OpenCV Implementation of Grid-based Vertical Safe Landing for UAV using YOLOv5

Hrusna Chakri Shadakshri V[1]*
Electronics & Communication Engineering
BMS College of Engineering
Bangalore, India

Veena M. B[2]
Senior IEEE Member
Electronics & Communication Engineering
BMS College of Engineering
Bangalore, India

Keshihaa Rudra Gana Dev V[3]
AXG Group
Intel Technology India Pvt Ltd
Bangalore, India

***Abstract*—The challenge of proving autonomous landing in practical situations is difficult and highly risky. Adopting autonomous landing algorithms substantially minimizes the probability of human-involved mishaps, which may enable the use of drones in populated metropolitan areas to their full potential. This paper proposes an Unmanned Aerial Vehicles (UAV) vertical safe landing & navigation pipeline that relies on lightweight computer vision modules, able to execute on the limited computational resources on-board a typical UAV. In this work, a grid-based mask technique is proposed for selecting the safe landing zones where each grid is parameterizable based on the size of the UAVs, which is implemented using OpenCV. A custom trained YOLOv5 model is the underlying building block for safe landing algorithm which is trained for aerial views of pedestrians, cars & bikes to identify as obstacles. The nearest obstacle-free zone algorithm is applied over the YOLOv5 output where boundary box locations are identified using Hue Saturation Value (HSV) filtering and then split into grids for safe landing zones where maximum coverage is taken into account while analyzing each scene. It performs a 2-level operation to prevent collisions while descending at different altitudes. Since UAV is expected to be processing only at predetermined altitudes, which will shorten the processing time, generating a PID signal for UAV actuators to navigate to the required safe zone with utmost safety and accuracy.**

Keywords*—Autonomous UAV system; computer vision algorithm; YOLOv5; safe landing site selection; Haversine equations

I. Introduction

Unmanned Aerial Vehicles (UAVs) is that uses navigation and control software powered by artificial intelligence (AI) and do not need a human pilot to fly them. These aircraft carry out activities and make decisions on their own, from takeoff and landing to conducting aerial site inspections and surveys. The utility of an autonomous UAV hinges on its ability to navigate with acceptable positional inaccuracy. The term "autonomous UAV" means a UAV that can fly without external guidance with help of onboard sensors and processors. The main problem is to build UAVs strong enough to fly independently and land safely in open fields without harming people, as in automated package delivery applications [17].

The law prohibits UAV operation over a crowd, but in practice, UAVs may fly over an unwary crowd, compromising people's safety in the event of a failure, such as a communication loss, a power shortage, or a human error. Providing every UAV with safety rules to prevent injuring people in an emergency and to select landing zones autonomously is vital. If every UAV had an emergency autonomous landing system [15], it would minimize harming people during drone accidents and enhance their urban deployment potential, especially in crowded circumstances.

Different drones use obstacle avoidance sensors such as stereo vision, ultrasonic (Sonar), time-of-flight, lidar, infrared, and monocular vision, either singly or in combination, thus fusing data for complex computations [18]. The data from these numerous obstacle avoidance sensors is fed back to the flight controller, which further uses algorithms and software to detect obstacles. The role of the flight controller is diverse, one of which is the real-time processing of visual data from the environment that is scanned by the obstacle detection sensors that will be employed in our model.

UAVs' limited processing capability owing to weight and battery power limits is one major challenge and developing UAV-compatible algorithms is difficult too. Also, Deep Neural Networks (DNNs) attaining state-of-the-art computer vision results demands a lot of processing resources for real-time operation. Utilization of single-stage object detection algorithm [8] like YOLOv5 along with OpenCV functions in the proposed architecture aids in surpassing the above concerns thus maximizing the desired outcome with minimal resources.

This paper is organized as follows. Section II described the related work for this study. In Section III, a proposed architecture for autonomous UAV safe landing algorithm is discussed. Simulation results are described in Section IV and Section V concludes the paper.

II. Related Work

Human recognition based on live input is crucial to the safety of autonomous UAV flying [1]. In order to avoid hurting people in the event of a failure, UAV safe landing demands that the UAV visually detect people [2] nearby the landing place; airspace above/near humans should be treated as a no-fly zone. Such detectors place a properly sized rectangular bounding box around each item they find on the image feed and assign it a discrete class label.

*Corresponding Author.

Early deep neural techniques [3] on human detection employed CNNs or R-CNN [4] object detection architecture, achieves a high-quality externally offered recommendations to attain good performance. In later efforts [5], such YOLOv2, employ single-stage detectors model. While speed is significantly increased by these detectors end-to-end construction in comparison to Faster R-CNN, accuracy is slightly decreased [6]. RetinaNet [7] is a different single-stage detector with detection performance relatively as good as to two-stage methods. A Feature Pyramid Network acts as the backbone on top of a ResNet architecture. Two separate sub-networks classify anchor boxes and modify values in relation to the default anchors. In this study [8], most effective model for identifying the kind of vehicle, each algorithm went through a training dataset of cars and then examined its performance. According to a study, YOLO v3 has advantages in detection speed while maintaining certain MAP i.e. 80.17% MAP (Mean Average Precision) surpasses competing approaches such as Faster R-CNN, SSD where frames per second (FPS) was more than eight times than that of Faster R-CNN. In [9], the author has conducted an experiment to check the viability of utilizing object detection methods to identify safe landing spots in case the UAV suffers an in-flight failure and compared different versions of YOLO model and it shows that YOLOv5l algorithm outperforms YOLOv4 [11]and YOLOv3 in terms of accuracy of detection while maintaining a slightly slower inference speed also a light-weight algorithm to execute worry-free procedure for power-limited UAVs.

This paper [10] inspired the idea of employing HSV color space masking to avoid water bodies and dense vegetation. This technique has no restrictions compared to prior studies. The aerial photos are segmented using color and texture cues to determine acceptable landing places.

A. Motivation

The abundance of commercial camera drones served as the motivation for this study. These drones are capable of "return to home" (RTH) and vertical landing. A number of environmental conditions, such as a sudden break in connection between the drone and the controller, instability brought on by a strong wind when the drone is in flight, or lack of power between drone parts, can cause emergencies. In this research, a model is employed for landing camera enabled drones securely without using a target as a reference landing or flying the drones back to their home location, which would be difficult and power intensive. A drone can do a vertical landing using the suggested architecture. According to the survey, YOLOv5 appears to be the most appropriate for real-time processing with a lightweight model for object detection in case of power-limited applications and using grid-based architecture maximizes the area coverage for site selection, as opposed to SLZ candidates, which are obtained as circular regions [15], where valuable portions of an obstacle-free zone may be missed. Our main objective was to offer most commercial drones a stable dynamic landing approach without the use of additional hardware or sensors.

III. Methodology

In this paper, an architecture is proposed for autonomous UAV safe landing algorithm. In the event when drone is unable to reach its home location due to low power or emergency, the purpose of our algorithm is to choose a safe landing zone inside populated areas so that it does not cause injury to any of the people within vicinity of drone and also minimal impact to the drone. Specifically, a UAV with a camera mounted on it is being considered. First, the camera is used to detect obstacles using YOLOv5 model. Second, the location of the closest safe landing zone is determined using grid-based architecture. These decisions are then relayed to the PID control block, which generates PID signals for the drone actuators so that they can navigate to the desired location as shown in Fig. 1.

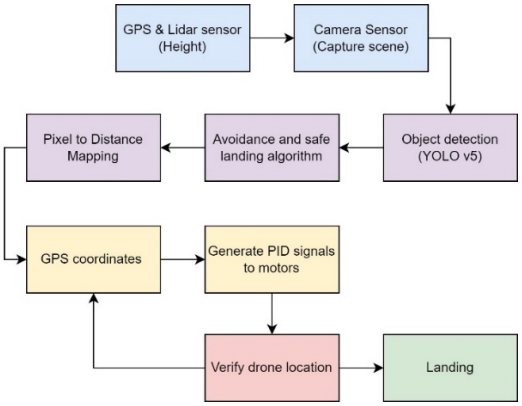


Fig. 1. Block Diagram of Overall Safe Landing Algorithm.

A. Camera FOV Distance Calculation

To calculate aerial view coverage of drone using fixed Field of View (FOV) camera is given below.

$$\tan\left(\frac{FOV}{2}\right) \times D \times 2 = \mathrm{N} \quad (1)$$

The coverage, denoted by "N," is obtained by using the field of view (FOV) of the camera and the working distance of the drone, "D". Keeping image formats as 1:1 aspect ratio, this results in a 1:1 ratio for the working distance and coverage as shown in Fig. 2(a). The fundamental highlight is the ability to determine the coverage distance based on the height of the drone. Scene 1 in Fig. 2(b) indicates that level-1 operation is being considered at an altitude of 27 meters, and coverage is 27m ×27m. Similarly, at level-2 operation at an altitude of 9m is considered will have a coverage of 9m×9m shown as scene 2. The aforementioned equation (1) is verified using the practical specifications of a drone camera listed in Table I.

B. Object Detection – YOLOv5

Object detecting methods like the single-stage detectors YOLOv5 model are utilized to avoid people, cars, and bikes on metropolitan areas. This model is trained using VisDrone datasets [12] and the Stanford Drone Dataset (SDD) [13]. Basically, YOLO models have three architectural blocks [14].

- YOLOv5 Backbone: It is used to extract image features. YOLO v5 uses CSP (Cross Stage Partial Networks) to obtain useful features from an input image.

- YOLOv5 Neck: It is used to construct feature pyramids. Feature pyramids help scaling models generalize. It helps identify objects in different sizes and scales. Feature pyramids help models perform well on new data. FPN, BiFPN, and PANet use feature pyramids. PANet generates a feature pyramids network to aggregate features and passes it to Head for prediction.
- YOLOv5 Head: Responsible for final detection. It employs anchor boxes to construct class probabilities, objectness scores, and bounding boxes.

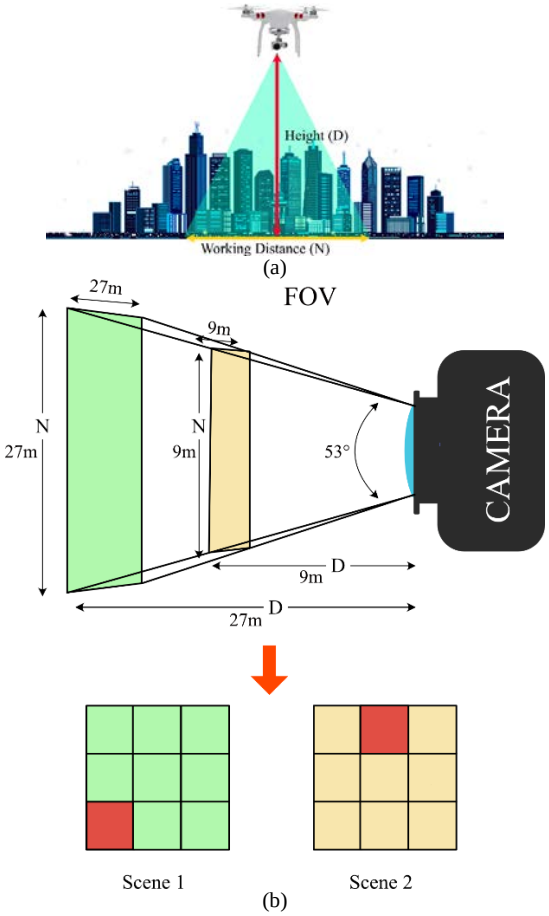


Fig. 2. (a) Drone Representation of Working Distance, (b) Working Distance Calculation.

TABLE I. PARAMETERS FOR DRONE CAMERA

Parameters	Value
Image sensor	1/4"
Image format	1:1 aspect ratio (square)
Image Pixel	1024x1024
FOV	53°
Focal length	3.2 mm
working distance	27m
Converge	27m x 27m

C. Avoidance and Identification of Safe Landing Location

The YOLOv5 output image is then provided to the avoidance system after object detection is done. Here, the image is converted to the HSV format in order to identify YOLO boundary boxes, green vegetation, water bodies such as pool, lake etc. HSV green threshold is then applied as a mask to represent the green vegetation and boundary boxes, while HSV blue represents the water bodies. The HSV range is fine-tuned using threshold of dominant color of image. The area within the boundary boxes is filled to represent an obstacle and then the image is inverted to show obstacles as being black.

The flow diagram as shown in Fig. 3, to determine the safe landing zone, a 3×3 grid is applied over the inverted image (scene-1), and at an altitude of 27 meters, each grid has a sufficient area of 9m×9m for level-1 operation as show in Fig. 2(b). After a set of safe landing zones (SLZ) are determined and stored in LUT for subsequent use. Therefore, the nearest safe landing area is considered. The geolocation is then determined by doing a pixel to distance mapping and then converting the distance to GPS coordinates. The PID control block receives these GPS coordinates and uses them to provide the necessary signals for the drone actuators to navigate to the specified safe landing site. After descending to 9m from an altitude of 27m, the above set of steps are repeated to locate a sub-zone for the drone to land securely. At altitude 9m with a 3×3 grid applied over the scene-2, the drone still has a sufficient space of 3m×3m for each grid to land safely. This can be dynamically modified depending on the size or class of the drone.

The below Table II shows that the algorithm is flexible enough to reconsider the decisions while descending to verify if the SLZ selected are truly obstacle free at lower altitudes.

TABLE II. SAFE LANDING DECISION MODES

Mode	No. of SLZ_{27m} (At level - 1)	No. of SLZ_{9m} (At level - 2)	Response
1	SLZ == 0	Don't care	TRAVERSE to New Site
2	SLZ == 1	SLZ == 0	ASCEND to Level -1; Then TRAVERSE to New Site
3	SLZ > 1	SLZ == 0	ASCEND to Level -1; Then TRAVERSE to Next nearest SLZ
4	SLZ > 0	SLZ > 0	LANDING

D. Pixel to GPS Coordinates

- Considering the captured scene is of 1:1 aspect ratio selected for computation. Firstly, a distance of 27 meters is mapped onto an image of 1024 pixels by using the map function in Python. Secondly, find the distance from the reference point (center of image) to the desired safe landing zone and calculate the destination GPS coordinate based on the distance to be travelled.

- For distance to GPS coordinates, GPRMC was utilized from the GPS NEMA sentence to obtain coordinates (longitude and latitude information) which decimal degree (DD) format.
- The following formula can be used to convert GPS data in the form of degrees, minutes, and seconds (DMS) to signed decimal degree (DD).

$$\text{GPS coordinate} = \text{degrees} + \frac{\text{minutes}}{60} + \frac{\text{seconds}}{3600} \quad (2)$$

The GPS waypoint distance is calculated using distance module in python using Algorithm 1.

Algorithm 1 Distance to GPS coordinate Mapping

Input: origin: latitude and longitude of current location

D: Distance from origin to detestation

sel: To select latitude or longitude

dir: Direction in either West/East or North/South

Output:

1: Find destination GPS coordinate lat2 or lon2.

2: Set radius of earth in km, radius = 6371

3: Compute central angle, c = D/radius of earth

4: **if** (sel is longitude) **then**

5: Compute,

$$dlon = degrees\left(\cos^{-1}\left[1 - \frac{2 \times \tan\left(\frac{c}{2}\right)^2}{\cos(lat1)^2\left[1 + \tan\left(\frac{c}{2}\right)^2\right]}\right]\right)$$

6: **if** (dir == North)

7: lon2 = dlon + lon1

8: **else if** (dir == South)

9: lon2 = lon1 – dlon

10: coordinate = lon2

11: **else if** (sel is latitude) **then**

12: Compute,

$$dlat = degrees\left(\cos^{-1}\left[1 - \frac{2 \times \tan\left(\frac{c}{2}\right)^2}{\left[1 + \tan\left(\frac{c}{2}\right)^2\right]}\right]\right)$$

13: **if** (dir == East)

14: lat2 = dlat + lat1

15: **else if** (dir == West)

16: lat2 = lat1 – dlat

17: coordinate = lat2

10: **return** coordinate

This algorithm is inspired using the Haversine equations [16]. This helps to create pre-defined points in grid to send drone to desired location as the scene captured is static.

$$\text{origin} = [\text{lat1}, \text{lon1}] \quad (3)$$

$$\text{destination} = [\text{lat2}, \text{lon2}] \quad (4)$$

$$\delta\text{lat} = \text{radian(lat1)} - \text{radian(lat2)} \quad (5)$$

$$\delta\text{lon} = \text{radian(lon2)} - \text{radian(lat2)} \quad (6)$$

Using haversine formula,

$$\alpha = \sin^2(\delta\text{lat}/2) + \cos(\text{lat1}) * \cos(\text{lat2}) \sin^2(\delta\text{lon}/2); \quad (7)$$

$$\varphi = 2 * \text{atan}\left(\sqrt{\frac{\alpha}{1-\alpha}}\right) \quad (8)$$

$$\delta = \text{radius} * \varphi \quad (9)$$

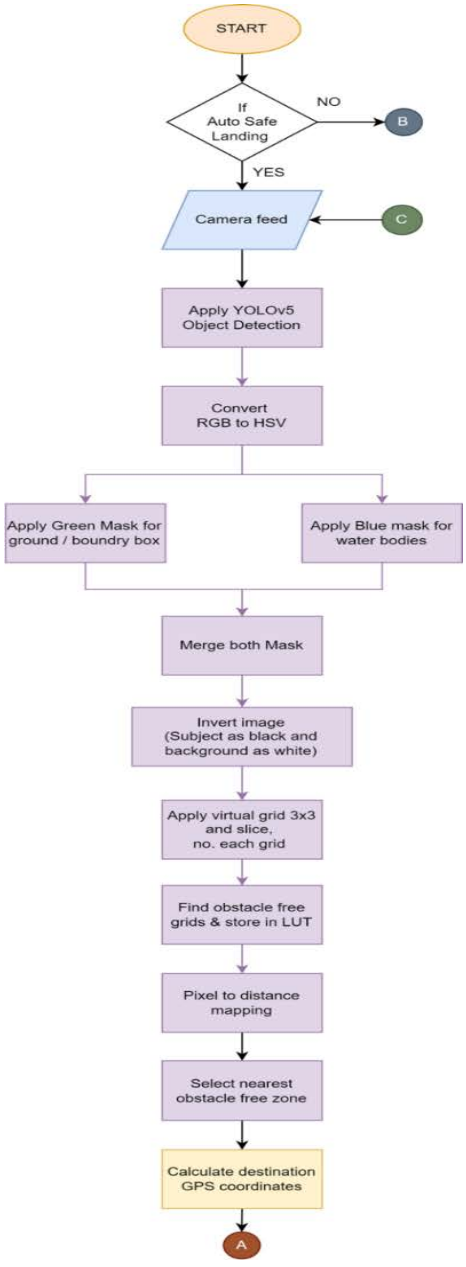


Fig. 3. Functional Flow Diagram of Safe Landing Algorithm.

E. Generate PID Signals

The command to send the drone to the desired location is done using only roll (x) and pitch (y) parameters. The current GPS location is assumed and motor control command is initiated upon receiving the safe landing zone GPS coordinates as shown in Fig. 4. In practice, drone angle is calculated from

the gyro rates of the IMU sensor and sent to the PID module to check the angle error to be corrected in the (x,y) region and calculate motor input speed. Then, PWM signals sent to the appropriate drone motors to perform actions such as roll or pitch to cruise to the desired location until the GPS coordinates latitude and longitude matches with destination coordinates. Once the drone reaches the target point, it descends to level-2 height (9m) and repeats the avoidance and location operation for scene 2.

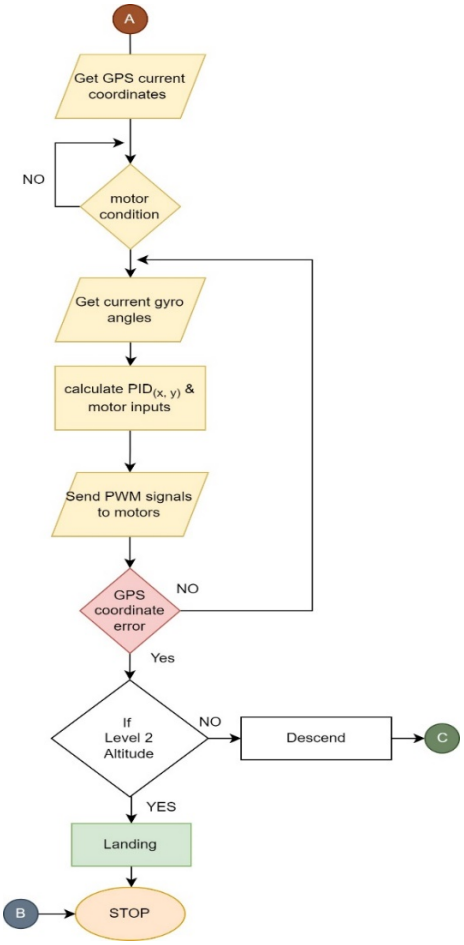


Fig. 4. Functional Flow Diagram of PID Control Block.

IV. Results

Utilizing test datasets from VisDrone and SDD pictures as input feed, the proposed safe landing technique is simulated on PC. The following figures illustrate the intermediate results for achieving the safe landing control signals generated for the drone:

A. Simulation Results of Avoidance and Safe Landing Location Algorithm

The raw image is fed into YOLOv5 model such that the objects can be identified, and the output of YOLO is shown in Fig. 5 for the objects identified with boundary boxes(green).

Fig. 5. YOLOv5 Output.

In Fig. 6, it shows the output of YOLO is then converted from RGB to HSV image using OpenCV for further image processing. Later, to mask the range of green color a threshold set for HSV image to identify the YOLO's Boundary boxes as well vegetation as shown in Fig. 6(b) where white indicates the masked colors which identifies the boundary boxes and vegetation locations within the captured scene.

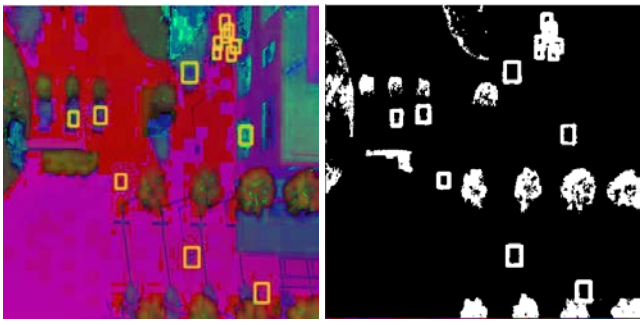


Fig. 6. (a) RGB to HSV Color Space (b) Identify Boundary Boxes.

The next step is that identified contours of the boundary boxes are filled. Then image is inverted, such that to identify the obstacles in black color as shown in Fig. 7.

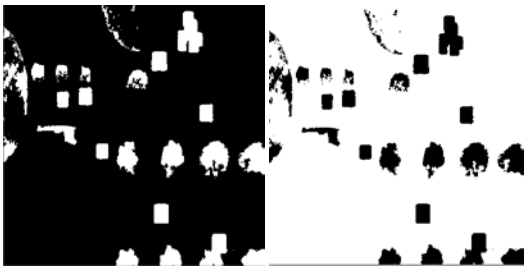


Fig. 7. (a) Fill in Objects Identified (b) Invert Image.

As shown in Fig. 8 the image is applied with grid such that to indicate each grid is sufficient area for drone to land in that select grid for safe landing area/zone. In this model, 3×3 grid is considered and size of each grid is dependent on the altitude of UAVs. At 27m altitude (level 1 scan), each grid will have as sufficient as 9m×9m grid space. And at 9m altitude (level 2 scan), grid size of 3m×3m is given for safe landing zone.

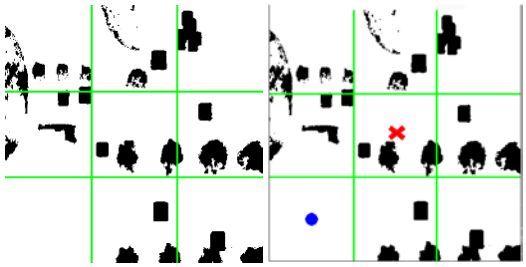
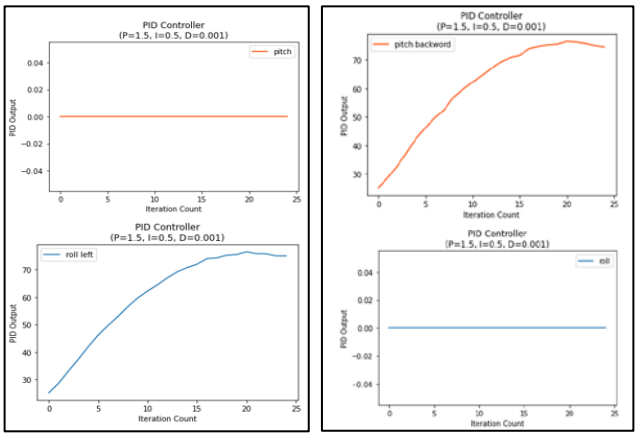


Fig. 8. (a) Apply Grids (b) Safe Landing Zone.

Each grid is split to identify if obstacles are present and is it safe to land. Here all the grids are taken individually and find if any obstacles are there by capturing the black pixels in each grid space and accordingly store all safe landing zones (SLZ) in LUT (Look-up-table). In Fig. 8(b) identifies the nearest SLZ indicated by blue spot which is the shortest distance from the reference point (red cross; resides at the center of image).

B. Simulation Results of PID Control Signals

The dimensions of image are taken as 1024 ×1024 and assuming that the drone is at 25m height and according to focal length to working distance ratio set to 1:1 ratio the Field of View will be 25m × 25m, and these values are mapped using map () function. A set of pre-defined values are given for roll/pitch angle (here, roll_angle = 12˚) which determines the speed of drone, and set until reaches destination waypoint. The below Fig. 9 shows output of each PID for drone motors i.e. roll (x) and pitch (y) plot which indicates that the drone will be navigated to desired safe landing zone location.



(a) (b)

Fig. 9. PID Control Signal – (a) x-axis(roll) (b) y-axis (Pitch).

V. CONCLUSION

In this paper, the proposed vision-based safe landing algorithm for UAVs intended for urban regions is simulated in Spyder IDE and has been verified using a real-life scene which is taken from SDD and VisDrone test datasets instead of a virtual environment. The custom YOLOv5 is trained for an aerial view of tiny objects such as humans, cars, and bikes and is carried out on a PC with an i9 processor and RTX 3060 GPU specification, which is identified successfully, especially for human detection. The grid-based decision nature of the algorithm for a safe landing will enable maximum coverage of area without missing valuable portions of obstacle-free zone. By feeding the real-life scenes, the model is verified and successfully works for all possible situations as shown in Table II where the algorithm is flexible enough to make re-decisions if unexpected obstacles occur while descending.

Since the YOLOv5 object detection model allows for the differentiation of objects, this model has a lot of potential for the future, as it can be implemented with a more intelligent system to choose and prioritize where the drone can land without creating any hazards to living beings. In addition, adequate data for tiny object detection allows the decision to be taken from higher altitudes, facilitating the selection of a suitable location for detection and landing.

REFERENCES

[1] Symeonidis, Charalampos & Kakaletsis, Efstratios & Mademlis, Ioannis & Nikolaidis, Nikos & Tefas, Anastasios & Pitas, Ioannis. “Vision-based UAV Safe Landing exploiting Lightweight Deep Neural Networks,” International Conference on Image and Graphics Processing (ICIGP),pp. 31-19 2021.

[2] Kakaletsis, E., Tzelepi, M., Kaplanoglou, P. I., Symeonidis, C., Nikolaidis, N., Tefas, A. & Pitas, I. “Semantic map annotation through UAV video analysis using deep learning models in ROS,” International Conference on Multimedia Modeling, vol 11296, pp. 328-340,2019.

[3] Zhang, S., Benenson, R., Omran, M., Hosang, J., & Schiele, B. “How far are we from solving pedestrian detection?,” IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1259-1267, 2016.

[4] Girshick, R., Donahue, J., Darrell, T., & Malik, J. “Rich feature hierarchies for accurate object detection and semantic segmentation,” IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 580-587, 2014.

[5] Lan, W., Dang, J., Wang, Y., & Wang, S. “Pedestrian detection based on YOLO network model,” IEEE International Conference on Mechatronics and Automation (ICMA), pp. 1547-1551, 2018.

[6] Ren, S., He, K., Gurshick, R. & Sun, J. “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”, Advances in Neural Information Processing Systems (NIPS), vol 28, pp. 91-99, 2015.

[7] Lin, T.-Y., Goyal, P., Girshick, R., He, K., & Dollar, P. “Focal loss for dense object detection,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 42, no. 2, pp. 318-327, 2017.

[8] J a. Kim, J. -Y. Sung and S. -h. Park, "Comparison of Faster-RCNN, YOLO, and SSD for Real-Time Vehicle Type Recognition," IEEE International Conference on Consumer Electronics - Asia (ICCE-Asia), pp. 1-4, 2020.

[9] Nepal, U.; Eslamiat, H., “Comparing YOLOv3, YOLOv4 and YOLOv5 for Autonomous Landing Spot Detection in Faulty UAVs,” Sensors 2022, vol. 22, 464, 2022.

[10] Dehshibi, M.M., Fahimi, M.S., Mashhadi,M. “Vision-Based Site Selection for Emergency Landing of UAVs,” Advances in Intelligent Systems and Computing, Springer Cham,vol 361,pp. 397-402, 2015.

[11] Meena Deshpande, Veena M.B., “License Plate Detection and Recognition using YOLO v4,” Webology, vol.18, no.4, 2021.

[12] P. Zhu et al., "Detection and Tracking Meet Drones Challenge," IEEE Transactions on Pattern Analysis and Machine Intelligence, 2021.

[13] A. Robicquet, A. Sadeghian, A. Alahi, S. Savarese, “Learning Social Etiquette: Human Trajectory Prediction In Crowded Scenes” in European Conference on Computer Vision (ECCV), vol 9912, pp. 549-565, 2016.

[14] YOLOv5 Glenn Jocher, Ayush Chaurasia, Alex Stoken, Jirka Borovec, NanoCode012, Yonghye Kwon, TaoXie, Kalen Michael, Jiacong Fang, imyhxy, Lorna, Colin Wong, Zeng Yifu, Abhiram V, Diego Montes, Zhiqiang Wang, Cristi Fati, Jebastin Nadar, Laughing, … xylieong. (2022). ultralytics/yolov5: v6.2 - YOLOv5 Classification Models, Apple M1, Reproducibility, ClearML and Deci.ai integrations (v6.2). Zenodo. https://doi.org/10.5281/zenodo.7002879.

[15] J. González-Trejo, D. Mercado-Ravell, I. Becerra and R. Murrieta-Cid, "On the Visual-Based Safe Landing of UAVs in Populated Areas: A Crucial Aspect for Urban Deployment," in IEEE Robotics and Automation Letters, vol. 6, no. 4, pp. 7901-7908, Oct. 2021.

[16] Gong, Yikai & Deng, Fengmin & Sinnott, Richard ”Identification of (near) Real-time Traffic Congestion in the Cities of Australia through Twitter”, ACM First International Workshop, pp. 7-12, 2015.

[17] Anne Goodchild & Jordan Toy, “Delivery by drone: An evaluation of unmanned aerial vehicle technology in reducing CO2 emissions in the delivery service industry,” Transportation Research Part D: Transport and Environment, vol 61, Part A, pp. 58-67, 2018.

[18] Arfaoui, Aymen. “Unmanned Aerial Vehicle: Review of Onboard Sensors, Application Fields, Open Problems and Research Issues,” International Journal of Image Processing, vol 11, pp. 12-24, 2017.