Simulation and Real-Life Implementation of UAV Autonomous Landing System Based on Object Recognition and Tracking for Safe Landing in Uncertain Environments

Ranjai Baidya and Heon Jeong *

Posted Date: 12 January 2024

doi: 10.20944/preprints202401.1037.v1

Keywords: Intelligent autonomous system; Autonomous Landing; Obstacle avoidance; Object detection; Quadrotors; Deep SORT; YOLOv5; Distance Transform; PID control

Article

Simulation and Real-Life Implementation of UAV Autonomous Landing System Based on Object Recognition and Tracking for Safe Landing in Uncertain Environments

Ranjai Baidya [1] and Heon Jeong [2,*]

[1] Kpro System, 673-1 Dogok-ri, Wabu-eup, Namyangju-si, Gyeonggi-do, [12270] South Korea; ranjai123baidya@gmail.com

[2] Department of Fire Service Administration, Chodang University, 80, Muanro, Muaneup, Muangun, Jeollanamdo, [58530] South KoreaMuan

* Correspondence: hjeong@cdu.ac.kr; Tel.:+82-61-450-1229

Abstract: The use of autonomous Unmanned Aerial Vehicles (UAVs) has been increasing, and the autonomy of these systems and their capabilities in dealing with uncertainties is crucial. Autonomous landing is pivotal for the success of an autonomous mission of UAVs. This paper presents an autonomous landing system for quadrotor UAVs with the ability to perform smooth landing even in undesirable conditions like obstruction by obstacles in and around the designated landing area and inability to identify or the absence of a visual marker establishing the designated landing area. We have integrated algorithms like version 5 of You Only Look Once (YOLOv5), DeepSORT, Euclidean distance transform, and Proportional-Integral-Derivative (PID) controller to strengthen the robustness of the overall system. While the YOLOv5 model is trained to identify the visual marker of the landing area and some common obstacles like people, cars, and trees, the DeepSORT algorithm keeps track of the identified objects. Similarly, using the detection of the identified objects and Euclidean distance transform, an open space without any obstacles to land could be identified if necessary. Finally, the PID controller generates appropriate movement values for the UAV using the visual cues of the target landing area and the obstacles. To warrant the validity of the overall system without risking the safety of the involved people, initial tests are performed, and a software-based simulation is performed before executing the tests in real life. A full-blown hardware system with an autonomous landing system is then built and tested in real life. The designed system is tested in various scenarios to verify the effectiveness of the system. The code is available at this repository: https://github.com/rnjbdya/Vision-based-UAV-autonomous-landing.

Keywords: Intelligent autonomous system; Autonomous Landing; Obstacle avoidance; Object detection; Quadrotors; Deep SORT; YOLOv5; Distance Transform; PID control

1. Introduction

Unmanned Aerial Vehicles (UAVs) are gaining wider acceptance due to their advantages over manned flights [1]. UAVs are versatile and used in fields such as transportation, search and rescue, military, surveillance, agriculture, and delivery [1]. They are also cost-effective and require fewer resources to operate and maintain. In scenarios like disaster response, firefighting, and hazardous material spills, UAVs are safer because no human lives are at risk. Furthermore, autonomous UAVs can transform various sectors, making them a topic of increasing interest [2].

Achieving full autonomy in UAVs requires a robust control system [3]. One of the notable challenges to achieving full autonomy is the ability to land at a marked spot. Even a minor error during UAV movement can cause significant damage to the UAV itself and its surroundings, necessitating research into methods to reduce these risks. Additionally, implementation of autonomous systems outside of a controlled environment into real-life scenarios could be much more challenging due to the

 doi:10.20944/preprints202401.1037.v1

unpredictability of the world. So, it is necessary to make these systems robust to some undesirable situations in the real world. Some of these situations are detection of multiple landing pads, the presence of obstructive objects near the landing target or the inability to find the visual marker indicating the landing target. These scenarios can be seen in Figure 1. Figure 1a shows the output of an object detection algorithm where two landing targets are identified by the object detection algorithm, with one of them being wrongly classified. Figure 1b shows a scenario of an obstacle in unsafe proximity of the landing target. Figures 1c and 1d both show conditions where the landing pad is not detected at all. While in Figure 1c the landing pad is fully absent, in Figure 1d the landing pad is not properly visible due to reflection of light. These kinds of undesirable situations can be considered explicitly to build systems around them to deal with these cases.

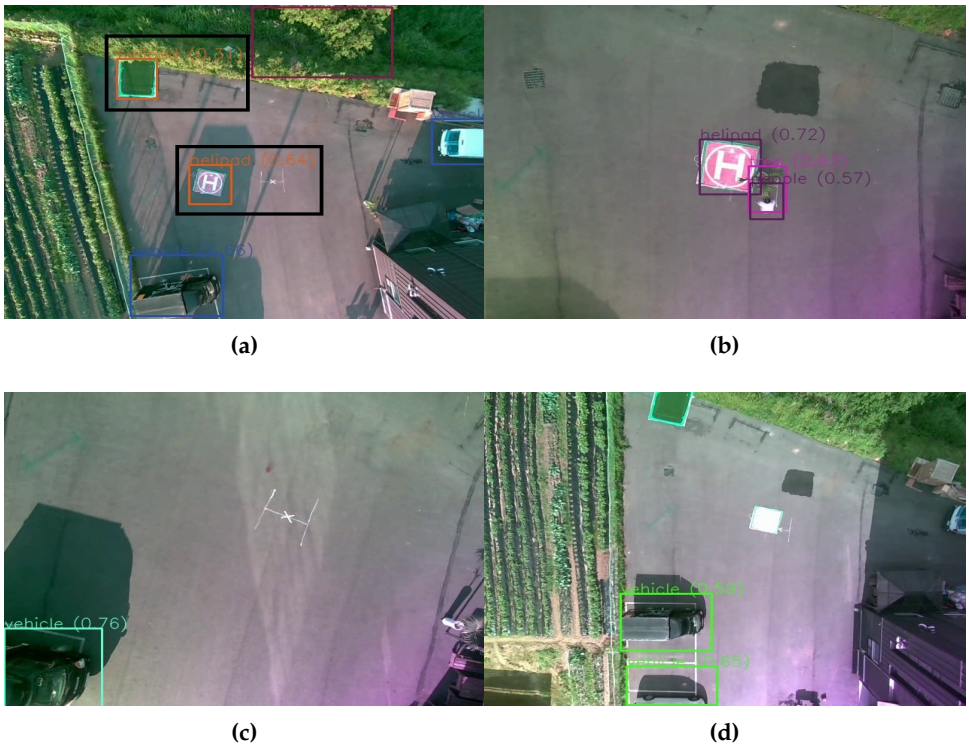
This paper first presents a simple method to perform autonomous landing with a quadrotor UAV using visual cues on top of a designated marker. Then the discussed undesirable scenarios are accounted for to increase the reliability. The considered scenarios are:

- The presence of multiple markers indicating the landing pad.
- The presence of obstacles near the marker.
- The inability to detect the designated visual marker or the absence of the marker altogether.

The overall system is built based on the Proportional-Integral-Derivative (PID) control, the You Only Look Once (YOLOv5) algorithm for object detection using a camera input, the DeepSORT algorithm for tracking, and a simple algorithm based on distance transform to find the open space. Two PID controllers generate control values for the left/right and forward/backward movement of the UAV, which utilize the discrepancies between the center of the landing target and the center of the image frame obtained from the camera pointed toward the ground. We trained the YOLOv5 algorithm on images of classes such as 'people', 'vehicle,' 'helipad,' and 'tree' for recognizing the visual marker and the obstacles. Before deploying the system in autonomous flights some initial tests are performed. First, we evaluate the capability of the system to accurately measure the distance between two points in a camera frame. To do this, we manually fly the UAV while capturing camera frames and the altitude of the UAV at the corresponding time. The distance between a point and a visual marker is then measured using a measuring tape. Then we calculate corresponding distances. After ensuring the difference between the measured distance and calculated distance is negligible, we perform a simulation of the landing process using the Software in the Loop simulator of Ardupilot. After finetuning the system in both ways, we deployed the system in the real world. The final system successfully performs autonomous landing in the designated spot in case of absence of any unfavourable scenarios and in a safe spot which may or may not be the designated spot in presence of some uncertainty. The main contributions of this paper can be listed as follows:

- Suggest an algorithm for safe autonomous landing of quadrotor UAVs.
- Introduction of highly likely but undesirable scenarios during vision-based autonomous landing of a quadrotor UAV and work around to deal with those scenarios.
- Integration of advanced algorithms like YOLOv5, DeepSORT, PID control, and Euclidean distance transform for better robustness of the overall system.
- Designing initial tests and simulation environment for the initial testing of the system, prior to real-life tests.
- Build and test a full-fledged hardware system with safe autonomous landing capabilities in real-life scenarios.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 12 January 2024 doi:10.20944/preprints202401.1037.v1



(a) **(b)**

(c) **(d)**

Figure 1. Exhibition of different undesirable scenarios during vision-based autonomous landing in UAV: (a) Object detection algorithm misclassifying another object as a landing pad (black box is manually drawn to help better identify the detections), (b) presence of obstacle nearby the landing pad, (c) absence of the visual markers denoting the landing pad, (d) the present landing pad is not properly identified due to reflection of light on the pad.

2. Related Works

The escalating demand for autonomous UAVs has led to several research efforts toward designing autonomous landing and obstacle avoidance systems for these vehicles. These systems are necessary in multiple academic and industrial fields, including search and rescue, agriculture, and package delivery. Various works in the fields of autonomous landing, obstacle avoidance, and object detection have been performed.

2.1. Autonomous Landing

Several previous attempts have been made to enhance the capabilities of autonomous UAVs for autonomous landing. One such work utilized a combination of Canny edge detection; Hough transform and Hu invariant moment to detect the landing platform [4] and then perform the attitude estimation. They performed testing on a 'T-Wing' V.T.O.L, and were able to get R.M.S. errors of 4.8°, 4.2°and 4.6°during attitude estimation. Another work utilizing classical image processing technique was [5]. This work utilized Hu invariant moment to resolve the position along with differential GPS to perform the autonomous landing on an UAV helicopter , while getting an average orientation error of 7°. [6] deployed a system to perform noise removal using median filtering, then enacted imaged segmentation using a fixed threshold on Zernike moments to find the landing platform. Their experiments demonstrated that Zernike moments are superior to Hu moments in identifying the landing point robustly, even in the event of landing pad rotation. They also demonstrated R.M.S error of 4.21 cm, 1.21 cm and 0.56°in x-position, y-position and orientation respectively. [7] presented a method to trace a unique arrangement of infrared lights on a platform using an infrared camera to locate the landing point. While this method was successful 90% of the time it was only suitable for indoor use. Another work utilized optical flow estimation to track a textured platform for non-linear

 doi:10.20944/preprints202401.1037.v1

controller of vertical take-off and landing (VTOL) UAV equipped with a simple sensors setup of only camera and inertial measurement unit (IMU) [8]. [9] utilized image-based vision servoing (IBVS) to track a landing platform in a two-dimensional image and estimated the required velocity to be used as an input to the adaptive sliding mode controller. The disadvantage of this work was its reliance on a single marker to mark the landing platform, making it susceptible to loss of the marker. There has been other work to solve the problem of losing the marker by using fish-eye cameras. However, [10] utilizes simple color-based tracking, which is highly susceptible to false detections, even more so in the outdoor environments. Also [11] utilizes prior knowledge of the expected path of the target. Additionally, many works also utilize fiducial markers for a vision-based navigation during the landing. [12] has an average error of 13 cm from the landing pad but is only tested in indoor environments. [13] fails when there is a large change in speed or direction.

In recent years, deep-learning algorithms have been increasingly used for landing target detection in vision-based autonomous landing systems. [14] used faster regional neural networks (Faster R-CNN) to detect the landing pad and obtained an average error of 2.23°, 1.18 cm, 1.31 cm and 1.29 cm while estimating yaw, orientation in x-axis, y-axis and z-axis respectively. In [15], LightDenseYOLO was used instead of template matching, and the accuracy was further improved by implementing Profile Checker. [16] suggested utilizing deblur generative adversial network (DeblurGAN) to deal with non-unifrom motion-blurred inputs and furthermore YOLOv2 for the detection of the landing area.

2.2. Obstacle Avoidance

Numerous research has been performed for detecting and avoiding the obstacles around the UAV. Multiple such research is based on traditional methods. [17] detect obstacles based on the relative size changes of image patches. They utilize speeded-up robust features (SURF) for feature matching alongside template matching to compare relative obstacle sizes with different image spacing. Many works also utilize optical flow for performing obstacle avoidance. [18] utilizes optical flow for obstacle avoidance in an autonomous mobile robot. [19] utilizes optical flow alongside inertial information to avoid collisions in an autonomous helicopter. [20] combine the depth cues with optical flow to detect the obstacles in a quadrotor system.

2.3. Object Detection

Conventional approaches for object detection utilize feature extraction methods such as Histogram of Oriented Gradients (HOG) [21] or Scale Invariant Feature Transform (SIFT) [22], which require a significant amount of manual input and time. In recent years, research on object detection has shifted towards deep learning approaches that can be broadly categorized as anchor-based or anchor-free object detection architectures, based on their use of pre-defined sliding windows.

Anchor-based methods involve the classification of object boxes into distinct bins, followed by box rectification. Region-CNN (RCNN) [23], Spatial Pyramid Pooling (SPP) network [24], Fast RCNN [25], Single Shot multibox detector (SDD) [26], the You Only Look Once series of models: YOLOv1 [27], YOLOv2 [28], YOLOv3 [29], YOLOv4 [30], YOLOv5 [31] are some examples of anchor-based methods. Conversely, anchor-free methods do not engage with computations pertaining to anchor boxes, instead employing alternative methodologies. Fully Convolutional One stage (FCOS) [32], Feature Selective Anchor Free Module (FSAF) [33], YOLOX [34] are some examples of the anchor-free methods.

Object detection in general images is much different than object detection in the images captured from the field of view of UAVs. In the context of UAV-captured images, object detection poses significant challenges, chiefly due to the wide variation in shape and size of the objects of interest, as well as the potential for a high number of objects to be detected. Additionally, computing resources on UAVs are inherently limited, further complicating the task of object detection. Works like Peele [35], ClutDet [36], and DMNet [37] focus on the object's size and deploy coarse-to-fine frameworks. M-CenterNet was also introduced to deal with minimal sized objects in frames captured from aerial devices [38]. Transformer Prediction Head YOLOv5 (TPH-YOLOv5) modified YOLOv5

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 12 January 2024 doi:10.20944/preprints202401.1037.v1

architecture to accommodate the needs of object detection in UAV images [39]. TPH-YOLOv5 added an extra prediction head to solve the issue of object detection of small objects and employ self-attention mechanism in the prediction head. Additionally, convolutional attention model (CBAM) has also been utilized in TPH-YOLOv5 to To locate the region of interest in scenarios characterized by densely packed objects. Another work further modified the architecture of TPH-YOLOv5 to using ConvMixers instead of transformers in the prediction heads, to make the architecture more computationally efficient [40].

3. Methods

In this section, we will discuss the details of the individual components used in the suggested algorithm and finally the overall algorithm. The content discussed in this section will be in the following order: YOLOv5, DeepSORT, meters per pixel calculation, PID Controller, algorithm to find the empty space, undesirable scenarios and their solutions, and the overall algorithm.

3.1. You Only Look Once version 5 (YOLOv5)

The YOLOv5 [31] object detection framework is based on a single shot detection (SSD) approach that processes the entire input image in a single feed-forward pass. The architecture consists of three main components: backbone network, neck, and head. The backbone network, a feature extractor, is responsible for extracting high-level features from the input image. YOLOv5 uses the CSPNet [41] architecture as the backbone network, which is an optimized version of the ResNet architecture. The general architecture of YOLOv5 can be visualized in Figure 2.

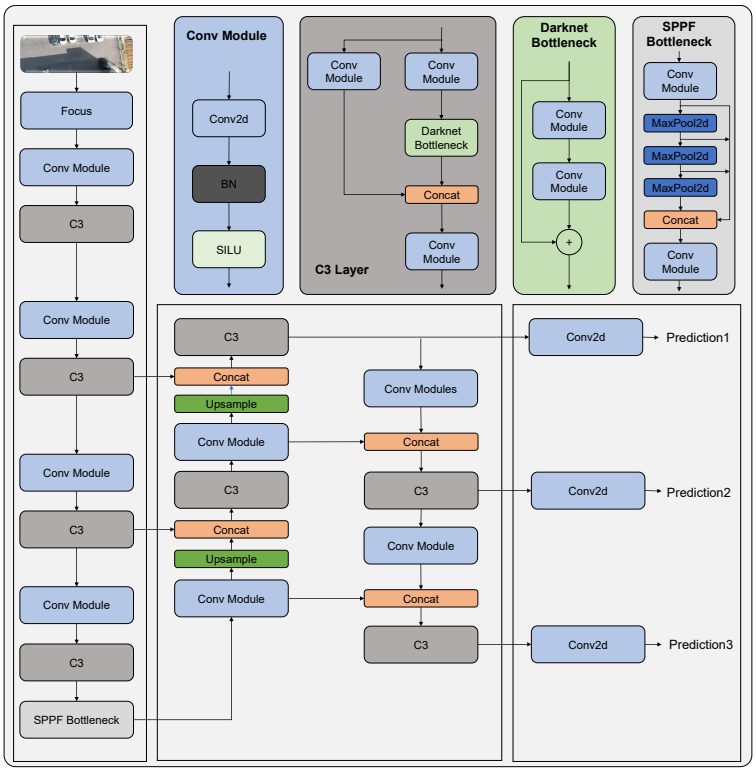


Figure 2. Model Architecture of YOLOv5.

The CSPNet [41] is composed of a stem layer, a series of CSP blocks, and a global pooling layer. The stem layer processes the input image and extracts initial features, which are then passed through a series of CSP blocks. The CSP block is a residual block that divides the input features into two branches, one with fewer channels and another with more channels. The low-channel branch is then processed through a series of convolutional layers and concatenated with the high-channel branch, which is processed through a shortcut connection.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 12 January 2024 doi:10.20944/preprints202401.1037.v1

The neck connects the backbone network to the head and is composed of a combination of convolutional and pooling layers. The neck uses the SPP (Spatial Pyramid Pooling) [24] module, which divides the input features into different scales and applies max pooling to each scale, allowing the network to capture features at different scales.

The head is responsible for predicting the bounding boxes, class probabilities, and confidence scores of the objects in the input image. The head architecture is composed of multiple convolutional and linear layers that perform the final prediction. The head uses the PAN (Path Aggregation Network) [42] module, which aggregates features from different scales and refines them to produce the final prediction.

YOLOv5 also introduces several new techniques, such as anchor-free detection, that eliminates the need for anchor boxes, which improves the accuracy of object detection. Additionally, YOLOv5 uses a hybrid approach for training, which combines both supervised and unsupervised techniques, resulting in improved model generalization and robustness.

3.2. Simple Online and Real-time Tracking with a Deep metric association (Deep SORT)

We incorporate the Deep SORT [43] algorithm for tracking the detections generated by the YOLOv5 algorithm. It capitalizes on a deep convolutional neural network to extract discriminative features from raw image data. These features contain essential information about the object's appearance and spatial information; hence, they tend to be reliable representations of these objects during tracking. Then Kalman filter-based approach enables the data association process. Here, the features extracted in the previous step and the object's state estimations are integrated by factoring motion, position uncertainties, and the previously predicted state. The tracks from the previous and the current steps are matched using the intersection over union (IoU) measure between predicted tracks and the actual detections, along with the deep feature similarities. The Hungarian algorithm is used for the best assignment of detections and the tracks for accurate correspondences. Finally, the created tracks are continuously updated based on their states, with new states being created when necessary and old ones being deleted to maintain efficiency. The algorithm adapts to changing scenarios while maintaining tracking consistency [46].

3.3. Meters per pixel calculation

Controlling the UAV based on the analysis of the input image frame in terms of pixels values could be highly inaccurate. For more precise control of the autonomous UAVs, it is necessary that the distances are in meters. In this section the formula to convert the horizontal and vertical meters per pixel is discussed. The horizontal meters per pixel value can be obtained from equation (1).

$$\text{Horizontal Meters per pixel} = \frac{2D\tan(\frac{H_{FOV}}{2})}{H_{RES}} \tag{1}$$

Similarly, the vertical meters per pixel value can be obtained from equation (2).

$$\text{Vertical Meters per pixel} = \frac{2D\tan(\frac{V_{FOV}}{2})}{V_{RES}} \tag{2}$$

Where, D is the distance between the camera and the object in frame, which we assume to be equal to the altitude measurement obtained from the lidar, HFOV and VFOV are the horizontal and the vertical field of view of the camera respectively, HRES and VRES are the vertical and horizontal resolution of the camera respectively.

3.4. Proportional Integral Derivative (PID) Controller

The precise movement of the quadrotor UAV is controlled by two PID controllers. The first controller generates the control values to control the 'Left/right movement' using the discrepancy

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 12 January 2024 doi:10.20944/preprints202401.1037.v1

between the x-coordinate of the center of the obtained image frame and the x-coordinate of the center of the landing target. The second controller takes in the difference between the y-coordinate of the center of the obtained image frame and the y-coordinate of the center of the landing target to output the control values to control the 'forward/backward movement' of the UAV. Throughout the landing, the 'yaw' of the drone is constant, and the altitude is slowly decreased by 0.5 m only if the error between the center of the image frame and the landing target is negligible and if there are no obstacles nearby the landing pad.

The overall structure of the two controllers is that of a basic PID controller. The controllers adjust the system output based on the error between the desired setpoint and the actual value of the process variables [44]. In this scenario, the system outputs are the 'left/right' and the 'forward/backward' movement of the UAV. Furthermore, the desired set points are the center coordinates of the image frames, and the process variables are the center coordinates of the landing target. The error in each step is the difference between the center coordinates of the image frame and the center coordinates of the landing target. The two controllers adjust the 'left/right' and 'forward/backward' movement utilizing the summation of proportional (K_p), integral (K_i) and derivate (K_d) components of the error. The structure of the two controllers can be visualized in Figure 3. The equation of the PID controller for the 'left/right movement control' is given by equation (3).

$$Output_{LR} = K_P * e_X(t) + K_i * \int_0^t e_X(t)dt + K_d * \frac{de_x(t)}{dt} \tag{3}$$

Similarly, the equation of the PID controller for the 'forward/backward movement control' is given by equation (4)

$$Output_{FB} = K_P * e_Y(t) + K_i * \int_0^t e_Y(t)dt + K_d * \frac{de_Y(t)}{dt} \tag{4}$$

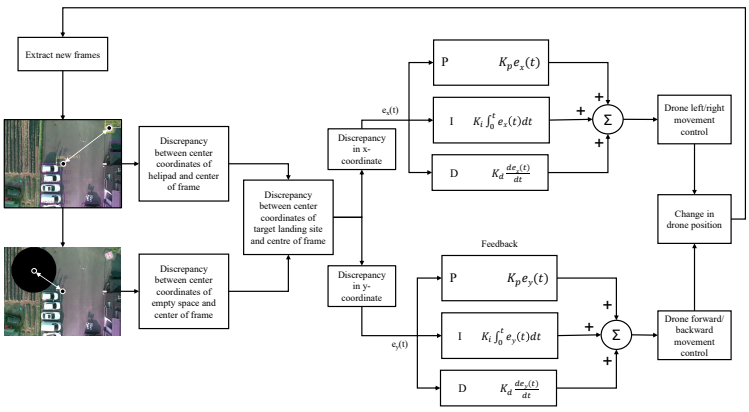


Figure 3. Visualization of the PID controllers in context of the suggested algorithm.

In equations 1 and 2, the $Output_{LR}$ and $Output_{FB}$ are the control values generated by the two controllers for controlling the left/right and the forward/backward movement of the UAV and e_x and e_y are the error in x-coordinate and y-coordinate respectively.

3.5. Finding Safe Alternate Landing Space

Performing autonomous landing on a landing target using visual cues may not always be a feasible option. The possibility of coming across some undesirable scenarios like the inability to locate the landing target, absence of the visual cues denoting the landing target, and unsafe landing conditions like the presence of other objects near the landing target, makes it necessary for adding a reliable alternate method to perform the landing. We suggest a simple algorithm using input visual

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 12 January 2024 doi:10.20944/preprints202401.1037.v1

cues to find an alternate landing spot without any obstacles near the drone. The overall process is shown in Figure 4.

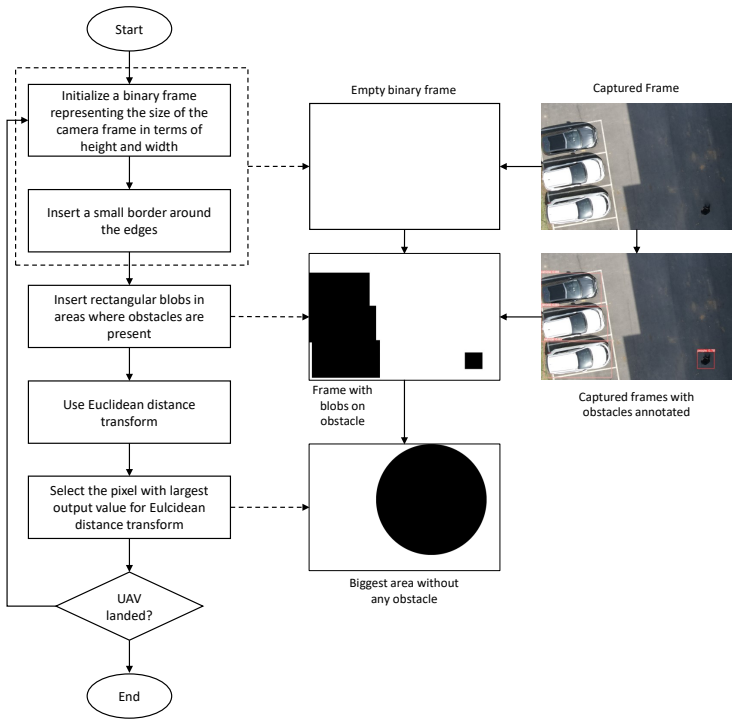


Figure 4. Flow chart for the algorithm to find the safe alternate landing space.

To find a safe alternate landing space, first an empty image frame with all white pixels and a small border of black pixels is initialized. The size of the initialized image frame is the same as that of input camera frames. Based on the position of the obstacles identified by the YOLOv5 on each input frame, the algorithm draws rectangular black blobs on the initialized image frame where the obstacles are present. Euclidean distance transform [45] is applied to the obtained image. The distance transform gives individual values to each pixel of the input frame, which represents the distance from that pixel to the closest black pixel. The biggest output value of the distance transform represents the largest open space in the image frame without any obstacles. The corresponding position of that value represents its center in terms of pixels.

3.6. Undesirable Scenarios and Their Solutions

We propose different methods to deal with the undesirable but highly plausible scenarios during vision-based autonomous landing. Here we will discuss how the targeted undesirable scenarios are dealt with.

- False detection of landing pad: Object detection techniques may not always be reliable and sometimes there may be instances where false detections are encountered. The system is designed such that it only considers the detections of the landing pad where the confidence score is above 50 %, the probability of failure of the system due to such false detections are reduced.
- Absence of a marker denoting landing pad: There are possibilities that the visual marker denoting the landing pad is not detected by the object detection algorithm. This could be caused by the reasons such as presence of the landing pad in an area out of field of view of the camera, obstruction of the landing pad, absence of the helipad altogether or sometimes even the inability of the object detection algorithm to detect the visual marker. In any of such scenarios, the designed system is capable to land in a safe landing spot. After reaching the end of the mission, in case the landing pad is not recognized, the UAV is elevated slowly to 5m higher than the

 doi:10.20944/preprints202401.1037.v1

altitude at the end of the mission. If the landing pad is recognized at any point while raising the altitude of the UAV then the UAV proceeds towards landing in the landing pad, otherwise a nearby alternate safe landing spot is considered based on the algorithm described in section 3.5.

- Presence of multiple landing pads: The object detection algorithm can also sometimes detect two landing pads in a single frame due to actual presence of such landing pads or due to some false detection. Our system only considers only the landing detection with the highest confidence and the one closest to the location at the end of the mission. Additionally, landing pad choosen are also kept in track using the Deep Sort algorithm.
- Presence of obstacles nearby the landing target: There are possibilities of presence of obstacles nearby the landing pad, which may move or remain stationary during the landing process. Eitherway, our system can proceed with the safe autonomous landing. Upon detection of obstacles nearby landing spot the system will sound a buzzer so as to notify that the drone is proceeding with landing. While sounding the buzzer, the system waits for 10 seconds so to observe whether or not the landing pad is devoid of any obstacles. If the landing pad is cleared within 10 seconds then the system proceeds with landing. Otherwise, the system proceeds to land in an alternate safe landing spot as per the algorithm discussed in section 3.5.

3.7. Algorithm for Vision-based Autonomous Landing with Adaption to Perform Safe Landing During Undesirable Scenarios

In this section we have described the overall algorithm for the vision-based autonomous landing with adaptions to perform safe landing during undesirable scenarios as described in section 3.6.

The suggested algorithm for autonomous landing requires information regarding the obstacles, the landing pad and the alternate safe landing space. These information are continually extracted and saved such that they are accessible to the rest of the program. The process of extracting these information and storing them is shown in Pseudo code Algorithm 1. The YOLOv5 and the Deep SORT algorithm are applied to each frames obtained from the camera. Based on the output of the YOLOv5 algorithm, some alternate safe landing spots are also identified, using the method mentioned in section 3.5. The locations of the landing target detection, the alternate safe landing spot, and the obstacles are saved to a queue, which is accessible by the rest of the program while performing the autonomous landing.

Pseudo code Algorithm 2 shows the overall landing algorithm. The algorithm initially monitors whether the mission has been completed and whether the autonomous landing can be started. Once the mission is complete, the PID controllers are initialized, and the algorithm searches for the detection of the landing pad in the queue described above. If there are no landing pad detections in the queue, the algorithm raises the altitude of the UAV for 10 seconds by 0.5 m after each time the detections are not present. Even after that time, if the landing pad is not found, the UAV will proceed to landing to the nearest largest empty area. If the location of the landing pad is found, then the UAV proceeds to land normally. After that, the algorithm continuously searches for obstacles near the landing spot. The algorithm proceeds to land normally if there are no obstacles detected throughout the landing process. In case some obstacles are seen within 1m of radius of the landing spot, then the UAV will wait for 10 seconds while sounding an alarm to see if the obstacle will move. When the obstacle moves away from the landing spot, the alogrithm will continue with landing. During the entire process, the algorithm finds the discrepancy between the image frame center and the landing target center. Based on whether this value is large, the algorithm decides whether to lower the altitude of the UAV. If the discrepancy is larger than 2m, the algorithm only sends the control values generated by the PID controllers for forward/backward and left/right movement of the UAV. Otherwise, the algorithm lowers the altitude of the UAV by 0.5m while also sending the control values generated by the PID controllers. We consider an altitude of 1m to be safe to perform normal landing so, once the altitude of the UAV is less than 1m, the algorithm sends "LAND" command to the FC.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 12 January 2024 doi:10.20944/preprints202401.1037.v1

Algorithm 1 Algorithm to save information regarding the landing pad, obstacles and the alternate safe landing space

Input: Queue storing captured camera frames **'q_frames'**; YOLOv5 model loaded with pre-trained weights **'yolo'**

Output: list of the landing pad detections with their position and confidence scores **'helipad'**; list of obstacle detections with their position and confidence scores **'obstacles'**; position of the largest open space available **'safe'**

```
while landing is not complete do
    pred ← yolo(q_frames.get())  ▷ yolo returns the detections with their class labels, confidence scores, and positions
    empty ← binary image frame with a small border around the edges
    for each detection p in pred do
        if p.class is one of obstacle class then
            obstacles.append(p)
            draw rectangular blob of size of p on empty at the position of p
        else if p.class is of landing pad then
            helipad.append(p)
    safe ← DistanceTransform(empty)
```

Algorithm 2 Algorithm for safe autonomous landing

Input: list of obstacles information **'obstacles'**; list of landing pad information **'helipad'**; information of largest open space available **'safe'**

```
while mission is running do
    if mission is completed then
        Break
PID ← PID controller initialization to estimate required drone movement
Target _is _heli ← True
while UAV is not landed do
    if UAV altitude is greater than 1 meter then
        Get helipad and obstacles from Algorithm 1
        if Target _ is _ heli is True then
            if helipad is not None then
                if len(helipad)>1 then
                    target ← item with highest confidence on helipad
                else
                    target ← helipad[0]
                error ← discrepancy between camera frame center and target
                movement ← PID(error)
                if movement > 2 then
                    send command to flight controller for horizontal movement
                else
                    if obstacles is not None then
                        if any o in obstacles is in close proximity to target then
                            if the counter time_of_wait is not started then
                                time_of_wait ← timer to denote the presence of obstacle o
                            Wait for obstacle to move away
                            if time_of_wait > 10 seconds then
                                Target_ is_heli ← False
                        else
                            send command to FC for horizontal + vertical movement
                    else
                        send command to FC for horizontal + vertical movement
            else
                send command to flight controller to raise Altitude by 0.2 meter
                if alt is greater than altitude at end of mission by 5 meters then
                    Target_is_heli ← False
        else
            error ← discrepancy between camera frame center and target
            movement ← PID(error)
            if movement > 2 then
                send command to FC for horizontal movement
            else
                send command to FC for horizontal + vertical movement
    else
        Send LAND command to FC
```

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 12 January 2024 doi:10.20944/preprints202401.1037.v1

4. Setups and Experiments

The implementation of the overall method was done in various steps. Initially, a test was performed to check whether the distance in meters calculated by the system is accurate. To compare the performance some existing object detection models, these models were intially trained on a publicly available dataset. Lateron based on this, YOLOv5s model was selected and trained on a self-collected dataset. A simulation of the landing conditions was performed in software before testing the scenarios in real life. This section will provide the details regarding the test to verify the accuracy of the distance calculation, selection of the object detection model, training of YOLOv5s, simulation setup, and the hardware setup.

4.1. Test for accuracy of pixels to meter conversion

One of the crucial steps to ensure the autonomous landing is to verify the accuracy of conversions while the algorithm is running. We performed a test to check whether the conversion of distance from pixels to meters is accurate. For this, we took distance measurements between various marked points using a measuring tape. The pictures of these points are taken from the camera in the UAV during a flight, and the altitude of the UAV is continuously recorded for each corresponding frame. Using the images and the corresponding altitude, the distance between the marked points is calculated using the formulae in section 3.4. The difference between the actual measurements and the calculations is then checked. Table 1 presents the results of this test. The table shows the images captured from the drone, the altitude from which the image was taken, the picture instance of the measurement being taken for the corresponding points, the calculated distance, the measured distance, and the error. It can be verified from Table 1 that the distance calculated in terms of meters based on the images captured from the UAV is accurate enough for the same method to be used in the autonomous landing algorithm.

4.2. Selection of Object Detection Model

For this we considered two object detection architectures designed specifically for use in UAV collected images and also the YOLOv5 [31] family. These models were trained on the VisDrone dataset and there performance are compared here. Additionally, these models were run on a Jetson Xavier NX board to compare the number of frames per second (FPS) that can be processed by the device for each of the models. These results have been presented on Table. While TPH-YOLOv5 [39] and CMPH-YOLOv5 [40] tend to perform better than rest in terms of precision, recall and mAP, the number of FPS possible to be processed is ideal in YOLOv5s. So we proceed with YOLOv5s model.

Table 1. Comparison results between different architectures.

Method	P(%)	R(%)	$mAP_{0.5}$(%)	mAP(%)	FPS
YOLOv5n	0.36189	0.28197	0.26161	0.13335	12
YOLOv5s	0.45251	0.3377	0.33179	0.18063	12
YOLOv5m	0.50072	0.37865	0.37837	0.21837	10
YOLOv5l	0.51648	0.39725	0.40004	0.23653	8
YOLOv5x	0.57061	0.39677	0.41358	0.24901	6
TPH-YOLOv5	0.66588	0.54958	0.59177	0.38152	5
CMPH-YOLOv5	**0.67406**	**0.55746**	**0.60015**	**0.38612**	5

4.3. Details of YOLOv5 training

Here we will discuss the details of the training process of the customized YOLOv5 model. This will include the data set used, the evaluation metrics and the results of training.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 12 January 2024 doi:10.20944/preprints202401.1037.v1

Table 2. Results from the test for verifying the accuracy of pixels to meter conversion.

ine S.N.	Image from UAV	Measurement taken	Altitude	Measured Distance	Calculated Distance	Error
1			20.0	4.3	4.5	0.2
2			20.0	3.2	3.2	0.0
3			20.1	2.5	2.7	0.2
4			20.0	2.6	2.6	0.0
5			20.0	2.0	2.0	0.0
6			20.0	2.3	2.2	0.1
7			20.1	2.4	2.5	0.1
8			20.1	2.2	2.2	0.0
9			20.0	2.7	2.6	0.1
10			20.0	3.0	3.0	0.0

4.3.1. Dataset

The dataset used in this study was self-collected through various means. A total of 683 images were collected, with most of the pictures taken by the researchers themselves, while some were generated using a generative AI model DALL-E 2 [46], and the rest are from the web. The dataset is

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 12 January 2024 doi:10.20944/preprints202401.1037.v1

composed of 4 classes: 'Helipad,' 'People,' 'Vehicle,' and 'Tree.' The 'Helipad' class represents the landing pad and the rest are the obstacles.

4.3.2. Evaluation Metrics

The study primarily emphasizes two key performance metrics for evaluating the effectiveness of the proposed model, namely, the Mean Average Precision at intersection over union (IoU) threshold of 0.5 (mAP0.5) and the overall mean Average Precision (mAP). The Mean Average Precision at intersection over union (IoU) threshold of 0.5 (mAP0.5) and IoU threshold ranging from 0.5 to 0.95 (mAP0.5:0.95) are commonly used to evaluate object detection tasks. The mAP measures the average precision of the model across multiple IoU thresholds, which determines the level of overlap required between the predicted and ground-truth bounding boxes to consider them as true positives. The mAP calculates the area under the precision-recall curve for different IoU thresholds, where the precision is the ratio of correctly predicted objects to the total number of predicted object, and the recall is the ratio of correctly predicted object to the total number in ground-truth.

The mAp is given by the following equation.

$$mAP = \frac{1}{N_{class}} \sum_{i=1}^{N_class} AP_i \tag{5}$$

Where N_{class} is the total number of classes.

Furthermore, $mAP_{0.5}$ only considers the IoU threshold of 0.5, which is a widely used threshold in object detection benchmarks. This metric measures the model's ability to correctly predict bounding boxes that have an IoU overlap of at least 50% with the ground-truth bounding boxes. The equation for $mAP_{0.5}$ is given by the following equation.

$$mAP_{0.5} = \frac{1}{N_{class}} \int_0^1 P(R)dR \tag{6}$$

Where N_{class} is also the number of classes and P and R represent precision and recall respectively at IoU threshold 0.5. Precision and recall are given by equations 7 and 8 respectively.

$$P = \frac{TP}{TP + FP} \tag{7}$$

$$R = \frac{TP}{TP + FN} \tag{8}$$

Where TP means true positives, FP means false positives, and FN means false negatives.

Likewise, $mAP_{0.5:0.95}$ considers the IoU threshold in the range of 0.5:0.95. It can be computed by taking the mean of the AP value across all classes and all IoU thresholds between 0.5 and 0.95 with a step of 0.05 (0.5, 0.55, 0.6, 0.7, 0.75, 0.8, 0.85, 0.9, and 0.95).

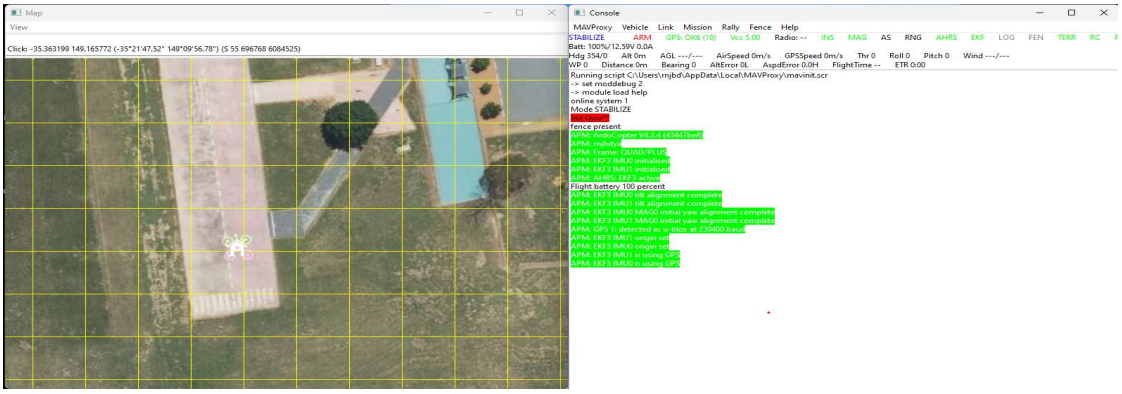
4.3.3. Results of training

The results obtained after training the YOLOv5 model on our dataset have been presented in Table 1. The trained YOLOv5 model obtained a precision value of 0.7498, which means that out of all the objects detected by the model, 74.298% of detection was correct. Furthermore, a recall of 0.60224 was obtained which means that the model was able to detect 60.224% of all the objects present in the test images. Also, the mAP0.5 value of 66.158% and $mAP_{0.5:0.95}$ was obtained.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 12 January 2024 doi:10.20944/preprints202401.1037.v1

4.4. Simulation Setup

The landing simulation was performed using a Python script that converts real-world 3D coordinates to 2D image coordinates and generates a simulated camera frame based on the Global Positioning System (GPS) and UAV orientation information of the UAV simulated on the Ardupilot Software in The Loop (SITL) software. First, an instance of Ardupilot SITL needs to be launched using which the Python script connects to the simulated UAV and continuously receives information like latitude, longitude, altitude, pitch, roll, and yaw of the simulated UAV. An image of the landing area can be passed to the script, and the image can be changed based on the desired scenario to be tested. The script simulates image frames as if they were received from the camera attached to the bottom of the UAV. The position of the landing pad is fixed to a particular location so that it would be visible in the image frames if it was in the field of view of the simulated camera. Figure 5 shows the snippets of the different parts of the simulation performed. In Figure 5, (a) is the snippet of the Ardupilot SITL software, (b) is the input helipad image, and (c) is the simulated output of the camera image frame which is based on the location and orientation of the UAV simulated in SITL software, and the input image of the landing pad. First, both the 3D real-world coordinates of the landing pad are projected onto the 2D image plane of a hypothetical camera attached to the bottom of the UAV and pointed towards the ground. The projected 2D coordinates are matched to the image coordinates, and the frames are rendered by applying perspective transformations. The script continuously receives the UAV position and orientation information and generates the camera frames till the UAV landing is complete.



(a)

(b) **(c)**

Figure 5. Snippets of different parts of the simulation software: (a) snippet of Ardupilot SITL, (b) input image of the landing pad, (b) simulation of the camera image frame based on the position and orientation of the SITL UAV and the input landing pad image.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 12 January 2024 doi:10.20944/preprints202401.1037.v1

The written script then controls the position of the simulated UAV based on the simulated frames of the attached camera and the algorithm described in section 3.5. Additionally, different landing scenarios can be simulated based on the input image of the landing pad. For example, an empty frame without any visual marker denoting a landing pad can be input to simulate the situation where there is no landing pad visible. Similarly, an input image with an object nearby can be input to the software to simulate the scenario of the presence of an unwanted object near the landing pad. In all these conditions, a simulation of the autonomous landing can be performed using Ardupilot SITL and the written Python script. The Pseudo code for this is given by Algorithm 3

Algorithm 3 Algorithm for simulating the camera frames based on SITL information

Input: location of SITL UAV in terms of metres from a reference point **'vehicle_location'**; attitude of SITL UAV **'vehicle_attitude'**;landing pad image **'target'**

vehicle_location ← location of SITL UAV
vehicle_attitude ← attitude of SITL UAV
target_location ← location of landing pad
while the program is running **do**
  vehicle_location ← updated location of SITL UAV
  vehicle_attitude ← updated attitude of SITL UAV
  vehicle_location_p ← **vehicle_location** converted in terms of pixels
  corners ← coordinates of corners of**target**
  new_corners ← **corners** translated in terms of vehicle location and attitude
  new_corners ← **new_corners** translated in terms of camera frames
  perspective_transform ← matrix for translation of corners to new_corners
  warp perspective of **target** based on **perspective_transform**
  display warped **target**

4.5. Hardware Setup

Figure 6 shows the schematic diagram of the UAV hardware setup. The parts used in the hardware setup are as follows:

- Frame: Quadcopter
- Flight Controller: Pixhawk 5x
- GPS: Pixhawk4 GPS module
- Telemetry radio: HoIybro 433MHz 100mW
- Propellers: 22-inch, pitch: 11
- Motors: MN605-S kv170
- Controller: Taranis x9d
- FRSky x8r

The companion computer, Jetson Xavier NX is where all the computation occurs. The companion computer is loaded with a script to connect to the flight controller Pixhawk 5x, extract images from the Intel Realsense 455D camera, perform the required processing, and then send the control commands to the flight controller. The script loaded on the companion computer also has the YOLOv5 algorithm loaded on it with the trained weights for the recognition of the landing pad and other objects to perform autonomous landing. For getting the accurate altitude of the UAV, a Benewake TF03 lidar is also connected to the flight controller. Human interference may also be necessary during emergencies, so a remote controller is connected to the flight controller using an FRSky x8r receiver. The status of the UAV may be monitored using a ground control station with software like Mission Planner. The UAV after complete assembly can be seen in Figure 7.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 12 January 2024 doi:10.20944/preprints202401.1037.v1

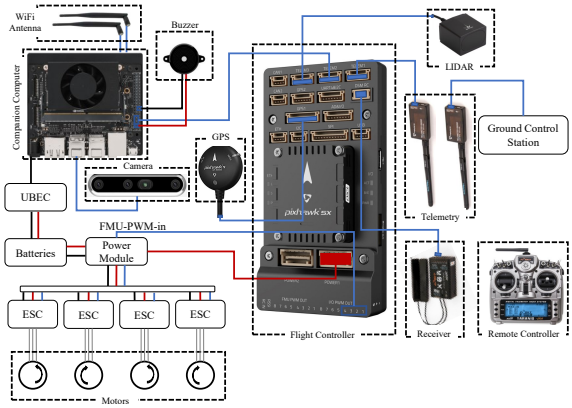


Figure 6. Schematic diagram of the hardware setup.

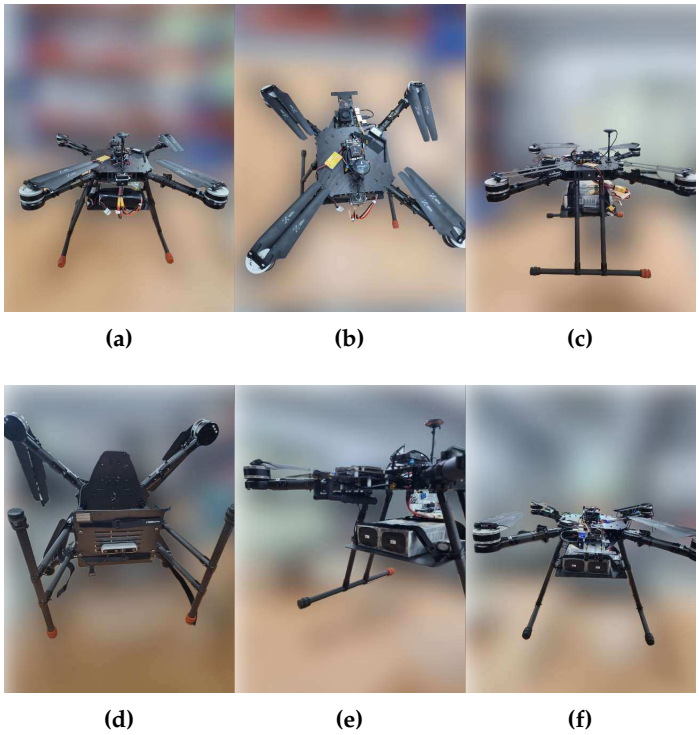


Figure 7. Fully assembled UAV from different filed of views (a) front view (b) top view (c) side view (d) bottom view (e) tilted view (f) back view.

5. Results of real-life implementation of autonomous landing in different scenarios

In this section, we present the results obtained in various scenarios. First this test, two different ways points are inputted into the UAV mission. The final waypoint is set to be near the position of the landing pad. The script running in the companion computer continuously monitors the state of the mission. Once the mission is completed, the script starts the autonomous landing based on the algorithm mentioned in section 3.5. The real-life tests of the autonomous landing were performed in the following scenarios:

- The first scenario is the most optimum landing condition when the landing pad is clearly visible once the mission is completed and there are no obstacles nearby the landing pad throughout the landing process. The results for this have been shown in Figure 8.
- The second scenario is created by not placing any visual markers that denote the landing target. The results for this have been shown in Figure 9.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 12 January 2024 doi:10.20944/preprints202401.1037.v1

- The third scenario is created by placing a plant and a human near the landing pad after it has been a while since the landing has started. The results for this have been shown in Figure 10.
- The final one is created by moving the position of the visual markers denoting the landing pad multiple times during landing procedure. The results for this have been shown in Figure 11.

Figure 8. Output for the optimum landing scenario. The images show the (a-o) image frames captured by the camera during different states of flights, graphs for the: (p) altitude, (q) buzzer state, and the control values of (r) left/right and (s) forward/backward movement of the UAV with time throughout landing process. The different state of flight in (a-o) are: (a) after take-off is completed (b) after first way-point is reached (c) after final way-point is reached (d) when autonomous landing is started (e-f) while the UAV is moving to make landing pad appear at center (g-n) after the center of frame and position of landing pad have descrepancy of less than 2m (o) when the UAV altitude is less than 1m and "LAND" command is sent to flight controller

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 12 January 2024 doi:10.20944/preprints202401.1037.v1

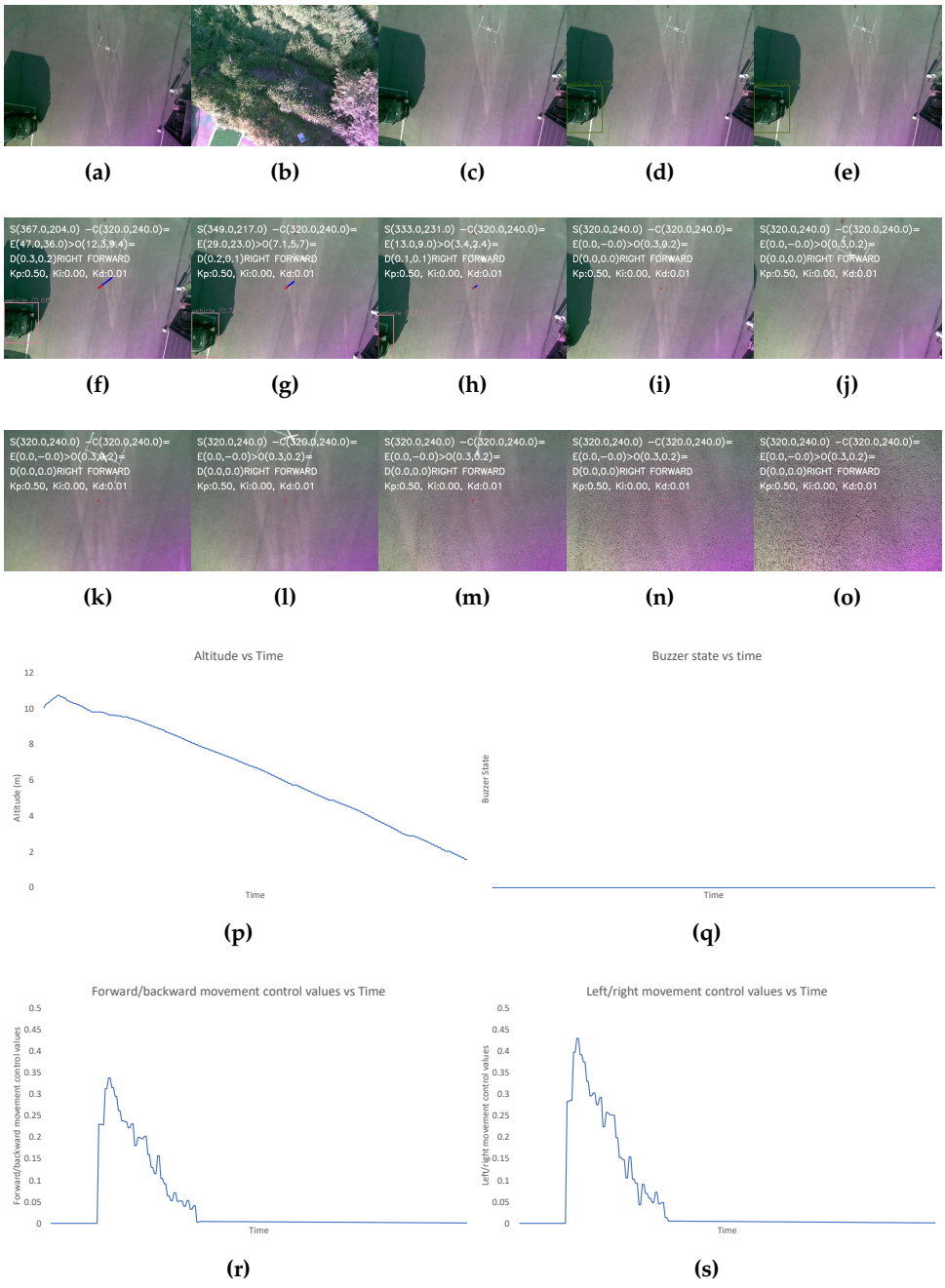


Figure 9. Output for the scenario where the visual marker denoting the landing pad is not placed. The images show the (a-o) image frames captured by the camera during different states of flights, graphs for the: (p) altitude, (q) buzzer state, and the control values of (r) left/right and (s) forward/backward movement of the UAV with time throughout the landing process. The different states of flight in (a-o) are: (a) after take-off is completed (b) after first way-point is reached (c) after final way-point is reached (d) when autonomous landing is started (e) when landing pad is not detected (f-h) while the UAV is moving to make the empty spot appear at the center (i-n) after the center of frame and position of empty spot have discrepancy of less than 2m (o) when the UAV altitude is less than 1m and "LAND" command is sent to flight controller.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 12 January 2024 doi:10.20944/preprints202401.1037.v1

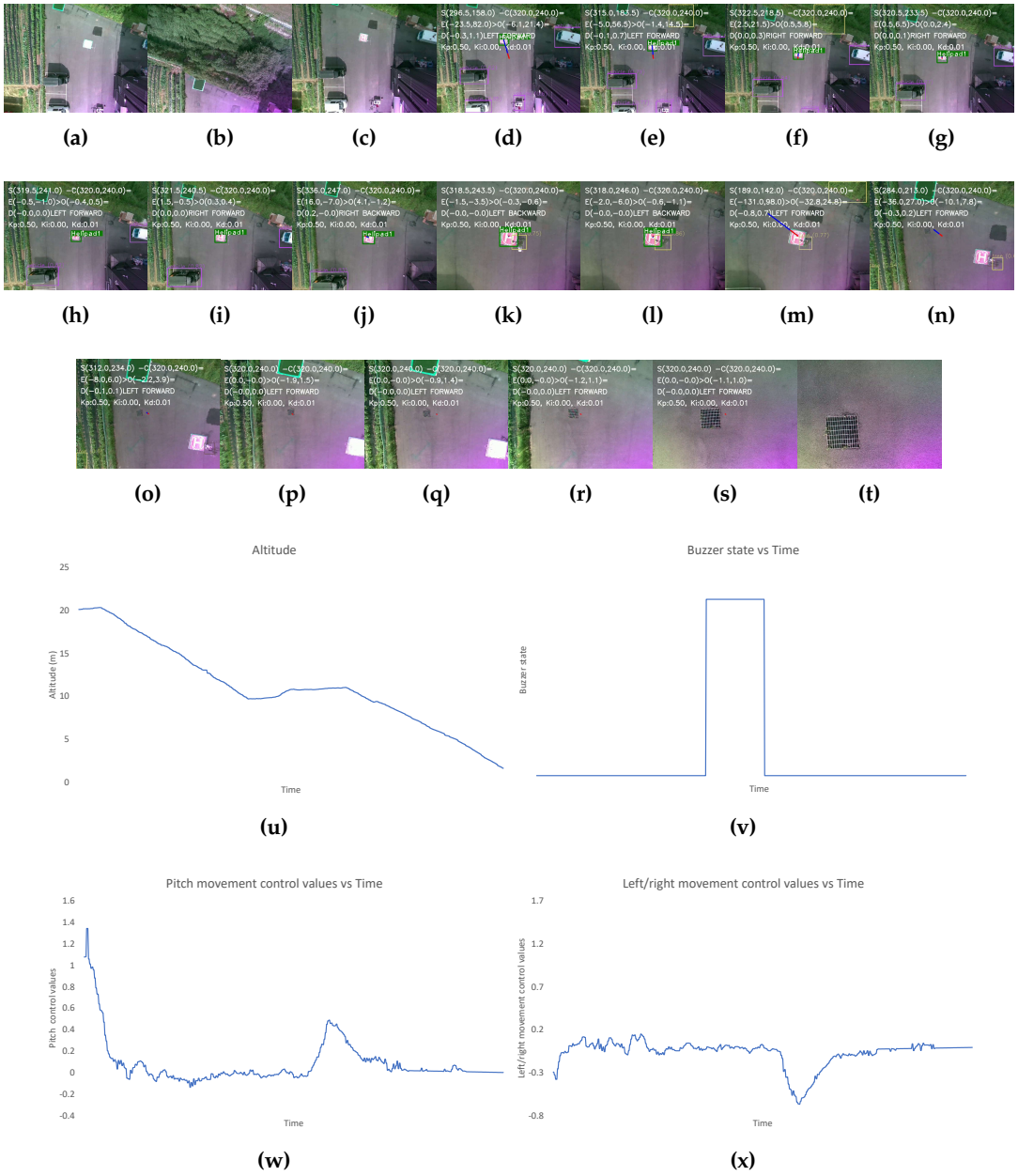


Figure 10. Output for the landing scenario when obstacles appear mid-way during the autonomous landing and donot move for more than 10 seconds. The images show the (a-t) image frames captured by the camera during different states of flights, graphs for the: (u) altitude, (v) buzzer state, and the control values of (w) left/right and (x) forward/backward movement of the UAV with time throughout the landing process. The different states of flight in (a-o) are: (a) after take-off is completed (b) after first way-point is reached (c) after final way-point is reached (d) after autonomous landing is started (e-h) while the UAV is moving to make landing pad appear at center of camera frame (i-j) after the center of frame and position of landing pad have discrepancy of less than 2m (k-l) while obstacle is detected nearby the vicinity of landing pad (m) after it is decided that the alternate spot with largest empty space on frame is the target landing spot, (n-o) while the UAV is moving to make the empty spot appear at the center (p-s) while the center of frame and position of empty spot have discrepancy of less than 2m (t) after the UAV altitude is less than 1m and "LAND" command is sent to flight controller.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 12 January 2024 doi:10.20944/preprints202401.1037.v1

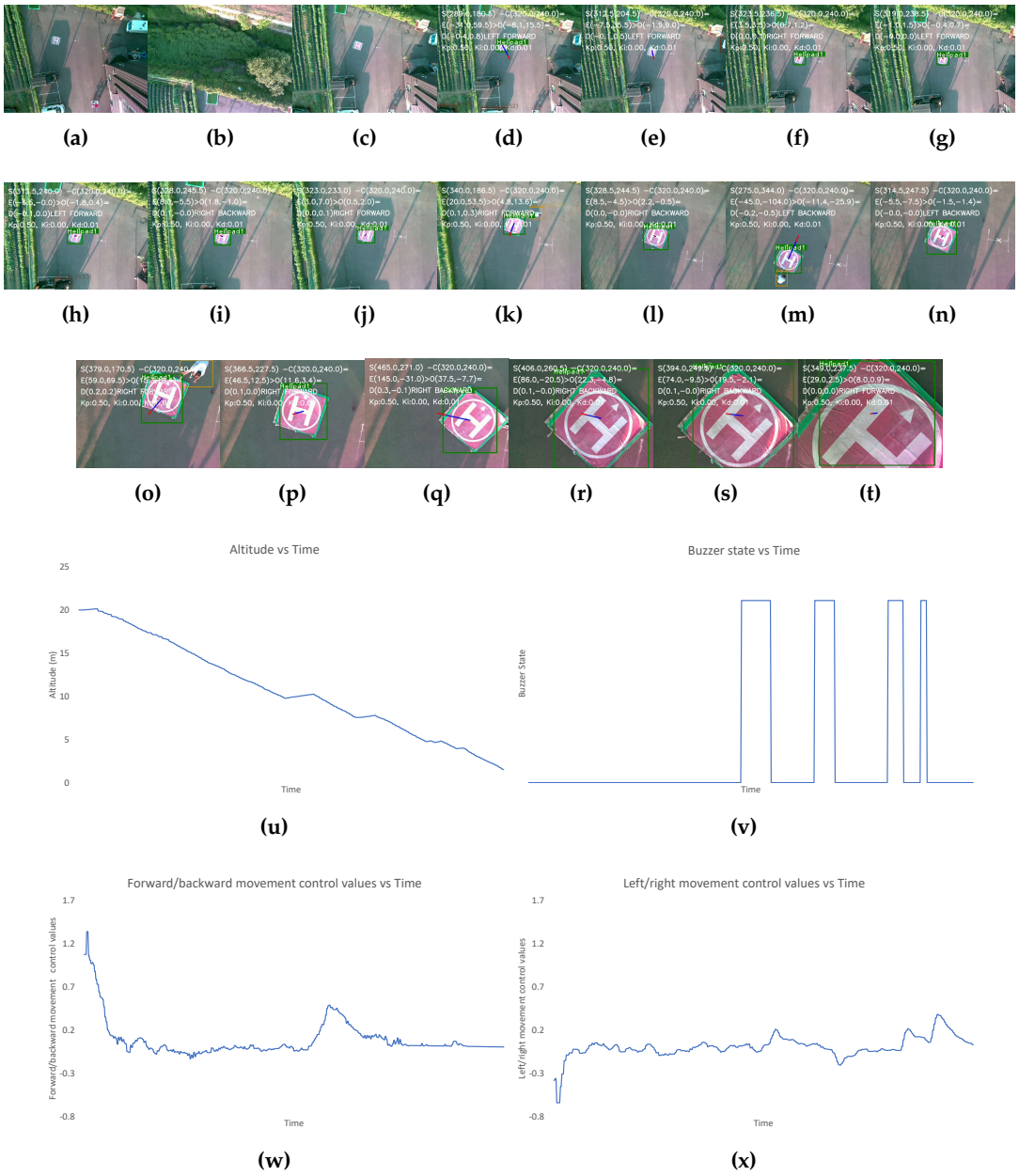


Figure 11. Output for the landing scenario where the landing pad position is changed multiple times during landing and an obstacle also appears temporarily. The images show the (a-t) image frames captured by the camera during different states of flights, graphs for the: (u) altitude, (v) buzzer state, and the control values of (w) left/right and (x) forward/backward movement of the UAV with time throughout the landing process. The different states of flight in (a-o) are: (a) after take-off is completed (b) after first way-point is reached (c) after final way-point is reached (d) after autonomous landing is started (e) while the UAV is moving to make landing pad appear at center of camera frame (f-j) after the center of frame and position of landing pad have discrepancy of less than 2m (k) when the location of landing pad is changed, an obstacle appears nearby the landing pad and the UAV should reposition itself (n) while the UAV is moving to make landing pad appear at center of camera frame (o) when the location of landing pad is changed for the second time (p-q) while the UAV is moving to make landing pad appear at center of camera frame (r-s) after the center of frame and position of landing pad have discrepancy of less than 2m (t) after the UAV altitude is less than 1m and "LAND" command is sent to flight controller.

 doi:10.20944/preprints202401.1037.v1

The results shown in Figures 8, 9, 10, and 11 include the image frames of key moments captured from the camera, the graphs of altitude, control values sent for the left/right and forward/backward movement of the UAV, and the buzzer state throughout the landing process for the four scenarios.

In Figure 8, the altitude continuously decreases due to optimum conditions and the buzzer state is also constantly low due to the absence of obstacles near the landing target. The control values for the left/right movement seem erratic and change continuously, but these values are small, so the system is still very stable. The graph for the forward/backward movement control values is smooth, indicating smooth control of the UAV forward/backward movement.

In Figure 9, initially, the altitude of the UAV seems to be increasing, which indicates that the landing pad is not identified after the completion of the mission. After 10 seconds, the algorithm deviates the UAV to land in an alternate safe spot. Then, the altitude gradually decreases. During the initial phase where the landing spot is not identified, the control values are generated for the left/right and forward/backward movement of the UAV.

In Figure 10, initially, the altitude is gradually decreasing however, after a while a constant altitude is maintained. At this point, obstacles are present near the landing pad. During this period the buzzer state is also high. Even after a certain period, the obstacle is still present, so the UAV is maneuvered toward an alternative landing spot. During this, the control values for the forward/backward and left/right movement initially change drastically since the safe landing spot without obstacles must be farther away from the UAV position because of the present obstacles.

In Figure 11, the altitude is constantly decreasing except for a few periods where the UAV maintains a constant altitude. During this, a human changes the position of the landing pad, and the human appears as an obstacle. The buzzer states are also high during these periods.

6. Discussion and Conclusion

In recent years, the usage of autonomous UAVs has seen a significant rise in various applications, such as aerial photography, surveying, and monitoring. One of the critical aspects of autonomous UAVs is their ability to perform autonomous landings even in adverse conditions of the real world. In this paper, we present a system for vision-based autonomous landing system with robust capabilities to perform autonomous landing in real-world undesirable scenarios like the inability to detect the designated visual marker or the absence of the marker altogether, the presence of multiple such markers, and the presence of obstacle nearby the marker. The proposed landing system encompasses multiple key algorithms that are integrated to collectively strengthen the system's performance. The integration of version 5 of You Only Look Once (YOLOv5), DeepSORT, Euclidean distance transform, and a Proportional-Integral-Derivative (PID) controller forms the foundation of this robust autonomous landing solution. YOLOv5 is employed to address the task of identifying both the designated landing area's visual marker and potential obstacles such as pedestrians, vehicles, and trees. The DeepSORT algorithm plays a vital role in tracking identified objects. The utilization of Euclidean distance transform in conjunction with object detection provides the system with the ability to discern open spaces devoid of obstacles within the designated landing area. The PID controllers form the control strategy of the system, generating precise movement commands for the UAV based on the visual cues of the target landing area and the detected obstacles. This controller ensures smooth and controlled maneuvers, enhancing the accuracy of the landing procedure. To establish the efficacy and safety of the proposed system, a comprehensive approach to testing and validation is adopted. Initial tests are conducted to evaluate the system's functionality, followed by a software simulation to further analyze its performance in a controlled environment. This stepwise validation strategy mitigates potential risks and allows for refining the system before real-world tests. Subsequently, a hardware system is developed, incorporating the autonomous landing system, and rigorously tested in real-life scenarios. The hardware testing validates the feasibility of implementing the proposed solution in practical applications and offers insights into its performance under dynamic conditions. In conclusion, this paper presents a comprehensive and innovative autonomous landing system tailored for quadrotor

autonomous UAVs. The integration of YOLOv5, DeepSORT, Euclidean distance transform, and a PID controller forms a synergistic approach to address the challenge of precise landings in varying conditions, including the presence of obstacles and the absence of visual markers. The system's effectiveness is established through a rigorous testing process, which encompasses initial functional tests, software simulations, and real-life hardware testing. The outcomes of these tests demonstrate the system's ability to successfully navigate complex landing scenarios, confirming its robustness and reliability. The presented autonomous landing system holds significant potential for enhancing the autonomy and adaptability of UAVs in critical missions, including search and rescue, surveillance, and package delivery. As UAV applications continue to expand, the advancement of such autonomous landing solutions becomes pivotal for ensuring safe and efficient autonomous operations.

Author Contributions: Conceptualization, R.B. and J.H.; methodology, R.B.; software, R.B.; validation, R.B. and J.H.; formal analysis R.B. and J.H.; writing, R.B.; review and editing, R.B. and J.H.; funding acquisition J.H. All authors have read and agreed to the submitted version of the manuscript.

Funding: This work was supported by Institute of Information and communications Technology Planning and Evaluation (IITP) grant funded by the Korea Government (MIST) (No. 2022-0-00530) and the "Regional Innovation Strategy (RIS)" through the National Research Foundation of Korea (NRF) and funded by the Ministry of Education (MOE) (No. 2021RIS-002).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The code and the dataset shall be made available upon request via email to the corresponding author.

Acknowledgments: In this section you can acknowledge any support given which is not covered by the author contribution or funding sections. This may include administrative and technical support, or donations in kind (e.g., materials used for experiments).

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

MDPI	Multidisciplinary Digital Publishing Institute
DOAJ	Directory of open access journals
TLA	Three letter acronym
LD	Linear dichroism

References

1. Mohsan, S.A.H.; Othman, N.Q.H.; Li, Y.; Alsharif, M.H.; Khan, M.A. Unmanned aerial vehicles (UAVs): Practical aspects, applications, open challenges, security issues, and future trends. *Intelligent Service Robotics* **2023**, *16*, 109–137.
2. Hassanalian, M.; Abdelkefi, A. Classifications, applications, and design challenges of drones: A review. *Progress in Aerospace Sciences* **2017**, *91*, 99–131.
3. Chen, H.; Wang, X.m.; Li, Y. A Survey of Autonomous Control for UAV. 2009 International Conference on Artificial Intelligence and Computational Intelligence, 2009, Vol. 2, pp. 267–271. doi:10.1109/AICI.2009.147.
4. Tsai, A.C.; Gibbens, P.W.; Stone, R.H. Terminal phase vision-based target recognition and 3D pose estimation for a tail-sitter, vertical takeoff and landing unmanned air vehicle. Advances in Image and Video Technology: First Pacific Rim Symposium, PSIVT 2006, Hsinchu, Taiwan, December 10-13, 2006. Proceedings 1. Springer, 2006, pp. 672–681.
5. Saripalli, S.; Montgomery, J.F.; Sukhatme, G.S. Vision-based autonomous landing of an unmanned aerial vehicle. Proceedings 2002 IEEE international conference on robotics and automation (Cat. No. 02CH37292). IEEE, 2002, Vol. 3, pp. 2799–2804.
6. Fan, Y.; Haiqing, S.; Hong, W. A vision-based algorithm for landing unmanned aerial vehicles. 2008 International Conference on Computer Science and Software Engineering. IEEE, 2008, Vol. 1, pp. 993–996.
7. Wenzel, K.E.; Masselli, A.; Zell, A. Automatic take off, tracking and landing of a miniature UAV on a moving carrier vehicle. *Journal of intelligent & robotic systems* **2011**, *61*, 221–238.

8. Herissé, B.; Hamel, T.; Mahony, R.; Russotto, F.X. Landing a VTOL unmanned aerial vehicle on a moving platform using optical flow. *IEEE Transactions on robotics* **2011**, *28*, 77–89.
9. Lee, D.; Ryan, T.; Kim, H.J. Autonomous landing of a VTOL UAV on a moving platform using image-based visual servoing. 2012 IEEE international conference on robotics and automation. IEEE, 2012, pp. 971–976.
10. Kim, J.; Jung, Y.; Lee, D.; Shim, D.H. Outdoor autonomous landing on a moving platform for quadrotors using an omnidirectional camera. 2014 International Conference on Unmanned Aircraft Systems (ICUAS). IEEE, 2014, pp. 1243–1252.
11. Baca, T.; Stepan, P.; Saska, M. Autonomous landing on a moving car with unmanned aerial vehicle. 2017 European conference on mobile robots (ECMR). IEEE, 2017, pp. 1–6.
12. Araar, O.; Aouf, N.; Vitanov, I. Vision based autonomous landing of multirotor UAV on moving platform. *Journal of Intelligent & Robotic Systems* **2017**, *85*, 369–384.
13. Salagame, A.; Govindraj, S.; Omkar, S. Precision Landing of a UAV on a Moving Platform for Outdoor Applications. *arXiv preprint arXiv:2209.14436* **2022**.
14. Chen, J.; Miao, X.; Jiang, H.; Chen, J.; Liu, X. Identification of autonomous landing sign for unmanned aerial vehicle based on faster regions with convolutional neural network. 2017 Chinese Automation Congress (CAC). IEEE, 2017, pp. 2109–2114.
15. Nguyen, P.H.; Arsalan, M.; Koo, J.H.; Naqvi, R.A.; Truong, N.Q.; Park, K.R. LightDenseYOLO: A fast and accurate marker tracker for autonomous UAV landing by visible light camera sensor on drone. *Sensors* **2018**, *18*, 1703.
16. Truong, N.Q.; Lee, Y.W.; Owais, M.; Nguyen, D.T.; Batchuluun, G.; Pham, T.D.; Park, K.R. SlimDeblurGAN-based motion deblurring and marker detection for autonomous drone landing. *Sensors* **2020**, *20*, 3918.
17. Mori, T.; Scherer, S. First results in detecting and avoiding frontal obstacles from a monocular camera for micro unmanned aerial vehicles. 2013 IEEE International Conference on Robotics and Automation. IEEE, 2013, pp. 1750–1757.
18. Souhila, K.; Karim, A. Optical flow based robot obstacle avoidance. *International Journal of Advanced Robotic Systems* **2007**, *4*, 2.
19. Muratet, L.; Doncieux, S.; Briere, Y.; Meyer, J.A. A contribution to vision-based autonomous helicopter flight in urban environments. *Robotics and Autonomous Systems* **2005**, *50*, 195–209.
20. Peng, X.Z.; Lin, H.Y.; Dai, J.M. Path planning and obstacle avoidance for vision guided quadrotor UAV navigation. 2016 12th IEEE International Conference on Control and Automation (ICCA). IEEE, 2016, pp. 984–989.
21. Dalal, N.; Triggs, B. Histograms of oriented gradients for human detection. 2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05). Ieee, 2005, Vol. 1, pp. 886–893.
22. Lowe, D.G. Distinctive image features from scale-invariant keypoints. *International journal of computer vision* **2004**, *60*, 91–110.
23. Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. Proceedings of the IEEE conference on computer vision and pattern recognition, 2014, pp. 580–587.
24. He, K.; Zhang, X.; Ren, S.; Sun, J. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence* **2015**, *37*, 1904–1916.
25. Girshick, R. Fast r-cnn. Proceedings of the IEEE international conference on computer vision, 2015, pp. 1440–1448.
26. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.Y.; Berg, A.C. Ssd: Single shot multibox detector. Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14. Springer, 2016, pp. 21–37.
27. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You only look once: Unified, real-time object detection. Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 779–788.
28. Redmon, J.; Farhadi, A. YOLO9000: better, faster, stronger. Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 7263–7271.
29. Redmon, J.; Farhadi, A. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767* **2018**.
30. Bochkovskiy, A.; Wang, C.Y.; Liao, H.Y.M. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934* **2020**.

31. Jocher, G.; Alex, S.; Borovec, J.; NanoCode012.; Chauarasia, A.; TaoXie.; Changyu, L.; V, A.; Laughing.; tkianai.; yxNONG.; Hogan, A.; lorenzomammana.; AlexWang1900.; Hajek, Jan amd Diaconu, L.; Marc.; Kwon, Y.; oleg.; wanghaoyang0106.; Defretin, Y.; Lohia, A.; ml5ah.; Milanko, B.; Fineran, B.; Khromov, D.; Yiwei, D.; Doug.; Durgesh.; Ingham, F. Yolov5. opensourcecode.
32. Tian, Z.; Shen, C.; Chen, H.; He, T. Fcos: Fully convolutional one-stage object detection. Proceedings of the IEEE/CVF international conference on computer vision, 2019, pp. 9627–9636.
33. Zhu, C.; He, Y.; Savvides, M. Feature selective anchor-free module for single-shot object detection. Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2019, pp. 840–849.
34. Ge, Z.; Liu, S.; Wang, F.; Li, Z.; Sun, J. Yolox: Exceeding yolo series in 2021. *arXiv preprint arXiv:2107.08430* **2021**.
35. Ozge Unel, F.; Ozkalayci, B.O.; Cigla, C. The power of tiling for small object detection. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, 2019, pp. 0–0.
36. Yang, F.; Fan, H.; Chu, P.; Blasch, E.; Ling, H. Clustered object detection in aerial images. Proceedings of the IEEE/CVF international conference on computer vision, 2019, pp. 8311–8320.
37. Li, C.; Yang, T.; Zhu, S.; Chen, C.; Guan, S. Density map guided object detection in aerial images. proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops, 2020, pp. 190–191.
38. Ding, J.; Xue, N.; Xia, G.S.; Bai, X.; Yang, W.; Yang, M.Y.; Belongie, S.; Luo, J.; Datcu, M.; Pelillo, M.; others. Object detection in aerial images: A large-scale benchmark and challenges. *IEEE transactions on pattern analysis and machine intelligence* **2021**, *44*, 7778–7796.
39. Zhu, X.; Lyu, S.; Wang, X.; Zhao, Q. TPH-YOLOv5: Improved YOLOv5 based on transformer prediction head for object detection on drone-captured scenarios. Proceedings of the IEEE/CVF international conference on computer vision, 2021, pp. 2778–2788.
40. Baidya, R.; Jeong, H. YOLOv5 with convMixer prediction heads for precise object detection in drone imagery. *Sensors* **2022**, *22*, 8424.
41. Wang, C.Y.; Liao, H.Y.M.; Wu, Y.H.; Chen, P.Y.; Hsieh, J.W.; Yeh, I.H. CSPNet: A new backbone that can enhance learning capability of CNN. Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops, 2020, pp. 390–391.
42. Liu, S.; Qi, L.; Qin, H.; Shi, J.; Jia, J. Path aggregation network for instance segmentation. Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 8759–8768.
43. Wojke, N.; Bewley, A.; Paulus, D. Simple online and realtime tracking with a deep association metric. 2017 IEEE international conference on image processing (ICIP). IEEE, 2017, pp. 3645–3649.
44. Bennett, S. The past of PID controllers. *Annual Reviews in Control* **2001**, *25*, 43–53.
45. Rosenfeld, A.; Pfaltz, J.L. Sequential operations in digital picture processing. *Journal of the ACM (JACM)* **1966**, *13*, 471–494.
46. Ramesh, A.; Dhariwal, P.; Nichol, A.; Chu, C.; Chen, M. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125* **2022**, *1*, 3.