$
I_{yy}	$0.1kg - m^2$
I_{zz}	$0.08kg - m^2$

The initial state of the quad-copter is set as [1,2,0] instead of [0,0,0] as the first target way point and the trajectory of the quad-copter as planned in the mat. File begins at [0,0,0] at $t=0$ and thus for the purposes of the simulation the quadcopter needs to be specified at a different location to the first position for the same. The initial values for the angular position and angular rates of the drone are kept to be zero for the purposes of our simulation.

This XYZsignal. m script will generate the actual signal that is fed into the drone control simulation mode. This program changes the waypoints given to the drone in the mat. file into a command signal which is provided as a function block in Simulink to give the set of reference outputs for the drone with regard to time.

An example file of the waypoints is as follows(Fig-10): The values in the first 3 columns signify the x, y, z co-ordinates of the drone with the time specified in the last column.



1	2	3	4
0	0	0	1
0	0	3	2
3	0	3	3
6	0	3	7
9	0	3	9
12	0	3	13
15	0	3	15
15	0	6	19
12	0	6	21
9	0	6	25
6	0	6	26
3	0	6	30
0	0	6	33
0	0	9	36
3	0	9	39
6	0	9	42
9	0	9	45
12	0	9	48
15	0	9	51
15	0	12	54

Fig-:10 Table showing the values of waypoints

Editing of these “Waypoints.mat” file we can make the drone go to any point in the domain defined with respect to time. One has to remember however to change the grid limits and the time for the simulation so as for the time available for the simulation should always be greater than the final time provided in the “Waypoints.mat” file. The Animation_direct file utilizes the code to generate the path of the drone in a domain and shows the track of the drone. It updates the position of drone with respect to time and also joins the dots between the previous and current position.

Dynamic generation of way points:

Previously, the waypoints of the drone had to be inputted into the MATLAB. Mat file individually. However, in real world application this is a tedious and impossible task specially for a drone which will be used in varied environments i.e various parking garages /structures. This file allows us to generate a series of waypoints via Mathematical equations in the cartesian co-ordinate system and for the purposes of showing that the quadcopter can follow a given trajectory the equations used in our MATLAB code is that of a helix. The equations are as follows:

$$\begin{aligned} x &= 2 * \cos(2 * t) \\ y &= 2 * \sin(2 * t) \\ z &= 2 * t \end{aligned}$$

In reality the Set of commands fed to the drone for path/Trajectory planning will be combination of waypoints given by us and a path predefined by mathematical equations.

Simulation steps:

For the Drone control via Waypoints:

1. Run the being “setdronecontrol_copy.m”.

2. Load the “Waypoints.mat” file into the workspace
 3. Run “XYZsignal_copy.m”, and then subsequently the associated Simulink file of the same folder.
 4. Run “animation direct.m”.
- For the Drone control via “Generate_A.m”:
1. Run the being “setdronecontrol_copy.m”.
 2. Run “Generate_A.m” to generate the set of path points via mathematical equations.
 3. Run “XYZsignal_copy.m”, and then subsequently the associated Simulink file of the same folder.
 4. Run “animation direct.m”.

Simulation results

To complete the flight simulation, first we performed a vertical take-off from a starting position, and then use the suggested technique to determine the remaining waypoints. The quadrotor flies through the created waypoints as new ones are generated at the same time. The waypoints are computed to prevent colliding with any barriers and to help you get closer to desired landmark.

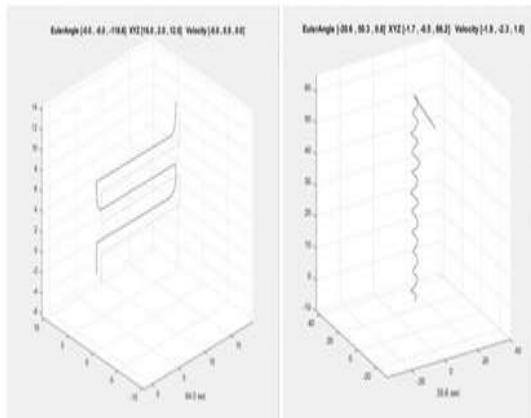


Fig-11: Simulation showing trajectory and localization based on camera axis towards a targeted point.

The Fig- below depicts the quadcopter’s three-dimensional trajectory. Fig The quadcopter’s position along the X and Y axes is also shown over time in figure where the parking space is located between black blocks the red curve is the trajectory generated with blue points as waypoints specified.

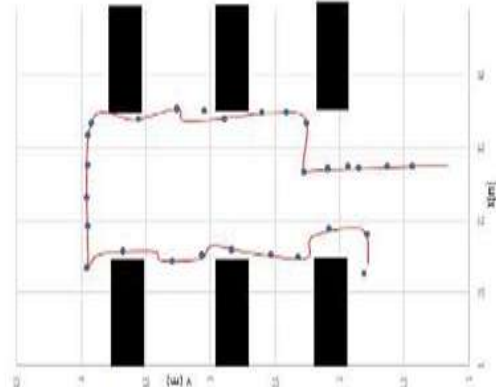


Fig-12: Quadcopter trajectory (in red) in the designed parking lot space with Waypoints shown in X-Y plane.

Controller results

Torque scope

The control structure is designed in such a way that Euler rates are given directly as input. By doing so, it reduced the noise of the Torque output signal, making it a more practical simulation when checked the torque scope. This has shown a more realistic thrust torque output with minimal oscillation. The sampling rate is set to 0.01 sec to have more practical controller signal o/p later which can be compared this to the deep neural network controller (DNN). To compare we need to have the same sampling rate and the oscillation.

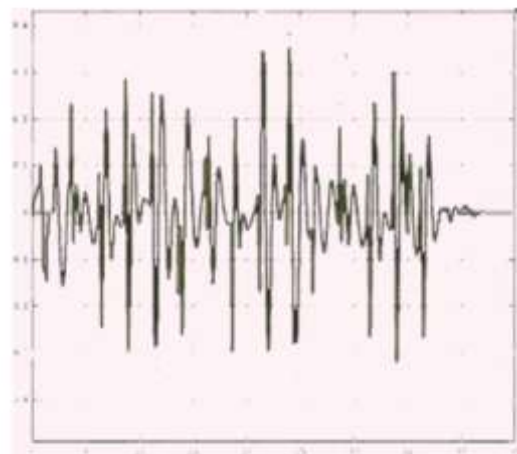


Fig-13 Torque scope obtained Using PD controller.

LPR Setup and result

The OpenCV code was implemented in MATLAB where the available images of car were taken and used for our case flowing are the sample results shown below.



Fig-15: LPR Taken original license plate images with final character output

V. CONSLUSION

The implemented architecture was found to be suitable for the indoor parking lot application. We see that the trajectory of the quad is optimized as well as the coupling of the LPR model works well with our PD control structure. The trajectory obtained successfully utilizing the PD controller along with specified way points

VI. FUTURE PROSPECTS

The idea of sign. In addition to accuracy, it can also propose another evaluation metric, which is to use confusion matrix and F1-score to evaluate the generalization performance of the valence algorithm. An F1-score close to 1 indicates that the algorithm generalizes well. For different input images, the accuracy of the output rectified image is different, and the threshold value needs to be adjusted manually. The use of different images in template matching has a slight impact on the recognition results, which may be achieved by using CNN. The CNN network is trained to assign appropriate weights to different templates to improve. Performing CFD analysis as seen in case of airfoils and supersonic nozzle studies [17]

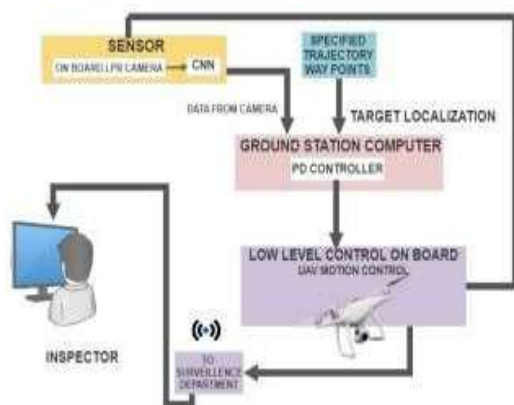


Fig-16: Autonomous surveillance

Fig-16 shows the intended autonomous LPR enabled surveillance drone which includes a pretrained CNN sending its output to the pd controller which also receives the output from specified waypoints from the data collected by LPR camera. The detailed control architecture of the intended autonomous drone is shown in the fig-17.

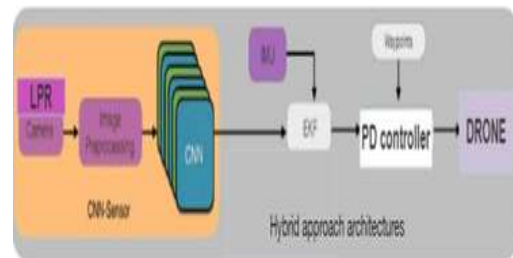


Fig-17 control architecture

REFERENCES

- [1] Opromolla, Roberto, et al. "LIDAR-inertial integration for UAV localization and mapping in complex environments." 2016 International Conference on Unmanned Aircraft Systems (ICUAS). IEEE, 2016.
- [2] Zhao, Pinjie, et al. "Design of 3D reconstruction system on quadrotor Fusing LiDAR and camera." 2020 39th Chinese Control Conference (CCC). IEEE, 2020.
- [3] Zhou, H., Xiong, H.-L., Liu, Y., Tan, N.-D., & Chen, L. (2020). Trajectory planning algorithm of UAV based on system positioning accuracy constraints. *Electronics*, 9(2), 250. <https://doi.org/10.3390/electronics9020250>
- [4] Wang, F., Cui, J., Phang, S.K., Chen, B.M., and Lee, T.H. (2013). A monocular and scanning laser range finder based UAV indoor navigation system. 2013 International Conference on Unmanned Aircraft Systems (ICUAS), IEEE, 694–701.
- [5] Paull, L., Saeedi, S., Seto, M., and Li, H. (2014). AUV navigation and localization: A review.
- [6] Georgiev, A. and Allen, P.K. (2004). Localization methods for a mobile robot in urban environments. *IEEE Transactions on Robotics*, 20(5), 851–864
- [7] Cui, J.Q., Lai, S., Dong, X., Liu, P., Chen, B.M., and Lee, T.H. (2014). Autonomous navigation of UAV in forest. 2014 International Conference on Unmanned Aircraft Systems (ICUAS), IEEE, 726–733.

- [8] Soundararaj, S.P., Sujeeth, A.K., and Saxena, A. (2009). Autonomous indoor helicopter flight using a single onboard camera. *Intelligent Robots and Systems, 2009. IEEE/RSJ International Conference on IROS 2009, IEEE*, 5307–5314.
- [9] Tournier, G.P., Valenti, M., How, J.P., and Feron, E. (2006). Estimation and control of a quadrotor vehicle using monocular vision and more patterns AIAA Guidance, Navigation and Control Conference and Exhibit, 21–24.
- [10] Roberts, J., Stirling, T., Zufferey, J.-C., and Floreano, D. (2007). Quadrotor using minimal sensing for autonomous indoor flight. *Proceedings of the European Micro Air Vehicle Conference and Flight Competition (EMAV2007)*.
- [11] He, R., Prentice, S., and Roy, N. (2008). Planning in information space for a quadrotor helicopter in a GPS-denied environment. *Robotics and Automation, 2008. IEEE International Conference on ICRA 2008, IEEE*, 1814–1820
- [12] Steder, B., Grisetti, G., Stachniss, C., and Burgard, W. (2008). Visual slam for flying vehicles. *IEEE Transactions on Robotics*, 24(5), 1088–1093.
- [13] Williams, B., Cummins, M., Neira, J., Newman, P., Reid, I., and Tardós, J. (2008). An image-to-map loop closing method for monocular slam. *Intelligent Robots and Systems, 2008. IEEE/RSJ International Conference on IROS 2008, IEEE*, 2053–2059.
- [14] Achtelik, M., Bachrach, A., He, R., Prentice, S., and Roy, N. (2009b). Stereo vision and laser odometry for autonomous helicopters in GPS-denied indoor environments. *SPIE Defence, Security, and Sensing, International Society for Optics and Photonics*, 733219–733219.
- [15] Sarkar, S., Totaro, M. W., & Elgazzar, K. (2019). Intelligent drone-based surveillance: Application to parking lot monitoring and detection. *Unmanned Systems Technology XXI*. <https://doi.org/10.1117/12.2518320>
- [16] Manish Kumar, Mohammad Sarim, Alireza Nemat, 8 - Autonomous Navigation and Target Geo-Location in GPS Denied Environment, Editor(s): Franck Cazaurang, Kelly Cohen, Manish Kumar, Multi-Rotor Platform-based UAV Systems, ISTE, 2020, Pages 153-175, ISBN 9781785482519, <https://doi.org/10.1016/B978-1-78548-251-9.50008-X>.
- [17] Khan, M. S. (2021). Numerical Analysis of de Laval nozzle under surrounding zone and compressed flow. *International Journal for Research in Applied Science and Engineering Technology*, 9(1), 98–105. <https://doi.org/10.22214/ijraset.2021.32720>