iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 9 **Issue:** XI **Month of publication:** November 2021

DOI: https://doi.org/10.22214/ijraset.2021.38831

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 9 Issue XI Nov 2021- Available at www.ijraset.com

Cascaded PID Control System for UAV with Gain Factor Prediction Using ML

Shubhankar Goje[1], Poorva Bedmutha[2], Prajwal Shah[3], Jagruti Agrawal[4]
[1, 2, 3]*Electronics & Telecommunications Engineering, Pune Institute of Computer Technology, Pune, Maharashtra, India*
[4]*Computer Engineering, Pune Institute of Computer Technology, Pune, Maharashtra, India*

Abstract: Drones are not inherently stable, necessitating the use of a flight controller. If the UAV is properly tuned, the drone will fly steadily; otherwise, it won't. Hence, we have used a PID (proportional, integral, differential) controller for a stable flight. A well-functioning PID controller should enable amazing climbs and long-range flights. But, when used singly, PID controllers can provide poor performance, resulting in a long settling time, overshoot, and oscillation.
Here, we propose a new approach to maneuver UAVs using a PID control system and overcome the shortcomings of using PID controllers in UAVs. This disadvantage is resolved using the Machine Learning polynomial regression model. The gain factors in a PID control system, which is otherwise ideally constant, should be changed in order to reduce the minor instabilities for a smooth flight. Our method has been elaborated and illustrated with suitable diagrams in the following work. When simulated in Gazebo on a Robot Operating System (ROS), our technique is proven to be successful.
Keywords: Control Systems, PID, UAV, Drones, Polynomial Regression, Gain Factor, Prediction Algorithm.

I. INTRODUCTION

Unmanned Aerial Vehicles (UAVs) have gained a lot of attention and study in the military, commercial, and agricultural industries in today's technologically evolved society. The employment of autonomous unmanned aerial vehicles (UAVs) in various complicated working contexts is becoming more common. It is critical for these vehicles to be able to generate collision-free mobility as well as to be able to adjust accordingly during implementation to deal with obstacles/hindrances that may arise during operation. Ideally, a UAV should be able to make decisions in dynamic surroundings with negligible human intervention along with a stable and reliable flight.

Drone technology has come a long way from remote-controlled drones to autonomous drones. There have been a lot of advancements in the technology used for the efficient designing of UAVs. PID controllers have been used for flying drones for a long time now. While PID control is the best controller for an unmanned aerial vehicle, better performance can be obtained by overtly modeling the process without resorting to any human intervention. The most major improvement is to use feed-forward control using system information instead of the PID for error control. PIDs can be tweaked in minor ways, such as changing the parameters (either gain scheduling in different use cases or adaptively modifying them based on performance), improving measurement (higher sampling rate, precision, and accuracy, as well as low-pass filtering if needed), or cascading multiple PID controllers.

In today's world, where machine learning has become an inseparable part of recent technologies, we intend to use it for a steady UAV flight too. We have come up with a polynomial regression model that tactfully varies the values of the gain factor in the PID control system, which otherwise are constant in an ideal PID control system. This helps in reducing the value of the error accordingly, which has a major part in altering the drone orientation, which in turn is responsible for flying the drone.

The subsequent work is arranged as follows: Section 2 discusses the associated background research in this area. Section 3 contains information on the system model of the UAVs and the assumptions made in the proposed algorithm. The suggested technique is explained in section 4. Finally, section 5 concludes with the successful simulation of the flight using the proposed algorithm.

II. BACKGROUND WORK

In the past years, many algorithms were implemented to make drones autonomous and also allow drones to deal with any obstacles that come in their way or to generate collision-free mobility. Constantly, research workers are working on these algorithms to increase their efficiency by finding an exemplary solution.

The concept of some of the works is focused on how to make drones autonomous, i.e. how to make them pilot-free. [1] Model-Based Design is used for creating a semi-autonomous control system for a drone. For communicating between the drone and the autonomous controller, Robotic Operating System is utilized.

©IJRASET: All Rights are Reserved

This gives some relief to the pilot of the drone and allows them to focus on data collection. [5] The vision system is used for landing area detection. Marker detection is done using Canny's edge detection algorithm with contour shape analysis algorithm. To detect tracking patterns, Lucas-Kanade optical flow algorithm is used. Robot Operating System (ROS) is used for system design. [6] Stabilization of the drone is achieved by setting the roll, pitch, and yaw. PI and PID controllers were used for altitude lock and position of the drone. Trajectory tracking is achieved by implementing ROS. The autonomy was achieved using [7] Deep Reinforcement Learning. [8] Using computer vision, drones can be controlled using hand gestures as well. AdaBoost Classifier is used for recognizing gestures. [9] Quadcopter, [10] Self-Tunable Fuzzy Inference System (STFIS). [16] The study in this paper is based on a simple two-neuron recurrent network with synaptic plasticity. It uses two small and light-weight LiDAR sensors for encountering obstacles and allowing the drone to circumvent them on its own. [18] Multi-sensor perception; The main objective of this paper is to make advancements in the UAS to achieve autonomous flights by the use of the latest development in deep learning and edge computing. [19] The objective of this paper is to increase the flight time of the drone. The iterative Feedback Tuning method is used for optimizing PID parameters. [20] Solid works simulator is used for the structural model of the drone.
The methods that were advised for control systems are [2] Redundant drone, [3] Wireless Network Systems, [4] Predictive models. Autonomous drones can be used for indoor purposes as well. But it may have to face a few challenges where a drone's control performance can be declined by wireless network interference. [11] This paper aims to determine how wireless network interference affects the drone's positioning controllability. [12] The drones can only be used when they are secure and safe. Drones' need for indoor environments is increasing. For indoor environments, they are supposed to be used without a global navigation satellite system (GNSS). [14] Using high accuracy hybrid markers and AR markers, guidance and accurate landing or drones are achieved. Autonomous drones have many applications like the military, commercial and agricultural industries. [13] The main objective of this paper is to figure out the footpaths among trees where the drones can follow. It uses artificial neural networks of deep learning or convolutional networks. [15] This paper uses drones for delivering mails for the mail office in the university. CAD software was used to draw the prototype and then the model generated was compared with the actual prototype. A finite state machine is used to demonstrate the design of the code. The analysis of the stability of the drone and the code is done by means of simulation. [17] This paper uses autonomous drones for repelling sharks to protect swimmers and surfers from their attacks. It uses an intercept algorithm. There are many more applications of autonomous drones.

III.SYSTEM MODEL AND ASSUMPTIONS

Following are the assumptions that are necessary for understanding the algorithm.

A. Model Assumptions

1) The technology used is an unmanned aerial vehicle (UAV) that can freely rotate in the roll, pitch, and yaw axes, allowing it to travel in any direction on a two-dimensional plane, stay still in mid-air, and vary its height with the help of throttle.
2) The UAV has four rangefinders looking front, right, rear, and left, each capable of detecting any obstacle within a range of R meters.
3) An IMU (Inertial Measurement Unit) sensor is installed on the UAV, which measures and reports the UAV's orientation. Three orientations are calculated by the IMU: roll, pitch, and yaw. The yaw is measured in degrees and starts at 0o when the UAV is facing North, increases clockwise, and ends at 180o when the UAV is facing South.
4) The GPS sensors on the UAV determine the location of the UAV in terms of latitude, longitude, and altitude.

B. Environmental Assumptions

1) The environment is a 2D plane that is located above ground and has a goal point that can be reached.
2) A finite number of 2D obstacles may exist in the environment, which can be imagined to be closed curves with finite bounds and area.
3) The position, size, and form of barriers in the surroundings are unknown unless the UAV's rangefinders can detect them, in which case the UAV can only identify a single point of the whole obstacle.

C. Variable Description:

1) Yaw: The UAV's angular displacement measured clockwise from the North axis.
2) Pitch: The UAV's angular displacement that indicates its forward tilt.
3) Roll: The UAV's angular displacement indicates its sideways tilt.

©IJRASET: All Rights are Reserved

4) Throttle: The UAV's linear displacement indicates its upward/downwards motion.
5) drone_altitude: Stores the altitude of the drone above sea level as received by GPS sensor.
6) setpoint_altitude: Stores the destination altitude of the drone.
7) Error: Measure of difference between desired altitude and current altitude of the UAV.
8) PID Controller - PID stands for Proportional-Integral-Derivative, together producing a control signal. This type of control is used to drive a system in the direction of an objective location otherwise level. It uses a closed-loop feedback device to regulate all the process variables.
9) P- Controller: The output of a proportional or P-controller is proportionate to the present error. The targeted or set point is compared to the actual or feedback process value. The output is obtained by multiplying the resultant error by a proportional constant. This controller output is 0 if the error value is zero.
10) I-Controller: Because of the constraints of the P-controller, where there is always an offset between the process variable and the setpoint, an I-controller is required to eliminate the steady-state error. It integrates the error over time until the value of the error approaches zero. It stores the value of the final control device where the error is zero.
11) D: The I-controller does not have the capacity to forecast incorrect behavior in the future. As a result, when the setpoint is modified, it reacts normally. D-controller solves this problem by predicting the error's future behavior. Its output is determined by the rate of change in error over time multiplied by the derivative constant. It increases system responsiveness by providing a kick start for the output.
12) Proportional tuning involves correcting a target proportional to the difference. Thus, the target value is never achieved because as the difference approaches zero, so too does the applied correction.
13) Integral tuning attempts to remedy this by effectively cumulating the error result from the "P" action to increase the correction factor. Rather than stopping when the target is reached, "I" attempts to drive the cumulative error to zero, resulting in an overshoot.
14) Derivative tuning attempts to minimize this overshoot by slowing the correction factor applied as the target is approached.
15) Steady State Error: Steady-State error is the final difference between the process variable and setpoint.
16) Setpoint: The desired position is called the setpoint
17) max_error: Difference between setpoint and initial done position
18) Kp: The PID gain factor for Proportional.
19) Ki: The PID gain factor for Integral.
20) Kd: The PID gain factor for Derivative.
21) Pro: Proportional output from PID.
22) Int: Integral output from PID.
23) Der: Derivative output from PID.
24) d_input: Measure of the differential input.
25) output: Overall output calculated from the PID control system for the given purpose.
26) last_input: Last Input to the PID to calculate the derivative term
27) prop1: Propeller speed for front left propeller
28) prop2: Propeller speed for front right propeller
29) prop3: Propeller speed for rear right propeller
30) prop4: Propeller speed for rear left propeller

IV. THE PROPOSED ALGORITHM

Our goal is to propel the UAV in a given environment to perform various tasks. This is done by receiving information from the sensors on the drone that provide us with a priori information of current conditions of the external environment and internal status. Such information is provided to us by the onboard sensors like IMU (Inertial Measurement Unit) and GPS sensors that provide us with details of UAV's current orientation and location respectively. A Control system processes this information to generate individual propeller speeds, which locomotes the drone from the current position to the desired position through a series of processes. It does so by obtaining optimum performance from the system in the form of maximum productivity, profit, high accuracy, and minimum cost. The modularity of such a digital system is evident as it can easily be modified by changing only a few program instructions.

©IJRASET: All Rights are Reserved

Using feedback in a control system improves control by automatically adjusting the controller output to reduce the steady-state error. This helps reduce the effects of dynamic disturbances. In addition, feedback adds stability to an unstable process, ensuring a repeatable and reliable control loop.

By considering the aforementioned assumptions and variables, the algorithm for the drone to navigate the environment and reach the destination can be defined using four steps.

A. Takeoff

The UAV takes off when we provide it with a positive throttle, i.e., rotating all four propellers with equal speeds. Once the throttle surpasses gravitational force and air resistance, the UAV takes off. The more the propeller speed, the faster the UAV's altitude increases. The desired altitude should be reached swiftly, without overshooting and negligible oscillations.

B. Yaw

Before the UAV begins to move in the direction of its target destination, it needs to have one of its rangefinders observing the direction it is moving in, so as to avoid any collisions. This is achieved by selecting the nearest rangefinder and rotating the UAV with minimum angular displacement in the yaw axis.

C. Travel

Once the yaw is set, the UAV should begin traveling to the destined latitude and longitude by simultaneously rolling and pitching with proportionate intensities so as to traverse in a straight line along with the displacement of starting and ending locations.

D. Landing

Once reached the target location, the drone lands by reducing its throttle gradually.

PID control problems, and often perform satisfactorily without any improvements or only coarse tuning, they can perform poorly in some applications and do not, in general, provide optimal control. The fundamental difficulty with PID control is that it is a feedback control system, with constant parameters, and no direct knowledge of the process, and thus overall performance is reactive and a compromise. While PID control is the best controller in an observer without a model of the process, better performance can be obtained by overtly modeling the actor of the process without resorting to an observer.

The most significant improvement is to incorporate feed-forward control with knowledge about the system and using the PID only to control error. Alternatively, PIDs can be modified in more minor ways, such as by changing the parameters (either gain scheduling in different use cases or adaptively modifying them based on performance), improving measurement (higher sampling rate, precision, and accuracy, and low-pass filtering if necessary), or cascading multiple PID controllers.

The UAV sends its current altitude, drone_altitude, to the control system, and knowing its target altitude, setpoint_altitude, we first calculate the error term as:

error=setpoint_altitude-drone_altitude

This error term needs to be converted into suitable actuator commands which in turn reduces the error term to zero. This can be achieved by a PID control system.

Derivative input, d_input, is calculated first as it depends on the previous input of the PID system.

d_input = drone_altitude - last_input; if last_input0

d_input =0; if last_input=0

For initial state the value of last_input is zero

Having calculated the input for the PID system, we can now calculate the Proportional, Integral and Derivative values.

*Pro=Kp * error*

*Int=Int + (KI * error)*

Calculating the derivative term that controls overshoot and slows the drone down when it is about to reach the desired position.

*Der=Kd*d_input*

Now, by this, we can get the overall output from the PID system. As the derivative term reduces the P and I terms, it is negative.

output=Pro + Int - Der

The current drone_altitude will be used for the next iteration of PID as last_input

last_input=drone_altitude

©IJRASET: All Rights are Reserved

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 9 Issue XI Nov 2021- Available at www.ijraset.com

The output calculated is directly given to the propellers equally as all propellers spinning with equal speeds would provide throttle to the drone. Once the desired altitude is reached, the PID controller automatically adjusts its output so that the drone hovers at that altitude until the next task is executed.

The working of a PID controller depends on its gain factors, Kp, Ki, and Kd. The value of these factors depends on the UAV's physical properties and external environmental factors like air resistance, gravitational force, etc. It is unique for each system and can be determined by the method of trial and error.

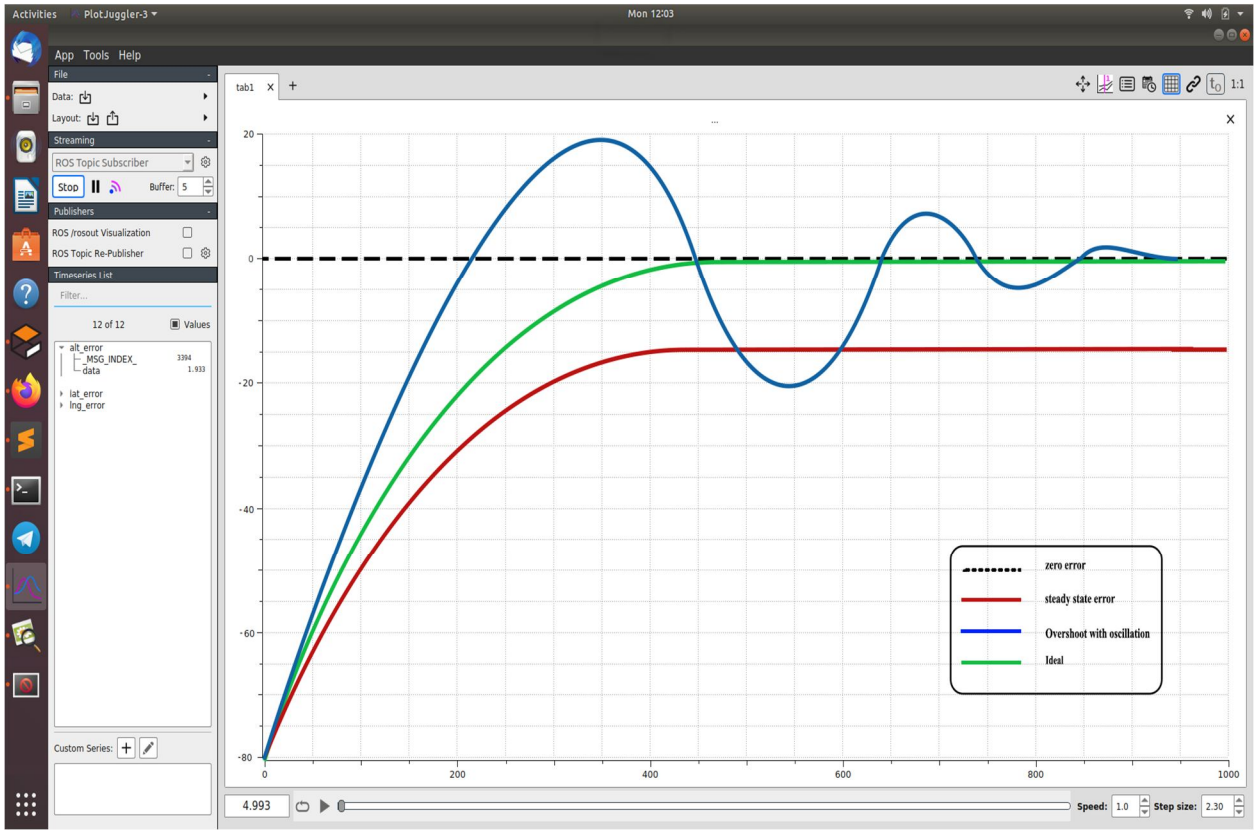


Fig.1 Error v/s Time PID graph

To find ideal Kp, Ki, Kd values, first Kp needs to be adjusted to get error v/s time graph as the blue line shown in Fig 3, that indicates overshoot with oscillations. With high Kp, the drone would reach the setpoint quickly but with overshoot and the UAV would oscillate about the setpoint. If the amplitude of oscillation increases with time, Kp is too high and should be reduced.

Once the error v/s time graph generates a damped oscillation, Kd needs to be increased till the overshoot is negligible. Too low Kd would result in oscillation and too high Kd would slow down the UAV's settling time.

Sometimes this may result in a steady-state error where the UAV would not stabilize at the exact setpoint but near it. This steady-state error is represented by the red line in the error v/s time graph. To reduce this Ki must be increased very gradually. As Int is an integral term, which accumulates over time, it must be clamped at an appropriate value.

The ideal value must result in a graph similar to the green line as shown in Fig 3 and UAV is said to be tuned.

PID controllers, when used alone, can give poor performance leading to slow settling time, overshoot, and oscillation. They also have difficulties in the presence of nonlinearities, not reacting to changing process behavior, and have lagged in responding to large disturbances. Due to this, the PID controller might work well only for a given range of max_error. For instance, the Kp, Ki and Kd values could be tuned for a 30 m increase in height of UAV (max_error = 30 m) but might result in overshoot and damped oscillation when the max_error is 10 m or slow settling time when the max_error is 50 m.

This drawback can be overcome by the Machine Learning polynomial regression model. For this, we find different gain factors of PID for different max_error(s) , by turning it on a set of max_error(s) with constant intervals (like 10m, 30m, 50m, 70m, so on..). Using these gain factor values, we train a polynomial regression model to predict a suitable gain factor for a given max_error.

©IJRASET: All Rights are Reserved

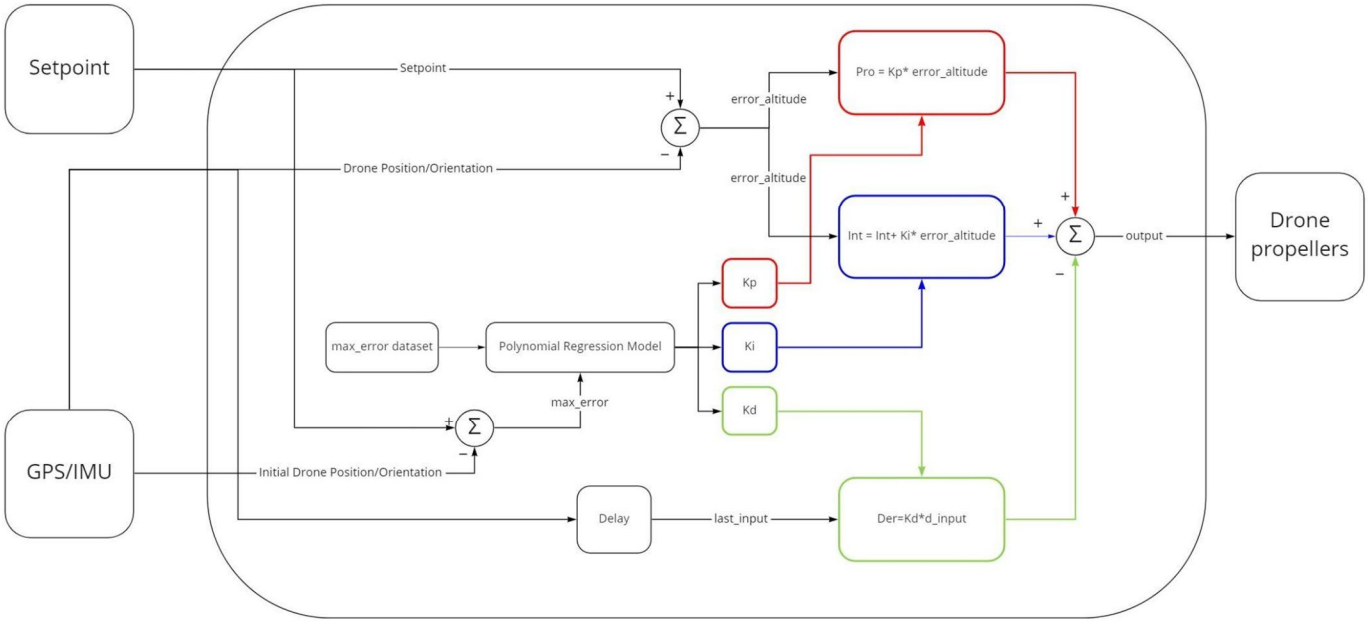


Fig. 2 PID Controller Block

For a UAV to travel in an environment after it takes off, it needs to displace linearly along latitude and longitude. Similar to altitude, a PID controller is needed to convert linear displacement of UAV along latitude and longitude to angular displacement along roll and pitch axis. This PID-generated output would denote the amplitude of pitch and roll of UAV according to the following formula. By using a Latitude Longitude to Roll Pitch (L2RP) Converter.

*pitch =(cos (drone_yaw)*output_longitude)+(sin(drone_yaw) *output_latitude)*

*roll =(cos (drone_yaw)*output_latitude)-(sin(drone_yaw) *output_longitude)*

Pitch and roll angular displacement needs to be converted to propeller speed, akin to calculating throttle, using another PID. This PID takes the above-calculated roll and pitch along with drone_roll and drone_pitch as inputs and calculates roll_error and pitch_error. This is then converted into output_roll and output_pitch which denotes the amplitude of propeller speeds as shown below. By using an RPYT (Roll Pitch Yaw Throttle) to Propeller speed Converter.

prop1=throttle+(output_roll4)-(output_pitch4)+setpoint_yaw

prop2=throttle+(output_roll4)+(output_pitch4)-setpoint_yaw

prop3=throttle-(output_roll4)+(output_pitch4)+setpoint_yaw

prop4=throttle-(output_roll4)-(output_pitch4)-setpoint_yaw

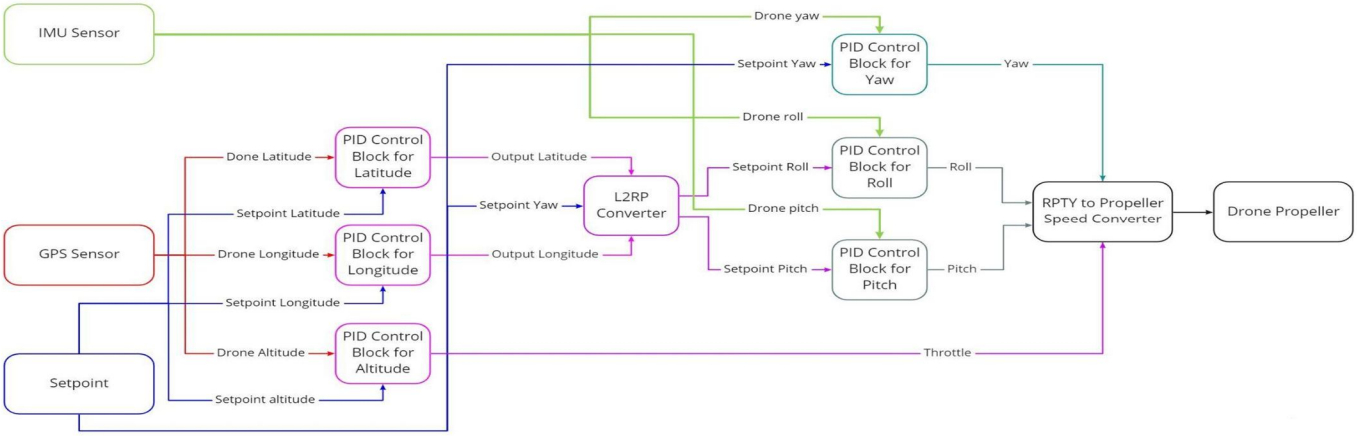


Fig. 3 Cascaded PID System Flow Chart

©IJRASET: All Rights are Reserved

V. EXPERIMENTATION AND RESULTS

The proposed 2D path planning algorithm for UAVs is implemented and simulated. The simulation environment consists of Gazebo Simulator with Robot Operating System (ROS). “ROS is a flexible framework for writing robot software. In addition to the core middleware components, ROS provides common robot-specific libraries and tools” that helped us communicate with our UAV by sending it information like propeller speeds and receiving information like proximity sensor data, GPS location, and its camera data for computer vision applications. Gazebo is an open-source 3D robotic simulator that enabled the rendering of the UAV model in a virtual city. The simulator has a reliable physics engine that is able to simulate forces like gravity, inertia, and thrust. The models are compatible with Python 2 which was used to program the UAV.

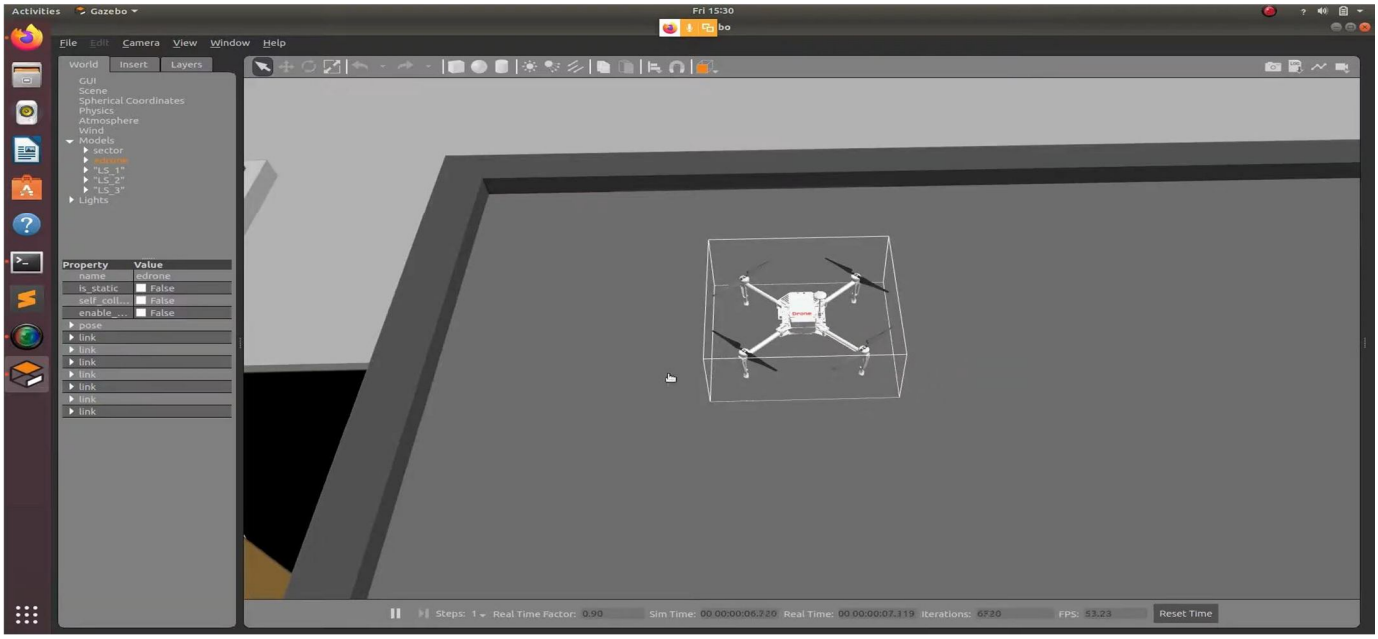


Fig.4 Takeoff

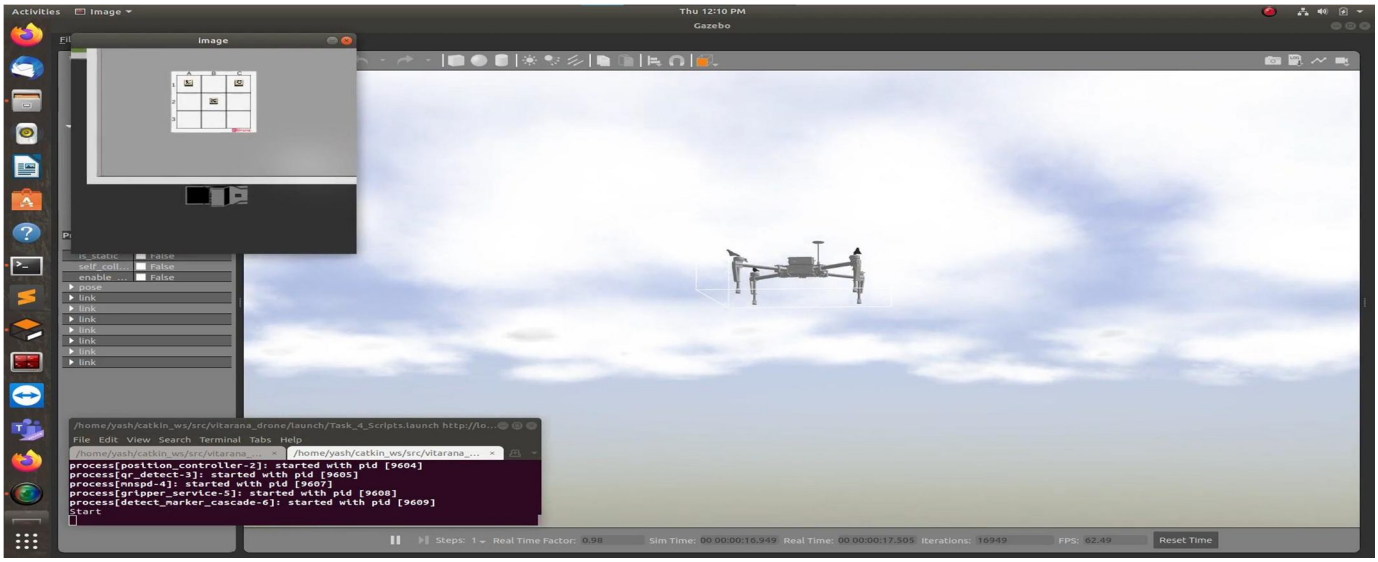


Fig.5 Hovering

As the drone takes off (Fig. 4). PID first throttles the drone to increase its height to 10 meters. Once the drone is about to reach the desired height, the derivative slows down the throttle such that the drone hovers steadily 10 meters above the ground. The integrator term removes any steady-state error if present. This hovering state is represented (Fig. 5).

©IJRASET: All Rights are Reserved

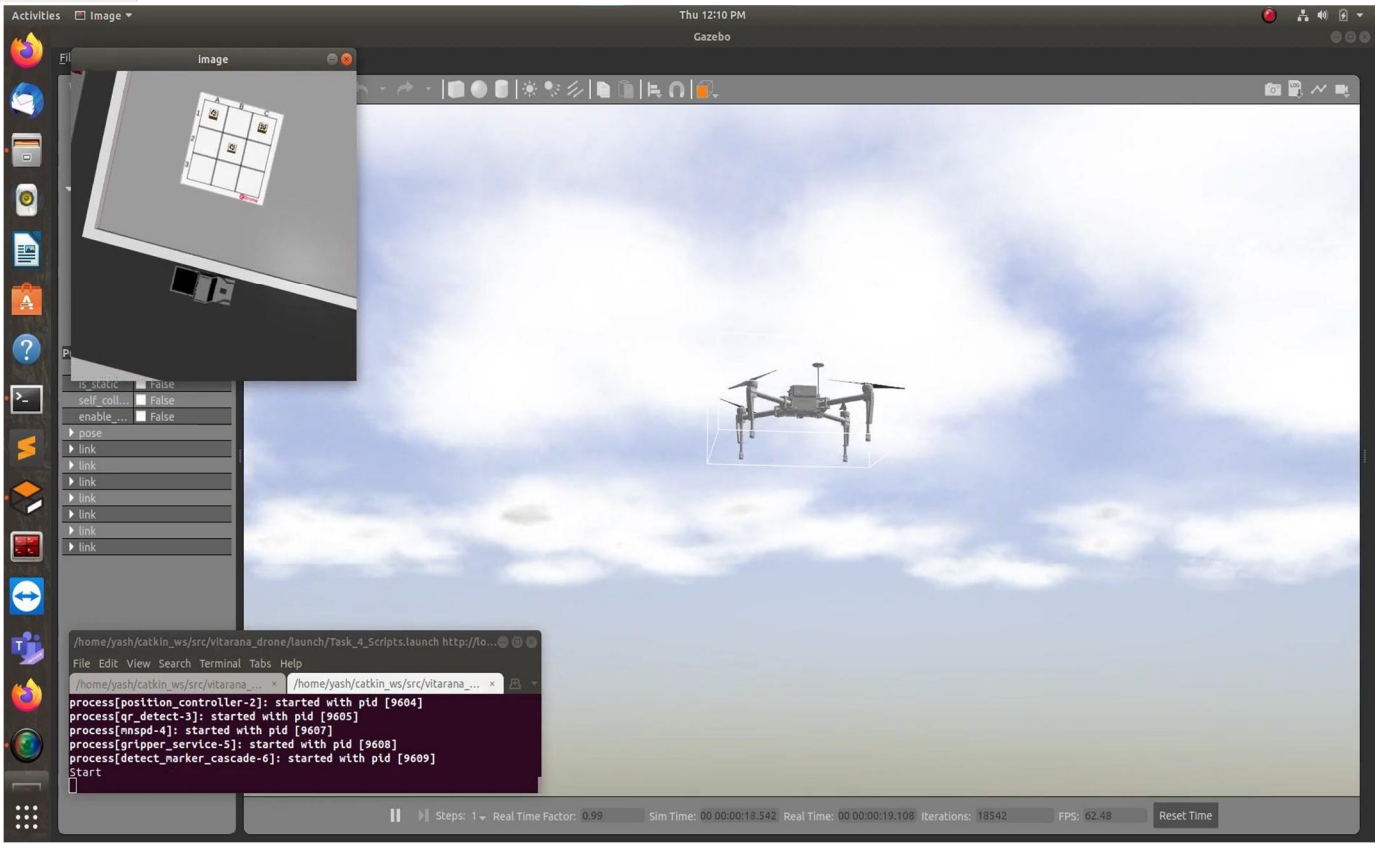


Fig.6 Yaw

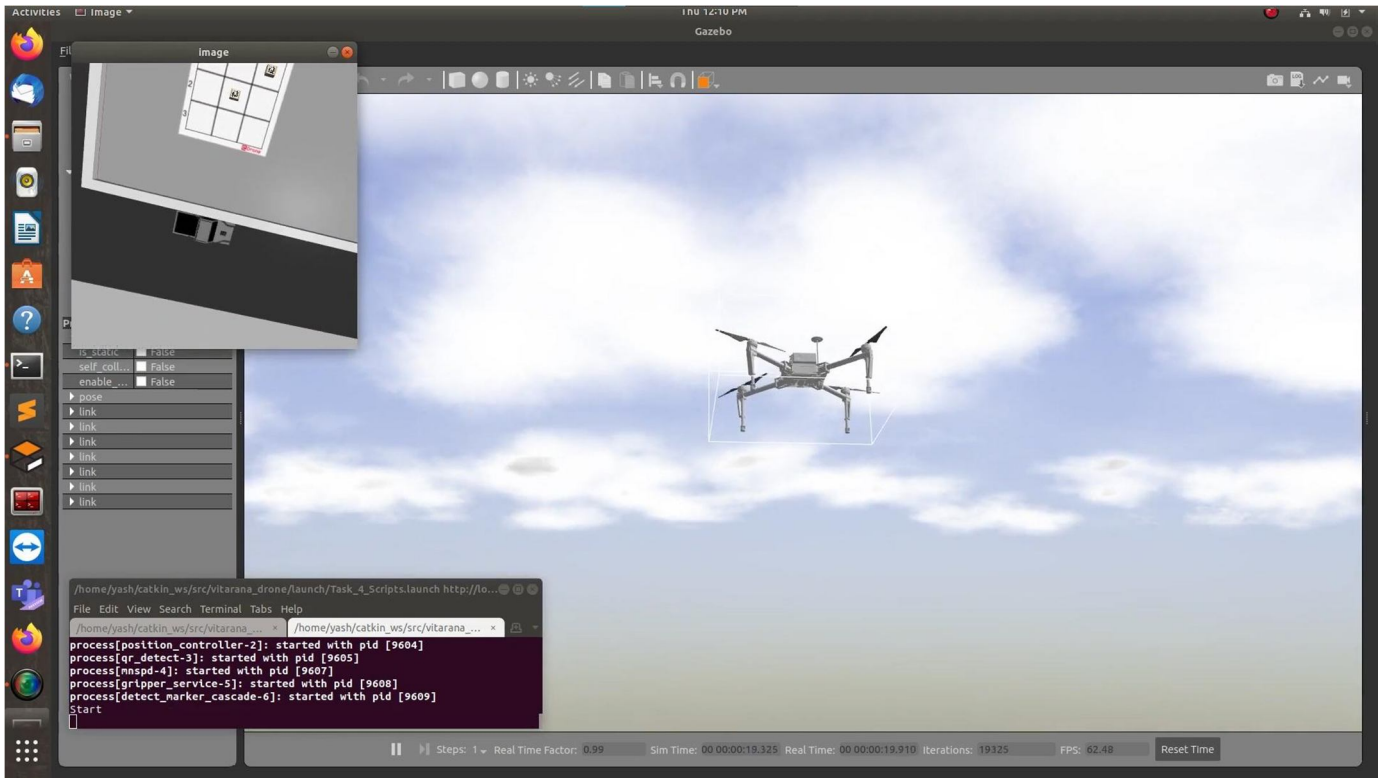


Fig.7 Forward Pitch

©IJRASET: All Rights are Reserved

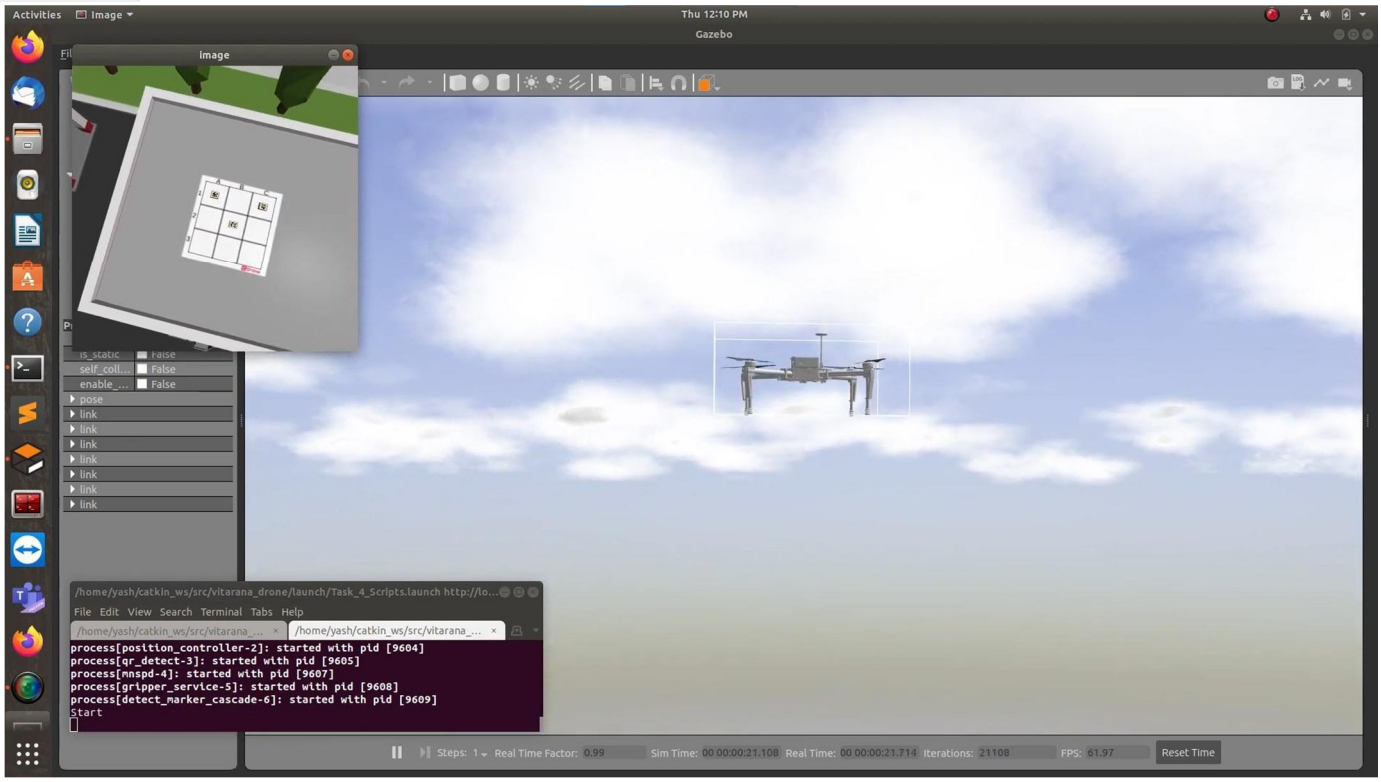


Fig.8 Backward Pitch

After reaching the desired height, the drone rotates about its yaw axis so that one of the range-finders points towards the destination (Fig. 6). The drone then starts to pitch, so as to move in the direction of the destination (Fig. 7). Once the drone is about to reach its destination, the drone pitches in the opposite direction to slow itself down as shown (Fig. 8). The drone finally lands by gradually decreasing its throttle.

VI. CONCLUSIONS

As a result of this study, we were able to successfully simulate a stable drone flight using a cascaded PID control system. Appropriate gain factors were selected using the polynomial regression model during the flight that significantly reduced the effect of environmental and system disturbances resulting in a smooth and oscillation-free flight.

REFERENCES

[1] O. McAree, J. M. Aitken and S. M. Veres, "A model based design framework for safety verification of a semi-autonomous inspection drone," 2016 UKACC 11th International Conference on Control (CONTROL), 2016, pp. 1-6, doi: 10.1109/CONTROL.2016.7737551.

[2] H. Fesenko and V. Kharchenko, "Reliability Models for a Multi-fleet of Drones with Two-level Hot Standby Redundancy Considering a Control System Structure," 2019 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), 2019, pp. 1030-1035, doi: 10.1109/IDAACS.2019.8924404.

[3] N. Yamamoto and N. Uchida, "Improvement of Image Processing for a Collaborative Security Flight Control System with Multiple Drones," 2018 32nd International Conference on Advanced Information Networking and Applications Workshops (WAINA), 2018, pp. 199-202, doi: 10.1109/WAINA.2018.00087.

[4] V. N. Taran and V. A. Detistov, "Optimal Drone Control Based on Predictive Model," 2018 International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM), 2018, pp. 1-4, doi: 10.1109/ICIEAM.2018.8728887.

[5] P. Smyczyński, Ł. Starzec and G. Granosik, "Autonomous drone control system for object tracking: Flexible system design with implementation example," 2017 22nd International Conference on Methods and Models in Automation and Robotics (MMAR), 2017, pp. 734-738, doi: 10.1109/MMAR.2017.8046919.

[6] B. Abhishek, K. Keshav, S. Gautham, D. V. R. R. Samuel and S. R. Nair, "Low cost ROS based semi-autonomous drone with position and altitude lock," 2017 IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI), 2017, pp. 2109-2112, doi: 10.1109/ICPCSI.2017.8392087.

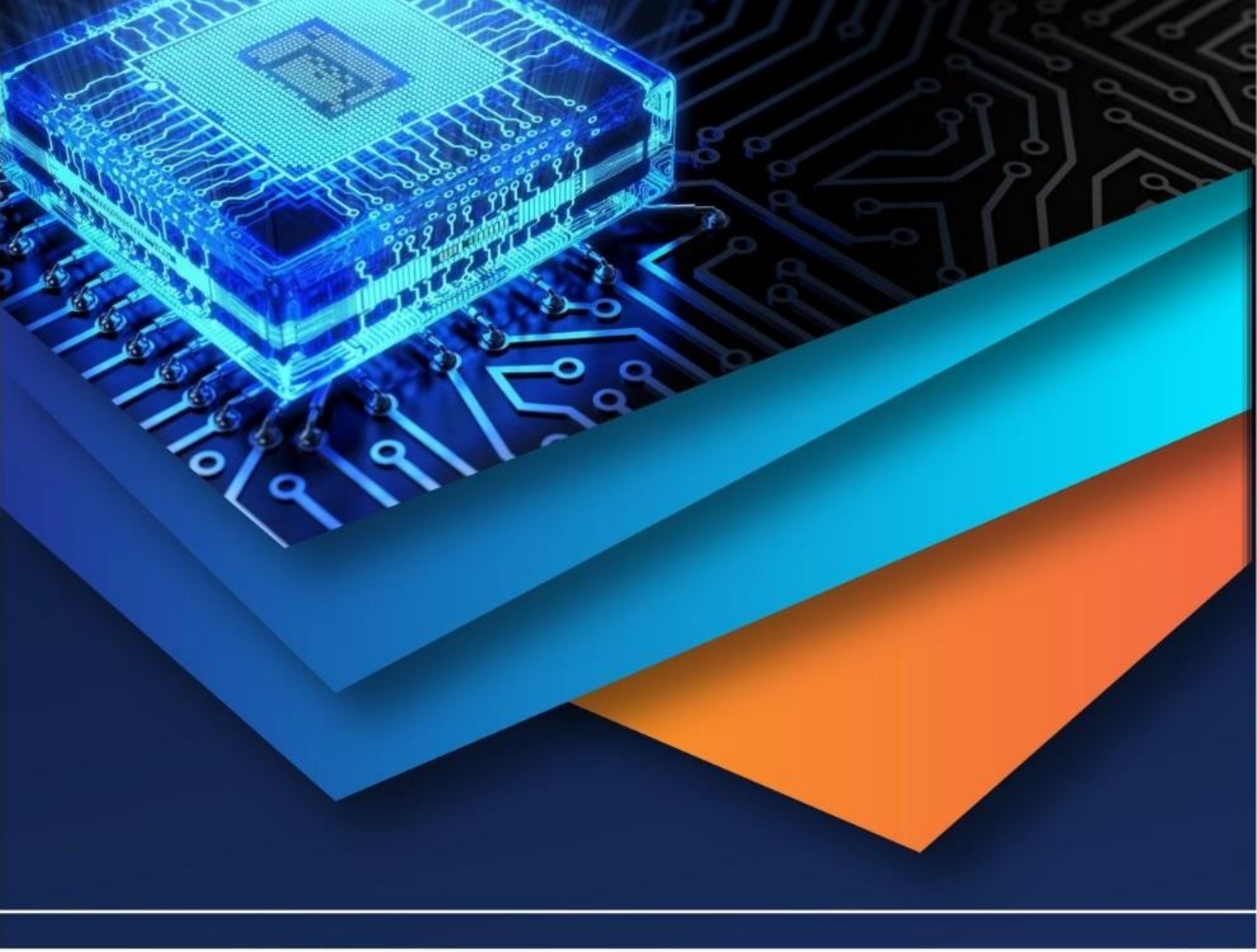
[7] K. Kersandt, G. Muñoz and C. Barrado, "Self-training by Reinforcement Learning for Full-autonomous Drones of the Future*," 2018 IEEE/AIAA 37th Digital Avionics Systems Conference (DASC), 2018, pp. 1-10, doi: 10.1109/DASC.2018.8569503.

[8] K. Natarajan, T. D. Nguyen and M. Mete, "Hand Gesture Controlled Drones: An Open Source Library," 2018 1st International Conference on Data Intelligence and Security (ICDIS), 2018, pp. 168-175, doi: 10.1109/ICDIS.2018.00035.

©IJRASET: All Rights are Reserved

[9] M. M. Vihari, U. R. Nelakuditi and M. P. Teja, "IoT based Unmanned Aerial Vehicle system for Agriculture applications," 2018 International Conference on Smart Systems and Inventive Technology (ICSSIT), 2018, pp. 26-28, doi: 10.1109/ICSSIT.2018.8748794.
[10] K. M. Zemalache and H. Maaref, "Intelligent control for a drone by self-tunable Fuzzy Inference System," 2009 6th International Multi-Conference on Systems, Signals and Devices, 2009, pp. 1-6, doi: 10.1109/SSD.2009.4956805.
[11] K. M. Yap, K. S. Eu and J. M. Low, "Investigating Wireless Network Interferences of Autonomous Drones with Camera Based Positioning Control System," 2016 International Computer Symposium (ICS), 2016, pp. 369-373, doi: 10.1109/ICS.2016.0081.
[12] R. Nouacer, H. Espinoza Ortiz, Y. Ouhammou and R. Castiñeira González, "Framework of Key Enabling Technologies for Safe and Autonomous Drones' Applications," 2019 22nd Euromicro Conference on Digital System Design (DSD), 2019, pp. 420-427, doi: 10.1109/DSD.2019.00067.
[13] A. A. Zhilenkov and I. R. Epifantsev, "System of autonomous navigation of the drone in difficult conditions of the forest trails," 2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus), 2018, pp. 1036-1039, doi: 10.1109/EIConRus.2018.8317266.
[14] H. Tanaka and Y. Matsumoto, "Autonomous Drone Guidance and Landing System Using AR/high-accuracy Hybrid Markers," 2019 IEEE 8th Global Conference on Consumer Electronics (GCCE), 2019, pp. 598-599, doi: 10.1109/GCCE46687.2019.9015373.
[15] J. C. Angarita Noriega, S. R. Prada and C. E. Moncada Guayazán, "Campus priority delivery system for the mail office of a university using an autonomous drone," 2020 IX International Congress of Mechatronics Engineering and Automation (CIIMA), 2020, pp. 1-6, doi: 10.1109/CIIMA50553.2020.9290322.
[16] A. Devos, E. Ebeid and P. Manoonpong, "Development of Autonomous Drones for Adaptive Obstacle Avoidance in Real World Environments," 2018 21st Euromicro Conference on Digital System Design (DSD), 2018, pp. 707-710, doi: 10.1109/DSD.2018.00009.
[17] X. Li, H. Huang and A. V. Savkin, "Autonomous Drone Shark Shield: A Novel Shark Repelling System for Protecting Swimmers and Surfers," 2020 6th International Conference on Control, Automation and Robotics (ICCAR), 2020, pp. 455-458, doi: 10.1109/ICCAR49639.2020.9107984.
[18] K. Liu, S. Chauhan, R. Devaraj, S. Shahi and U. Sreekumar, "Enabling Autonomous Unmanned Aerial Systems via Edge Computing," 2019 IEEE International Conference on Service-Oriented System Engineering (SOSE), 2019, pp. 374-3745, doi: 10.1109/SOSE.2019.00063.
[19] S. Suganuma, D. H. Nguyen, Y. Nishioka, K. Shimamura, K. Mori and S. Yokota, "Autopilot Drone with Rectenna for 28GHz Microwave Irradiation Measurement," 2018 Asia-Pacific Microwave Conference (APMC), 2018, pp. 1420-1422, doi: 10.23919/APMC.2018.8617142.
[20] K. M. Hasan et al., "Design and Development of an Aircraft Type Multi-functional Autonomous Drone," 2020 IEEE Region 10 Symposium (TENSYMP), 2020, pp. 734-737, doi: 10.1109/TENSYMP50017.2020.9230929.

©IJRASET: All Rights are Reserved



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)