ROBUST REINFORCEMENT LEARNING CONTROL STRATEGY FOR VISION-BASED SHIP LANDING OF VERTICAL FLIGHT AIRCRAFT

A Thesis

by

VISHNU SAJ

Submitted to the Graduate and Professional School of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

December 2021

Major Subject: Computer Engineering

ABSTRACT

This study discusses a fully autonomous vertical flight aircraft ship landing procedure in presence of wind disturbances. The proposed study closely follows the established Navy helicopter ship landing procedure wherein the pilot utilizes the ship as the visual reference for long-range tracking; however, upon coming closer, the pilot follows a gyro-stabilized horizon bar installed on most Navy ships to approach and land vertically independent of deck motions. This was accomplished by developing a unique vision system and a hybrid control system validating its performance in simulations and flight tests.

The vision system serves the purpose of a pilot's eye by obtaining the visual information required for a safe approach and landing. The vision system can be engaged from 250 meters away from the landing pad, initially utilizing machine learning strategies to detect the ship for long-range tracking and switches to a unique combination of classical computer vision techniques to detect the horizon bar to precisely estimate the aircraft position and orientation relative to the bar during the final approach and landing. The distance and attitude estimations were validated using the measurements from an accurate 3D motion capture system (VICON), which demonstrated sub-centimeter and sub-degree accuracy.

Finally, a hybrid control system is developed to control the aircraft using the perceived visual information. The hybrid control system is a combination of a non-linear controller and a Deep Reinforcement Learning(RL) controller. The non-linear controller demonstrated robust tracking capability even in presence of estimation noise and varying time delays between successive control actions. The RL controller is developed exclusively for disturbance rejection. When conducted flight testing in presence of 5 m/s wind, the RL controller shows a 100% reduction in drift and a 10 times faster rate of correction compared to a conventional control system. The vision and hybrid control system was implemented on a quadrotor UAV and extensive flight tests were conducted to demonstrate accurate tracking in challenging conditions and safe vertical landing on a sub-scale ship platform undergoing 6 degrees of freedom deck motions.

DEDICATION

To my mother, father, teachers and all friends.

ACKNOWLEDGMENTS

Throughout my almost two years of I have received a great deal of support and assistance.

First and foremost, I owe a large debt of gratitude to Dr. Dileep Kalathil and Dr. Moble Benedict. Without their careful guidance and supervision, development of the system presented in this dissertation would have been nearly impossible. Their expertise was invaluable in formulating the research questions and methodology. Every time I lost the way to go, both professors always shed light on the path to follow. They listened to my opinions with open mind and shared great insights with me. Their insightful feedback pushed me to sharpen my thinking and brought my work to a higher level.

I would also like thank Dr. Bochan Lee who was my mentor for this research from the beginning. Without his insights and guidance, it wouldn't have been success as it stands now. He always pushed me in the right direction when I am lost at some point.

I am also grateful to all my friends Gargi, Amogh, Desik, Akshay, etc that I met here at Texas A&M University for their help and sharing ideas. It is not possible to remember all, and I apologize to those I have left out.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a thesis committee consisting of Professors Dileep Kalathil, P. R. Kumar and Raffaella Righetti in the department of electrical and computer engineering and Moble Benedict in the department of aerospace engineering

Dr.Bochan Lee in the department of aerospace engineering contributed to the development of the non-linear control system and has active participation in flight testing and collecting data. All the other work for the dissertation was completed by the author.

Funding Sources

This study has been funded by the Center for Unmanned Aircraft Systems (C-UAS), a National Science Foundation Industry/University Cooperative Research Center (I/UCRC) under NSF award Numbers IIP-1161036 and CNS-1946890, along with significant contributions from C-UAS industry members.

NOMENCLATURE

A	Action Space
A_{int}	Camera intrinsic matrix
a_t	Action at time t
b	Close-range controller constant
c_u	Corner position-column
c_v	Corner position-row
D	Replay Buffer
d	Deviation from target, meter
$de(t_k)$	Error difference between time t_k and t_{k1}
dt_k	Time difference between time t_k and t_{k1}
$e(t_k)$	Error at time tk
f_x	Focal length of camera in horizontal direction
f_y	Focal length of camera in vertical direction
K_D	Derivative gain
K_I	Integral gain
K_P	Proportional gain
$L(\phi, D)$	Mean squared Bellman error function
m	Long-range controller constant
p	Indicator to check whether a terminal state or not
$Q(s, a)$	Q-value function
R	Rotation matrix
R_{basic}	Basic rotation matrix

R_t	Cumulative Rewards
S	State Space
s	Scaling factor
s_t	State at time t
t	Translation vector
u	Image pixel position-column
u_0	Center of image-column
$u(t_k)$	Control law
v	Image pixel position-row
v_0	Center of image-row
$V(s)$	Value function
x	Forward relative distance, meter
y	Sideward relative distance, meter
v_x	Forward relative speed, m/s
v_y	Sideward relative speed, m/s
α	Relative yaw angle, rad
β	Relative pitch angle, rad
γ	Relative roll angle, rad
Γ	Discount factor
θ	Aircraft pitch angle, deg
μ	Mean value
Π	Policy Space
π	Policy
π^*	Optimal policy
ϕ	Q-value network parameter

ρ_u	Parameter to convert pixels to SI units-column
ρ_v	Parameter to convert pixels to SI units-row
$\nabla f(u,v)$	Image gradient

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION AND LITERATURE REVIEW

[1]

Landing a helicopter on a small ship at rough sea states is an extremely challenging task due to the small landing space, six degrees of freedom ship deck motions, limited visual references for pilots, and lack of alternative landing spots. It's a very difficult task even for a human pilot. To date, many studies have focused on automating helicopter ship landing by utilizing a wide array of sensors such as GPS, vision sensors, motion sensors, LIDAR, etc. This study investigates a novel solution that falls under the category of a vision-based control system that does not use GPS signals and thus ensures its functionality in GPS-denied/-spoofed environments.

Previous efforts toward autonomous ship landing involve a common process that is to estimate or track ship deck motions first, and then control the aircraft attitude to match the ship motions for landing. In order to extract the ship deck motion information, various methods have been introduced such as tracking H landing marking [5], T landing marking [6], points dispersed on the deck [7], lights [8], and infrared cooperated targets on a ship [9, 10]. These vision-based methods have shown the limited capability for autonomous UAVs landing on ships undergoing significant movements that represent rough sea conditions. Fundamentally, the method of visually tracking deck movement is limited to vertical space where the deck can be captured. Hence is not ideal for Vertical Take-Off and Landing (VTOL) capable Unmanned Aerial Vehicles (UAVs) that approach a ship horizontally at low altitudes. In addition, the active control of the UAV to accommodate complex deck movements can result in unstable UAV attitude dynamics. This is even more dangerous if the aircraft is in close proximity to the moving deck, as even small control errors can lead to catastrophic accidents due to the impact by the deck. Moreover, none of the previous methods have been based on the Navy helicopter landing procedures, which have been established for decades and have been successfully carried out by pilots. In fact, visually tracking the moving landing deck is exactly the opposite of what Navy helicopter pilots are trained to do.

[1]The chapter has been adapted or reprinted from recent publications [1] [2] [3] [4].

The present landing method is developed based on an in-depth understanding of the Navy helicopter ship landing procedure, which is discussed in Refs. [11, 12]. Contrary to intuition, Navy pilots are trained not to follow ship deck motions for two main reasons. The spatial disorientation can occur when a pilot has no fixed, visible horizon to refer to, which is a critical element for maintaining a proper sense of helicopter attitude independent of ship motions. The key visual aid that helps pilots to land safely is a "horizon reference bar" shown in Fig. 1.1, which is gyro-stabilized to indicate a perfect horizon regardless of ship motions and is widely used in most modern Navies [13, 14]. Thus, pilots can land a helicopter by referencing the horizon bar without responding to ship motions.

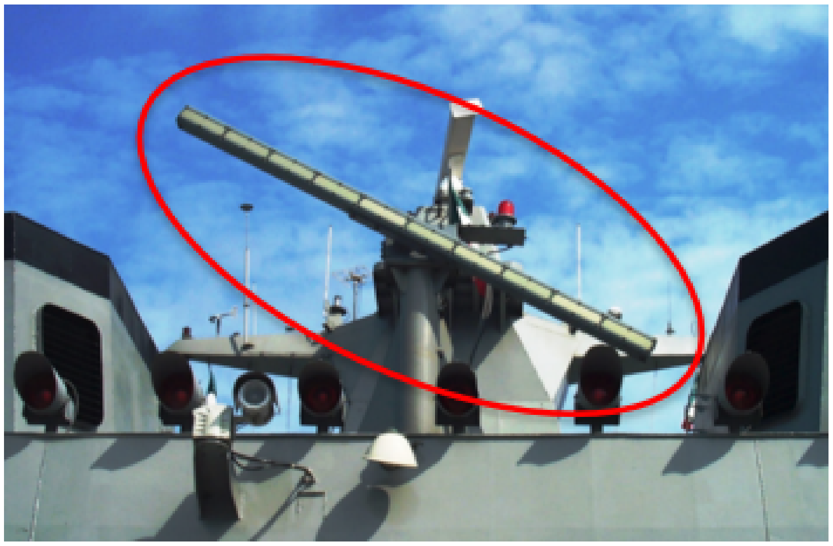


Figure 1.1: Horizon reference bar

Another factor is that constant changes in helicopter attitude to ship deck movement can lead to unstable helicopter dynamics that pose a serious potential danger. The pilot attempts to steer the helicopter in a stable manner, regardless of the ship's roll or pitch, and then lands vertically. It is also advisable to land quickly to prevent the ship's deck from hitting the skid / wheel of the helicopter landing gear causing rollover. This vertical landing has proven to be safe within the helicopter's operational limits and is currently in practical use. The present technical approach towards automating such a landing procedure consists of machine vision to obtain the relative position and heading of the aircraft and a control system to execute the approach and landing maneuvers.

As shown in Fig. 1.2, referring to the visual cue allows the UAV to approach horizontally similar to the helicopter ship landing procedure. Therefore, it has a wider horizontal range and it can be used for both approach and landing phase. However, the operating range for a platform-motion-tracking approach is limited to the vertical space, and therefore, it can be used only for the final landing phase and requires another method for approach. Therefore, the method presented in this study is fundamentally different from previous UAV landing techniques and can be applied to all types of VTOL aircraft.

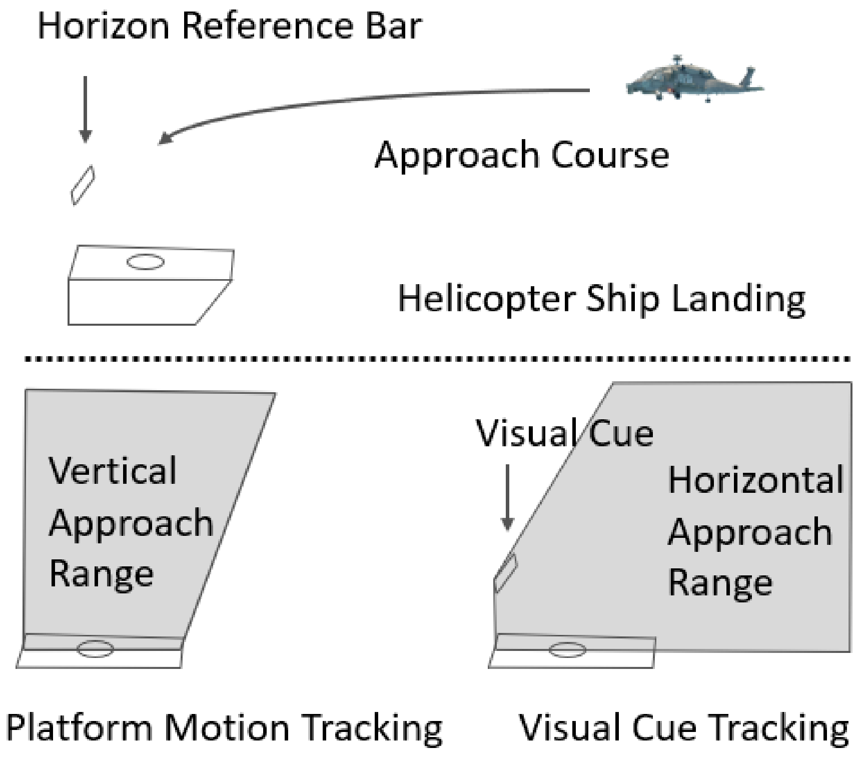


Figure 1.2: Comparison of Previous Platform Motion Tracking with Present Visual Cue Tracking Method [1] [2] [3]

In the first half of the study, instead of the standard horizontal bar, a checkerboard pattern is used as a visual cue which is easily distinguishable from its surroundings and also offers redundancy with multiple corners that can be detected. Gain scheduled Proportional-Integral-Differential (PID) controllers are configured to command UAV roll, pitch, yaw, and throttle controllers. When the UAV moves to the landing point, multiple controller layers are activated depending on the relative distance from the landing platform. This allows you to approach, slow down if the UAV is close to landing deck and move faster if the UAV is far away. It reaches near the landing site and eventually performs the landing operation as a remote helicopter pilot does.

Later the visual cue was replaced by a miniature version of the horizon bar used in navy helicopter ship landing. The vision system also gets modified and is hybrid in nature with two different methods, a machine learning object detection and a classical computer vision method, each of which is designed to operate depending on the relative distance to the landing pad. In the long-distance, the machine learning object detection method is applied to identify the landing platform (the ship), and an image-based control is utilized in the autonomous flight control system. In recent years, there have been many studies to develop algorithms that guarantee fast detection as well as higher accuracy, which are essential to reliable UAV operations. Various algorithms and architectures of Convolutional Neural Network (CNN), a class of deep neural networks, have been proposed such as Region-based CNN (R-CNN) [15, 16, 17], Single Shot Detector (SSD) [18], and You Look Only Once (YOLO) [19, 20, 21]. R-CNN is classified as a two-stage detector that combines region-proposal algorithms with CNN to extract 2,000 regions via a selective search, then classifies the selected regions on the image. Later, its variant Faster R-CNN is introduced to improve the detection speed by replacing the slow selective region search process. Meanwhile, SSD and YOLO are classified as a one-stage detector that regards object detection as a regression problem by taking an input image and simultaneously learning the probability of an object class and bounding box coordinates. According to the studies that compared the state-of-the-art algorithms, YOLOv3 demonstrated faster detection performance than Faster R-CNN and SSD [21, 22]. Hence, the YOLOv3 algorithm is selected to train an object detector that is able to detect a ship and a horizon bar in real-time. It is not only visually tracking the object but that information is relayed in real-time to the autonomous flight control system. Once the object is detected, it provides the object position and its bounding box in the image to the autonomous flight control system. Even though the actual relative distances are not estimated, the size of the object and its position in the image are sufficient information to control a UAV to approach the ship from a long distance. The verified maximum range is approximately 250 meters (820 feet) when the object occupies an area of 1.8 x 1.8 meters (6 x 6 feet). Considering the range is proportional to the object's occupying area in the image, a typical small ship where the rear-side occupies 15 x 15 meters (50 x 50 feet) area can be detected from 17.3 kilometers (9.3 nautical miles) away.

On the other hand, obtaining accurate relative position and orientation from a captured image is crucial for the final approach and even more important for precise landing on the ship deck. Instead of detecting an object as a whole using machine learning, computer vision techniques are applied to extract particular points of interest. To name a few, edge and line detection [23, 24], corner detection [25, 26], and contour detection [27] are previously applied to aircraft landing applications. In the present system, the visual cue to track is a horizon bar that has a rectangular shape and green color. To detect distinctive points on the bar, multiple processes such as image filtering, contour detection, corner points detection, and screening are conducted in this order. From the detected points in the 2D image, UAV relative positions and orientations are estimated using Perspective-n-Point (PnP) method. The accuracy of the present vision system to sub-centimeter and sub-degree levels have been previously demonstrated by the authors [1].

To demonstrate the safety of the vertical landing maneuver, which in this case is independent of the ship motions, realistic ship motions are implemented on a six Degrees Of Freedom (DOF) motion platform. The first ship motion case is from the Oliver Hazard Perry Class FFG Frigate which is a small ship with a single landing deck. The ship motions at the sea state of 6 that refers to a wave height of 4 to 6 meters with the wave direction of 60° are scaled down for the 1.22 x 1.22 meters (4 x 4 feet) platform and are provided in [5], which are also similar to measured ship motion data presented in [28]. The second ship motion case is the FFG 7 Class ship motion limits which are 3° of pitch and 8° of roll as defined in the Naval Air Training and Operating Procedures Standardization (NATOPS) [29]. The period of pitch and roll motions are selected as 10.1 seconds and 6.5 seconds according to the study of typical small ship motions conducted by the Sandia National Lab [30]. Vertical landings are conducted at random instances of deck motions.

By using the information provided by the vision system, the flight controller manipulates the UAV to approach and land. To this end, various control systems that uses vision sensors without GPS have been investigated in the literature, such as Proportional-Derivative (PD) control [31, 8], a Proportional-Integral-Derivative (PID) control [32, 7], gain-scheduled PID control [1], Linear Quadratic Regulator (LQR) [33, 34, 35], adaptive control [36, 37, 38], discrete-time nonlinear model predictive control [39], and reinforcement learning based control [40]. In the present system, a nonlinear control system with the Kalman filter is uniquely designed to operate accurately and robustly in the presence of time delays and sensor noise. The Kalman filter reduces the noise in estimation, however, small noise can be amplified when incorporated into the derivative controller due to the numerical differentiation process. A novel nonlinear controller is developed on top of the Kalman estimator to prevent the controller from responding to unrealistic estimations (or large fluctuations). It multiplies the estimation difference by the probability of its occurrence that follows a normal distribution. In this manner, the controller probabilistically perceives if the estimation is physically possible or not and then determines how to respond with a control input.

While these approaches have been partially successful in solving the problem, they often have two significant weaknesses that limit them to very specific situations. Firstly, the performance of the PD/PID type controllers solely depends on the design of its gain parameters, which are typically difficult to fine-tune, especially in the simulator setting. PD/PID controllers have also limited transient response capabilities. Moreover, PD/PID controllers are not robust against uncertainties and disturbances such as wind gusts and parameter mismatches between the simulator model and the real-world system model. Secondly, MPC and LQR approaches often require a very sophisticated analytical model of the real-world UAV. Obtaining such a model can be very challenging in practice. Moreover, for computational tractability, the design of the optimal control policies using these approaches is often limited to simplified settings such as linear policy and quadratic costs. This often results in poor performance in the real world.

In this study a deep reinforcement learning (RL)-based control algorithm for autonomous vision-based ship landing of UAVs is also proposed. Using the representation power of neural networks, RL provides an attractive model-free approach for developing nonlinear control algorithms for high dimensional systems in an automated and computationally tractable way. We propose and develop a clever modification of the standard off-the-shelf RL approach to adapt it to the vision-based autonomous UAV landing problem and to make it robust against adversarial disturbances in the environments such as wind gusts. Multiple previous works have developed RL algorithms to address the autonomous landing of UAVs. RL-based high-precision, time-critical flight attitude control that could operate in unpredictable and harsh environments is discussed in [41]. It also discusses different policy gradient RL algorithms such as deep deterministic policy gradient (DDPG), proximal policy optimization (PPO), and trust region policy optimization (TRPO). An RL approach for UAV landing task on a moving platform using a variant of the DDPG algorithm is discussed in [42]. An actor-critic RL framework used to fly a UAV by following designated waypoints is presented in [43]. A variant of the DDPG algorithm is used to recover a UAV attitude quickly from an out-of-trim flight state [44]. However, these works do not address the problem of designing RL controllers that are robust against adversarial disturbances such as wind gusts. We adapt the state-of-the-art RL algorithm called twin delayed DDPG (TD3) and use the idea of domain randomization [45, 46] to make it robust. The disturbance rejection capability of the RL controller is compared to the previously developed nonlinear control system with the Kalman filter [47] through extensive simulations and flight tests. During flight testing, to verify the performance of the RL controller, wind gusts from different directions are suddenly imposed while the UAV attempts to approach and land on a sub-scale ship platform undergoing 6 degrees of freedom (DOF) deck motions. The UAV used is Parrot ANAFI quad-rotor that streams live video to a base station computer, which processes the image frames, generates control inputs, and then transmits the control commands back to the UAV through WIFI.

The following are the key contributions of this work.

- Automate Navy helicopter ship landing procedure for VTOL UAVs and verify the safety of vertical landing while realistic and challenging ship motions present.
- Develop a state-of-the-art machine/deep learning based vision system that can identify and track objects of interest (ship platform and horizon bar) in a long distance.
- Develop a nonlinear control system that can be effective for real-time autonomous flight in the presence of time delay caused by the machine/deep learning based detection.
- Demonstrate a fast and reliable method to extract points of interest from an image by combining classical computer vision and screening algorithms.
- Develop a novel nonlinear controller along with a probabilistic algorithm to prevent large incorrect control inputs due to non-physical estimations to enable robust and smooth tracking.
- Considering deck motions, UAV size, and safety margin, determine a safe landing boundary and demonstrate the vertical landing accuracy via flight tests.
- Developed an RL-based control strategy that is robust against adversarial disturbances such as wind gusts.
- Demonstrated the superior disturbance rejection capability of the RL-based controller when strong wind gusts are suddenly imposed in different directions.
- Demonstrated the precise approach and landing on a sub-scale ship platform undergoing challenging 6-DOF deck motions using a novel approach that does not track deck movements.

2. MACHINE VISION SYSTEM

[1] The objective of the vision system is to provide the necessary visual information required in a given situation for the control system. A helicopter pilot, from a far distance, identifies the ship and sets its initial course and once getting closer to the ship, refers to gyro-stabilized horizon bar for safe approach and landing irrespective of the deck motions. Hence in this study, different detection strategies are developed depending on the distance. This chapter gives a detailed explanation on how to detect different objects of interest and estimate the relative position and orientation of the UAV and finally validate its accuracy. Throughout the study, a gimballed camera of the VTOL UAV that can mechanically compensate for the roll and pitch motions of the UAV is used to capture images.

2.1 Detection Methodologies

The detection methodologies are developed and configured according to how a Navy helicopter pilot perceives and acts during different situations while approaching and landing. From a long-distance, the pilot visually confirms the ship's location and sets the course and speed for the approach. Once the helicopter comes closer to the ship, the horizon bar becomes visible and is gyro-stabilized to indicate the true horizon independent of the ship's motions. From that point, the pilot stably controls the helicopter by referring to the horizon bar for a safe landing. The same set of strategies are utilized for automating VTOL UAVs by taking advantage of state-of-the-art machine learning-based object detection for long-range tracking and classical computer vision techniques for close-range tracking.

Fig. 2.1 shows the primary and secondary objects of interest for detection between 0 to 250 m away from the landing pad. Initially, the whole ship is used as a reference by the control system, then the horizon bar is detected and finally, points on the horizon bar are used for precise estimation of relative positions and orientations of the UAVs for a safe landing. If any of the primary target

[1]The chapter has been adapted or reprinted from recent publications [1] [2] [3] [4].

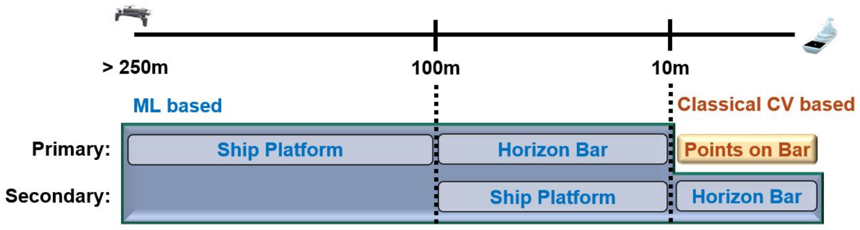


Figure 2.1: Tracking object depending on distance [2] [3]

objects are not detected, the secondary objects are detected as a backup.

2.1.1 Machine Learning Based Object Detection

Machine learning-based object detectors are developed to detect the ship as a whole for long-range and horizon bar for mid-range tracking. Classical computer vision may achieve the same task however it requires explicit algorithms for detection which can be extremely complicated and challenging to capture every aspect of the object. On the other hand, machine learning object detectors learn the characteristics of an object thoroughly using neural networks. The speed and accuracy of such detectors can be significantly improved by using Graphics Processing Units (GPUs).

Developing an object detector involves three steps, which are, collecting images(data collection), labeling objects in the images(training set), and finally training the object detector. First, 2000 images of the ship platform and 1000 images of the horizon bar are collected by the UAV's onboard camera. To include various object figures in training sets, images are captured from different perspectives and distances, and in varying lighting and weather conditions. Second, the object in the collected images are labeled by drawing a bounding box around it and assigning the object class name. This data is stored as pixel positions and the object identification number. Third, the set of labeled images are used for training via a state-of-the-art machine learning technique.

To implement the machine learning object detection in a real-time flight system, computational speed is the primary factor to consider. With the advancement of deep learning, several algorithms have been developed for the purpose of fast detection as well as higher accuracy such as Region-based CNN (R-CNN) [78, 79, 80], Single Shot Detector (SSD) [81], and You Look Only Once

(YOLO) [82, 83, 84]. They commonly have the architecture of convolutional neural network (CNN) which is a class of deep neural networks. According to the recent studies that compare the processing time of fast detecting algorithms [22, 18, 21], YOLOv3 is faster than other algorithms such as SSD and Faster R-CNN while having similar prediction accuracy. For this reason, the YOLOv3 algorithm is selected for the system.

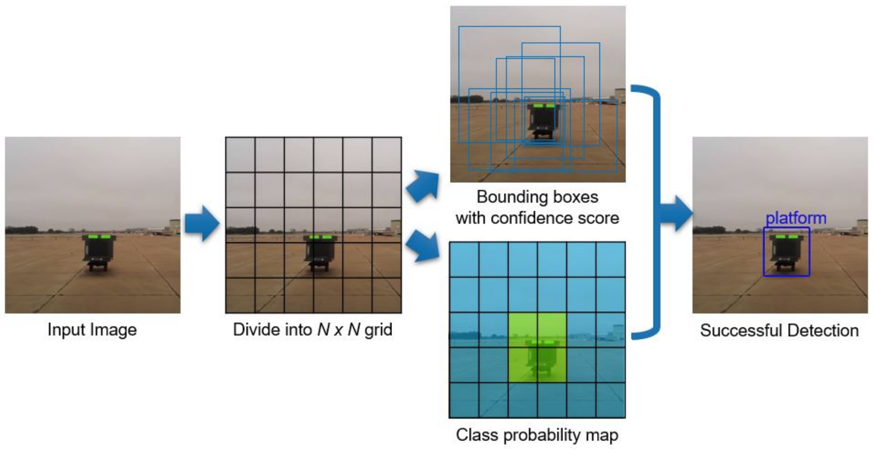


Figure 2.2: Ship platform detection process by YOLOv3 algorithm [2] [3]

The object detection task consists of object classification and localization. Typically, an object detector is developed to detect multiple objects and classify them; however, in the present approach, two separate single object detectors are developed to detect the ship platform and horizon bar, respectively. By doing so, each detector only needs to find a particular object in the image, and the object class is automatically assigned without incurring the risk of false classification. Once it detects an object, it returns the position of the object and the size of its rectangle bounding box in the image. The position and size are used to determine approach course and speed, respectively. Since the detection range is proportional to the area occupied by the detected object in the image, during the early phases of approach the whole ship is detected to maximize the detection range. As the UAV gets closer to the ship, it also detects the horizon bar which provides more specific position and size information. Instead of having one object detector that identifies two objects (ship and horizon bar), two separate object detectors are developed for each object so that they do not have to distinguish one object from the other. The control system takes the information from the object detectors and takes the most appropriate control action in a given situation.

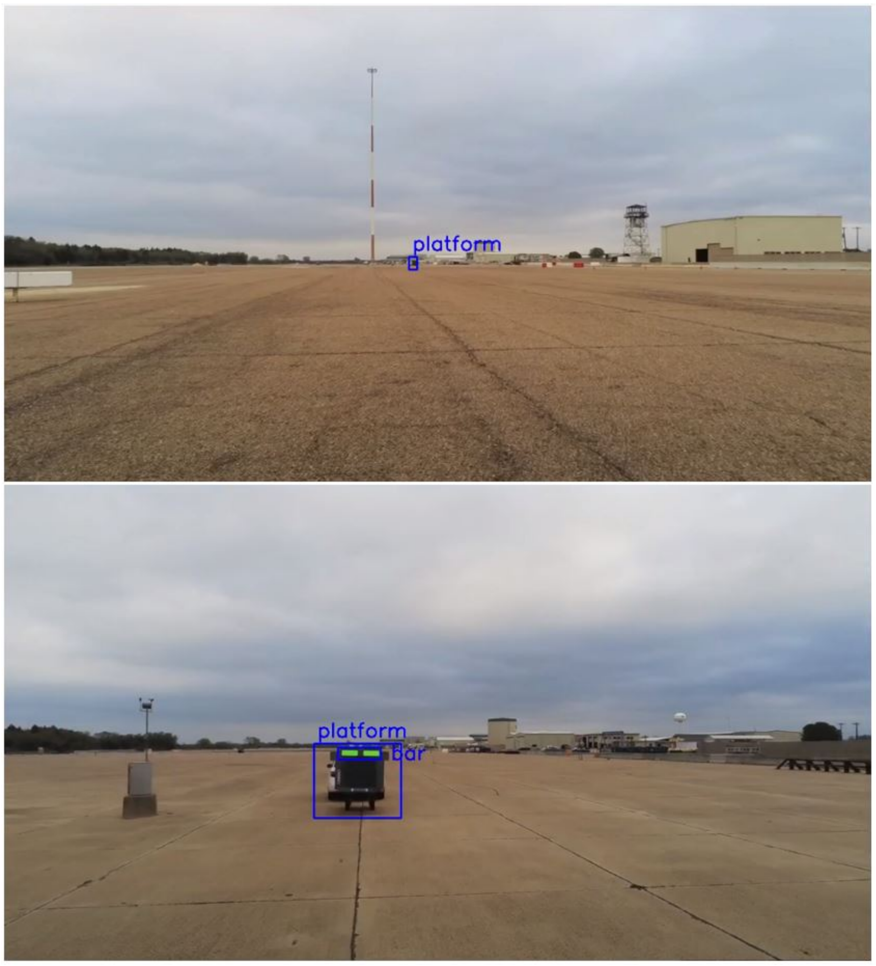


Figure 2.3: Long and mid-range real-time object detection result [2] [3]

The one-stage detector YOLOv3 regards the detection as a regression problem and uses a single neural network. It analyzes the entire image to predict the object-bounding box. As shown in Fig. 2.2, the input image is divided into an N x N grid and each grid cell predicts bounding boxes with a confidence score and class probability. To better detect the object in different sizes, it predicts bounding boxes at three different scales, which helps to detect the object from a far distance. The predictions of developed detector are encoded as an $N \times N \times [3 * (4+1+1)]$ tensor for $N \times N$ grid cells, 3 different scales, 4 bounding box offsets, 1 object confidence score, and 1 class prediction. The sequential procedures in a YOLOV3 object detector is shown in Fig. 2.2

The observed maximum ranges for detecting the ship platform and horizon bar are approximately 250m and 100m, respectively. The real-time detection at different distances is shown in Fig. 2.3.

The maximum verified range for the machine learning-based detector is approximately 250 meters (820 feet) when tracking a 6 X 6 foot object (subscale ship platform). Because the detection area is proportional to the occupied area of the object in the image, a typical small ship, with the rear occupying an area of 50 X 50 feet, can be detected from a distance of 17.3 kilometers (9.3 nautical miles). This is 18 times greater than the distance to the Missed Approach Point (MAP), a point at which the pilot must visually identify the ship. As soon as the target object has been recognized, it provides the position of the object and its bounding box in the image to the control system. Although the actual relative position and orientation are not estimated, the developed control system achieves long-range autonomous flight using the size of the object and its position in the image.

2.1.2 Classical Computer Vision

The modern machine learning-based approach has the advantage that detection visual cue from afar. However, the training procedure in machine learning-based approach is non-deterministic, meaning detection may fail for unknown reasons. For example, the model might identify other objects as a visual cue, and the bounding box of the object might be larger or smaller than the original size of the object. These faults are unacceptable for close-range tracking and precision landing. In addition, the machine learning-based approach is computationally intensive, which can lead to delays in the control system. Therefore, to develop a short-range vision system, classical computer vision algorithms are used, which clearly indicate what needs to be done to detect and execute in the shortest possible time.

Once the UAV is in close proximity to the ship platform, it must recognize the visual cue and then estimate the relative position and orientation for the final approach and landing. The visual cue should be installed perpendicular to the landing pad and parallel to the UAV's line of sight, much like the horizon reference bar on a ship. At close range, robust detection and estimation is achieved by combination of computer vision techniques and developed detection algorithms. The image processing system first recognizes the desired points of interest and then estimates the relative position and orientation. The visual cue need not have a particular shape as long as the dimensions are known in advance. However, the unique characteristics that can be distinguished from the environment are beneficial for detection. A number of different visual cues are tested to validate the effective range and representative cases are shown in Fig.2.4

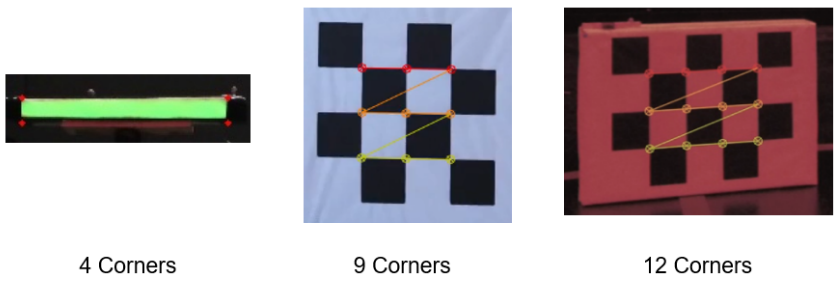


Figure 2.4: Available Representative Visual Cues [1] [3]

Among the candidate visual cues, checkerboard pattern visual cue is investigated first due to its distinctive features and a redundant number of corners. A 4x4 checker pattern was chosen for the study. Its geometry exhibits local image features such as distinctive edges, lines, and corners which are greatly helpful in detection. From the experiments that utilize checkerboard pattern visual cues, the effect of camera resolution and the size of squares in the checkerboard on the detection range is studied. The built-in front camera of the UAV has mechanical pitch and roll gimbals that compensate for the UAV's pitch and roll movements and keep the camera level with the horizon. This means that images recorded by the camera do not experience the effects of pitch and roll. This reduces the complexity caused by rotational movements. However, the effect of yaw is reflected in the images, which is utilized to extract the heading of the UAV with respect to the visual cue. The maximum resolution of the camera is 4k ultra-high definition, UHD (4096 x 2160), and the highest resolution helps to increase the detection range. The size of the squares on the checkerboard is the other factor that affects the detection range. As the size of the square increases, so does the effective detection range. However, there is a trade-off between the maximum effective range and the closest proximity to the visual cue, as with a larger square size, the camera must be positioned further away for visual cue recognition. The effective range for each case is found through experiments and are provided in Table 2.1.

Table 2.1: Valid Range for Varying Resolution & Square Size

Resolution	80 mm Square	120 mm Square
qHD(540p)	13 m	15.5 m
HD(720p)	14 m	17.5 m
FHD(1080p)	15 m	19 m
4K UHD(2160p)	17 m	25 m

.

According to the results, the size of the square was chosen to be 120mm because it can be accurately detected from a distance of up to 25 meters and the size of the square is small enough that it can be captured even when the visual cue is only is 1 meter away. Although a higher resolution increases the effective range, it also increases the amount of data processed. Therefore, HD 720p (1280 x 720) resolution was chosen taking into account the latency of the live streaming and image processing for a relatively high band-width integrated vision-based feedback control system as well as a good effective range. This visual cue was used in the first stage of development of a vision-based autonomous ship landing system. Finally, a visual cue that closely mimics the horizon reference bar on Navy ships is built and used for close range detection and pose estimation. It has two green rectangles on a gray background with known dimensions and splits as shown in the Fig. 2.5

Taking into account the characteristics of the installed horizon bar, the corner points of the

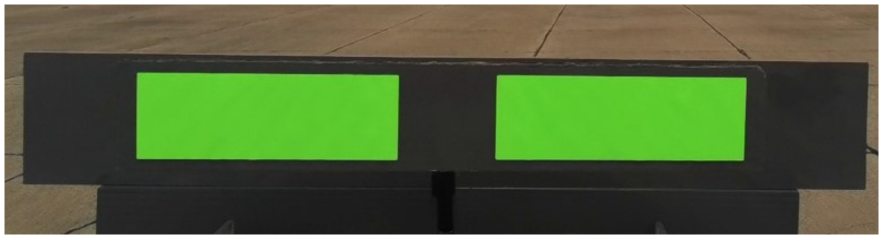


Figure 2.5: Installed horizon bar [2] [3]

green rectangles are identified as the target to be detected. To ensure robust detection and estimation in situations involving large UAV movements and different visibility conditions, the vision system is set up to perform sequential image filtering, contour and corner detection, detected points screening, and the estimation of position and orientation.

- Image Filtering

The initial approach involves applying a Hue-Saturation-Value (HSV) filter to sort out the green rectangles. The HSV filter is preferred over Red-Green-Blue (RGB) because the RGB color space is more sensitive to light conditions. To ensure the capture of the green rectangles in different light conditions, the greater range of HSV (H: 35 - 85, S: 70 - 255, V: 90 - 255) is assigned; however, this inevitably leads to the capture of unwanted portions. In the HSV filtered image, there possibly exists small white patches outside the rectangles and black voids inside the rectangles. The morphological opening technique is used to remove the white patches in the image. It first erodes an image removing any small white patches and then dilates the eroded image to preserve the original shape and size. Morphological closing is used to fill up small voids in those rectangles by dilating an image first then eroding the dilated image. By performing these operations, the rectangles are depicted as white regions on the black background. The watershed algorithm [48] is exploited to obtain clear boundaries of the rectangles. The two reference areas are obtained by eroding the processed image by 1% and dilating the processed image by 1%. The gap between the two areas is assumed to contain the boundaries of the rectangles. The algorithm simultaneously expands the area of the background and the rectangles toward each other until they meet at one pixel point. By connecting the points, clear boundaries of the rectangles are obtained. The images of each step are shown in Fig. 2.6.

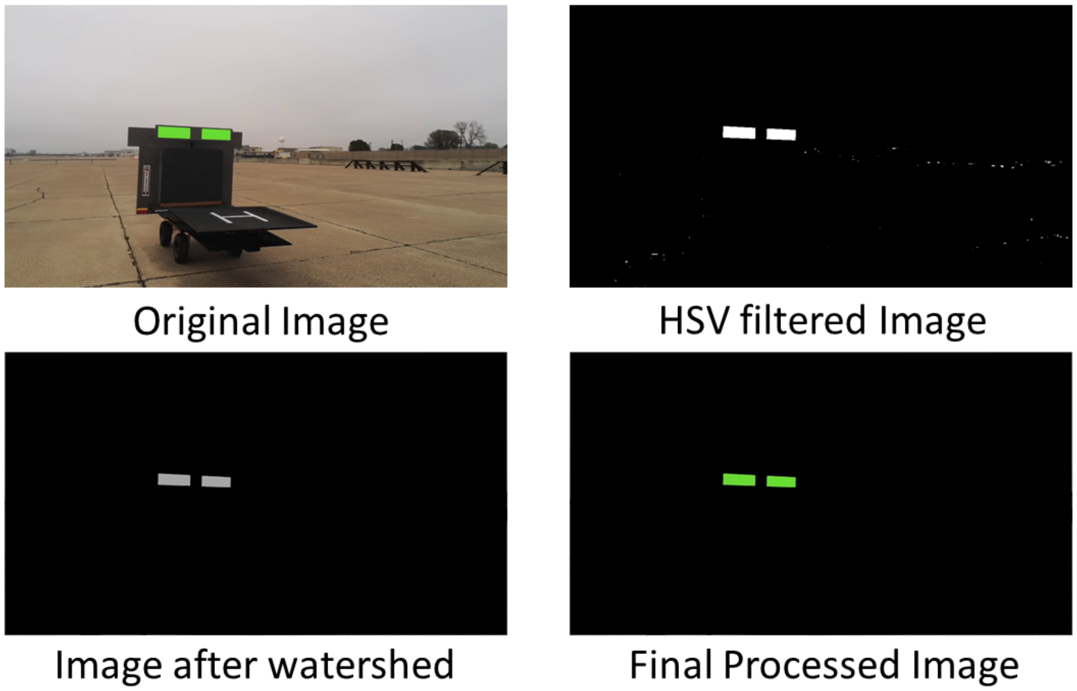


Figure 2.6: Image filtering process [2] [3]

- Contour and Corner Detection

Once the green rectangles are isolated by the image filtering, the detection of contours and corner points is conducted. Even after the filtering, directly implementing any pre-existing corner detection algorithms is prone to detect some false corners. To detect the eight corners precisely, the contours of the detected region are found and bounded in rectangles first. Thus, the size and shape of the detected areas are very close to the green rectangles and the corners of those bounding rectangles can be used as rough estimates of the actual corners. Second, the Förstner corner detection method is adopted to detect the corner points of the rectangles precisely based on the rough corners obtained by contour detection [49]. The Förstner corner detection increases the accuracy by sub-pixel refinement process. It is based on the fact that an ideal corner is a single point that tangent lines of the object cross perpendicular to each other as shown in Fig. 2.7. This way, false corner detection can be avoided.

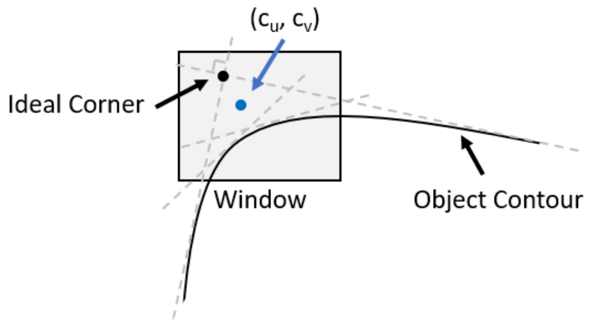


Figure 2.7: Förstner corner detection method [1] [3]

The pixel information around a corner is not perfectly clear. Therefore, an approximation process for defining the corner position (c_u, c_v) is required and is expressed in Eq. (2.1).

$$(\hat{c_u}, \hat{c_v}) = \arg\min_{c_u, c_v} \sum_{u,v \in N} ([\nabla f(u,v)]^T (u - c_u, v - c_v))^2 \tag{2.1}$$

The image gradient $\nabla f(u,v)$ at the image pixel position (u,v) is perpendicular to line from (u,v) to corner position (c_u, c_v). In order to obtain (c_u, c_v), a least square estimation in a small window chosen for each image. It cannot be a fixed window size because the UAV is always moving towards the landing pad. The size of the detected region keeps increasing as it approaches closer to the visual cue. Hence a variable window size which is a function of the width and height is assigned as expressed in Eq. (2.2).

$$sw(w(t_k), h(t_k)) = \frac{1}{5}\{w(t_k) \times h(t_k)\} \tag{2.2}$$

$sw(w(t_k), h(t_k))$ is the window size at time t_k where $w(t_k)$ and $h(t_k)$ are the width and height of bounding rectangles in pixels. The results of the contours and corners detection are shown in Fig. 2.8.

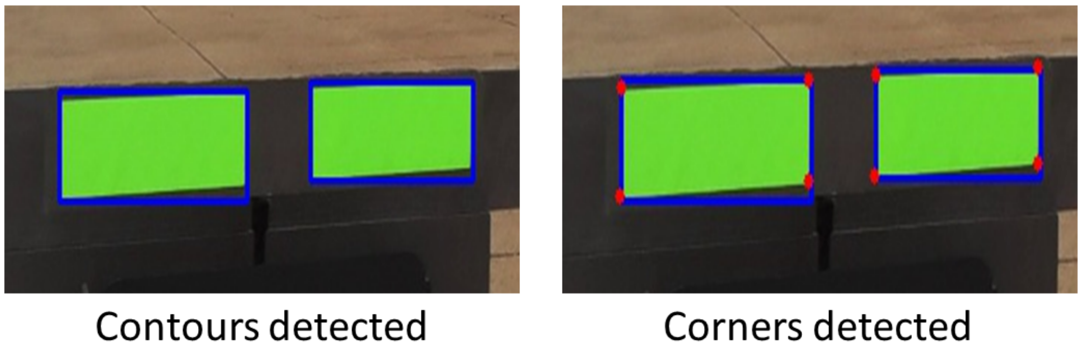


Figure 2.8: Detection of contours and corners [2] [3]

- Detected Points Screening

The screening procedure is established to assure that no false corners are present. All the corners are sorted in a particular order as shown in Fig. 2.9, which helps in finding the length and slope of each side of the rectangles in the image. Even though they are not perfect rectangles in the image, width 1, 2, 3, and 4 have similar lengths and slopes. The height 1, 2, 3, and 4 also have similar lengths and slopes. A $\pm 10\%$ tolerance level is set for the lengths and a $\pm 5\%$ tolerance level is set for the slopes.

2.1.3 Estimation of Relative Position and Orientation

The estimation is based on a single camera calibration method using a planar object [50, 51]. The geometric relation of the image and real-world coordinates is shown in Fig. 2.10.

A conventional pinhole camera model is used to derive the geometric relation. A 3D coordinate system can be defined with respect to the visual cue and there is a 2D coordinate system associated with the image frame. (X, Y, Z) is a point on the object described in the visual cue body-fixed frame, and (u, v) is a corresponding point on the image frame (pixel position in the image). Each

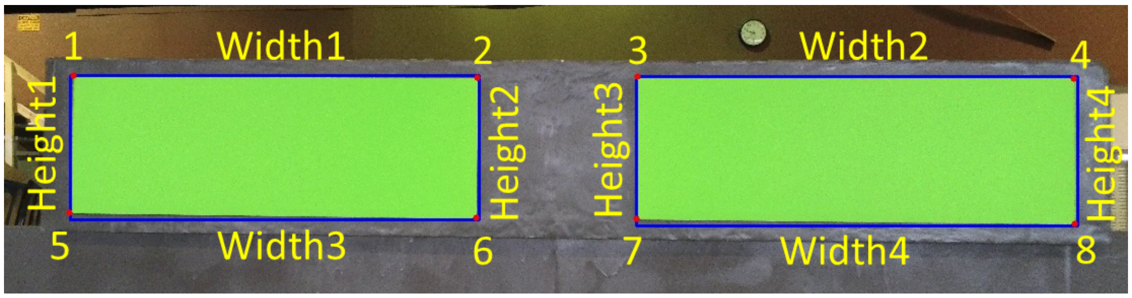


Figure 2.9: Detected corners in sorted order [2] [3]

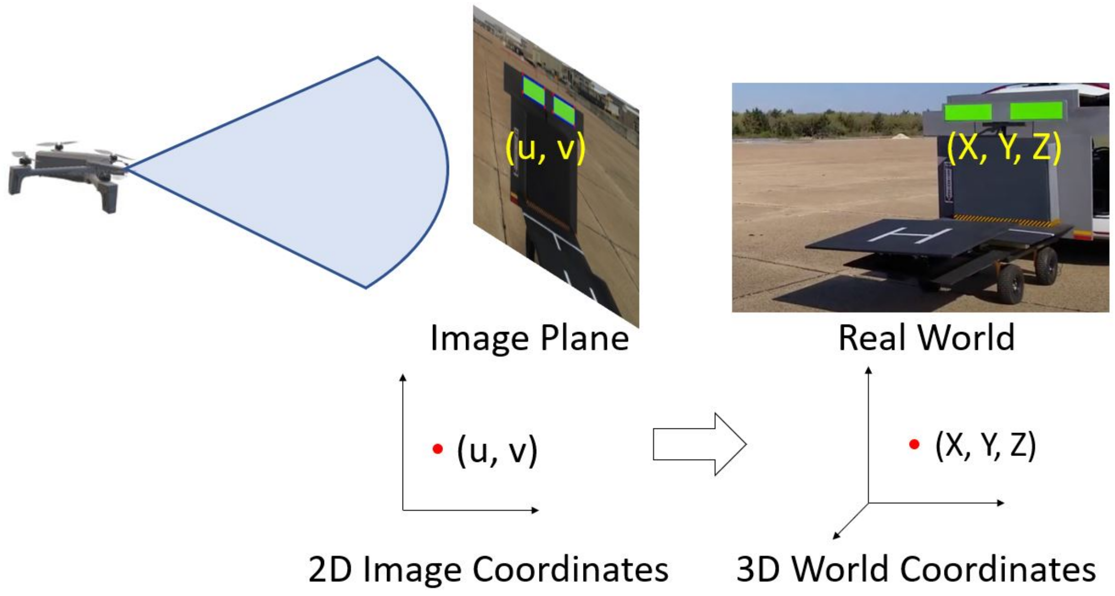


Figure 2.10: Geometric relation of image and real-world coordinates [2] [3] [4]

rectangle in the visual cue is 48 X 16 centimeters. There is an 18.5 centimeters relative separation between the two rectangles. The top left corner of the left rectangle is set to (0,0,0) and the 3-D world coordinates of the remaining corners can be assigned according to the known dimension. Since the object is planar, the Z-coordinate remains zero for all corners. The 2-D image coordinates of the corners can be found in the image. The 3-D coordinates remain the same throughout the flight while the 2-D image coordinates change from frame to frame. The relationship between the image coordinates and the body-fixed coordinates in matrix form is described in Eq. (2.3).

$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \boldsymbol{A_{int}} \begin{bmatrix} \boldsymbol{R} & t \\ 0_{1x3} & 1 \end{bmatrix}^{-1} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.3}$$

The equation is derived in a homogeneous coordinate system. In the homogeneous coordinate system, the 2D image point (u, v) becomes a 3D vector $(u, v, 1)$ and the 3D world coordinates (X, Y, Z) become 4D vector $(X, Y, Z, 1)$. The scaling factor s comes from transforming the homogeneous coordinates to cartesian coordinates. R is a 3 x 3 rotation matrix and t is a 3 x 1 translation vector. R matrix has the information about the camera orientation with respect to the visual cue and t provides the information on how far the camera is from the particular (X, Y, Z) point in the visual cue body-fixed frame. O_{1x3} is 1 x 3 zero matrix. The matrix which includes R, t, O_{1x3}, and 1 is called the camera extrinsic matrix, which varies from image to image. The matrix A is called the camera intrinsic matrix and its components are shown in Eq. (2.4).

$$\boldsymbol{A_{int}} = \begin{bmatrix} 1/\rho_u & 0 & u_0 \\ 0 & 1/\rho_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f_x & 0 & 0 & 0 \\ 0 & f_y & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{2.4}$$

$1/\rho_u$ and $1/\rho_v$ are parameters that scale the image coordinates, which are in pixels to values in the International System of Units (SI). u_0 and v_0 are the center position in the image plane. The first matrix shifts the center of the image plane to the top left corner point. The second matrix contains the information f_x and f_y, which are the focal lengths of the camera in the x and y directions, respectively. The product of two matrices forms the unique camera intrinsic matrix and it can be found using a camera calibration [52].

To determine the relative position and orientation of the camera, the PnP algorithm is applied [53]. Given a set of n 3D coordinates of an object and its corresponding 2D projections on the image, this algorithm solves Eq.(2.3) to obtain the rotation vectors R and the translation vectors

t. There are 6 DOF for a camera, which are 3 DOF in rotation (roll, pitch, yaw) and 3 DOF in translation (X, Y, Z). A minimum of 3 points are required to find a solution, but the solution is not unique. There should be a minimum of 4 points to obtain a unique solution; however, it can be more reliable and redundant when there are more points. An iterative method is used for the PnP algorithm since it is robust for objects which consist of a planar surface. The iterative method is based on Levenberg-Marquardt optimization [54, 55]. In this method, the function minimizes re-projection error, which is the sum of squared distances between the observed image points (u, v) and projected object points (X, Y, Z). By default, the iterative algorithm sets the initial value as zero and then updates during each iteration.

The RANSAC method [56, 57] is used to find a rough initial guess for the extrinsic matrix in an iterative approach. The RANSAC method identifies the outliers and removes them during the calculation. The initial guess and inliers are fed into the iterative algorithm to have a more accurate estimation. The solvePnP algorithm [52] returns a rotation and translation vector. The rotation vector can be converted to the rotation matrix using the Rodrigues function. Thus, once the rotation matrix (R) and translation vector (t) are obtained, $-R^{-1}t$ gives the relative distances in 3D from the camera to the origin of the visual cue body-fixed frame.

The present estimation method is modified from the existing algorithm to take advantage of the gimbal camera. Since the gimbal corrects for the roll and pitch motion of the UAV, the images only reflect the yaw motion (heading angle) with respect to the visual cue. Thus, roll and pitch angles computed by the image can be regarded as camera noise and gimbal correction errors. Therefore, the roll and pitch angles are excluded from the position estimation. The basic and modified rotation matrix are specified in Eqs. (2.5) and (2.6), respectively.

$$\boldsymbol{R_{basic}} = \begin{bmatrix} c\alpha c\beta & c\alpha s\beta s\gamma - s\alpha c\gamma & c\alpha s\beta c\gamma + s\alpha s\gamma \\ s\alpha c\beta & s\alpha s\beta s\gamma - c\alpha c\gamma & s\alpha s\beta c\gamma + c\alpha s\gamma \\ -s\beta & c\beta s\gamma & c\beta c\gamma \end{bmatrix} \tag{2.5}$$

$$\boldsymbol{R_{modified}} = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & -\cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.6}$$

$$\begin{bmatrix} X_{cam} \\ Y_{cam} \\ Z_{cam} \end{bmatrix} = -\boldsymbol{R_{modified}}^{-1} t \tag{2.7}$$

In Eq. 2.5, the s and c stand for sine and cosine, respectively. α, β, and γ represent yaw, pitch, and roll angles, respectively. Each row of the R matrix contributes to their respective coordinates (X, Y, Z). By setting pitch and roll angle to zero, the modified rotation matrix is obtained from the basic rotation matrix. Only the yaw angle remains which takes care of the relative heading angle. The third row of the modified rotation matrix shows that the Z-coordinate is independent of the yaw angle, α. Hence, the yaw angle has no contribution in the estimation of forward relative distance (Z-coordinate) between the UAV and the visual cue. However, the sideward relative distance (X-coordinate) and vertical relative distance (Y-coordinate) are heavily dependent on the yaw angle. The camera position with respect to the visual cue is given in Eq. (2.7)

It was apparent from multiple experiments that the yaw angle estimation becomes noisy as the camera gets farther away from the visual cue. Since the estimation is based on the number of visual cue pixels in the image, the changes in pixels with distance results in the noise. Despite this sensitivity issue, yaw estimation still shows a reasonable trend within the range that the forward relative distance is accurately estimated. To utilize the yaw estimation trend, instead of directly taking the noisy yaw angle estimation, a moving average filter is configured and the results are shown in Fig. 2.11.

The red line denotes the yaw angle estimation results by the basic algorithm and the blue line denotes the results after the moving average filter is applied. The moving average is calculated using the history of the previous estimation data. A lower and upper bound are defined for the next yaw angle estimation value with respect to the current average. Whenever the estimated yaw angle

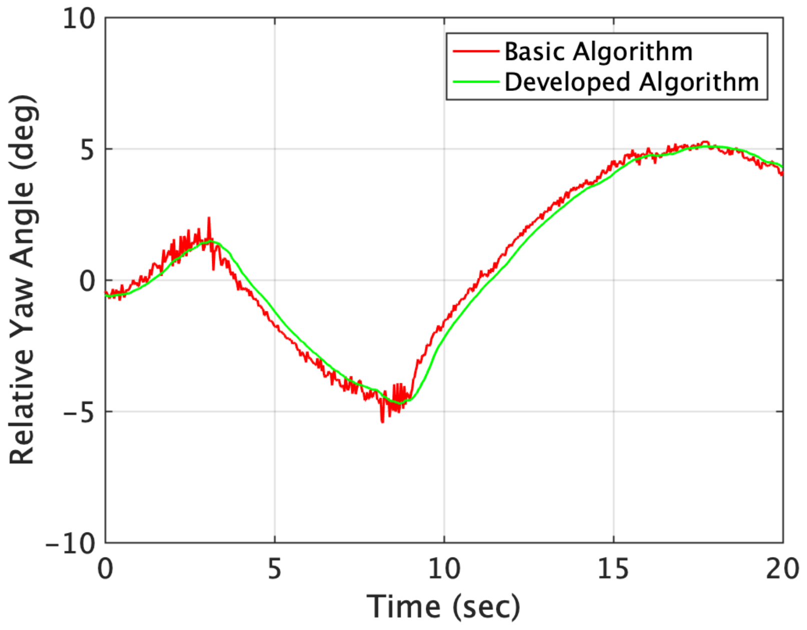


Figure 2.11: Effect of Moving Average Filter on Yaw[1] [3]

is out of the defined range, it rejects that value and takes an average value. In this way, the moving averaging filter provides stable yaw estimation by tracking a trend in a further distance, and the accuracy naturally increases as it gets closer to the visual cue due to having more pixel data in the image.

However, the moving average filter requires multiple accumulated data to predict the current estimation. Thus, a simple low pass filter is configured to reduce the noise level. One of the commonly used real-time data filters is Kalman Filter. Kalman filter predicts the current estimate by taking into account the current measured value, the previous estimate, and the noise level in the data. A single state Kalman filter was opted over other real-time filters because of the lesser number of variables required while computation. Gaussian noise is assumed to be present in the measurement data with a specified mean and variance. The underlying equations involved in estimating the yaw are:

$$\begin{aligned} CE &= PE + KG \times CM \\ KG &= PrE \times \frac{1}{PrE + RN} \\ PrE &= PE + QN \end{aligned} \tag{2.8}$$

where CE is the current yaw estimate, PE is the previous yaw estimate, KG is the Kalman Gain, CM is the current yaw measurement, PrE is the predicted probability estimate, RN is the process noise covariance, and QN is the measurement noise covariance. RN and QN values are experimental values found by analyzing different sets of yaw angle measurements and they are set to 0.005 and 0.05, respectively.

2.1.4 Validation

The results obtained from the developed computer vision system are validated through position and attitude measurements using a Vicon motion capture system shown in Fig. 2.12.

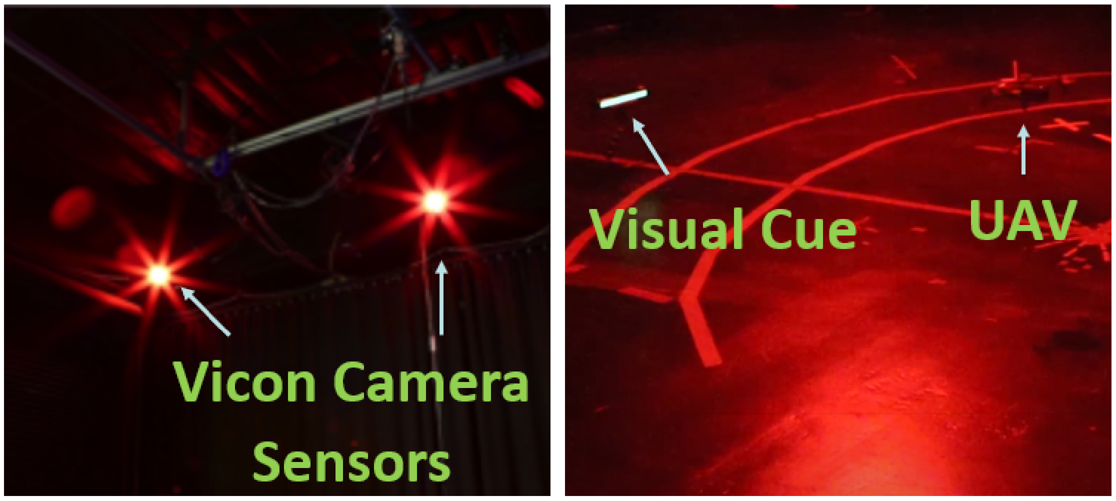


Figure 2.12: Experimental Setup in Vicon System [1] [3]

Vicon is a motion capture system widely used in the entertainment industry to track the motion of people and other objects even up to sub-millimeter resolution. The Vicon system used for this study can capture images at 120Hz using eight 16-megapixel digital video cameras. Camera-mounted strobes illuminate small, retro-reflective markers, which are identified and processed by the imagers. The Vicon cameras track reflective markers on both the UAV and the visual cue to obtain precise position and orientation data, which is then used as the ground truth for this study to validate the computer vision system. The comparisons of the forward relative distance and sideward relative distance are shown in Fig. 2.13 and 2.14, respectively.

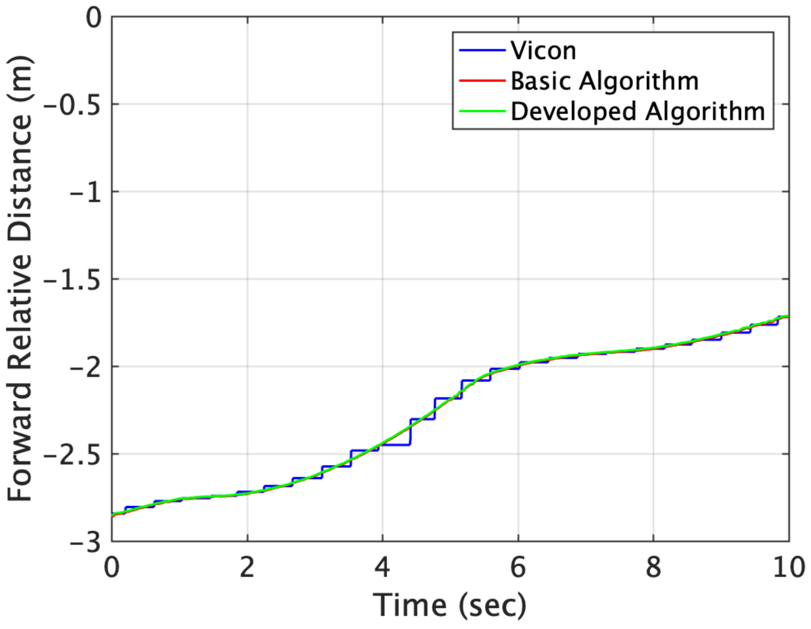


Figure 2.13: Validation of Forward Distance Estimation

The validation of the estimation method as well as the developed algorithm is implemented with a checkered visual cue with 9 corners. The detection method varies depending on the selection of visual cues, however, the estimation method that calculates the relative position and angle is same for any kind of visual cue. Moving average filter is used in the validation data which is later replaced with the Kalman filter. A comparison of the forward relative distances and the lateral relative distances is shown in Fig. 2.13 and 2.14, respectively.

The blue, red, and green lines denote the Vicon measurements, baseline algorithm estimations,

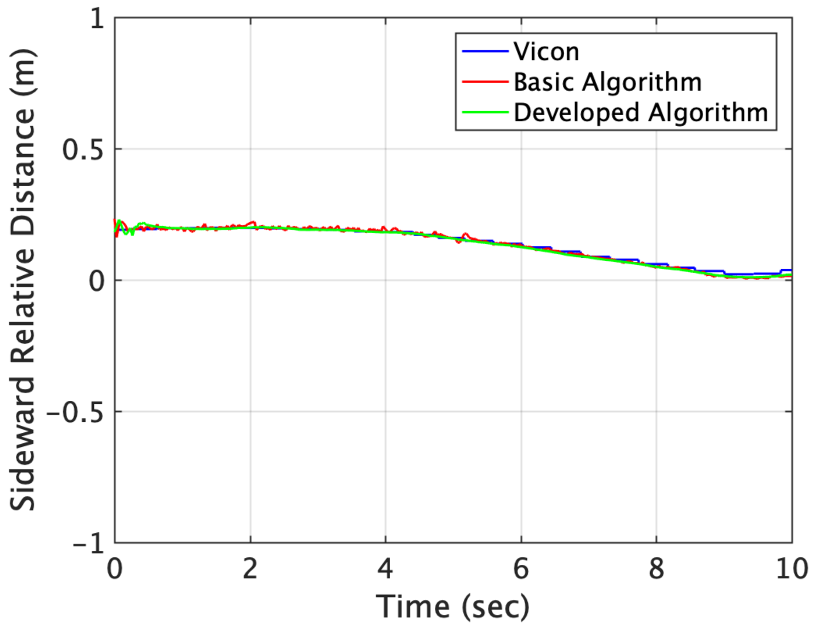


Figure 2.14: Validation of Sideward Distance Estimation[1] [3]

and estimations from the present improved algorithm, respectively. As seen from the figure, the baseline and the improved algorithms have no difference when it comes to the forward relative distance estimation, and both these methods can obtain the same level of accuracy as the Vicon measurements. The sideward distance estimation by the baseline algorithm has fluctuations; however, the improved algorithm shows smooth results due to the moving average filter. The maximum error in sideward distance estimation is one centimeter, when compared to the Vicon result. The comparisons of the vertical relative distance and relative yaw angle are as shown in Fig. 2.15 and 2.15, respectively.

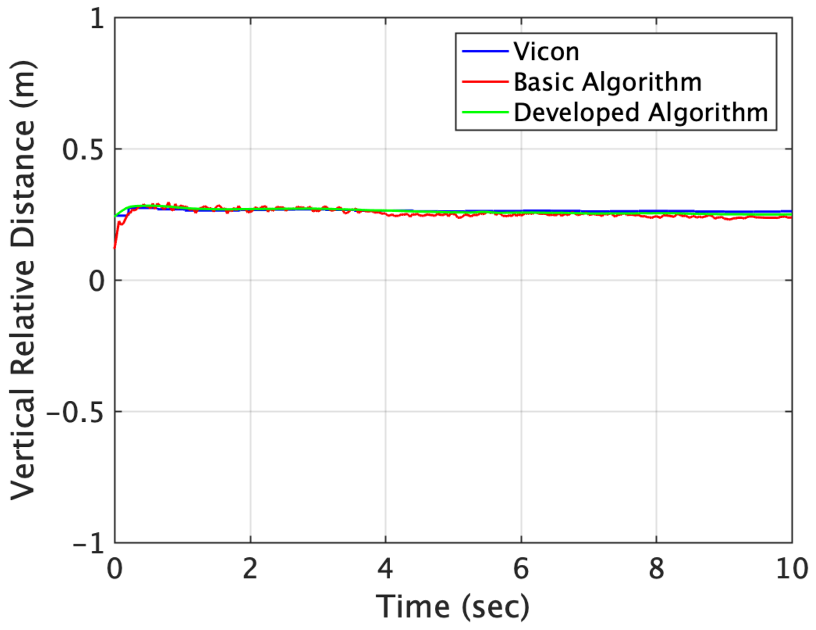


Figure 2.15: Validation of Vertical Distance Estimation[1] [3]

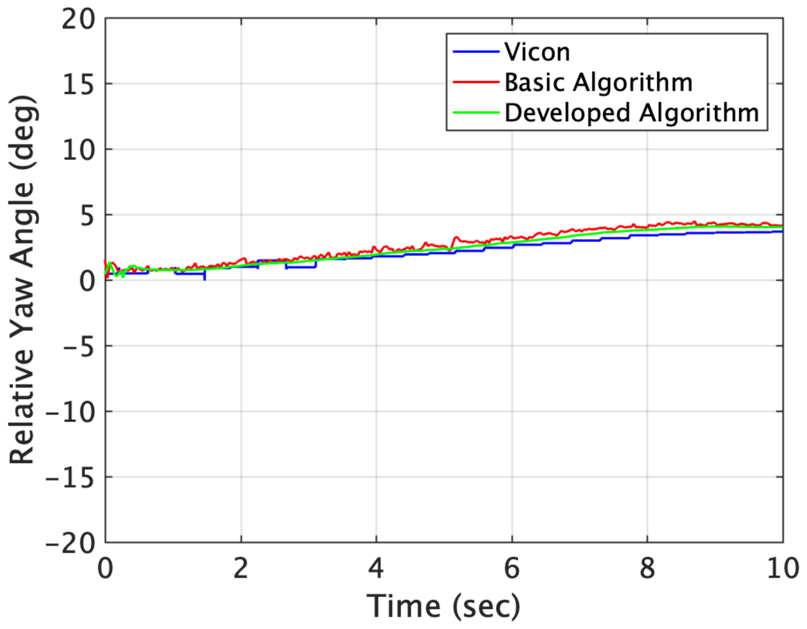


Figure 2.16: Validation of Yaw Angle Estimation[1] [3]

In both vertical relative distance and relative yaw angle comparison, the improved algorithm yields smooth estimation results. The maximum error in the vertical relative distance is below one centimeter and the maximum error in the relative yaw angle is one degree. Through the Vicon experiments, the ability of the present computer vision system to detect the visual cue and precisely estimate the position and orientation is demonstrated.

3. CONTROL SYSTEM

[1] This chapter includes a detailed description of the control system that has been used for the autonomous ship landing procedure is discussed. First, a conventional gain-scheduled controller was developed to test the concept, and then a non-linear conventional controller was developed for better long-range tracking and accurate landing. Finally, a Reinforcement Learning based control strategy is developed for superior disturbance rejection and accurate landing. All the control systems are comprehensively validated extensively simulated in the Gazebo simulation program before flight tests.

3.1 Gain-scheduled Control Strategy

As an initial step towards implementing the novel autonomous ship landing method in practice, a simple gain-scheduled conventional control system is designed to approach and land following a checkerboard pattern visual cue installed on the ship platform as shown in Fig.2.4.The vision-based controller will form the outer-loop of a cascaded feedback control system where the inner-loop will handle the basic attitude stability of the aircraft as well as generates the required pitch, roll, heave and yaw rates as demanded by the outer-loop. Proportional integral derivative (PID) controllers with scheduled gains are designed to achieve a realistic flight dynamic behavior as well as precise tracking and landing of the VTOL UAV. The standard form of the PID controller is represented in Eq. (3.1).

$$u(t) = K_P e(t) + K_I \int_0^\tau e(\tau)d\tau + K_D \frac{d}{dt} e(t) \tag{3.1}$$

$e(t)$ denotes an error between a setpoint and relative position data with respect to the visual cue. Since the positions and heading angle are computed from the vision system, disturbances and noises are already included. The update rate of the outer loop of the closed-loop feedback control system is 0.1 seconds including the detection, estimation, and live-streaming through the computer

[1]The chapter has been adapted or reprinted from recent publications [1] [2] [3] [4].

vision system. K_P, K_I, and K_D gains are tuned based on the effect of each parameter as shown in Table 3.1.

Table 3.1: Effect of K_P, K_I, and K_D

Characteristic	Increase K_P	Increase K_I	Increase K_D
Rise Time	Decrease	Small Decrease	Small Decrease
Overshoot	Increase	Increase	Decrease
Settling Time	Small Increase	Increase	Decrease
Steady-state	Decrease	Large Decrease	Minor Change
Stability	Degrade	Degrade	Improve

The different sets of gains are scheduled based on relative positioning data to control the behavior of a UAV as desired. It allows the UAV to fly at high-speed when it is relatively far away from the visual cue and then slow down the relative speed for cautious tracking when it gets closer to the landing point. The flight mode selector is also configured in order to adapt the UAV movements to the flight conditions and enhance safety. The states are the relative positions in three dimensions and the heading angle of the UAV with respect to the visual cue obtained from the computer vision system. The control inputs are pitch, roll, yaw, and throttle commands. Each control magnitude ranges from -100 to 100 as a percentage of the total input. The outputs are UAV attitude and raw images from the camera. The UAV attitude consists of roll angle ϕ, pitch angle θ, and yaw angle ψ.

3.1.1 Simulations

The objective is to verify every aspect of the vision-based flight control system under different scenarios and to demonstrate that extreme accuracy can be achieved by adopting this novel method of tracking a visual cue installed parallel to a UAV approach course instead of tracking a landing platform.

In the present study, the entire coding is done in the Python programming language and a realistic robot simulation program, Gazebo [58], is used to simulate the physical and visual surroundings of a UAV as shown in Fig. 3.1.

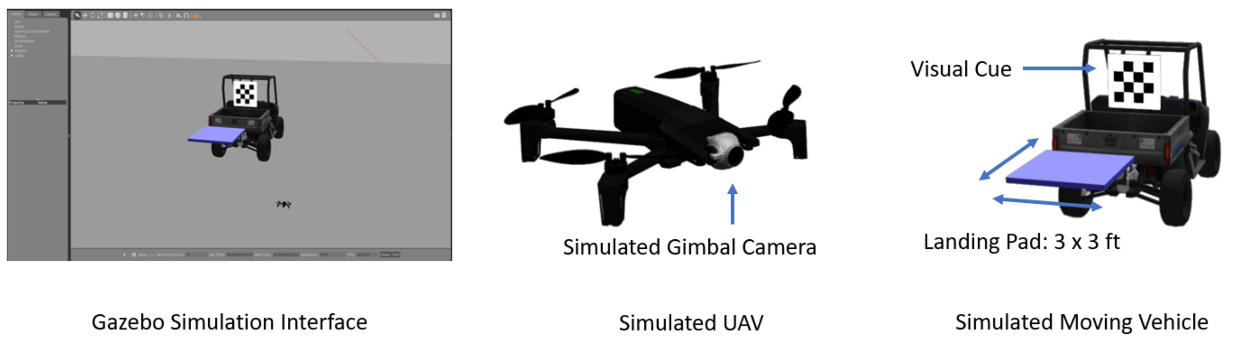


Figure 3.1: Simulated UAV and Moving Platform in Gazebo Simulation Program [1] [3]

The simulated UAV model has the same dimensions, mass moments of inertia, gimbal camera, and flight mechanism as the physical UAV, Parrot Anafi. The simulation also implements a moving vehicle that carries the visual cue and landing platform as shown in Fig. 3.1. As determined by vision system experiments, the visual cue has a checkerboard pattern consists of 4 x 4 squares with a side length of 120 mm each, and the landing platform size is 3 x 3 ft. These characteristics remain the same throughout simulations and flight tests.

The simulations are conducted in the same way as the actual flight tests including live-streaming, real-time vision processing, and feedback control loop. First, the simulated UAV is connected via WIFI to the base station computer. Once the UAV takes off, the images from the simulated camera are streamed to the computer. The images are processed one by one to obtain the relative position and heading information, which is utilized by the feedback controller running on the computer to generate the control inputs. Then, the computed control inputs are sent back to the UAV via the WIFI connection. This one cycle from streaming to sending commands back to the UAV takes 0.1 seconds. This simulated system, once verified, can be directly applied to a real flight test by simply switching an IP address from simulated UAV to physical UAV. The following simulation results show the tracking capability and landing accuracy in detail.

3.1.1.1 Landing Accuracy

To demonstrate the tracking and landing accuracy of the present vertical-visual-cue based method, multiple simulations are conducted with different landing thresholds and varying the speed of the moving platform. The final landing locations from 100 independent simulations are plotted in Fig. 3.2.

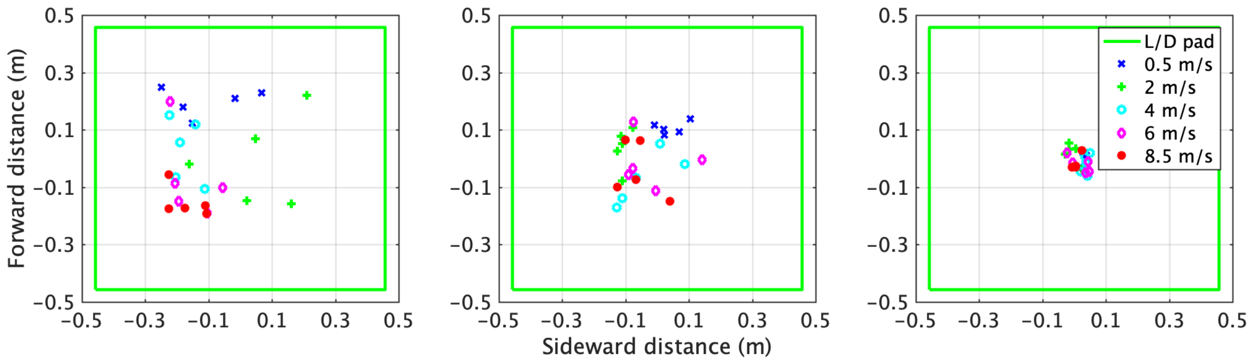


Figure 3.2: Landing Points with Landing Threshold of 25 x 25cm(left), 15 x 15cm(center), and 5 x 5 cm(right)[1] [3]

The green line depicts the landing pad and each mark denotes the landing point for different speeds of the moving platform. Note that landing anywhere within the 25 x 25 cm area can be regarded as a safe landing considering the UAV size. The present vision-based control system achieves precise landing within a 5 x 5 cm landing threshold for all platform speeds. When the platform is moving at 8.5 m/s, it requires the UAV to fly close to its maximum speed but it still can make accurate landings. The dispersion of landing points in the figure can be attributed to the assigned landing threshold. The time history of relative distance for each landing case is shown in Fig. 3.3.

Each line denotes the relative distance between the UAV and landing pad center at different

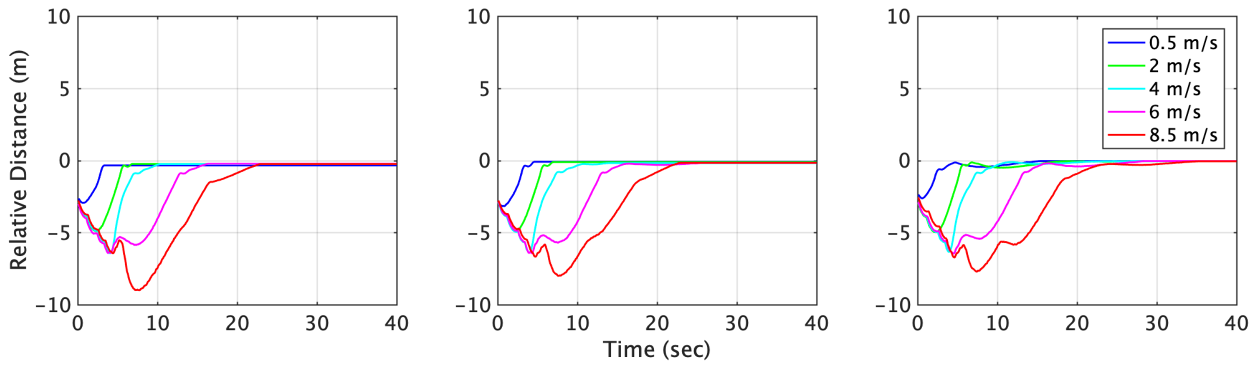


Figure 3.3: Relative Distance with Landing Threshold of 25 x 25cm(left), 15 x 15cm(center), and 5 x 5 cm(right)[1] [3]

platform speeds. By the time the UAV starts tracking after take-off, the platform is already accelerating forward up to its designated speed. Thus, the relative distance increases at the beginning of the tracking phase. Also, it takes more time to land as the landing threshold is made smaller. The average time to achieve a landing threshold in each case is specified in Table 3.2.

Table 3.2: Average Time to Achieve Landing Threshold

Vehicle Speed	5x5cm	15x15cm	25x25cm
0.5m/s	15.3s	4.6s	3.3s
2.0m/s	25.6s	7.0s	6.8s
4.0m/s	26.9s	13.8s	10.3s
6.0m/s	28.6s	16.4s	16.2s
8.5m/s	35.2s	23.0s	22.9s

In conclusion, the landing accuracy depends on an assigned landing threshold. The present system can achieve a precise landing, however, there is a trade-off between the time and landing accuracy. Thus, the landing threshold should be selected based on the priority of a mission, which could be accuracy or time. In the following simulations, the medium landing threshold of 15 x 15 cm is applied.

3.2 Nonlinear Control System

The sequential steps involved in the vision-based control system are live-streaming the video from UAV's onboard camera to an external base-station computer, image processing, feedback controller generating the control inputs, and transmitting the control inputs back to the UAV. The processing time is considerably affected by the type of detection method engaged in the vision system. The average time to complete one cycle is 0.5 seconds when the machine learning detection is engaged and 0.03 seconds when the detection of the rectangle corner points is engaged. In order to cope with the time delay as well as potential sensor and estimation noise, nonlinear controllers, which can adapt to different situations are developed. Particularly, five different flight modes are configured according to the situations perceived by the vision system as shown in Fig. 3.4.

In the case that the computer vision system provides estimated positioning data based on corner points detection at a close distance, mode 1 triggers vertical landing command upon the satisfaction of landing condition and mode 2 makes the UAV approach the designated landing point by engaging the probabilistic nonlinear control system. At a long distance where the corner points detection is unavailable, the machine learning-based vision system provides the object area and location in the image at a long distance. In this case, the exponential nonlinear control system is primarily engaged with mode 3 if the horizon bar is detected. If not, mode 4 is engaged based on ship platform detection. It is designed to utilize previously fed data when no vision data is available, which is activated by mode 5.

3.2.1 Long-range Tracking Controller

In the long-range where the entire landing platform is detected as a whole, the information provided by the vision system is the platform size and position in the camera view. The flight control system in this region deals with a relatively slow update rate that is 0.5 seconds on average. When the update rate is fast enough, the discrete system that receives sensor data at each update time can be a good approximation to the continuous system. Thus, integral and derivative controllers

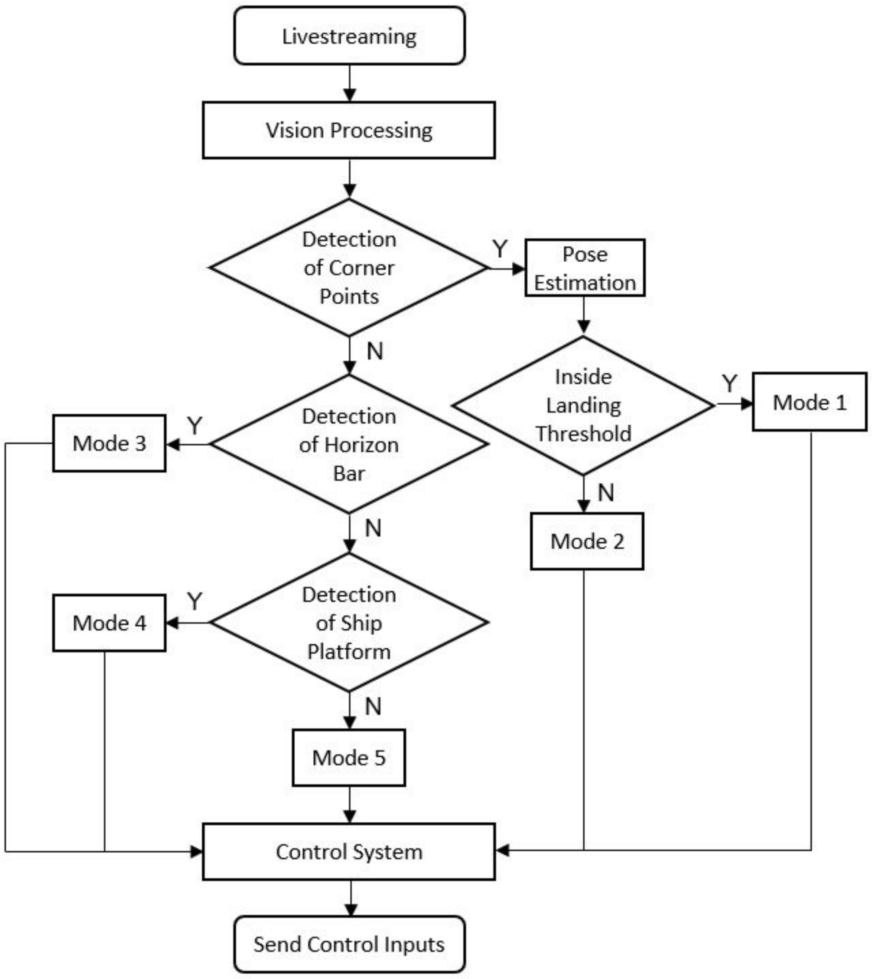


Figure 3.4: Flowchart showing vision-based control system [2] [3]

can be configured by using the summation and difference of error. However, the summation and difference are not good approximations for integral and derivative controllers when the update rate is slow. Hence, the nonlinear controller is designed to achieve the control task in the presence of the slow update rate by applying the nonlinear exponential gain $K_P\{e(t_k)\}$ as shown in Eq. (3.2).

$$\begin{aligned} e(t_k) &= r(t_k) - c \\ u(t_k) &= K_P\big\{e(t_k)\big\} \times e(t_k) \\ &= \begin{matrix} -m(e^{ae(t_k)} - d) \times e(t_k) & (e(t_k) < 0) \\ m(e^{ae(t_k)} - d) \times e(t_k) & (e(t_k) \geq 0) \end{matrix} \end{aligned} \tag{3.2}$$

In Eq. (3.2), $e(t_k)$ denotes the error at time t_k, $r(t)$ is the relative position at time t_k, and c is the setpoint. The control law, $u(t_k)$, has exponential term in the nonlinear gain $K_P\{e(t_k)\}$ to decay control magnitude exponentially near zero error. The constants m, a, c, and d selected for the ship platform and bar tracking are provided in Table 3.3.

Table 3.3: Selected constants for nonlinear exponential controller

Flight Mode	Controller	m	a	c	d
Ship Platform Tracking	pitch	0.008	0	5000	0
	roll	1.2	0.0158	640	1
	heave	3.0	0.0108	360	1
Bar Tracking	pitch	0.004	0	3400	0
	roll	1.0	0.0158	640	1
	heave	5.0	0.0108	360	1

The desired object size, the image center position in the horizontal direction, and the image center position in the vertical direction are the setpoints for the pitch, roll, and heave controllers, respectively. Unlike the roll and heave controllers, the pitch controller with the assigned parameters returns to a linear proportional controller, which is sufficient to approach the ship in the long-range. There is also a yaw controller that can control the heading angle to set the approach course. It is designed as a nonlinear probabilistic control system, which is the same as the close-range tracking yaw controller and detailed in the following section. The only difference is that a magnetometer is used to read the current heading in the long-range and the vision system provides the estimated heading angle during close-range tracking. The nonlinear roll control input is shown in Fig. 3.5.

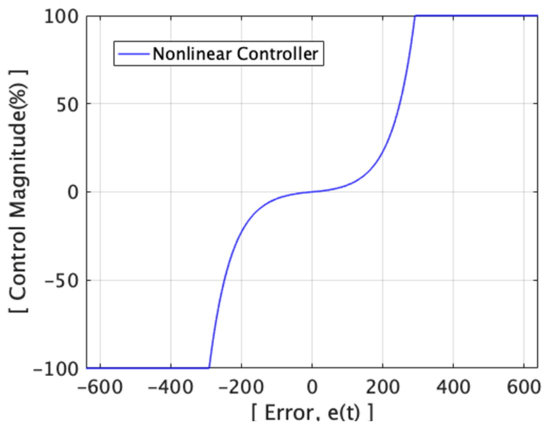


Figure 3.5: Exponential nonlinear roll control input [2] [3]

Depending on the constant value a, the rate of change of control magnitude varies. In the roll controller case, constant a is selected as 0.0158. The maximum and minimum control magnitudes are limited to 100 and -100, respectively. Even though it utilizes only the error at time t, it can minimize the overshoot around the setpoint where the error is zero by decaying quickly. On contrary, it yields a large control magnitude as the error becomes larger.

In the case of a slow update rate, the nonlinear controller is able to achieve stable setpoint tracking. The effect of the nonlinear roll controller is compared to the conventional linear proportional controller and PID controller as shown in Fig. 3.6.

It demonstrates that the nonlinear controller can prevent the system from overshooting the setpoint. However, the linear proportional controller cannot stabilize the oscillations because it computes the control input by multiplying the error with a fixed gain value. Even the linear PID controller is not able to stabilize the oscillations in an effective fashion due to the slow update rate.

However, as seen from Fig. 3.6, the nonlinear controller has some steady-state error. Considering the long distances at which the nonlinear controller is engaged, the steady-state error is not an

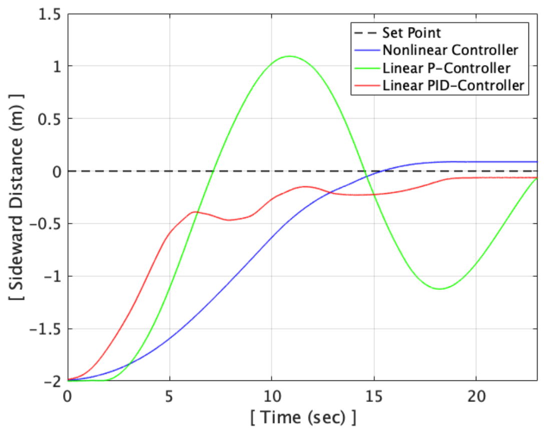


Figure 3.6: Effect of exponential nonlinear Controller [2] [3]

issue because this is not the final error of the entire approach and landing maneuver, but the initial error for the corner tracking system, which takes over the control at close distances. Therefore, it is more imperative to control the aircraft stably than to achieve the minimum steady-state error while the UAV is approaching from a long distance.

3.2.2 Close-range Tracking Controller

At close range, when the vision system can reliably detect the corners of the rectangles on the visual cue, the vision-based controller utilizes the estimated relative position and orientation data to yield control inputs. The average update rate is 0.03 seconds, which is significantly faster than the machine learning object detection that is applied at long distances. Having integral and derivative controllers along with the proportional controller enables precise tracking. Even though the estimation provides accurate position and orientation data with sub-centimeter and sub-degree error, a small error difference between subsequent time steps can yield large and noisy control magnitude since it depends on the difference in error divided by the small update rate of 0.03 seconds. To take advantage of the derivative controller with minimum noise, the Kalman filter and nonlinear derivative controller are designed.

A single state Kalman filter is applied with the unity feedback, which means that it does not require the prediction from the model. This model-free estimator reduces the noise, however, it uses error difference values for estimation. Therefore, it will be affected by incorrect and unrealistic error difference values that may occur from time to time. To reject this intermittent unrealistic estimation effectively, a novel nonlinear derivative controller with linear proportional and integral controllers is designed by utilizing the normal distribution concept as described in Eq. (3.3).

$$\begin{aligned} de(t_k) &= e(t_k) - e(t_{k-1}) \\ u(t_k) &= K_P e(t_k) + K_I \sum_{k=1}^{k} e(t_k) dt_k + K_D \big\{ de(t_k) \big\} \frac{de(t_k)}{dt_k} \end{aligned} \tag{3.3}$$

$de(t_k)$ is an error difference between time t_k and t_{k-1}, and $u(t_k)$ is a control law that has linear proportional and integral terms as well as the nonlinear derivative term. K_P and K_I are constant proportional and integral gains, and $K_D\{de(t_k)\}$ is the nonlinear derivative gain that is a function of error difference, $de(t_k)$, as specified in Eq. (3.4).

$$\begin{aligned} f(x) &= \frac{1}{\sqrt{2\pi\sigma^2}} e^{-0.5\frac{(x-\mu)^2}{\sigma^2}} \\ K_D\big\{de(t_k)\big\} &= b e^{-0.5\frac{(de(t_k)-\mu)^2}{\sigma^2}} \end{aligned} \tag{3.4}$$

$f(x)$ is the probability density function that forms a normal distribution. σ denotes the standard deviation and μ denotes the mean value. The area under the function indicates the probability that a certain range of deviation occurs. The probabilistic nonlinear derivative controller is constructed by taking the exponential term from the probability density function and multiplying constant b that determines the control magnitude. Based on the observation of aircraft movement, σ is determined as 0.04, which means the distance that the aircraft can move during 0.03 seconds has a 68.2 percent chance of being within 4 cm and a 95.4 percent chance of being within 8 cm. When b is 1 and μ is 0, the probabilistic nonlinear derivative controller computes derivative gains as shown in Fig. 3.7.

Depending on the error difference, $de(t_k)$, the corresponding nonlinear derivative gain, K_D, is selected and multiplied by the derivative term $de(t_k)/dt_k$. If the estimated error difference is too

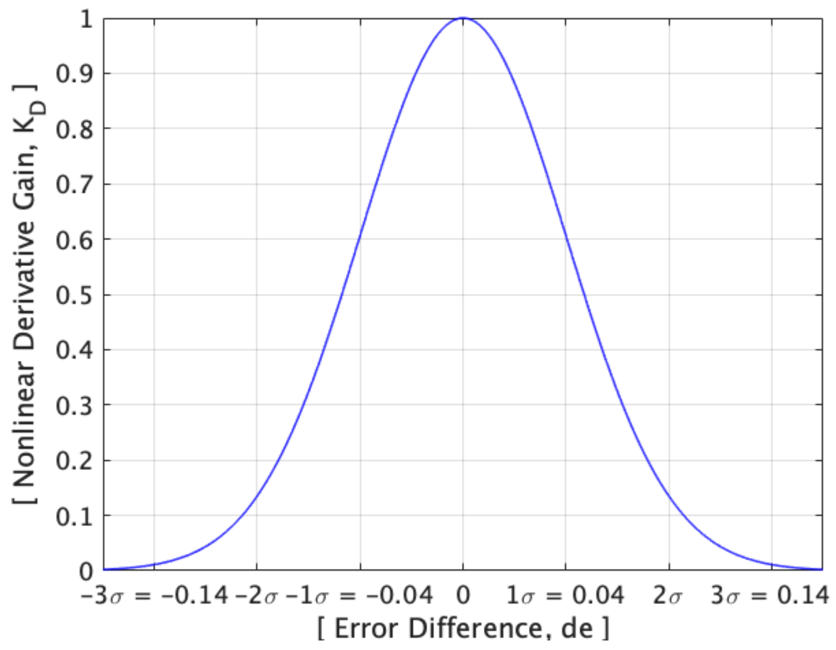


Figure 3.7: Probabilistic nonlinear derivative controller gain [2] [3]

high (or unrealistic), then it takes a small K_D value to significantly minimize the control input. In this way, the controller does not respond to the large noise in the error data, which can trigger undesired and unstable maneuvers. Also, the magnitude of gain can vary according to the selection of constant b in Eq. (3.4). The effects of the Kalman filter and the probabilistic nonlinear controller are shown in Fig. 3.8.

Red circles denote the control magnitude of the linear derivative controller without the Kalman filter. The high noise that occurs in the range of 0 to 10 seconds are due to the derivative term, $de(t_k)/dt_k$. This term is sensitive because the error is divided by a small dt_k, which has an average value of 0.03 seconds. Thus, even a small error in estimation can be magnified in the derivative controller. The green line is the result after applying the Kalman filter to the linear derivative controller. The noise is reduced, however, it still yields large control inputs in response to the incorrect estimation values. The blue line shows the control inputs generated by the probabilistic nonlinear controller with the Kalman filter and these inputs are relatively small and insensitive to the large unrealistic error differences. At the 6.5 second mark, the error difference de has a large

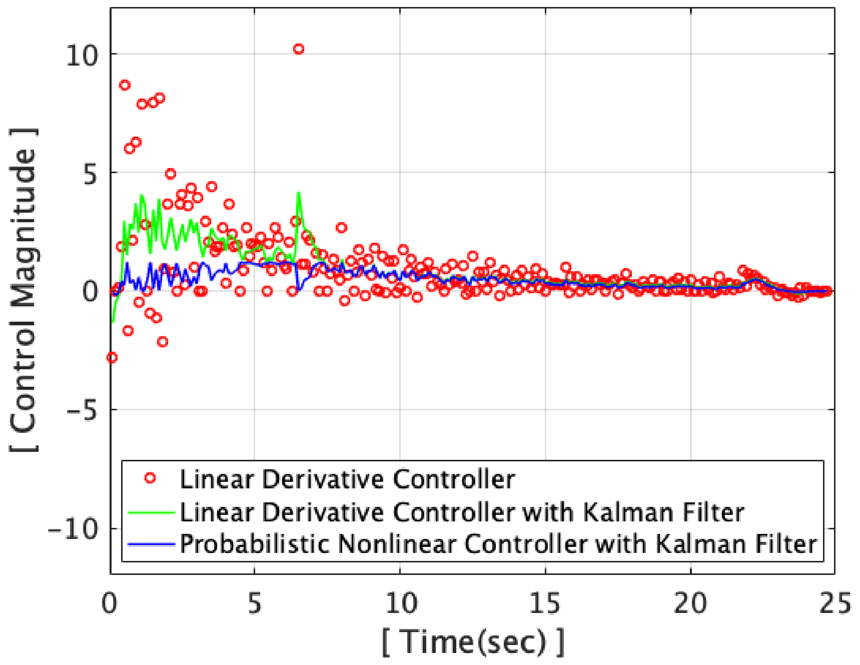


Figure 3.8: Effect of Kalman filter and probabilistic nonlinear derivative controller [2] [3]

value caused by incorrect estimation. In this case, the linear derivative controller with the Kalman filter reduced the magnitude to some extent but it is still affected by that particular spurious error value. However, the probabilistic nonlinear controller with the Kalman filter is able to screen out the wrong value, thereby minimizing the effect of incorrect error estimation. The finally selected gains through extensive flight tests and simulations are specified in Table 3.4.

Table 3.4: Selected gains for nonlinear controllers

Mode	Controller	K_P	K_I		K_D	
				b	μ	σ
Corner Points Tracking	pitch	7.5	0.05	4.5	0.02	0.04
	roll	7.5	0.01	8.5	0	0.04
	heave	15	0.01	7	0	0.02
	yaw	5.5	0.05	1.75	0	5

3.3 Reinforcement Learning Control Strategy

Designing a conventional feedback control system requires a detailed model of the underlying system and computationally expensive procedures for finding a control policy that can achieve the desired objectives. Adaptive control approaches [59, 60, 61] can learn a control policy without the explicit system models. However, they require approximations of the uncertainties that can occur, and this approximation adds additional parameters that make the algorithm more complicated. The more advanced way of finding the optimal control strategy is by learning approaches. Online learning methods [62, 63] can learn the flight dynamics in real-time. However, their control capabilities are limited to the situations they experienced during the training/learning session, thus any inexperienced events can degrade the performance. There are also offline learning methods such as supervised learning [64, 65] and reinforcement learning [41, 44, 66]. The supervised learning method requires a large number of states and action data to learn the proper control strategy for different scenarios. It is difficult to obtain such a large amount of quality data and the capability of the method cannot surpass the training set's control quality since it learns from the provided data. We propose an RL-based control strategy for VTOL UAVs.

3.3.1 Reinforcement Learning Overview

Reinforcement Learning is a very intuitive state-of-the-art machine learning-based approach for finding the optimal control policy for an unknown dynamical system. In this approach, an agent interacts with an environment to learn the best action for any given state. After executing each action, the agent receives a reward and the system evolves to the next state. The primary objective of the agent is to learn the optimal policy that can maximize the cumulative rewards.

Mathematically, a policy $\pi \in \Pi$ is a function that maps state $s \in S$ to an action $a \in A$, where Π is the policy space, S is the state space, and A is the action space. The transition probability model $P(s_{t+1} \mid s_t, a_t)$ determines the dynamics of the system. At each timestep, policy π determines an action to be taken and the agent receives a reward $r(s_t, a_t)$ from the environment. The policy can be stochastic $\pi(a, s)$ (probability of taking action a in state s) as well as deterministic $\pi(s)$. The agent tries to maximize the cumulative discounted reward $R_t(s) = \sum_{i=t}^{T} \Gamma^{(i-t)} r(s_i, a_i)$ of state s from time t to T. The T can be finite value (finite horizon episodes) or infinite value (infinite horizon episodes). The parameter Γ is know as discount factor that determines the importance of future rewards. The two other important terms in RL formulation is the value function V^π and action-value function or Q-function Q^π. Value function is the expectation of discounted cumulative reward for a state s, defined as $V^\pi(s) = \mathbf{E}_\pi[R_t(s) \mid s]$. The optimal policy π^* can be expressed as one which maximizes the value function, $\pi^*(s) = \arg\max_\pi V^\pi(s)$.

3.3.2 Algorithm Selection

Robotic problems in general are associated with continuous state space and continuous action space. With the advancement in deep neural networks over the years, combining RL with deep learning shows impressive learning results. A deep version of Q-learning (DQN) algorithms has been successful in playing video games [67, 68]. Two key features of DQN are the experience replay and a target network. The experience replay stores prior experience (s, a, r, s', a') in a buffer and randomly select one for updating Q-value network. Target network helps in target remaining unchanged during several gradient updates. However, DQN algorithms were not successful in continuous action-continuous states space. Policy gradient algorithms are more useful in continuous action-continuous states space problems. Deep deterministic policy gradient (DDPG) algorithm [69] can operate over continuous action and state spaces and it uses experience replay and slow-learning neural networks of the double Q learning. Particularly for the control of a UAV, DDPG has demonstrated its successful implementations [44]. In this study, an improved version of DDPG, twin delayed DDPG (TD3) [70, 71], is selected as the RL algorithm.

TD3 learns two Q-functions Q_{ϕ_1} and Q_{ϕ_2} by mean square Bellman error minimization. It adds clipped noise to actions that make the policy difficult to extract the Q-functions. The target action is obtained as described in Eq. (3.5).

$$a'(s') = \text{clip}(\pi_{\theta_{\text{targ}}}(s') + \text{clip}(\epsilon, -c, c), a_{\text{low}}, a_{\text{high}}), \epsilon \sim N(0, \sigma) \tag{3.5}$$

$\pi_{\theta_{\text{targ}}}(s')$ is the target policy and $a_{\text{low}} < a < a_{\text{high}}$ is the valid action range. This is known as target policy smoothing which essentially serves as regularizer for the algorithm. TD3 updates the policy less frequently than the Q-functions. Both Q-functions use the same target $y(r, s', p)$ (p indicates whether state s' is terminal state or not) which is the calculated using the smaller Q-value among the two Q-functions as given in Eq. (3.6).

$$y(r, s'p) = r + \Gamma(1-p) \min_{i=1,2} Q_{\phi_{\text{targ},i}}(s', a'(s')) \tag{3.6}$$

The two Q-functions are then optimized by using the following two loss functions (mean squared Bellman error function) given in Eqs. (3.7) and (3.8).

$$L(\phi_1, D) = \mathbf{E}[(Q_{\phi_1}(s,a) - y(r, s', p))^2 \mid (s, a, r, s', p) \sim D] \tag{3.7}$$

$$L(\phi_2, D) = \mathbf{E}[(Q_{\phi_2}(s,a) - y(r, s', p))^2 \mid (s, a, r, s', p) \sim D] \tag{3.8}$$

ϕ_1 and ϕ_2 are network parameters and D is the replay buffer.

3.3.3 Training Procedure

The objective of the RL control strategy is for the UAV to stably hover and fly as desired against sudden wind gusts. To obtain an RL control policy that is effective in such a challenging condition, training is conducted in a realistic Gazebo simulation environment along with the precise UAV model as shown in Fig. 3.9.

During the training session, an agent takes some set of actions in an environment to achieve a particular task. The agent is a UAV that decides actions based on the rewards by using neural networks. The quality of those actions is quantified by user-defined rewards that are assigned in the environment and the better sets of actions to achieve the task will acquire higher cumulative rewards. A particular period that the agent takes for a set of actions by the RL control policy is called an episode. As the episode continues, the control policy keeps getting updated according to an RL algorithm and the cumulative rewards will be converged to a certain value.

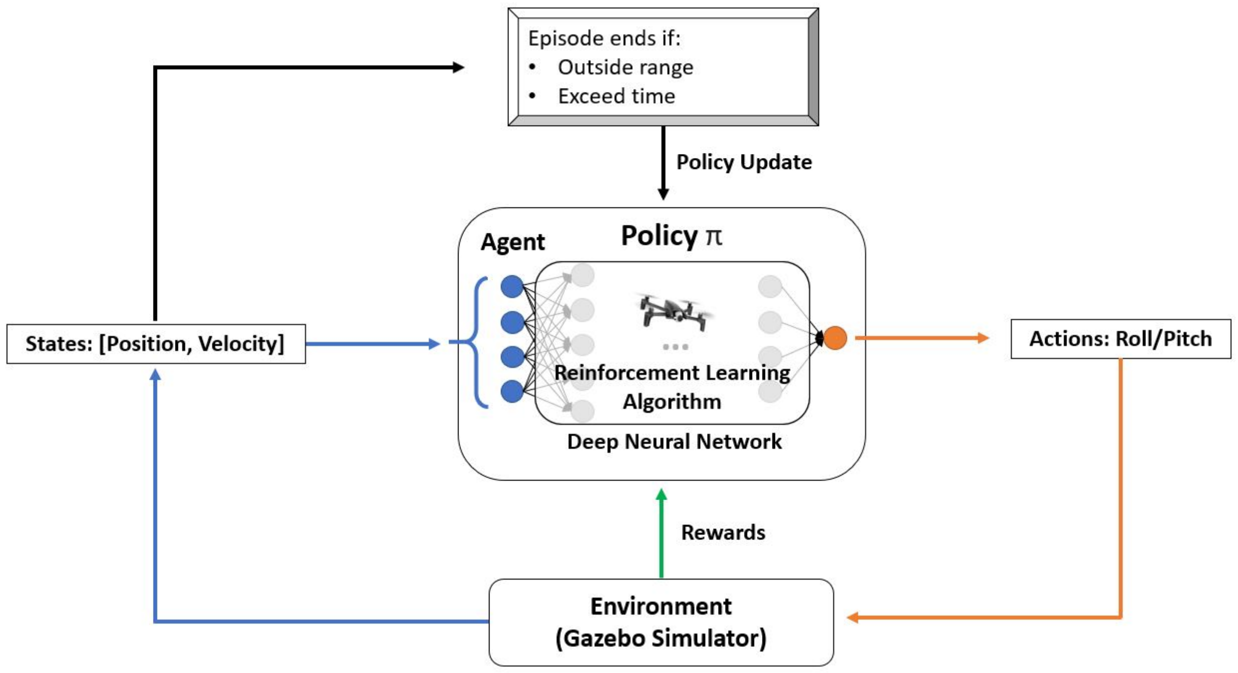


Figure 3.9: Schematic showing the training process for reinforcement learning [3] [4]

To develop a successful RL control strategy, there are key requirements which are: (1) a precise model of the agent, (2) a realistic simulation engine, (3) an appropriate selection of states, (4) careful reward engineering, and (5) a powerful RL algorithm.

First, a commercial off-the-shelf (COTS) quadrotor UAV, Parrot Anafi, is used for this study. Along with the physical UAV, the precise UAV model(agent) that can be used in the Gazebo simulation is provided. Second, Gazebo is a realistic simulation engine that is widely used for robotics applications. It has closely simulated the real flight dynamics. The agent can communicate with the simulation engine using a framework called Olympe. Olympe allows the control of UAV and access to its sensors through python scripts.

3.3.3.1 Action Space

The UAV has four different control actions: Roll, Pitch, Yaw, and Heave. It is observed in the simulation that roll and pitch actions were heavily influenced by the wind as compared to the yaw and heave actions. Hence the roll and pitch actions are chosen to be part of the action space. It is worth noting that for the given UAV, the roll and pitch actions are independent of each other. Roll action only makes the UAV move right and left, while pitch action only makes it move forward and backward. Taking advantage of this uncoupled behavior, it is decided to split the 2-D action space into two 1-D action spaces. Two independent RL models were trained for roll and pitch controllers, respectively. This not only helps in reducing the dimension of the action space but also reduces the dimension of the state space used for the independent models. The splitting of the two controllers helps in reducing the complexity of the policy network and hence resulted in faster training and better convergence.

3.3.3.2 State Space

As mentioned earlier, the mission objective of the UAV is to achieve a target point and hover at that position in the presence of wind. So the position of the UAV is an obvious choice as a state. Since the action space is 1-D, the position state is also 1-D that is the position along the respective action axis for each controller. The roll controller objective is to achieve a certain target on the roll axis and maintain that position without considering the pitch motion. Similarly, for the pitch controller, the objective is to maintain a position on the pitch axis independent of the roll motion. Also, velocities are included as states, which can take an action quickly responding to velocity changes before the drift becomes large. Finally, the history of position and velocity are also selected as states for both roll and pitch controllers, which allows determining actions based on current states and trends.

To verify the effect of selected states, three different sets of states as shown in Table 3.5 are modeled for pitch control and roll control individually and applied for learning.

Table 3.5: Models with different sets of states

Model No.	Model 1	Model 2	Model 3
States for pitch control	$[x]_t$	$[x, v_x, v_y]_t$	$[(x, v_x, v_y)]_{t-i}$ where i = 0 to 5
States for roll control	$[y]_t$	$[y, v_x, v_y]_t$	$[(y, v_x, v_y)]_{t-i}$ where i = 0 to 5

x and y are the deviations along the x-axis and the y-axis, respectively. v_x and v_y are the velocities along the x-axis and the y-axis, respectively. t is the current time, thus model 3 is also taking five previous deviations and velocities as states.

3.3.4 Domain Randomization

In most of robotics design, one of the difficult problem is how to make your model transfer to the real world. Due to the sample inefficiency of deep RL algorithms and the cost of data collection on real robots, we often need to train models in a simulator which theoretically provides an infinite amount of data. However, the reality gap between the simulator and the physical world often leads to failure when working with physical robots. The gap is caused by an inconsistency between physical parameters and by incorrect physical modeling. In this case, we need to model different kinds of wind so as to make it robust. Also the estimation error and delays caused in the vision system should be taken into consideration to make it more close to the real scenario.

During the training, the simulation engine imposes four different kinds of wind conditions: zero wind, constant magnitude wind (-10 m/s to 10 m/s), sudden wind magnitude change (-10 m/s to 10 m/s magnitude change), and a sinusoidal wind (amplitude of 5 m/s and time periods of 10 secs, 20 secs, 30 secs, 40 secs and 50 secs). These four different conditions were applied among different episodes. The idea here is that this makes the agent learn proper control actions for various wind scenarios. This is known as domain randomization and it has been an active area of research in RL [72]. It is observed that headwind affects the forward drift and crosswind affects the sideward drift. Hence crosswinds are applied for roll controller training and headwinds are applied for pitch controller training.

The estimation of relative distances and velocities from the vision system is prone to errors depending on how far the camera is located from the visual cue. Hence state randomization is included during the training. An addition Gaussian zero-mean noise is added randomly to each state estimation whose variance depends on the relative distance from the target point. And on top of this, the delays between each time-step are also randomized. For the vision system, there is a delay of 0.04s to 0.1s between two frames, and that information is encoded during the training.

3.3.4.1 Reward Function

Reward functions are very important in the RL framework. A proper design of the reward function can yield a better convergence rate and good performance in the testing phase. A normalized reward function is opted here, which penalizes every action based on the deviation from the target point as well as the action values and change in successive action values. The reward function is carefully designed to represent the control objective in numbers as shown in Eq. (3.9).

$$\text{Reward} = \begin{cases} -(1/20) \times |a_{diff}| - (1/10) \times |a| & \text{if } |d| \leq 0.1 \\ -2 \times |d| - (1/20) \times |a_{diff}| - (1/10) \times |a| & \text{if } 0.1 \leq |d| \leq 0.4 \\ -1 & \text{if } 0.4 \leq |d| \leq 4 \\ -(T_{\max} - T_{\text{inside}}) & \text{if } |d| > 2 \end{cases} \tag{3.9}$$

d denotes the deviation along the x-axis for the pitch control case and the deviation along the y-axis for the roll control case. a_diff is the difference in current action value and average of past 5 action value and a is the current action value. T_{max} is the maximum episode time and T_{inside} is the time that the UAV stays within 2 meters from the hovering point. Thus, it can accumulate higher rewards (less negative) as it stays closer to the point. In other words, it will get more penalties as it deviates from the point. One episode ends if the deviation is greater than 2 meters and a new episode is started. Thus, the reward written by the time factor, $-(T_{max} - T_{inside})$, is acquired only once in the unsuccessful case. The idea here is that rather than exploring unnecessary states, it is better to give the future cumulative rewards in one step. This helps in maintaining the consistency of episodic rewards. In each episode, the UAV is spawned at a random position near the origin. Apart from the other common RL control strategies, the episode does not end even though the UAV achieves the target position. Instead, it is designed to continue maintaining the position until the end of the episode time, which can allow sufficient time to learn the proper control actions.

Successful training is heavily dependent on the selection of an RL algorithm. One of the modern policy gradient algorithms, TD3 is used for the study. The applied hyper-parameters that control the learning process are given in Table 3.6.

Table 3.6: Selected hyper-parameters for training

Gamma	0.99	policy delay	2
learning rate	0.0001	target policy noise	0.2
buffer size	100000	target noise clip	0.5

3.3.4.2 Convergence

The training convergences of pitch and roll control cases are shown in Figs. 3.10 and 3.11, respectively.

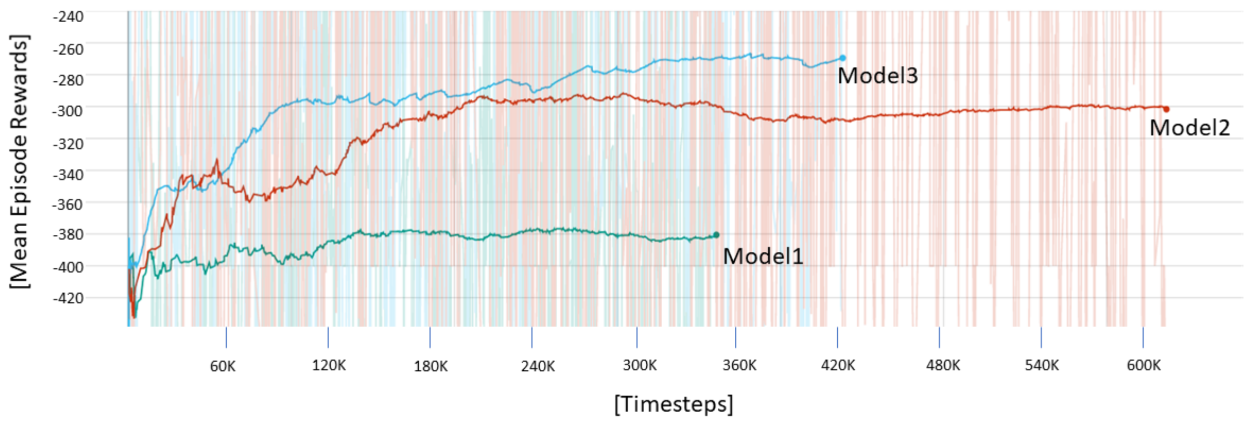


Figure 3.10: Training convergence of different models for the pitch control case [3] [4]

In both the pitch and roll control cases, model 3 converges to the higher accumulated reward value and learns faster than the other models on average. It demonstrates that taking previous positions and velocities as states is helpful, which is designed based on the intuition that neural networks can decode the wind gust information from those states. The obtained control policy is used in the simulations and flight tests.

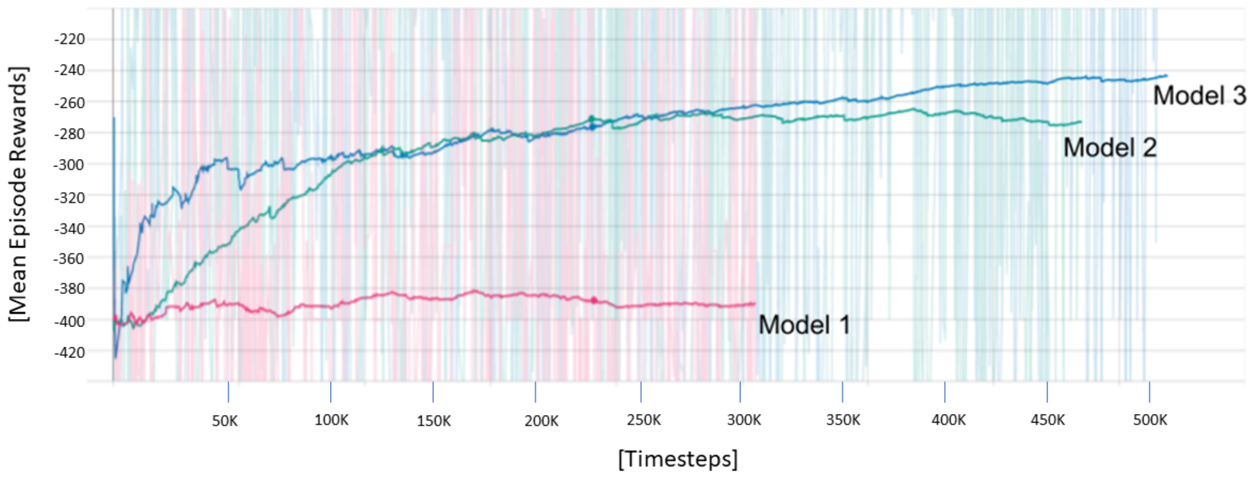


Figure 3.11: Training convergence of different models for the roll control case [3] [4]

The system used for training is LENOVO Legion Y740-15IRH, which is composed of Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz, 2592 Mhz, 6 Cores, and 12 Logical Processors. It features an integrated NVIDIA GeForce GTX 1660 Ti 6GB Graphics and 8GB of LPDDR4 memory with a 128-bit interface. The Ubuntu 18.04 OS with Nvidia driver version 440 and CUDA version 10.2 are used. The same system is used for flight simulations and testing.

3.4 Simulation Results

In order to verify the disturbance rejection capability of RL controller in challenging situations, different wind conditions are simulated in the simulation engine and are compare with a PID controller while the UAV is hovering. The sideward and forward drifts as well as the control inputs are compared here. In the next steps, wind conditions are imposed on the UAV while it is landing on a stationary platform as well as moving platform. The performance of the RL controller is compared with the previously developed nonlinear PID-based feedback control system with the Kalman filter based of the forward and sideward drift throughout the flight till landing. The simulated UAV and Gazebo simulation program is shown in Fig. 3.12.

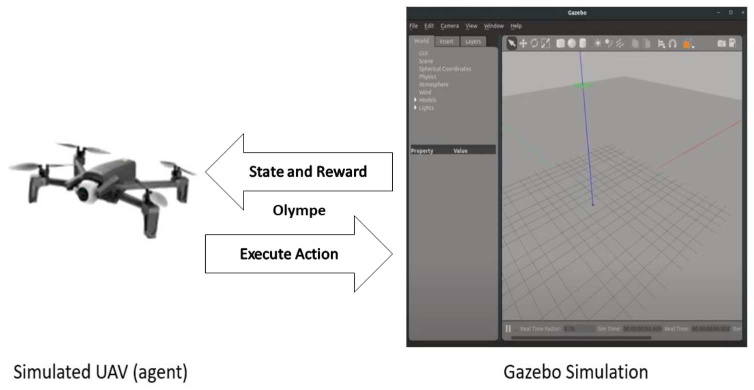
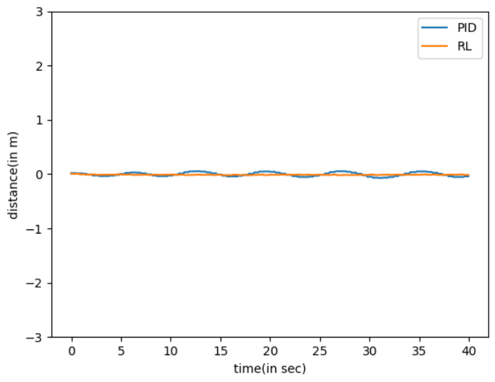


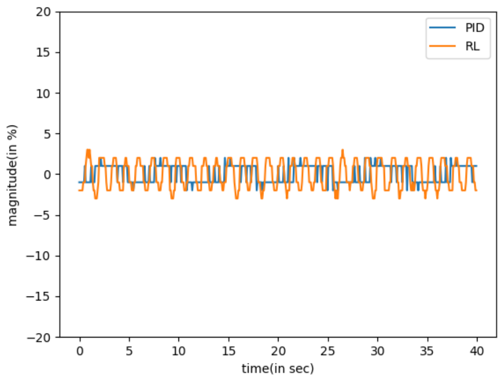
Figure 3.12: Simulated UAV (agent) and Gazebo simulation environment [3] [4]

3.4.1 Case-1: Hovering in absence of wind

The UAV is hovering and there are no wind or any kind of disturbance imposed in this case. The control magnitude and UAV is recorded for 40s of flight. As can be seen from the figures 3.13a and 3.14a, there is almost zero deviation just like the PID. The RL control magnitudes are also similar to PID control magnitudes shown in 3.13b and 3.14b.

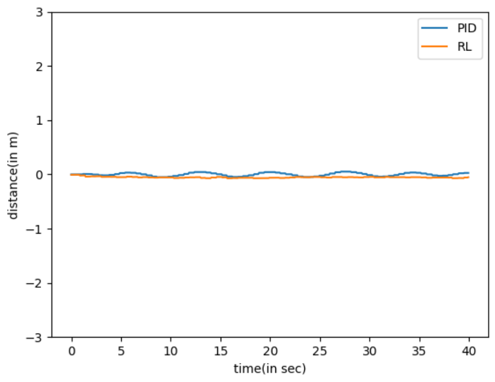


(a) Forward drift

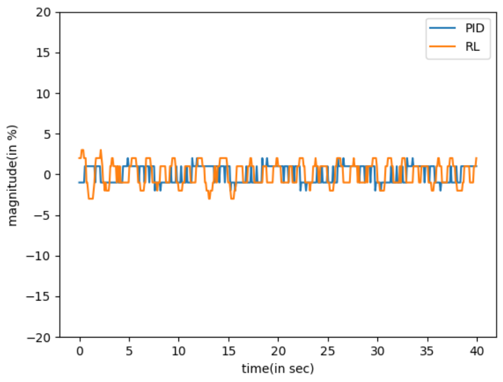


(b) Pitch control magnitude

Figure 3.13: Forward drift and Pitch magnitude for case-1



(a) Forward drift

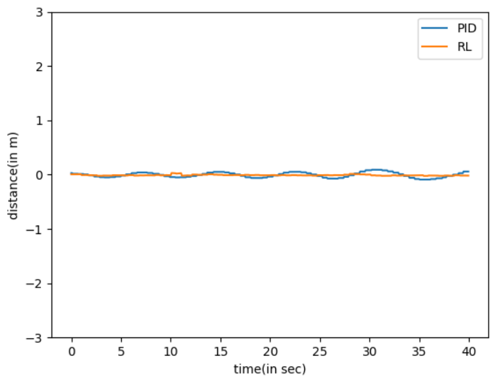


(b) Pitch control magnitude

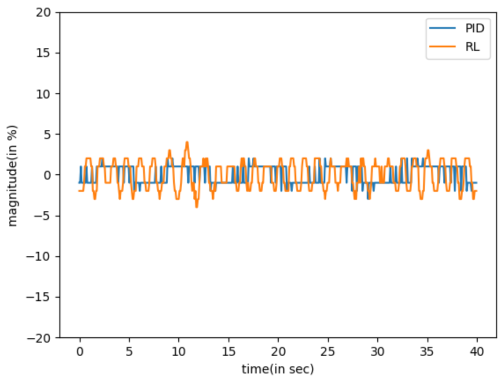
Figure 3.14: Sideward drift and Roll magnitude for case-1

3.4.2 Case-2: Hovering at sudden cross wind gust

In this case, a sudden 5m/s cross wind (90°) with respect to heading of UAV) is applied at 8 second mark and its stays the same for next 16 seconds. After that the wind goes back to zero. The wind function used here is a step function which instantaneously changes from 0 to 5 at 8 seconds and then back to zero at 24 seconds.

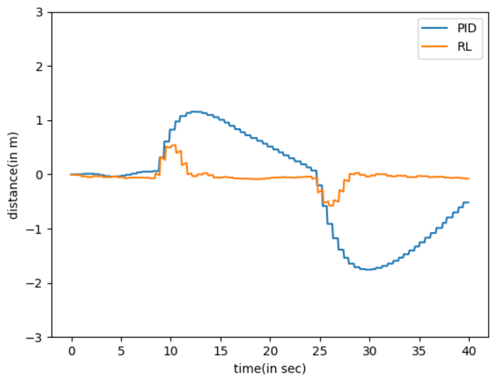


(a) Forward drift

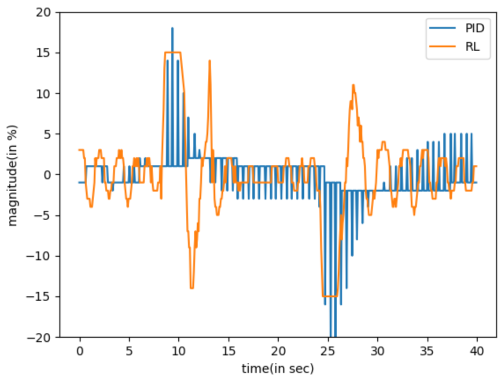


(b) Pitch control magnitude

Figure 3.15: Forward drift and Pitch magnitude for case-2



(a) Forward drift



(b) Pitch control magnitude

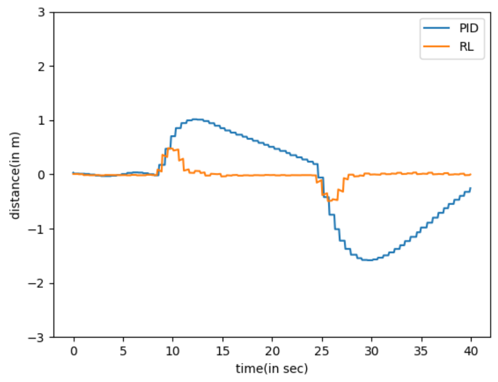
Figure 3.16: Sideward drift and Roll magnitude for case-2

The instantaneous change in wind caused 3 times lesser sideward drift in RL controller case than the PID controller. And also the RL immediately corrected the drift in mere 2 seconds while the PID controller couldn't do it in next 16 seconds. The roll control magnitudes also shoots up when the wind is applied. The forward drift and pitch controller magnitudes are similar to case-1.

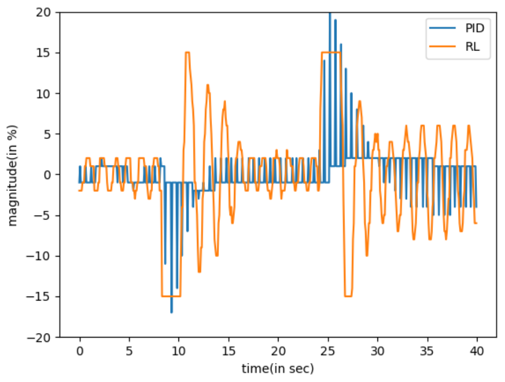
3.4.3 Case-3: Hovering at sudden head wind gust

In this case, a sudden 5m/s head wind (0°) with respect to heading of UAV) is applied at 8 second mark and its stays the same for next 16 seconds. After that the wind goes back to zero. The wind function used here is a step function which instantaneously changes from 0 to 5 at 8 seconds and then back to zero at 24 seconds.

The instantaneous change in wind caused 3 times lesser forward drift in RL controller case than the PID controller. And also the RL immediately corrected the drift in mere 2 seconds while the PID controller couldn't do it in next 16 seconds. The pitch control magnitudes also shoots up when the wind is applied. The sideward drift and roll controller magnitudes are similar to case-1.

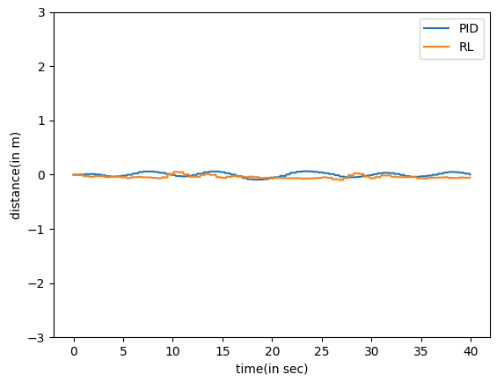


(a) Forward drift

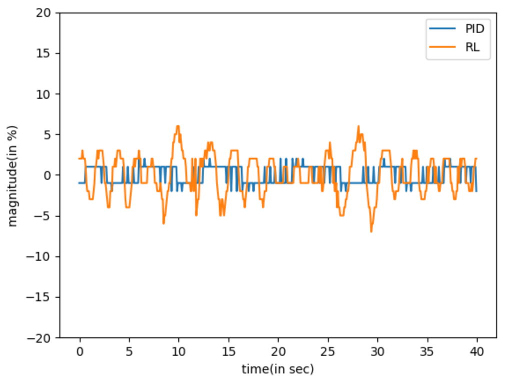


(b) Pitch control magnitude

Figure 3.17: Forward drift and Pitch magnitude for case-3



(a) Forward drift



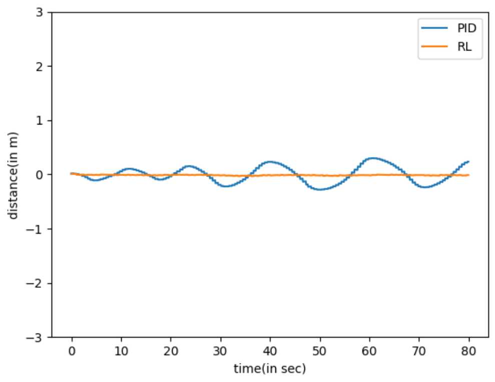
(b) Pitch control magnitude

Figure 3.18: Sideward drift and Roll magnitude for case-3

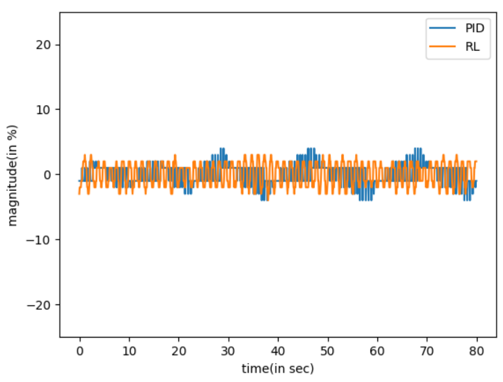
3.4.4 Case-4: Hovering at sinusoidal cross wind

The UAV is hovering the presence of sinusoidal cross wind (90°) which has an amplitude of 5m/s and a time period of 20 seconds. The wind is present throughout the flight. The total flight time is 80 seconds.

The maximum sideward drift in RL controller is less than 20 cm and it is 10 times smaller than the sideward drift happening in the PID controller. The RL roll control magnitude as well as PID

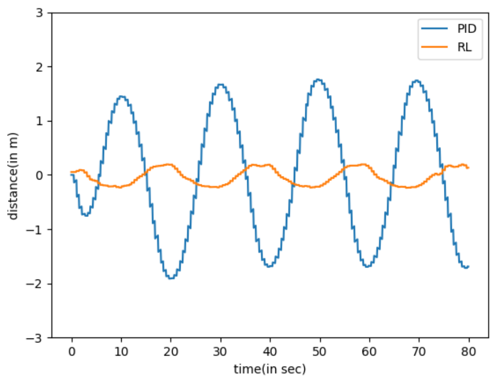


(a) Forward drift

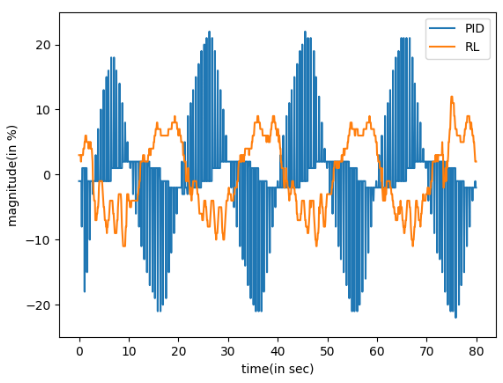


(b) Pitch control magnitude

Figure 3.19: Forward drift and Pitch magnitude for case-4



(a) Forward drift



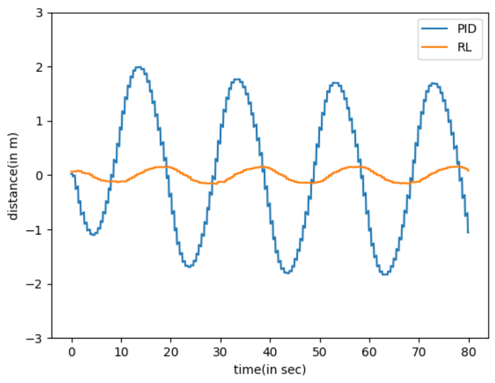
(b) Pitch control magnitude

Figure 3.20: Sideward drift and Roll magnitude for case-4

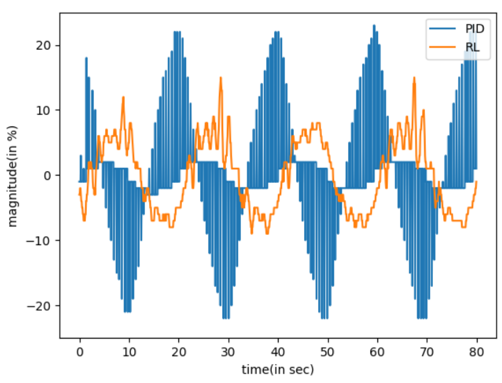
roll control magnitude follows a sinusoidal pattern just like the sinusoidal wind. The sinusoidal pattern in the control magnitude has a time period around 20 seconds which is same for the wind pattern. The forward drift and pitch controller magnitudes are similar to case-1.

3.4.5 Case-5: Hovering at sinusoidal head wind

The UAV is hovering the presence of sinusoidal cross wind (0°) which has an amplitude of 5m/s and a time period of 20 seconds. The wind is present throughout the flight. The total flight time is 80 seconds.

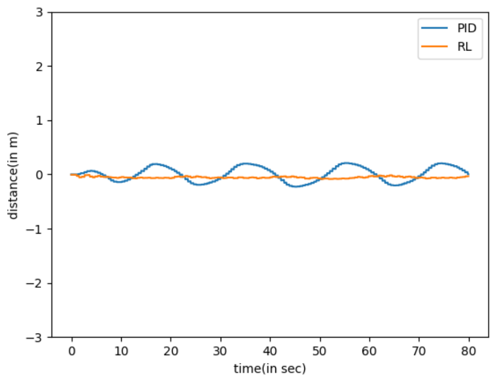


(a) Forward drift

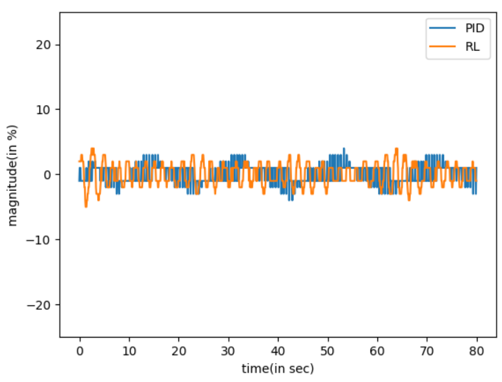


(b) Pitch control magnitude

Figure 3.21: Forward drift and Pitch magnitude for case-5



(a) Forward drift



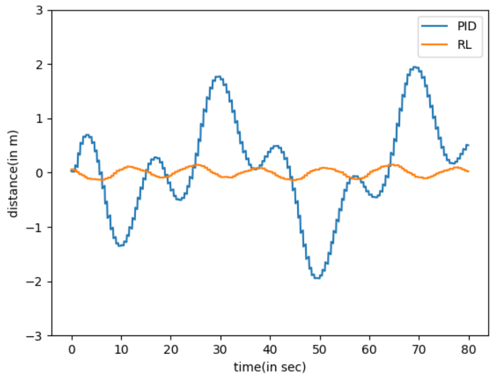
(b) Pitch control magnitude

Figure 3.22: Sideward drift and Roll magnitude for case-5

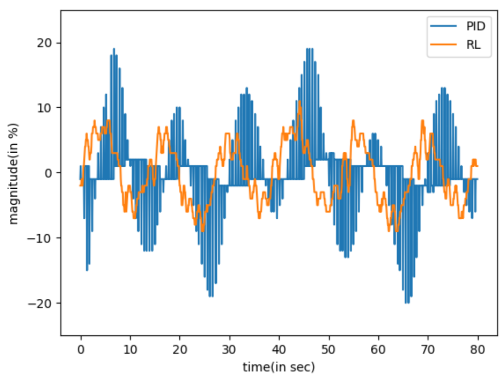
The maximum forward drift in RL controller is less than 20 cm and it is 10 times smaller than the forward drift happening in the PID controller. The RL pitch control magnitude as well as PID pitch control magnitude follows a sinusoidal pattern just like the sinusoidal wind. The sinusoidal pattern in the control magnitude has a time period around 20 seconds which is same for the wind pattern. The forward drift and pitch controller magnitudes are similar to case-1.

3.4.6 Case-6: Hovering at sinusoidal wind in varying direction

The UAV is hovering the presence of sinusoidal wind which has an amplitude of 5m/s and a time period of 20 seconds. The wind changes its direction from 0° to 360° in every 40 seconds. The wind is present throughout the flight. The total flight time is 80 seconds.

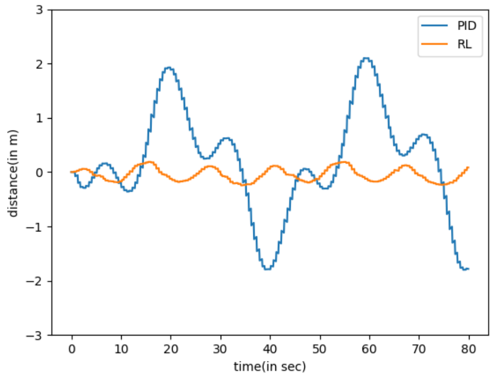


(a) Forward drift

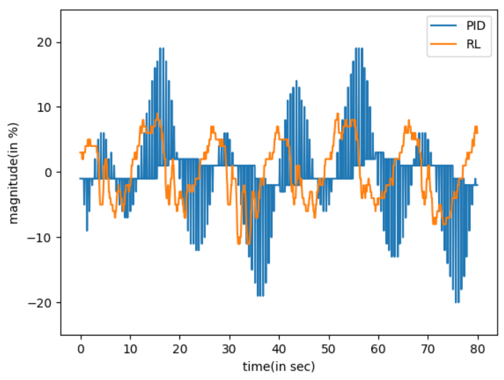


(b) Pitch control magnitude

Figure 3.23: Forward drift and Pitch magnitude for case-6



(a) Forward drift



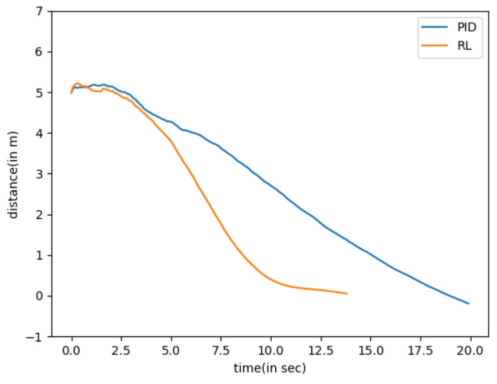
(b) Pitch control magnitude

Figure 3.24: Sideward drift and Roll magnitude for case-6

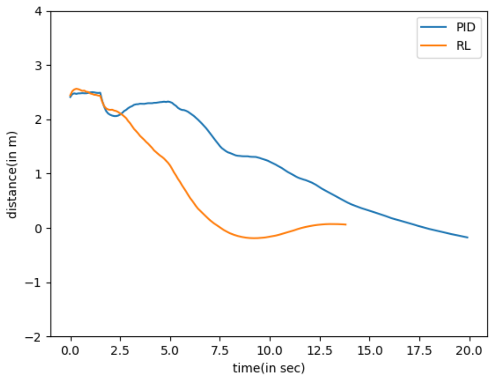
The maximum forward drift and sideward drift in RL controller is less than 20 cm and it is 10 times smaller than the forward drift and sideward drift happening in the PID controller. The RL pitch and roll control magnitude as well as PID pitch and roll control magnitude follows a sinusoidal pattern just like the sinusoidal wind. The sinusoidal pattern in the control magnitude has a time period around 20 seconds which is same for the wind pattern.

3.4.7 Case-7: Landing on a stationary platform

The developed vision system is used for the estimation of state in this case. The non-linear controller is configured for the landing on a stationary platform in the absence of wind. The non-linear controller is compared with RL controller in absence and presence of wind.



(a) Forward drift

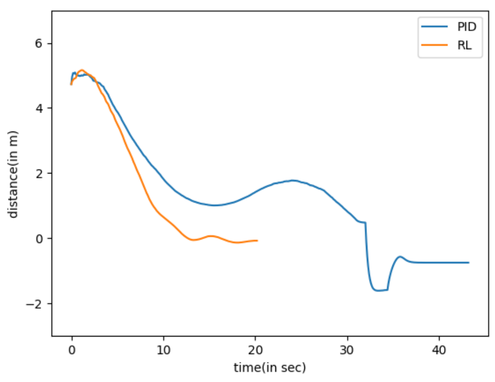


(b) Sideward drift

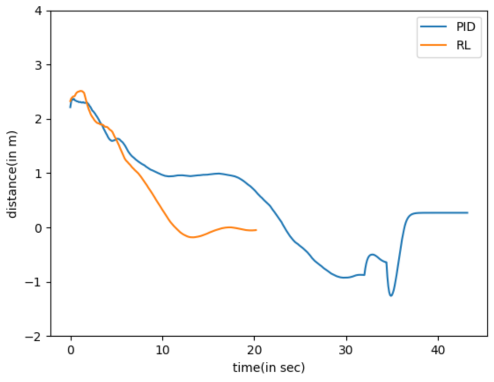
Figure 3.25: Forward drift and Sideward drift for landing on stationary platform in absence of wind

From Fig.3.25a and Fig. 3.25bshows RL controller makes landing much faster than the PID controller. Fig. 3.26a and Fig. 3.26b shows PID was not able to make the landing while RL controller easily makes the landing.

Fig. 3.27a and Fig. 3.27b shows that RL controller comfortably makes landing in all kinds of wind scenarios

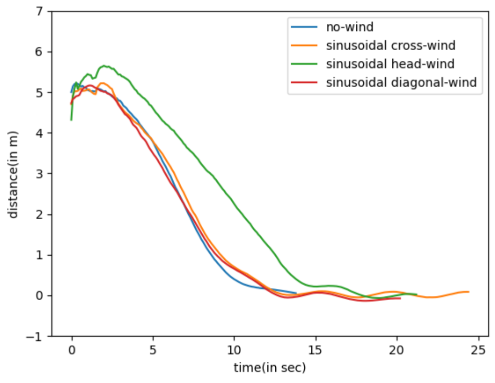


(a) Forward drift

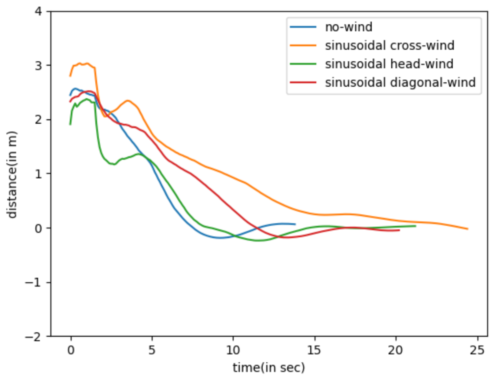


(b) Sideward drift

Figure 3.26: Forward drift and Sideward drift for landing on stationary platform in presence of wind



(a) Forward drift



(b) Sideward drift

Figure 3.27: Forward drift and Sideward drift for landing on stationary platform in different wind scenarios

4. FLIGHT TESTING

[1] Extensive flight tests are carried out to demonstrate the novel autonomous ship landing methodology in practice. The specific objectives are to prove the safety of a vertical landing, regardless of platform movement, and to demonstrate the tracking ability and precision of the landing. This chapter divides the flight testing into three sections based on the objectives and each section includes detailed results obtained from the implementation of gain-scheduled PID control system, nonlinear control system, and deep reinforcement learning-based control system. Throughout the flight tests, the quad rotor UAV that has a gimballed camera has been used. However, the visual cue and the landing platform have progressed during the course of the project. In the earlier phase of flight testing, a checkerboard pattern visual cue is used along with a landing platform that has a fixed landing pad. Later, the horizon bar visual cue is structured to the ship platform that has a motion deck. Regardless of visual cues, platforms and control systems, autonomous flight shares the same process as shown in Fig. 4.1. It has a built-in (on-board) internal loop autopilot that controls the speed of each propeller to achieve the commanded inputs generated by the outer loop's vision-based control system(off-board).

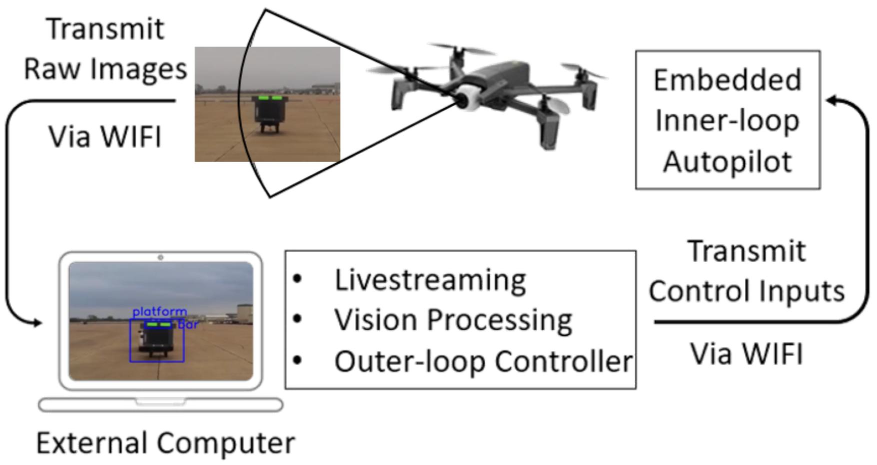


Figure 4.1: Process of autonomous flight system[1] [2] [3]

[1]The chapter has been adapted or reprinted from recent publications [1] [2] [3] [4].

The UAV is controlled by a Python script that runs on an external base station computer. The external computer communicates with the UAV through the WIFI connection. The UAV transmits raw images captured by the onboard camera to the computer in real-time. Then, the computer processes the images to provide perceived visual information. The image resolution used is 1280 x 720p and this affects the effective range for detection and estimation. Once the perceived visual information is fed into the feedback controller, it yields the corresponding command inputs which are roll, pitch, throttle (heave), and yaw. The commands are sent back to the UAV and then the embedded inner-loop autopilot controls the rotating speed of each propeller to achieve the commanded inputs.

The system used for processing is LENOVO Legion Y740-15IRH, which is composed of Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz, 6 Cores, and 12 Logical Processors. It features an integrated NVIDIA GeForce GTX 1660 Ti 6GB Graphics and 8GB of LPDDR4 memory with a 128-bit interface. The Ubuntu 18.04 with Nvidia driver version 440 and CUDA version 10.2 are used. The WIFI communication is established by an external TP-LINK TL-WN722N Wireless N150 High Gain USB Adapter.

4.1 Experimental Setup

To simulate the 6 DOF motions of the ship deck, a Servos and Simulation Inc Generic Motion System (model 710-6-500-220) with a 1.22 x 1.22 meters (4 × 4 feet) landing deck as shown in Fig. 4.2. The ranges of roll, pitch, and yaw are ± 13°, ± 15°, and ± 16°, respectively. The ranges of surge, sway, and heave are ± 1.02 meters, ± 1.02 meters, and ± 0.64 meters, respectively.

A sub-scale ship platform is constructed including the horizon bar and motion deck as shown in Fig. 4.3. The width, height, and length of the ship platform are 5 ft, 5 ft, and 10 ft, respectively. The horizon bar always indicates a perfect horizon, and the motion deck has its own 6 DOF motions in addition to the forward translational motion, which is similar to what would be experienced on a real ship. This setup has been used for the nonlinear control system and RL-based control system.

In addition, as shown in Fig. 4.4, a drum fan is installed for generating the wind gust to demonstrate the disturbance rejection capability of the developed RL control strategy. The fan

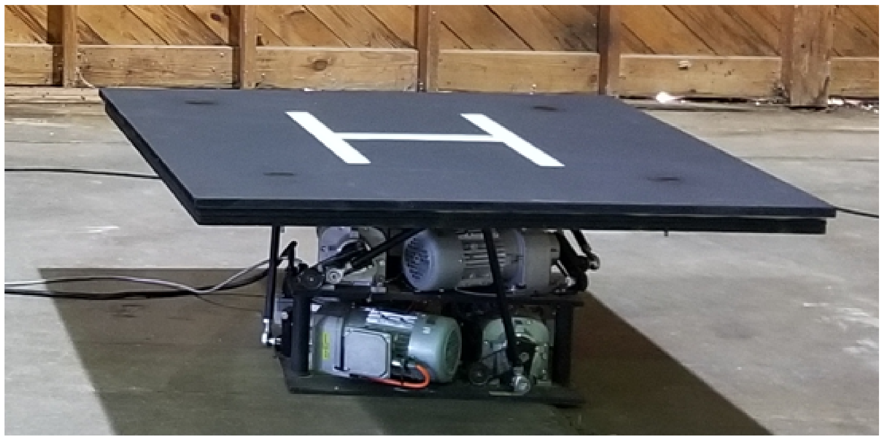


Figure 4.2: Servos and Simulation Inc. 6 DOF motion system with 1.22 x 1.22 meters(4 × 4 feet) Landing Deck [2] [3]

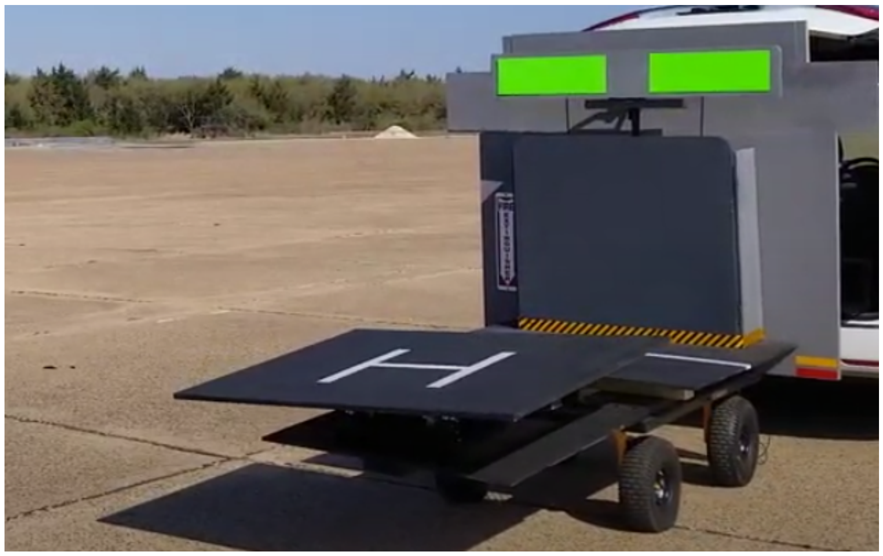
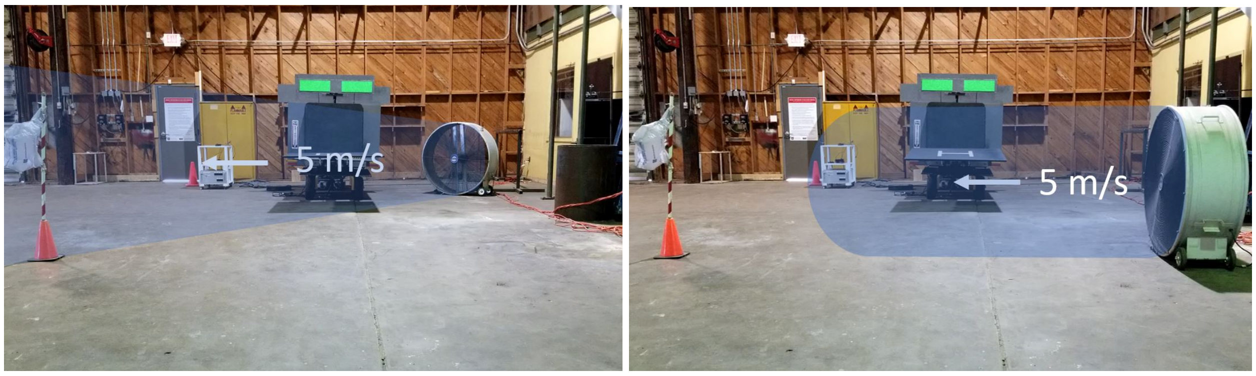


Figure 4.3: Constructed ship platform with horizon bar and motion deck [2] [3]

could generate a wind gust with a maximum speed of 5 m/s in any direction relative to the UAV heading and flight path.

4.2 Ship Motions and Vertical Landing Safety

In order to demonstrate that it is safe to land vertically without matching the UAV attitude dynamics to platform motion, landing tests are conducted while the 6 DOF platform is simulating



(a) Diagonal wind testing setup

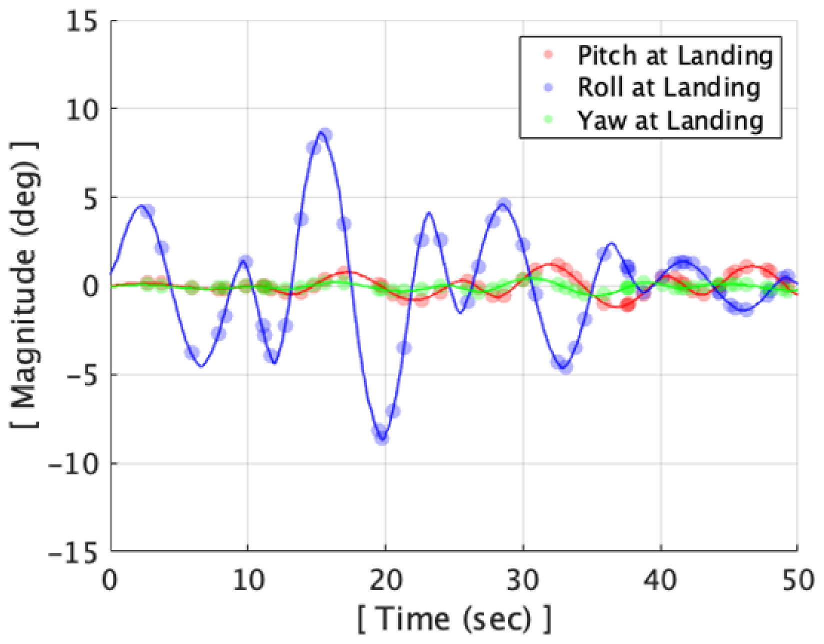
(b) Crosswind testing setup

Figure 4.4: Disturbance rejection experimental setup [3] [4]

two challenging ship motions. The first prescribed motion is the Oliver Hazard Perry Class frigate at the sea state of 6 and a wave direction of 60° introduced by Sanchez-Lopez, Jose Luis, et al. [5]. The frigate is 136 meters long and 14 meters wide and has a single flight deck. Sea state 6 is defined by the World Meteorological Organization (WMO) as a very rough condition that has a wave height of 4 to 6 meters [73]. While the platform is undergoing this complicated motion, vertical landings are executed at random time instances as shown in Fig. 4.5.

Solid red, blue, and green lines denote continuous angular (pitch, roll, and yaw) and linear (surge, sway and heave) motions of the 6 DOF platform. 50 landing tests are conducted in total and the motion of the platform at the time of each successful landing are depicted as dots. The randomly distributed landing timings demonstrate the vertical landing is safe at any moment of the Oliver Hazard Perry Class FFG frigate ship motions under the given conditions.

The second prescribed motion is the Navy helicopter ship landing limits defined by NATOPS. In the case of the FFG 7 Class Ships which the Oliver Hazard Perry frigate belongs to, the ship motion limits for landing are set as ± 8° of roll and ± 3° of pitch. Even though the limits are defined by the maximum roll and pitch magnitudes, the frequency of the motion is also a crucial factor for the motions. According to the report for a similar size ship (length: 152.4 m, width: 15.2 m) motions conducted by the Sandia National Laboratory [30], the roll period is 10.1 seconds and the pitch period is 6.5 seconds as shown in Table 4.1.



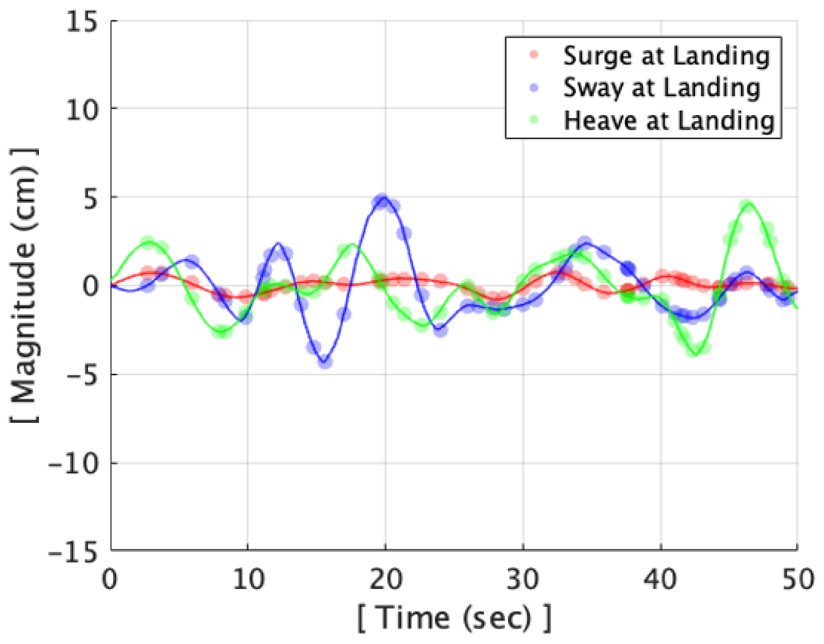


Figure 4.5: Oliver Hazard Perry class frigate motion and vertical landing moments [2] [3]

The maximum pitch and roll magnitude with the reported periods are applied to the platform and 50 vertical landings are conducted at random moments as shown in Fig. 4.6.

Solid red and blue lines denote continuous pitch and roll motions of the 6 DOF platform. The dots are the motions at the time of successful landings. The randomly distributed landing timings

Table 4.1: Typical ship characteristics by Sandia National Lab report

Ship Type	Length (m)	Width (m)	Roll Period (secs)	Pitch Period (secs)
Destroyer	152.4	15.2	10.1	6.5
Aircraft Carrier	304.8	38.1	15.8	8.8

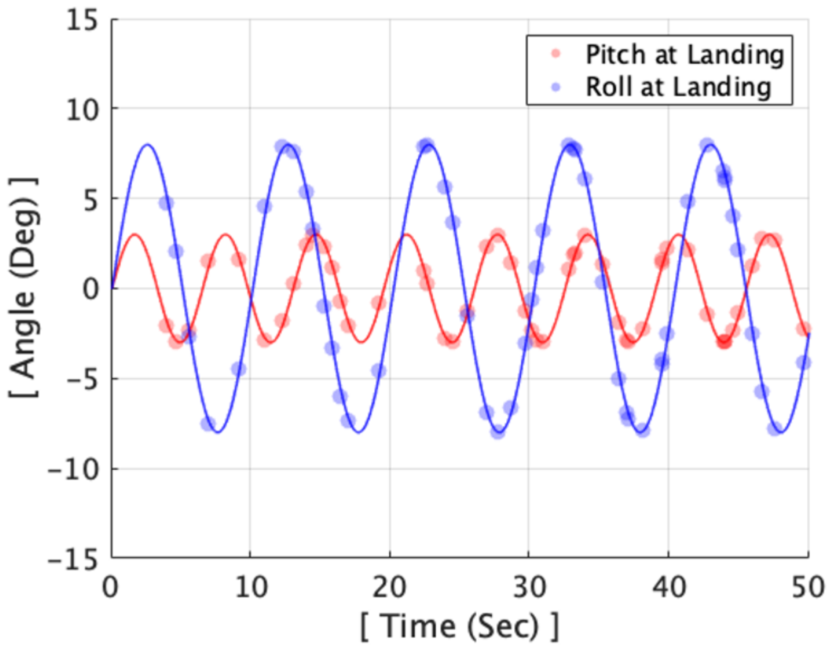


Figure 4.6: Motions of helicopter ship landing limits and vertical landing moments [2] [3]

demonstrate the vertical landings are safe independent of the motion as long as it is within the operational limits.

4.3 Tracking Capability and Landing Accuracy

The tracking capability and landing accuracy of the developed vision-based autonomous flight control system are verified in challenging situations such as random initial positions, maximum distance of 250 meters, realistic ship motions, communication latency, sensor noise, low visibility, and windy conditions. In this fully autonomous vision-based system, GPS and a magnetometer are used only to log positions and heading angles and not for control. During flight testing, the landing pad is mimicking the ship deck motions that are introduced in the previous section.

First, flight tests are conducted to verify the maximum range and smooth transition between flight modes. The trajectories are logged as shown in Fig. 4.7.

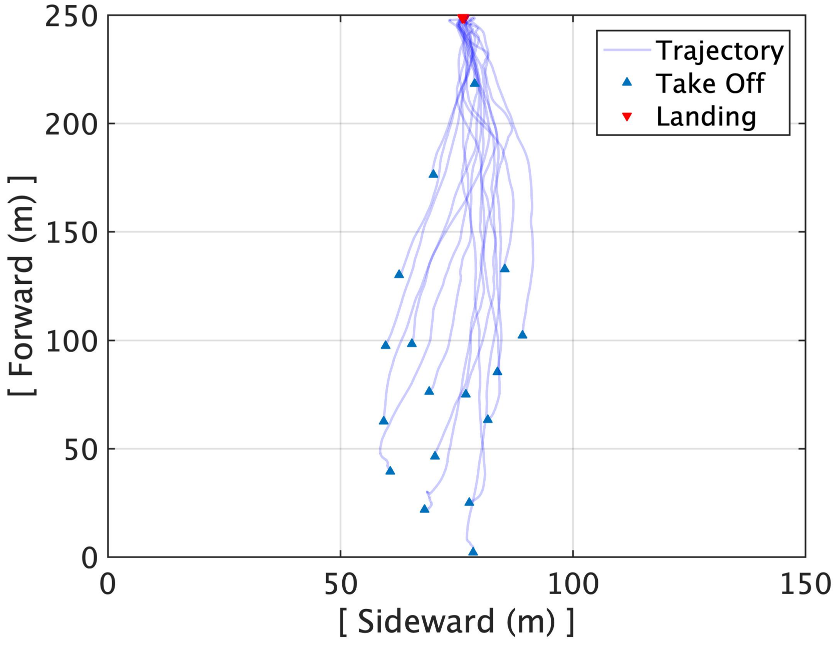


Figure 4.7: Trajectories of long-range tracking [2] [3]

The initial positions are widely distributed and the maximum distance between the UAV and ship platform is approximately 250 meters. During flights, the flight modes are switched from the ship platform/bar tracking to the corners tracking depending on the distance. The results demonstrate stable long-range tracking capability in the presence of wind up to 3 m/s. The time history of control inputs for a representative case is shown in Fig. 4.8.

It shows the control inputs, distances, and heading angle after take-off until the execution of vertical landing. Each line denotes pitch, roll, heave, and yaw control inputs described as percentages that range from -100 to 100 and generated based on forward, sideward, vertical relative distance, and relative heading angle, respectively. Zero control input means neutral control input that maintains current UAV pose such as pitch, roll, altitude, and heading. While the UAV approaches the ship platform from 250 meters away, the vision-based flight control system effec-

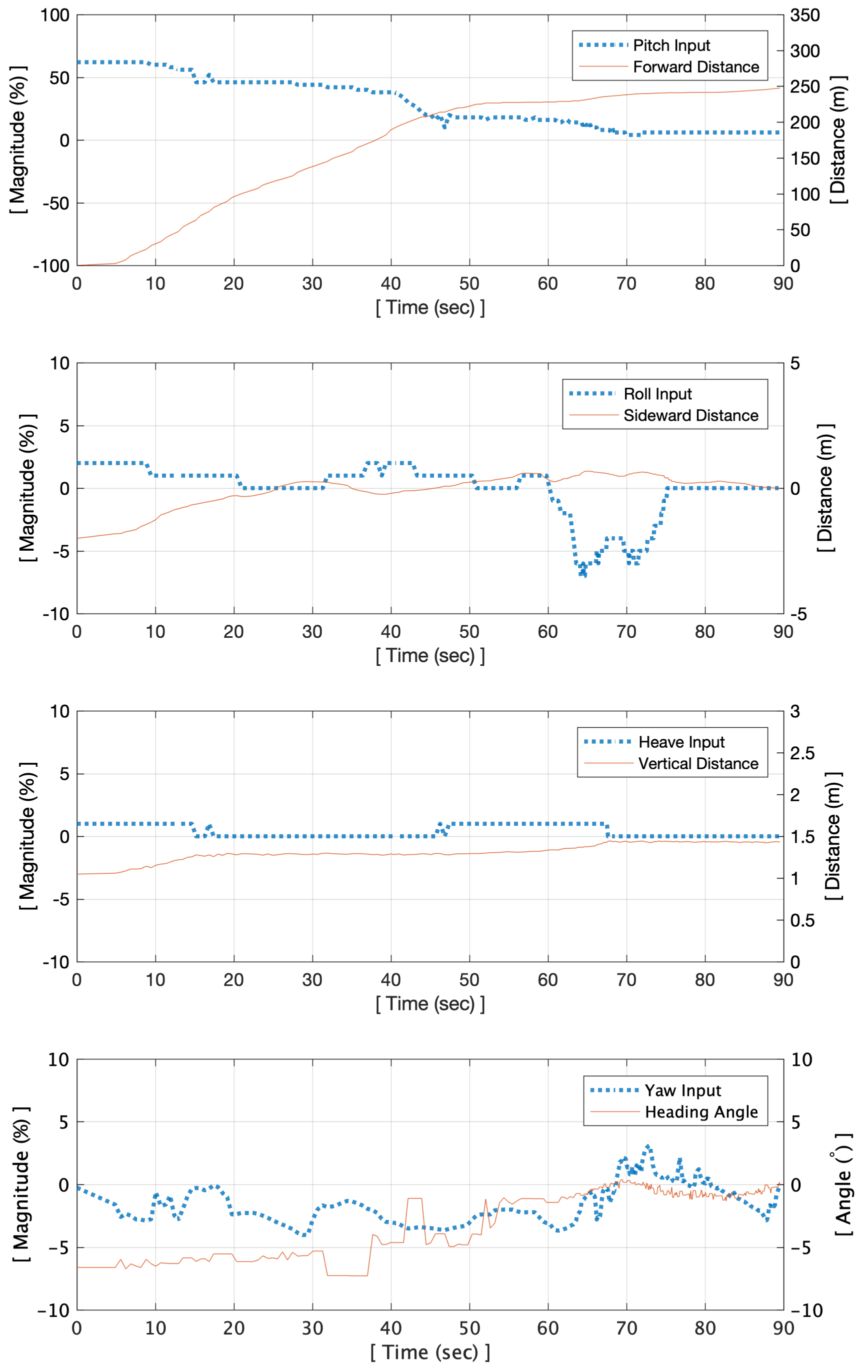


Figure 4.8: Control inputs, distances, and heading angle during long-range tracking in time [2] [3]

tively flies the UAV by smoothly switching between the flight modes from the machine learning object tracking for ship platform and horizon bar to the rectangle corner points tracking, depending on the relative distance.

Second, flight tests are conducted to verify the robust tracking while the ship platform is moving in different courses with its own 6 DOF motions. The three representative trajectories are shown in Fig. 4.9.

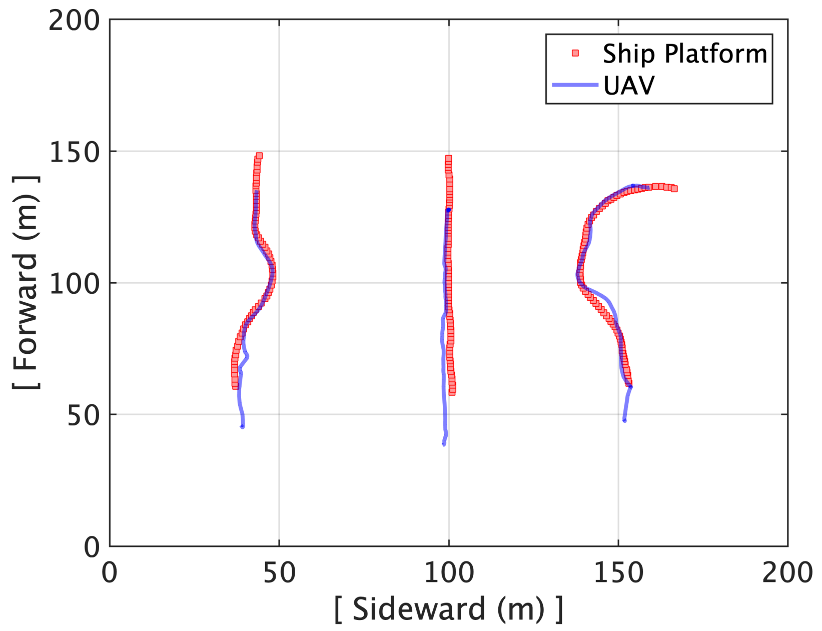


Figure 4.9: Trajectories of moving ship tracking [2] [3]

Red squares denote the trajectory of the ship platform and blue lines denote the trajectory of the UAV. It shows robust tracking while the ship platform is moving in straight, S-pattern, and 90° turn. The control inputs for S-pattern moving platform tracking are shown in Fig. 4.10.

It shows the control inputs, distances, and heading angle after take-off until execution of vertical landing while the UAV is tracking the ship platform moving in an S-pattern. The pitch control input is generated to approach the ship platform that varies its speed from 0 to 4.5 m/s (10 mph). Since the ship platform changes the course abruptly up to 130°, the relative sideward distance and heading also change to a great extent. Accordingly, the corresponding roll and yaw control inputs change aggressively to achieve zero relative sideward distance and heading. Vertical landings are conducted once the UAV flies into the 0.35 x 0.35 meters landing threshold while the ship platform is in motion. The different flight test cases are shown in the video [74].

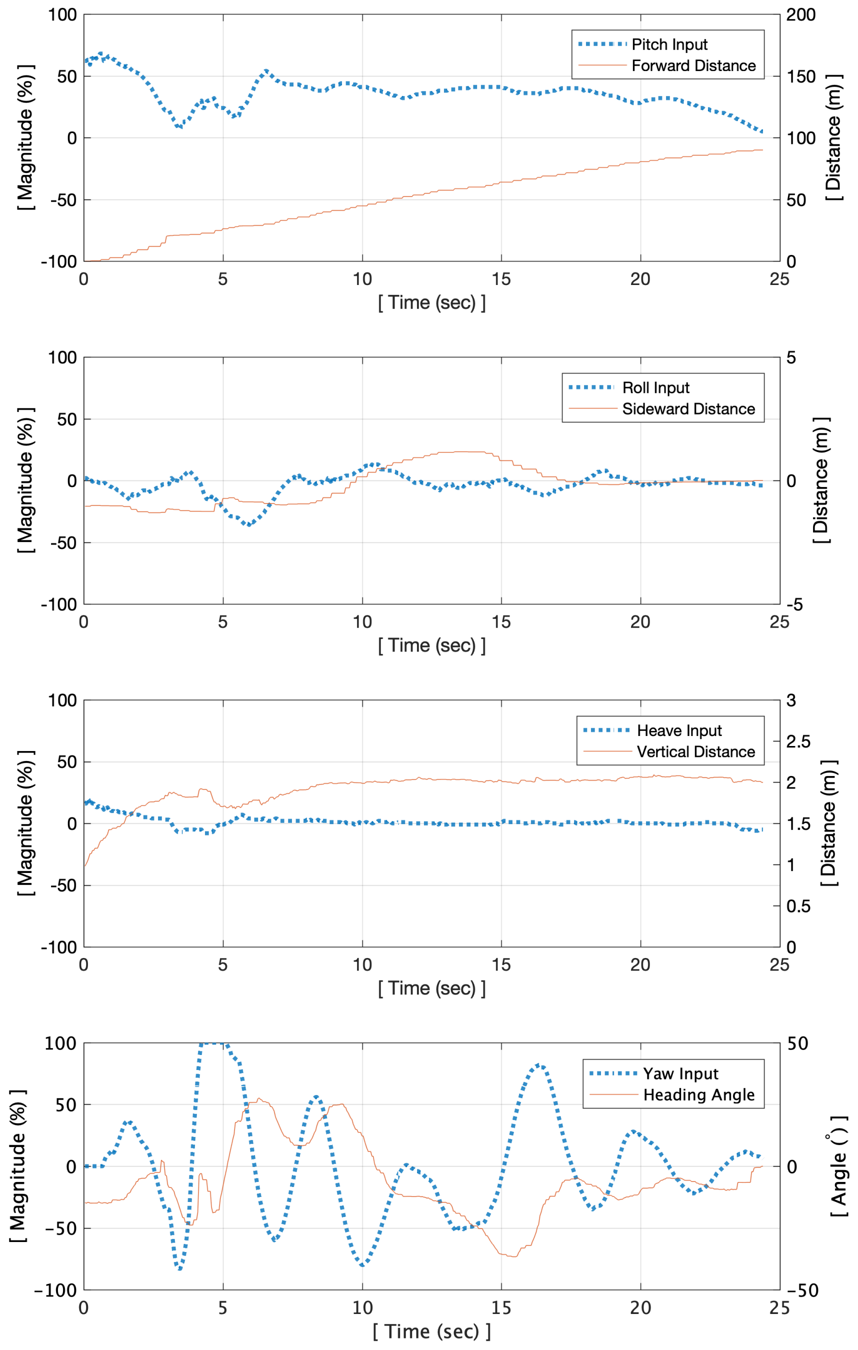


Figure 4.10: Control inputs, distances, and heading angle of moving ship tracking in time [2] [3]

4.4 Disturbance rejection for diagonal wind gust case

Flight tests are conducted to demonstrate the disturbance rejection capability of the RL control strategy when the wind gust is suddenly imposed. During flight testing, the fan generates a

5 m/s wind gust at an angle of 45 degrees with respect to UAV heading while the UAV tries to autonomously approach and land on the ship platform. The performances of the developed nonlinear PID-based feedback control system with the Kalman filter and the deep RL control strategy are compared based on the sideward drift. The control inputs and the sideward drift using the RL control strategy are also examined. The sideward drift while using the nonlinear PID control and the RL control for the diagonal wind gust case is shown in Fig. 4.11.

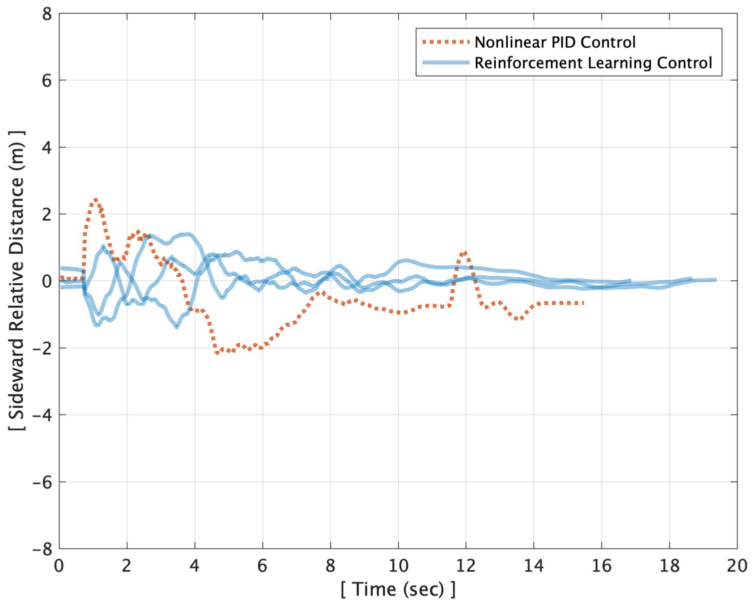
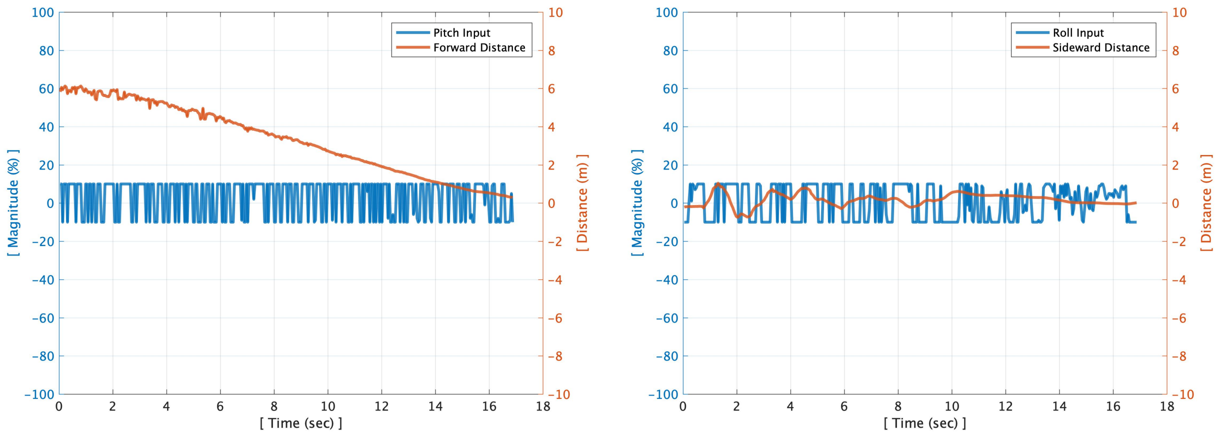


Figure 4.11: Sideward drift for the 5 m/s diagonal wind gust case [3] [4]

The flight tests using the nonlinear PID control experience more than 2 meters of sideward drift and fail to land on the landing deck. However, the RL control strategy demonstrates less than 1.5 meters of sideward drift and successfully lands on the deck repeatedly.

The relative distances and control inputs of successful RL control strategy are shown in Fig. 4.12. RL controller demonstrates robust tracking by rejecting the diagonal wind gust effectively from about 6 meters distance from the landing deck. Also, it keeps the sideward drift below 1 meter during approach and landing by actively commanding the roll control inputs of high magnitude and frequency. The trained control policy yields control inputs by deep neural networks that consider the current and 5 previous positions and velocities. Therefore, for an RL controller, it is not possible to explicitly correlate the relationship between the states and actions. However, the flight tests clearly demonstrated that the RL control effectively rejected the disturbance while approaching and landing.



(a) Forward relative distances and pitch control inputs (b) Sideward relative distances and roll control inputs

Figure 4.12: Disturbance rejection of reinforcement learning control for the 5 m/s diagonal wind gust case [4] [3]

4.5 Disturbance rejection for crosswind gust case

Flight tests are also conducted to demonstrate the disturbance rejection capability of the developed RL control strategy in a more challenging condition. During these flight tests, the fan is installed perpendicular to the approach course so that it experiences a sudden 5 m/s crosswind gust while approaching and landing on the ship platform. The performance of the deep RL-based control strategy is compared with the nonlinear PID and Kalman filter-based feedback control system in terms of the sideward drift and control inputs. The sideward drift of the nonlinear PID control and RL control in the sudden crosswind case is shown in Fig. 4.13.

This sudden crosswind affects the sideward drift in both cases. The nonlinear PID control fails

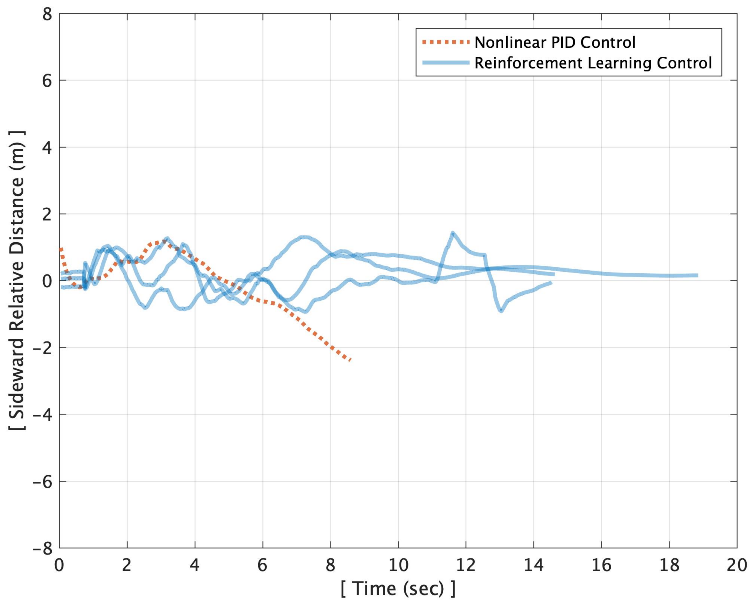
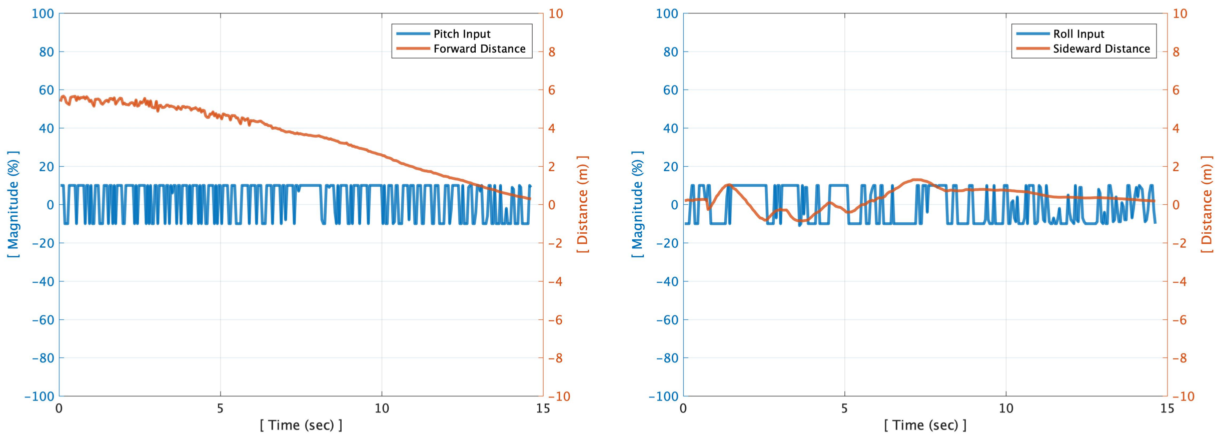


Figure 4.13: Sideward drift for the 5 m/s crosswind gust case [3] [4]

to reject the disturbance quickly enough, and therefore, the UAV loses the visual cue at approximately 8.5 seconds. On the other hand, the RL control keeps the sideward drift below 1.5 meters and successfully lands on the deck repeatedly.

The relative distances and control inputs of the RL control strategy are shown in Fig. 4.14. The RL controller demonstrates robust tracking by rejecting the crosswind gust effectively at 6 meters distance from the landing deck. It has a limited effect on forward drift but a distinguishable effect on sideward drift due to the direction of the wind. The pitch control inputs are mainly commanded to move forward in such a wind condition and the roll control inputs are commanded to minimized the sideward drift caused by the wind gust. It kept the sideward drift below 1.3 meters during the approach and landing phases.



(a) Forward relative distances and pitch control inputs (b) Sideward relative distances and roll control inputs

Figure 4.14: Disturbance rejection of reinforcement learning control the 5 m/s crosswind gust case [3] [4]

4.6 Landing Accuracy

In the cases of landings on the sub-scale ship platform, the safe landing boundary is set by a 35 x 35 centimeters square from the deck center considering the UAV size, landing deck size, and safety margin. Therefore, once the UAV reaches the landing boundary the controller commands vertical landing while the deck is in motion. The final landing points are collected from the multiple flight tests of tracking a ship in long distance with random initial positions, different initial heading angles, and different weather conditions. As shown in Fig. 4.15, they are distributed randomly within the set landing boundary.

The final landing points are also collected from the multiple flight tests of tracking a ship that is dynamically moving in forward, S-pattern, and $90°$ turn. They are distributed within the set landing threshold as shown in Fig. 4.16.

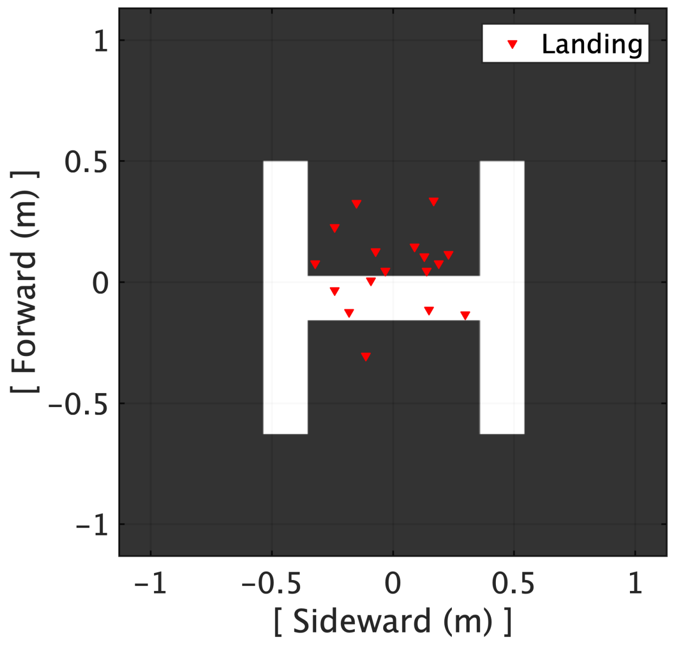


Figure 4.15: Landing points on pad of long-Range tracking [2] [3]

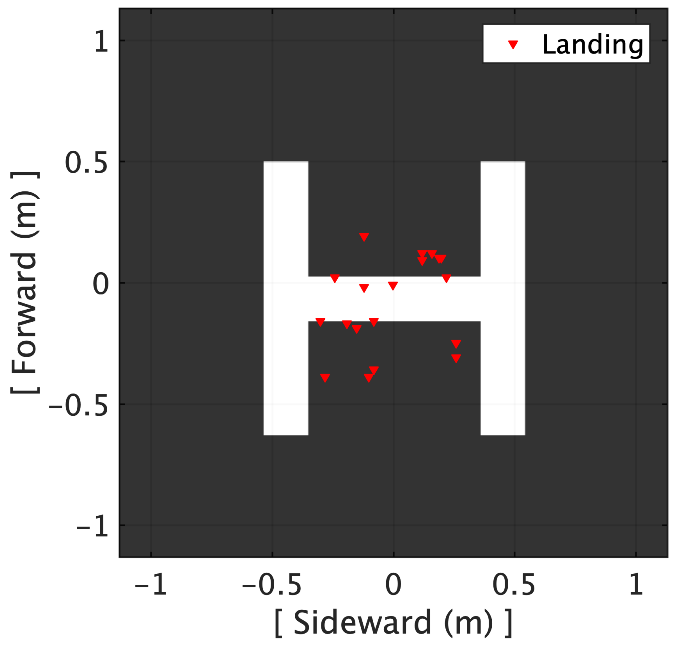


Figure 4.16: Landing points on pad of moving ship platform [2] [3]

The final landing points are also collected from the multiple flight tests of tracking a ship in the presence of wind gust where reinforcement learning control strategy is implemented. While approaching and landing, 5 m/s diagonal wind ($45°$) and cross-wind ($90°$) are suddenly imposed by a drum fan as shown in Fig. 4.4 of the experimental setup section. They are distributed within the set landing threshold as shown in Fig. 54.17.

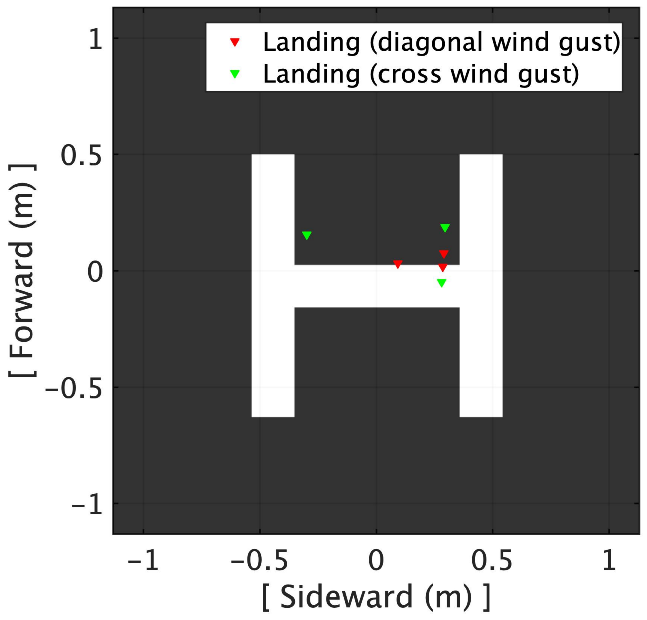


Figure 4.17: Landing points on the deck for the different wind gust flight test cases [3] [4]

5. SUMMARY AND CONCLUSIONS

[1] The goal of this study is to improve the tracking and disturbance rejection capability of a VTOL UAV while attempting to land on a ship deck by closely following the Navy helicopter ship approach and landing procedure. The automation has been achieved by using a single onboard camera without using GPS. The developed autonomous ship landing system consists of a hybrid machine vision system and a robust control system. The machine vision system is developed while taking advantage of state-of-the-art machine learning for long range tracking and classical computer vision techniques for precise estimation and landing. The control system is developed to generate situation-adaptive control inputs by introducing the idea of nonlinear gain variation and a probabilistic approach to limit the impact of incorrect estimations and later made robust to different kinds of disturbances by introduction of deep reinforcement learning strategy.

Extensive simulations and multiple flight tests were systematically conducted to verify the tracking and disturbance rejection capability as well as the landing accuracy. The successfully trained control policy was directly implemented on a quadrotor UAV and flight tests were conducted. UAVs land vertically regardless of ship movement, much like a Navy helicopter lands on a ship. Over 100 landing tests are successfully performed on a mobile deck that mimics the realistic and challenging movement of a ship with 6 degrees of freedom. The machine learning object detector begins identifying the 1.8 x 1.8 meters (6 x 6 feet) ship platform from 0.25 kilometers away. The classical computer vision techniques allows precise landing with sub-centimeter accuracy. The unique nonlinear control system demonstrates robust tracking capability during a wide range of realistic scenarios such as random initial positions, complicated ship motions, communication latency and sensor noise. The developed RL control demonstrated superior disturbance rejection capability compared to a nonlinear PID control system. The RL control strategy was made robust by introducing different wind scenarios and modelling the sensor noises and latency issues while training.

[1]The chapter has been adapted or reprinted from recent publications [1] [2] [3] [4].

Some of the key takeaways from this study are enumerated below.

1. The proposed VTOL UAV vertical landing approach, which is agnostic to the ship deck motions, was verified as a safe way of landing. This was demonstrated through multiple flight tests using a sub-scale landing platform mimicking challenging ship motions derived from the NATOPS helicopter ship landing operational limits and Sandia National Laboratory report (roll: ± 8, pitch: ± 3, roll period: 6.5 secs, pitch period: 10.1 secs).

2. The long-range vision system developed based on the state-of-the-art machine learning algorithm YOLOv3 demonstrated a 10 times greater detection range than the classical computer vision systems. The control system successfully utilized the detected object position and relative size as states for long-range tracking.

3. The biggest challenge to implement the machine learning based object detection on the real-time autonomous flight was the time delay. To cope with the time delay issue, a long-range controller was constructed that responded less sensitively to errors around the setpoint and aggressively to large errors, using an exponential variation of feedback gain with error. This approach enabled the UAV to stay in the appropriate flight course while approaching the ship platform.

4. The developed close-range vision system that combined the classical computer vision techniques and screening algorithms guaranteed fast and reliable detection of the visual cue and demonstrated precise relative position and orientation estimation. The update rate was approximately 15 times faster than the machine learning vision system and therefore, the control system was able to control the UAV more precisely using faster feedback.

5. Even after going through the configured Kalman estimator, large/false estimation error can still occur from time to time. To prevent responding to such non-physical estimations, the probabilistic nonlinear controller was developed. It probabilistically perceives if the estimation is physically possible or not, based on the normal distribution curve and known UAV

characteristics. Multiplying the estimation value by its probability can effectively reject responding to false estimations. By this approach, the controller never generated abrupt large control inputs even when the vision system provided inaccurate estimations. This greatly improved the robustness of tracking.

6. The roll and pitch control policies are individually trained by the RL algorithm. Using this approach, the control action taken by the deep neural networks was set as only one action (pitch or roll), which significantly reduced the training time and converged to higher a cumulative reward at a faster rate. This was only possible because roll and pitch control are uncoupled for a quad-rotor UAV.

7. The optimal set of states were identified by comparisons of different models. Out of the three models analyzed, the model which used the history of position and velocity as states has the best mean episodic reward convergence.

8. The normalized negative reward function that yields a value between -1 and 0 was finally derived after multiple attempts with different reward functions. Also, it was designed to receive a high negative reward only if the UAV drifts out of the boundary that is 2 meters away from the setpoint. It was learned that even a high reward value for successful action and a wide range of rewards could overestimate the actions taken and thus the training could fail. The successful training was heavily dependent on the reward function design.

9. To learn the proper actions by the RL algorithm, various kinds of wind conditions were imposed as the training episodes continued. The sensor noises and latency issues are also modelled whole training. This was done as a part of domain randomization. The final trained control policy obtained robust flight capability in challenging conditions.

10. In comparison with the nonlinear PID control system, the RL control demonstrated superior disturbance rejection capability, which reduced lateral drift by 100% at a 10 times faster rate. This is a crucial improvement that can ensure a safe approach and landing at the proximity

of the ship where the airflow is highly turbulent and unpredictable.

11. The landing accuracy depends on the pre-defined landing threshold. The accuracy can be increased by setting a smaller threshold. Considering the UAV and landing pad size, the appropriate landing threshold of 0.35 x 0.35 meters was set and the UAV successfully landed within this area every time during the 100+ flight experiments. The precise landing capability of the RL control strategy in 5 m/s of sudden wind gusts was also verified through the flight tests.

The results demonstrate conclusively the feasibility of this novel autonomous approach and landing strategy for VTOL UAVs, which is inspired by the Navy helicopter ship landing. This is a significant accomplishment since there are no known efforts in the literature, which focused on automating the real helicopter ship landing procedure. The results also demonstrate the capability of the deep RL control strategy for autonomous ship approach and landing for VTOL UAVs in the presence of disturbances such as wind gusts. The study showed that this RL-based methodology, if further developed, has the potential to significantly improve aircraft survivability in highly challenging and unpredictable ship landing situations.

5.1 Challenges and Further Study

First and foremost, the automation of ship landing should be developed with its unique characteristics in mind. This is a comprehensive task that requires not only landing on a moving platform with 6DOF movements, but also following established procedures such as horizontal approach, visual identification, and vertical landing without following deck movements. The perturbations to aircraft dynamics induced by the unsteady wake from the ship super-structure should also be considered. Through the research, the overwhelming capabilities of the proposed method compared to typical moving deck tracking methods have been demonstrated in terms of safety, accuracy, simplicity, compatibility and operational efficiency. However, there exists technical areas that can be improved as detailed below.

Current vision systems can recognize visual cues and estimate relative positions and heading of the UAV at close range very accurately, but you need to know the dimensions of the visual cues and camera specifications in advance. This vision system can be used for a variety of visual cues. However, if you want to use another visual clue, you need to change the parameters in the vision algorithm. Therefore, it is advisable to use additional sensors such as light detection and distance (LIDAR), radio detection and distance (RADAR), and infrared cameras. Lidar systems are useful in close proximity because you don't need to know the specific size and shape of a visual clue. In addition, recent advances have introduced several COTS Lidar systems that can be conveniently integrated into aircraft systems. To measure the relative distance, directional RADAR systems that transmit a short radio pulse with very high pulse power can also be used. Infrared cameras that use infrared (IR) radiation are effective at night or in poor visibility. Maritime VTOL-enabled aircraft such as the SH / MH60, AW159, and V22 are equipped with a Forward Looking Infrared (FLIR) system that provides visual information and distance to the target. Therefore, the use of infrared camera sensors can be integrated into existing helicopters without the need for additional sensors. Therefore, it is expected that the fusion of the current digital camera, Lidar, RADAR, and the thermal camera can expand its operational capability and reliability.

The developed machine learning-based vision system demonstrated excellent detection of a target ship in the long distance. Even though the YOLOv3 algorithm was carefully selected due to its fast detection rate and good accuracy; however, its performance was not compared with other algorithms. Therefore, a more systematic study on detection algorithms in such a ship landing environment is recommended. The detection accuracy of machine/deep learning-based vision algorithms surpassed the accuracy of human eyes in 2015 and the error was recorded as only 2.25% in 2017. Since then, new algorithms have been developed to increase the detection speed while maintaining the accuracy level. They typically measure the speed of detection from a recorded video, thus the rate is evaluated based on how fast it can detect an object of interest once the image frame is given. However, they still cause latency issues when integrated into the real-time flight control system. Therefore, in order to identify the best algorithm, it is recommended to compare the speed and accuracy of algorithms through flight testing. If the current machine learning based detection feedback rate of 0.5 seconds can be reduced to below 0.1 seconds, more rigorous control strategies can be constructed to enhance the flight performance in the long distance.

It is very powerful to use the state-of-the-art deep reinforcement learning control strategy at a close distance from the ship where the state feedback from the vision system is fast. It can quickly respond to a sudden wind gust, thus the drift can be minimized better than other conventional control systems. Also, this is an area where fast and accurate control is important, thus machines can perform better than human pilots by design. Even though the control policy obtained by training with TD3 algorithm with selected states and reward engineering demonstrated the improved disturbance rejection capability compared to the nonlinear PID control system, more extensive studies on deep reinforcement learning control algorithms, states, and rewards are recommended. Proximal Policy Optimization (PPO), Trust Region Policy Optimization (TRPO), Soft Actor-Critic (SAC), and Deep Deterministic Policy Gradient (DDPG) are some of the recommended candidate algorithms worth trying for training. The current states are selected as positions and velocities including their 5 previous values. However, the effect of taking the specific number of previous data as states is not fully investigated, thus the number of previous data can be optimized during further studies. Also, the other sets of states can be analyzed to identify the states that can obtain a more effective control policy. After going through multiple trial and error processes, the current reward function was developed. Nevertheless, It is expected that the current reward function could be improved if the rewards are designed more distinctively to achieve the control task.

Also there is research being done on combining RL and conventional controller. RL shows great potential in rejecting disturbances but it was never trained to track a moving platform. The RL and conventional controller can work together to achieve landing on moving platform in highly turbulent environments.The control system can be modified for obstacle avoidance and path planning. Currently the UAV is asked to move in a straight line to the target but this is not ideal in all cases. Hence path planning can be considered for future research.

It is recommended to take a different control strategy in the long-distance since it requires comprehensive decision making to set an approach course due to several factors such as weather condition, ship course, air and sea traffic, and so on. This is an area that the experienced pilot's decision-making ability is more important than quickly controlling an aircraft to achieve certain setpoints. Moreover, it is nearly impossible to develop an explicit algorithm that can take into account all possible scenarios. Thus, imitation learning-based control strategy can be a good approach for developing high-level control policies that encode the skills of human professionals. A well-trained control policy behaves like an experienced pilot because this can be obtained by learning from expert demonstrations in certain situations. Although the current experimental setup simulated ship landing environment closely, it could be improved for larger-scale flight tests. It is recommended to replace the current quadrotor UAV with a small helicopter UAV. The same autonomous ship landing system can be implemented with minimum modifications. Since the small helicopter UAV has similar configurations as existing helicopters such as rotors, landing gear, and center of gravity position, the testing results can be appropriately scaled for full-size helicopter ship landings. Thus, the effect of deck motions (magnitude and frequency) and wind gusts on full-size helicopters can be estimated by considering the differences in vehicle inertial properties and blade tip speeds. Once a small helicopter UAV successfully lands on a ship deck in properly scaled motions, there will be more confidence to implement the system on a full-size helicopter at sea.

To conclude, the application of developed novel autonomous vision-based approach and landing method can be extended broadly since it demonstrated successful landings in one of the most difficult moving platform landing cases, which is ship landing. Possible applications are landing on stationary and moving platforms such as cars, tanks, trains, submarines, and aircraft. It is also able to achieve precise approach and landing in GPS-denied environments such as inside a tunnel, inside a building, under a bridge or forest canopy, etc.. Moreover, the same autonomous flight method can also be applied to other purposes such as air refueling, vertical replenishment, and so forth with some modifications. It is expected that the novel autonomous ship landing method and the results of this study are expected to have potential for use in a variety of areas beyond the originally intended use.

REFERENCES

[1] B. Lee, V. Saj, M. Benedict, and D. Kalathil, "A vision-based control method for autonomous landing of vertical flight aircraft on a moving platform without using gps," *arXiv preprint arXiv:2008.05699*, 2020.

[2] B. Lee, V. Saj, and M. Benedict, "Machine learning vision and nonlinear control approach for autonomous ship landing of vertical flight aircraft,"

[3] B. LEE, *ON THE COMPLETE AUTOMATION OF VERTICAL FLIGHT AIRCRAFT SHIP LANDING*. PhD thesis, Texas A&M University, 2021.

[4] B. Lee, V. Saj, M. Benedict, and D. Kalathil, "A deep reinforcement learning control strategy for vision-based ship landing of vertical flight aircraft," in *AIAA AVIATION 2021 FORUM*, p. 3218, 2021.

[5] J. L. Sanchez-Lopez, J. Pestana, S. Saripalli, and P. Campoy, "An approach toward visual autonomous ship board landing of a vtol uav," *Journal of Intelligent & Robotic Systems*, vol. 74, no. 1, pp. 113–127, 2014.

[6] G. Xu, Y. Zhang, S. Ji, Y. Cheng, and Y. Tian, "Research on computer vision-based for uav autonomous landing on a ship," *Pattern Recognition Letters*, vol. 30, no. 6, pp. 600–605, 2009.

[7] Q. H. Truong, T. Rakotomamonjy, A. Taghizad, and J.-M. Biannic, "Vision-based control for helicopter ship landing with handling qualities constraints," *IFAC-PapersOnLine*, vol. 49, no. 17, pp. 118–123, 2016.

[8] W. K. Holmes and J. W. Langelaan, "Autonomous ship-board landing using monocular vision," in *Proc. 72nd Am. Helicopter Soc Forum*, vol. 2, p. 36, 2016.

[9] Y. Meng, W. Wang, H. Han, and J. Ban, "A visual/inertial integrated landing guidance method for uav landing on the ship," *Aerospace Science and Technology*, vol. 85, pp. 474–480, 2019.

[10] O. A. Yakimenko, I. I. Kaminer, W. J. Lentz, and P. Ghyzel, “Unmanned aircraft navigation for shipboard landing using infrared vision,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 38, no. 4, pp. 1181–1200, 2002.

[11] B. Lumsden, C. Wilkinson, and G. Padfield, “Challenges at the helicopter-ship dynamic interface,” 1998.

[12] J. Colwell, “Maritime helicopter ship motion criteria-challenges for operational guidance,” *Challenges for Operational Guidance-NATO RTO Systems Concepts and Integration Panel SCI-120. Berlin, Germany*, 2002.

[13] A. L. Stingl, “Vtol aircraft flight system,” Jan. 6 1970. US Patent 3,487,553.

[14] “Helicopter operations from ships other than aircraft carriers(hostac),” vol. I, NATO standard, 2017.

[15] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 580–587, 2014.

[16] R. Girshick, “Fast r-cnn,” in *Proceedings of the IEEE international conference on computer vision*, pp. 1440–1448, 2015.

[17] S. Ren, K. He, R. Girshick, and J. Sun, “Faster r-cnn: towards real-time object detection with region proposal networks,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 6, pp. 1137–1149, 2016.

[18] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, “Ssd: Single shot multibox detector,” in *European conference on computer vision*, pp. 21–37, Springer, 2016.

[19] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788, 2016.

[20] J. Redmon and A. Farhadi, "Yolo9000: better, faster, stronger," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7263–7271, 2017.

[21] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[22] B. Benjdira, T. Khursheed, A. Koubaa, A. Ammar, and K. Ouni, "Car detection using unmanned aerial vehicles: Comparison between faster r-cnn and yolov3," in *2019 1st International Conference on Unmanned Vehicle Systems-Oman (UVS)*, pp. 1–6, IEEE, 2019.

[23] W. X. P. S. S. Zishan and S. Weiqun, "Computer vision scheme for autonomous landing of unmanned helicopter on ship deck [j]," *Journal of Beijing University of Aeronautics and Astronautics*, vol. 6, 2007.

[24] X. Wang, S. Pan, Z. Song, and W. Shen, "Vision based analytic 3d measurement algorithm for the autonomous landing of unmanned helicopter on ship deck," *Optical Technique*, vol. 33, pp. 264–267, 2007.

[25] C. S. Sharp, O. Shakernia, and S. S. Sastry, "A vision system for landing an unmanned aerial vehicle," in *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164)*, vol. 2, pp. 1720–1727, Ieee, 2001.

[26] L. S. H. C. Z. Jihong, "A method for estimating position and orientation of an unmanned helicopter based on vanishing line information [j]," *Computer Engineering and Applications*, vol. 9, 2004.

[27] S. Lange, N. Sunderhauf, and P. Protzel, "A vision based onboard approach for landing and position control of an autonomous multirotor uav in gps-denied environments," in *2009 International Conference on Advanced Robotics*, pp. 1–6, IEEE, 2009.

[28] A. Lawther and M. J. Griffin, "Motion sickness and motion characteristics of vessels at sea," *Ergonomics*, vol. 31, no. 10, pp. 1373–1394, 1988.

[29] "Helicopter operating procedures for air-capable ships natops manual," 2003.

[30] A. W. Doerry, “Ship dynamics for maritime isar imaging.,” tech. rep., Sandia National Laboratories, 2008.

[31] J. M. Daly, Y. Ma, and S. L. Waslander, “Coordinated landing of a quadrotor on a skid-steered ground vehicle in the presence of time delays,” *Autonomous Robots*, vol. 38, no. 2, pp. 179–191, 2015.

[32] O. Araar, N. Aouf, and I. Vitanov, “Vision based autonomous landing of multirotor uav on moving platform,” *Journal of Intelligent & Robotic Systems*, vol. 85, no. 2, pp. 369–384, 2017.

[33] B. Lee, “Helicopter autonomous ship landing system,” Master’s thesis, Texas A&M University, 2018.

[34] B. Lee and M. Benedict, “Development and validation of a comprehensive helicopter flight dynamics code,” in *AIAA Scitech 2020 Forum*, p. 1644, 2020.

[35] K. A. Ghamry, Y. Dong, M. A. Kamel, and Y. Zhang, “Real-time autonomous take-off, tracking and landing of uav on a moving ugv platform,” in *2016 24th Mediterranean conference on control and automation (MED)*, pp. 1236–1241, IEEE, 2016.

[36] B. Hu, L. Lu, and S. Mishra, “Fast, safe and precise landing of a quadrotor on an oscillating platform,” in *2015 American Control Conference (ACC)*, pp. 3836–3841, IEEE, 2015.

[37] J. Kim, Y. Jung, D. Lee, and D. H. Shim, “Landing control on a mobile platform for multi-copters using an omnidirectional image sensor,” *Journal of Intelligent & Robotic Systems*, vol. 84, no. 1, pp. 529–541, 2016.

[38] K. Xia, S. Lee, and H. Son, “Adaptive control for multi-rotor uavs autonomous ship landing with mission planning,” *Aerospace Science and Technology*, vol. 96, p. 105549, 2020.

[39] P. Vlantis, P. Marantos, C. P. Bechlioulis, and K. J. Kyriakopoulos, “Quadrotor landing on an inclined platform of a moving ground vehicle,” in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2202–2207, IEEE, 2015.

[40] A. Rodriguez-Ramos, C. Sampedro, H. Bavle, P. De La Puente, and P. Campoy, “A deep reinforcement learning strategy for uav autonomous landing on a moving platform,” *Journal of Intelligent & Robotic Systems*, vol. 93, no. 1-2, pp. 351–366, 2019.

[41] W. Koch, R. Mancuso, R. West, and A. Bestavros, “Reinforcement learning for uav attitude control,” *ACM Transactions on Cyber-Physical Systems*, vol. 3, no. 2, pp. 1–21, 2019.

[42] A. Rodriguez-Ramos, C. Sampedro, H. Bavle, P. De La Puente, and P. Campoy, “A deep reinforcement learning strategy for uav autonomous landing on a moving platform,” *Journal of Intelligent & Robotic Systems*, vol. 93, no. 1-2, pp. 351–366, 2019.

[43] Y. Li, H. Li, Z. Li, H. Fang, A. K. Sanyal, Y. Wang, and Q. Qiu, “Fast and accurate trajectory tracking for unmanned aerial vehicles based on deep reinforcement learning,” in *2019 IEEE 25th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, pp. 1–9, IEEE, 2019.

[44] J. Hwangbo, I. Sa, R. Siegwart, and M. Hutter, “Control of a quadrotor with reinforcement learning,” *IEEE Robotics and Automation Letters*, vol. 2, no. 4, pp. 2096–2103, 2017.

[45] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, “Domain randomization for transferring deep neural networks from simulation to the real world,” in *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pp. 23–30, IEEE, 2017.

[46] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, “Sim-to-real transfer of robotic control with dynamics randomization,” in *2018 IEEE international conference on robotics and automation (ICRA)*, pp. 3803–3810, IEEE, 2018.

[47] B. Lee, V. Saj, and M. Benedict, “Machine learning vision and nonlinear control approach for autonomous ship landing of vertical flight aircraft,” in *Proceedings of the 77th Annual Forum*, (Virtual), The Vertical Flight Society, May 2021.

[48] M. Couprie and G. Bertrand, “Topological gray-scale watershed transformation,” in *Vision Geometry VI*, vol. 3168, pp. 136–146, International Society for Optics and Photonics, 1997.

[49] W. Förstner and E. Gülch, "A fast operator for detection and precise location of distinct points, corners and centres of circular features," in *Proc. ISPRS intercommission conference on fast processing of photogrammetric data*, pp. 281–305, Interlaken, 1987.

[50] Z. Zhang, "Flexible camera calibration by viewing a plane from unknown orientations," in *Proceedings of the seventh ieee international conference on computer vision*, vol. 1, pp. 666–673, Ieee, 1999.

[51] N. Araki, T. Sato, Y. Konishi, and H. Ishigaki, "Vehicle's orientation measurement method by single-camera image using known-shaped planar object," *Int. J. Innov. Comput. Inf. Control*, vol. 7, no. B, pp. 4477–4486, 2011.

[52] OPenCV, "Camera calibration and 3d reconstruction." `https://docs.opencv.org/master/d9/d0c/group__calib3d.html`, 2021 (accessed April 17, 2021).

[53] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[54] K. Levenberg, "A method for the solution of certain non-linear problems in least squares," *Quarterly of applied mathematics*, vol. 2, no. 2, pp. 164–168, 1944.

[55] D. W. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," *Journal of the society for Industrial and Applied Mathematics*, vol. 11, no. 2, pp. 431–441, 1963.

[56] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[57] A. M. Andrew, "Multiple view geometry in computer vision," *Kybernetes*, 2001.

[58] "Gazebo simulation program." `http://gazebosim.org/`.

[59] S. K. Kannan, G. V. Chowdhary, and E. N. Johnson, “Adaptive control of unmanned aerial vehicles: theory and flight tests,” in *Handbook of Unmanned Aerial Vehicles*, pp. 613–673, Springer Netherlands, 2015.

[60] Z. T. Dydek, *Adaptive control of unmanned aerial systems*. PhD thesis, Massachusetts institute of Technology, 2010.

[61] B. Wang, Z. A. Ali, and D. Wang, “Controller for uav to oppose different kinds of wind in the environment,” *Journal of Control Science and Engineering*, vol. 2020, 2020.

[62] T. Dierks and S. Jagannathan, “Output feedback control of a quadrotor uav using neural networks,” *IEEE transactions on neural networks*, vol. 21, no. 1, pp. 50–66, 2009.

[63] C. Nicol, C. Macnab, and A. Ramirez-Serrano, “Robust neural network control of a quadrotor helicopter,” in *2008 Canadian conference on electrical and computer engineering*, pp. 001233–001238, IEEE, 2008.

[64] J. F. Shepherd III and K. Tumer, “Robust neuro-control for a micro quadrotor,” in *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pp. 1131–1138, 2010.

[65] P. Williams-Hayes, “Flight test implementation of a second generation intelligent flight control system,” in *Infotech@ Aerospace*, p. 6995, 2005.

[66] P. Becker-Ehmck, M. Karl, J. Peters, and P. van der Smagt, “Learning to fly via deep model-based reinforcement learning,” *arXiv preprint arXiv:2003.08876*, 2020.

[67] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, “Playing atari with deep reinforcement learning,” *arXiv preprint arXiv:1312.5602*, 2013.

[68] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, “Human-level control through deep reinforcement learning,” *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[69] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *International conference on machine learning*, pp. 387–395, PMLR, 2014.

[70] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International Conference on Machine Learning*, pp. 1587–1596, PMLR, 2018.

[71] O. S. Up, "Twin delayed ddpg." `https://spinningup.openai.com/en/latest/algorithms/td3.html`, 2021 (accessed June 21, 2021).

[72] L. Weng, "Domain randomization for sim2real transfer," *lilianweng.github.io/lil-log*, 2019.

[73] R. A. Davis Jr and D. M. FitzGerald, *Beaches and coasts*. John Wiley & Sons, 2009.

[74] B. Lee, "Intelligent ship landing." `https://youtu.be/ExkyUOdgYaw`, 2021 (accessed April 17, 2021).