Final Technical Report of NAVAIR proposal, BAA No.: N00421-18-S-001

Autonomous Landing of a Rotary-Wing UAV on a Shipboard

Abhishek Kumar Shastry*, and Inderjit Chopra†
University of Maryland, College Park, Maryland, 20740

I. Introduction and Motivation

Autonomous unmanned aerial vehicles(UAV) are of interest to U.S. Navy because they provide unique capability for reconnaissance and surveillance missions when operating close to enemy shores, without significant endangerment to any human life. Automating the final landing approach of the UAV to the shipdeck, is the most difficult part of the entire flight profile of the vehicle. This is because the landing platform is continuously moving in 3D space, being stochastically perturbed by the ocean waves. And since stealth is of paramount importance for UAV operations, the ship cannot transmit its position or orientation, because that risks jeopardizing ship's location if the transmission is intercepted. Hence, use of passive sensors such as camera to estimate the ship's position and orientation for landing is imperative. This neccessiates the development and testing of control techniques for accurate tracking of a moving target with limited sensor information. This article focuses on the development of such control techniques and its testing with both high fidelity flight mechanics models in simulation and with real-time quadrotor hardware.

The complete autonomous shipboard landing problem is an extremely complicated task, as evident by the huge cognitive workload of the human pilots performing the task today for manned vehicles. When dealing with such a complicated problem with many technological barriers, it is often advantageous to split them into small problems, solutions of which will lay the foundation for a successful operation. This is the approach that has been used by us.

II. Landing On A Stationary Target

The most elementary problem close to the final shipboard landing task is that of landing on a stationary target in well controlled laboratory environment and without any gusts or wake of the ship. This section discusses our process of achieving that.

A. Vehicle

Flight hardware used for landing on a stationary target is shown in Figure 1. The entire vehicle has a footprint of about 240 mm $\times$ 360 mm, a gross take-off weight of about 620 grams and max thrust to weight ratio of 4. An internal 6-DOf Inertial Measurement Unit(IMU) serves as the sensor for estimation of orientation and angular velocity of the vehicle. An optical flow sensor combined with a sonar serves as the instrument for obtaining position and velocity estimates of the quadrotor. It is to be noted that while the frame is custom built, the flight controller, sensors and motors are taken from a commercially available bebop quadrotor. This is because tracking and landing on a stationary target is a guidance problem. Using a commercially available flight controller gives us a stable flying vehicle on which guidance can be easily implemented. For tracking and estimation of relative position and orientation of the ship-deck, an on-board custom monocular global shutter camera is used. The images are processed by an onboard flight computer(Up-board) containing Intel Atom processor. A dual band WIFI module connected to the flight computer is used to communicate with an offboard personal computer(PC) serving as the ground station.

*Graduate Research Assistant, Aerospace Engineering, University of Maryland
†Distinguished University Professor, Aerospace Engineering, University of Maryland

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
05/10/2020	Final	02/13/2019 - 05/13/2020

4. TITLE AND SUBTITLE
Autonomous Rotary-Wing UAS Flight in Gusty Shipboard Environment using Snapdragon Flight

5a. CONTRACT NUMBER
N004211920004

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Abhishek Kumar Shastry
Inderjit Chopra

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
University of Maryland
3112 Lee Building 7809 Regents Drive
College Park, MD 20742

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Naval Air Warfare Center Aircraft Division - Pax River, MD
21983 Bundy Road, BLDG 441
Patuxent, MD 20670

10. SPONSOR/MONITOR'S ACRONYM(S)
NAWC-AD

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

DISTRIBUTION A. Approved for public release: distribution unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The report details outlines methods and results of those methods towards advancing autonomous landing of an UAV on a shipboard. Landing on a stationary platform in laboratory environment is consistently demonstrated. Landing on a dynamic platform is currently undergoing flight testing, with preliminary results shown. To tackle gusts or changing vehicle dynamics in the wake of the ship, adaptive and reinforcement learning controls are explored in simulations with promising results.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Inderjit Chopra
			SAR	23	19b. TELEPHONE NUMBER *(Include area code)* (301)4 405-1122

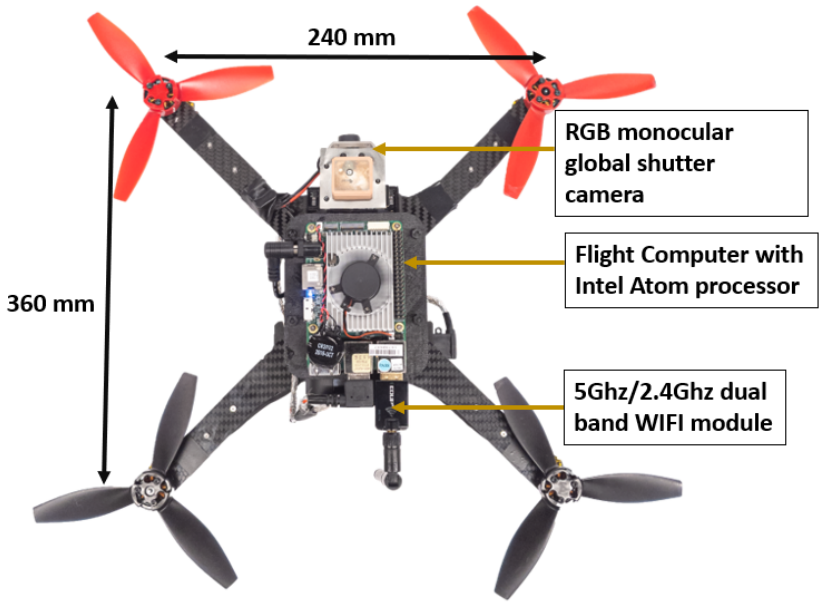


Fig. 1 Quadrotor hardware used for landing on a stationary target

B. Target tracking using Computer Vision

Classical computer vision techniques are used to recognize an Aruco marker placed on the landing target. The marker serves as a fiducial in camera images and is used to obtain position and orientation of the quadrotor relative to the target. It should be noted that if desired, in real world scenarios multiple markers can be used to improve the relative position and orientation estimates of the quadrotor.

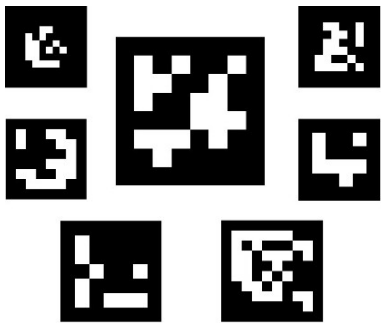


Fig. 2 Aruco markers used as a fiducial for ship-deck tracking

C. Control

Once relative position and velocity of the quadrotor with respect to the landing target is obtained, PD guidance is used to generate position setpoints for the vehicle to navigate to, and is sent to the bebop flight controller, as summarized by the following equation.

$$X_d = K_p * (0 - X) + K_d * (0 - \dot{X}) \tag{1}$$

(2)

Here, X is relative position obtained from camera while $\dot{X}$ is the velocity of the quad obtained from optic flow sensor. As stated in previous section, tracking and landing on a stationary target is a guidance problem and the above guidance law can be implemented on any stable rotary wing vehicle.

D. Results

The results of tracking and landing on a stationary target is shown in Fig 3. As observed, once the target is acquired in the camera frames, the quadrotor converges and lands on the target pretty quickly(in less than 10 seconds).

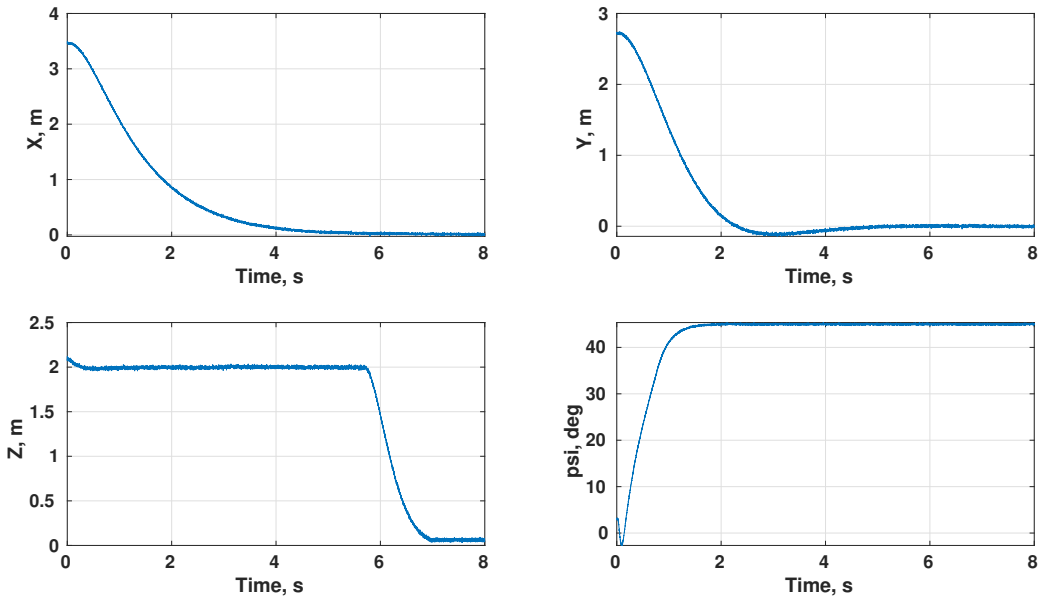


Fig. 3 Results of landing on a stationary target

III. Landing On A Moving Platform

A. Ship motion data (Analysis and Prediction)

Data of shipdeck motion under different sea states was provided by the Office of Naval Research (ONR). The ship was a generic surface combatant moving in a straight line as maintained by its internal Proportional-Derivative(PD) controllers. From the analysis of the data, the deviations of the shipdeck motion around the mean were found to be of Gaussian distribution. The standard deviation of the linear velocities and orientations of the ship-deck for a representative ship motion in extreme weather are presented in Table1.

Surge Velocity	V_x	0.9 ft/s
Sway Velocity	V_y	3 ft/s
Heave Velocity	V_z	2.9 ft/s
Roll Angle	ϕ	6.72°
Pitch Angle	θ	0.78°
Yaw Angle	ψ	0.92°
Roll rate	p	$3.53°/s$
Pitch rate	q	$0.67°/s$
Yaw rate	r	$0.33°/s$

Table 1 Representative standard deviations of ship motion

As observed from the table, deviations in heave, sway and roll motions of the ship are more dominant than other motions of the ship-deck. It should be noted that sway and roll are inherently coupled, as sway of the deck is the result of large ship roll and the deck being high above the sea-line.

Next we try to predict the ship motion using the popular fourier decomposition technique. The idea is that if it is possible to predict the ship-deck motions reasonably well, then the entire problem of landing vehicle on a dynamic platform is reduced to a tracking problem. If so, the quadrotor can aggressively track the ship deck or wait for the predicted deck states to reach a calm steady condition to land. The entire prediction algorithm can be summarized in the following steps:

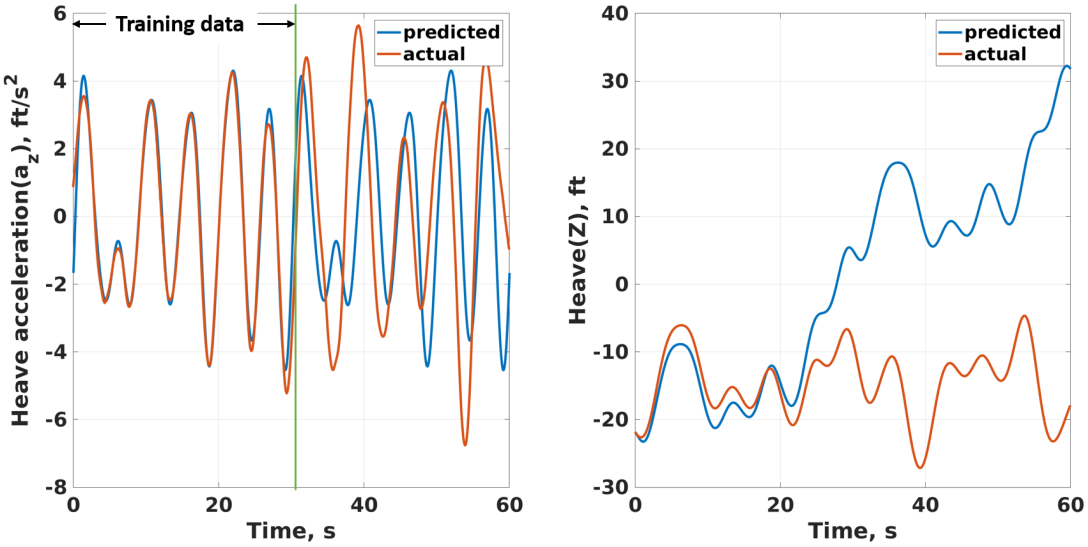
1) Decompose the time history of ship-deck motion into component frequencies through fourier transform.
2) Re-construct the motion from the best n-frequencies and project them into future to predict deck motion in future.
3) Update the component frequencies as new data about ship deck becomes available.

The best n-frequencies for reconstruction is defined as those that minimize the following function:

$$||f(t) - \sum_{i=1}^{n} A_i sin(\omega_i t + \phi_i)||_2^2 \tag{3}$$

where $f(t)$ is the time history of the deck state to be modelled and A_i, ω_i, ϕ_i are the parameters over which the minimization is done.

The results of the method on heave data is shown in Fig 4. Training is done for 30 second time history of heave acceleration and the trained function is then projected forward to predict heave acceleration for 30 seconds. The predicted acceleration is double-integrated to get predicted heave. As observed the method does not works well and hence is not carried forward. Infact, from here-on the disturbances by the sea-waves are assumed to be stochastic and have a gaussian white noise distribution.



(a) Heave acceleration **(b) Heave**

Fig. 4 Analysis and prediction of ship-deck heave with fourier decomposition method

B. Control

A baseline rotary-wing controller for tracking the stochastic motions of a ship is developed using classical control techniques. Utilising the timescale separation of the translational and attitude dynamics, the entire control is divided into two loops. The outer loop, controlling the translational dynamics of the vehicle, is responsible for generating desired thrust, roll and pitch angles for tracking the ship deck trajectory in 3D space. The inner loop controlling the rotational dynamics, generates desired body moments from the desired roll (ϕ), pitch (θ) and yaw (ψ) angles, received from outer loop. The inner control loop of the vehicle is designed as a Proportional-Integral-Derivative (PID) controller of the rotational dynamics linearized around hover. The outer control loop design based on nonlinear Dynamic Inversion theory is described next.

1. Outer loop control for tracking of ship motion

The translational dynamics of a quadrotor can be written in inertial frame (defined as +ve Z toward gravity) in the following form:

$$\begin{bmatrix} \ddot{X} \\ \ddot{Y} \\ \ddot{Z} \end{bmatrix} = \frac{R_b^i}{m} \begin{bmatrix} 0 \\ 0 \\ -\mathbf{T} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} \tag{4}$$

where R_b^i is the rotation matrix from body to inertial frame, $\mathbf{T}$ is the total thrust from all rotors (in our demo case, 4), m is the mass of the vehicle and g is acceleration due to gravity. Now since control appears in second derivative of position, a second order error dynamics for tracking a given trajectory $X_d(t)$, $Y_d(t)$, $Z_d(t)$ is designed as follows:

$$e_x \triangleq X_d - X \tag{5}$$

$$\ddot{e}_x + K_d \dot{e}_x + K_p e_x = 0 \tag{6}$$

$$\implies \ddot{X} = \ddot{X}_d + K_d(\dot{X}_d - \dot{X}) + K_p(X_d - X) \tag{7}$$

where K_p and K_d are tunable gains. Eq. (7) gives the desired acceleration along X for tracking the trajectory. Desired acceleration along Y and Z can be derived in a similar way. These desired accelerations are then converted into desired thrust, roll and pitch angles by inverting the dynamics given by Eq. (34) resulting in following equations:

$$\begin{aligned} \mathbf{T}_d &= m\sqrt{\ddot{X}^2 + \ddot{Y}^2 + (g - \ddot{Z})^2} \\ \theta_d &= \tan^{-1} \frac{\ddot{X}\cos\psi + \ddot{Y}\sin\psi}{g - \ddot{Z}} \\ \phi_d &= \tan^{-1} \frac{\ddot{X}\sin\psi - \ddot{Y}\cos\psi\cos\theta}{g - \ddot{Z}} \end{aligned} \tag{8}$$

These are outer loop control laws. It should be noted that even though the control laws are derived for a quadrotor, they can be easily extended to other rotary wing vehicle configurations as translational dynamics of most rotary wing vehicles, especially around hover are similar to Eq (34). Most rotary wing vehicles change the direction of rotor thrust through changes in body attitudes to achieve translation in 3D space. The complete baseline control algorithm is shown in Figure 5. The estimators are planned to be Extended Kalman Filter.

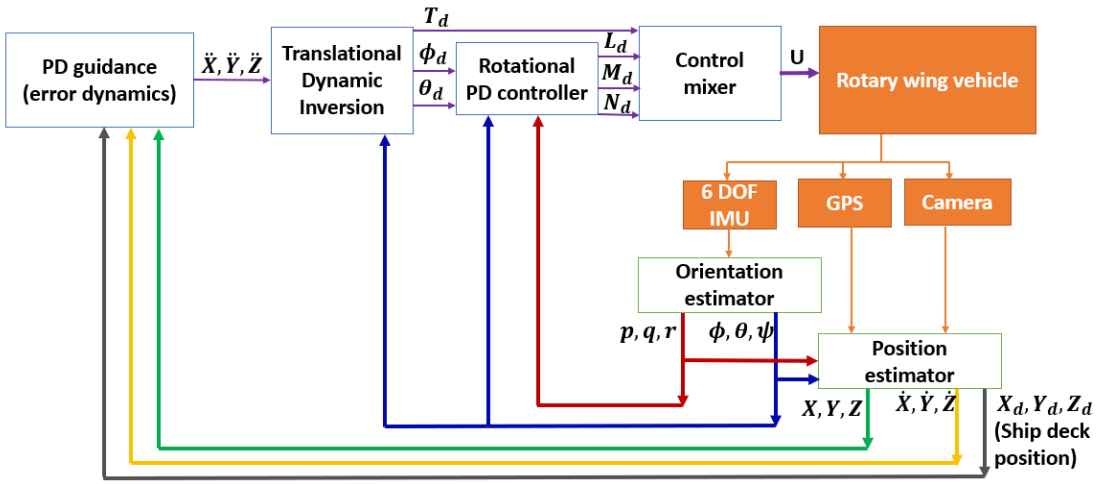


Fig. 5 Baseline rotary wing controller

C. Simulation Results

Fundamental results regarding validation of our technical approach are presented next. First we demonstrate the ability of our baseline tracking controller to track a ship deck in simulation, followed by results of validation of our freewake model.

The designed baseline controller is tested with the most elementary flight mechanics simulation model, which models rotors as simple thrust producing devices. A sample ship-deck motion trajectory provided by ONR is used as the desired trajectory to be tracked by the controller. First, we assume that knowledge of target's position (X_d), velocity ($\dot{X}_d$) and acceleration ($\ddot{X}_d$) are available to be fed forward into the tracking controller. The results are shown in Figure 6.

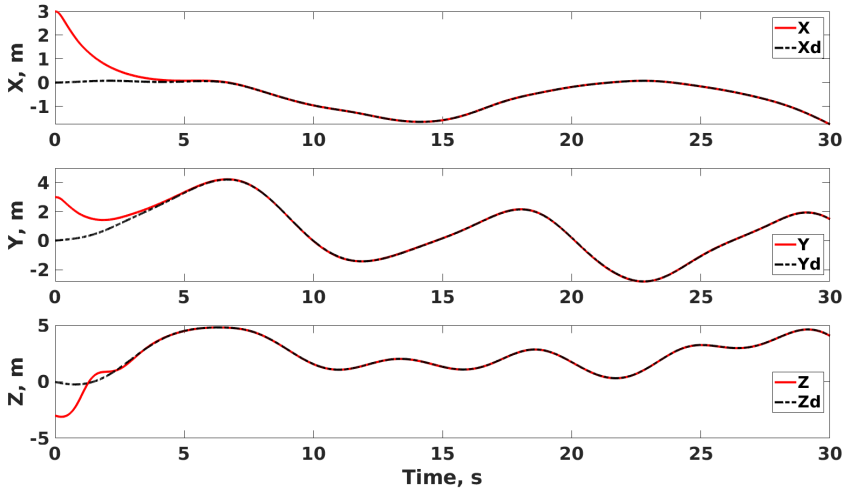


Fig. 6 Ideal tracking; dotted lines are ship deck trajectory, solid lines are trajectory of quadrotor tracking it

Clearly the controller is capable of perfect tracking with perfect knowledge of the target's trajectory. But in practice, it is difficult to estimate the acceleration ($\ddot{X}_d$) and velocity ($\dot{X}_d$) of the target, especially with sensors such as camera. Hence in this situation only position of the target (X_d) is available to be fed forward into the tracking controller. The result for this simulation is shown in Figure 7.

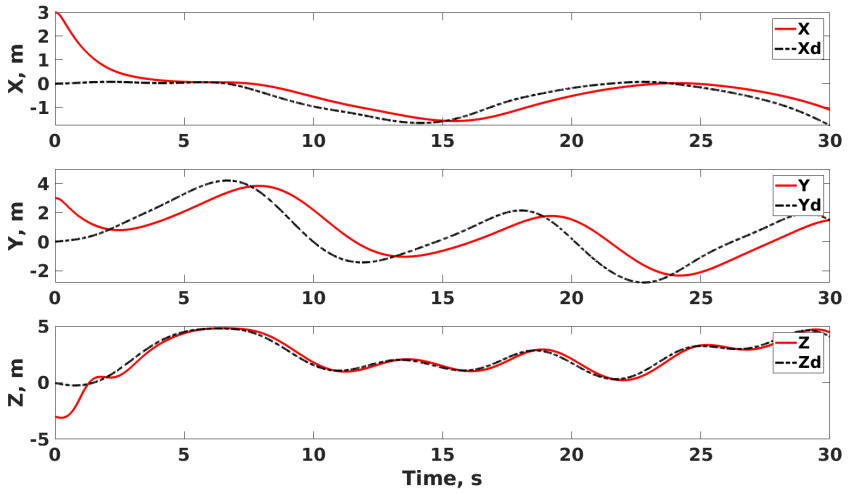


Fig. 7 Tracking with only position feed forward, large phase lag

As observed there is a lag in tracking of the target's trajectory which is the result of high K_d gain in error dynamics(Eq. (6)). If we reduce the gain, the lag reduces but vehicle now significantly overshoots the target's position. This is observed in Figure 8.

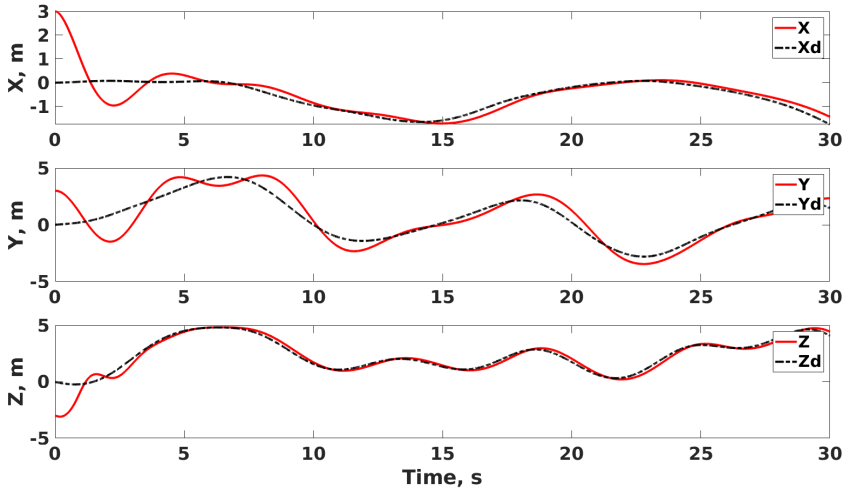


Fig. 8 Tracking with only position feed forward, large overshoot

These observations show that the overshoot and lag in tracking cannot be reduced simultaneously with only position information of the target. These observations are typical steady state behavior of second order systems, in which low damping leads to high amplitude, while high damping leads to large phase lag. This is consistent with the fact that in the absence of velocity and acceleration information of the target, the error dynamics in Eq (6) becomes

$$\ddot{e}_x + K_d\dot{e}_x + K_p e_x = \ddot{X}_d + K_d\dot{X}_d \tag{9}$$

As a result, the second order error dynamics now has a forcing term ($\ddot{X}_d + K_d\dot{X}_d$) on the right. This forcing term does not allows the tracking error to reduce to zero and is moreover dependent on K_d itself. The forcing term can be reduced significantly (and thereby improving tracking), if in addition to position, information regarding velocity of the target is also available, as shown in Figure9.

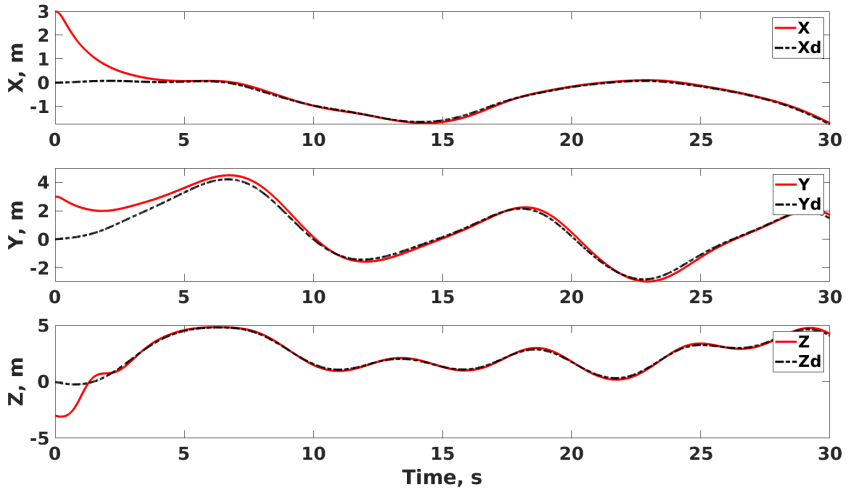


Fig. 9 Tracking with both position and velocity feed forward

These results show that along with position, estimates of target's velocity is also required for good tracking of a moving target like a shipdeck. We plan to estimate both using only camera as the sensor.

D. Vehicle

Flight hardware to be used to demonstrate landing on a dynamic platform is different from that used for landing on a static platform. The primary reason is that the one used for static platform used a proprietory flight controller from Parrot bebop and as a result accessing the lower level controller of the vehicle was not possible. The new flight hardware is shown in Figure 1. The entire vehicle has a footprint of about 280 mm $\times$ 280 mm, a gross take-off weight of about 750 grams and max thrust to weight ratio of about 4. An internal 6dof Inertial Measurement Unit(IMU) will serve as the sensor for estimation of orientation and angular velocity of the vehicle. For indoor experiments, an optical flow sensor combined with an optical distance sensor, instead of GPS will serve as the instrument for obtaining position and velocity estimates of the quadrotor. An onboard monocular global shutter camera will be used to estimate relative position and orientation of the shipdeck. An Aruco marker as shown in Figure 11, placed on the ship-deck will serve as a fiducial in camera images. A 32 bit ARM Cortex M4 microcontroller board will process IMU data and run the inner loop of the baseline controller at 72MHz. All other sensor data will be processed by a flight computer containing a quad core Intel Atom processor. The computer will also be responsible for running the outer loop of the baseline controller. Communication between the processor and the microcontroller is through a dedicated Universal Asynchronous Receiver Transmitter(UART) channel. A dual band WIFI module is to be used to communicate with an off-board personal computer(PC) serving as the ground station. The inner loop of the controller is currently being tested on the flight hardware.

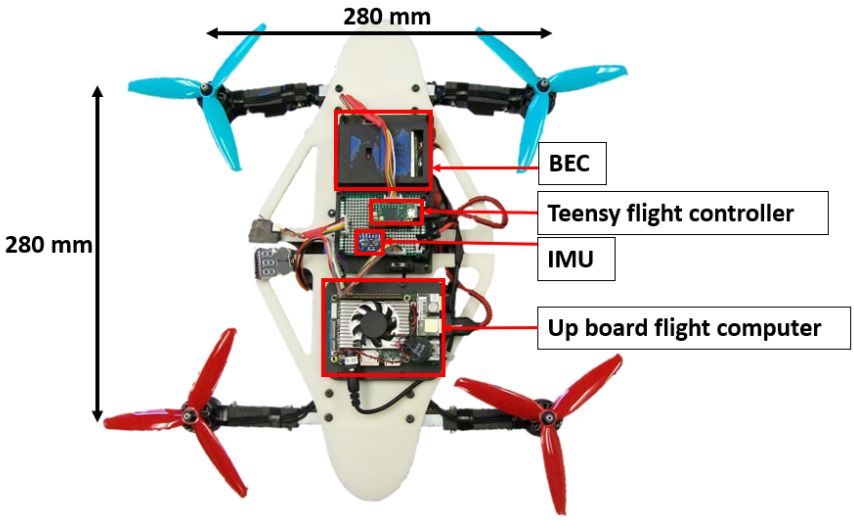


Fig. 10 Quadrotor hardware to be used for landing on a dynamic platform

E. Target tracking(Vision)

Similar to that used for stationbary shipboard landing, classical computer vision techniques are used to recognize an Aruco marker placed on the landing target. The marker serves as a fiducial in camera images and is used to obtain position and orientation of the quadrotor relative to the target. It should be noted that if desired, in real world scenarios multiple markers can be used to improve the relative position and orientation estimates of the quadrotor.

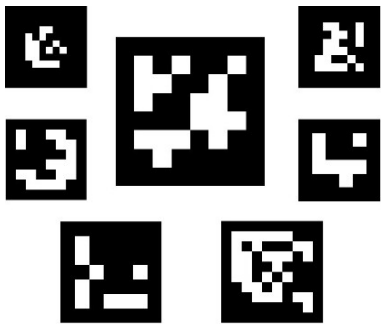


Fig. 11 Aruco markers used as fiducial for ship-deck tracking

F. 6-DoF motion platform for replicating ship deck motion

Final testing of the algorithm will be carried out on a platform that can replicate the 6-DOF stochastic motions of a shipboard. Replication of the 6-DOF motion is to be done using a stewart platform as shown in Fig 12. The benefit of using such a platform for 6-DOF motion replication is that its inverse kinematics is much simpler than its forward kinematics. In other words, it is much easier to obtain the actuator commands for a given 3D position and orientation of the platform than it is to do the other way around.

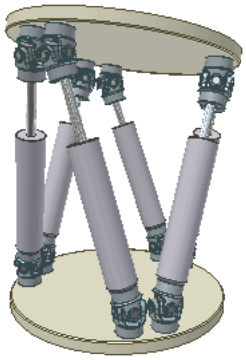


Fig. 12 Example of a 6-DOF stewart platform

Current Status

After re-opening of the campus facilities, design and building of the stewart platform for replicating 6-DOF shipdeck motion has started. The flight hardware is currently being tested for robustness using only an inner loop attitude controller, which will be completed soon and will lead to the testing of the outer navigation loop control using an optic flow sensor for position estimation. The final testing of the flight hardware on the 6-DOF ship motion simulation platform in laboratory environment is expected to be completed by the end of the year.

IV. Tackling Changing Vehicle Dynamics due to gusts

Another problem pertaining to shipboard landing task is flying in an environment with rapidly changing winds/gusts which is similar to flying in the wake of the ship which leads to changing vehicle dynamics. A way to tackle this is to adapt the control inputs accordingly. There are two approaches to control adaptation namely, model based adaptive control and model-free reinforcement learning. Both of these approaches applied to quadrotor are studied in this section.

A. Model-based Adaptive Control Theory

In this section, the derivation of adaptive control law for a control-affine dynamic system is presented. The controller is developed through the use of a dummy state variable which drives the state of the actual system toward the desired state.

Using Lyapunov theory, the stability of the controller is proved. For simplicity, a single state system is first considered. The theory developed for a single state system can be easily extended to multiple states system.

In general, the dynamics of a single state control affine system can be written in state-space form as

$$\dot{x} = f(x) + g(x)u + W_f \phi_f(x) + W_g \phi_g(x)u + \epsilon \tag{10}$$

Here, $W_f \phi_f(x)$ and $W_g \phi_g(x)$ are the terms used to model the uncertainties in $f(x)$ and $g(x)$ respectively. $\phi_f(x)$ and $\phi_g(x)$ are the basis functions which are assumed to be known *a priori*. W_f and W_g are the unknown weights which should be 'learnt' online. ϵ represent the terms in plant dynamics which cannot be modelled into $W_f \phi_f(x)$ or $W_g \phi_g(x)u$(ideal approximation error). ϵ is assumed to be bounded. It should be noted that the bounds on ϵ can always be decreased approaching toward zero(and the complexity of the system be increased) by including more basis functions and corresponding weights in the above equation.

Since W_f, W_g, and ϵ are not known *a priori*, Eq.(10) cannot be used for control derivation. The approximated dynamics equation used for control derivation is

$$\dot{x}_a = f(x) + g(x)u + \hat{W}_f \phi_f(x) + \hat{W}_g \phi_g(x)u \tag{11}$$

where $\hat{W}_f$ and $\hat{W}_g$ are the approximate weights which(by their designed dynamics) should move toward their ideal values W_f and W_g respectively. It is to be noted that x_a is a dummy state variable.
For driving x towards x_d, $\dot{x}_a$ is defined as follows,

$$\dot{x}_a \triangleq \dot{x}_d + K_x(x_d - x) \tag{12}$$

where K_x is a positive constant and is treated as a design parameter.
Defining error as $e \triangleq x - x_d$, the error dynamics equation is(Subtracting Eq.(11) from Eq.(10))

$$\dot{e} + K_x e = \tilde{W}_f \phi_f(x) + \tilde{W}_g \phi_g(x)u + \epsilon \tag{13}$$

where, $\tilde{W}_f = W_f - \hat{W}_f$ and $\tilde{W}_g = W_g - \hat{W}_g$. The equilibrium of the error dynamics is calculated to be

$$e^{id} = 0,\ \tilde{W}_f^{id} = 0,\ and\ \tilde{W}_g^{id} = 0. \tag{14}$$

For the dynamics given in (11), with real positive constants γ_f, and γ_g, the weight update law given by

$$\begin{aligned} \dot{\hat{W}}_f &= \gamma_f e \phi_f \\ \dot{\hat{W}}_g &= \gamma_g e \phi_g u \end{aligned} \tag{15}$$

estimates the uncertainties in $f(x)$ and $g(x)$ and asymptotically stabilizes the identity element of the error space, $\left(e^{id},\ \tilde{W}_f^{id},\ \tilde{W}_g^{id}\right) \equiv (0,\ 0,\ 0)$ if $\epsilon = 0$.
If $\epsilon \neq 0$ and is finite, all the errors remain bounded. Consider a Lyapunov canditate function given by

$$V = \frac{1}{2}e^2 + \frac{1}{2\gamma_f}\tilde{W}_f^2 + \frac{1}{2\gamma_g}\tilde{W}_g^2 \tag{16}$$

The function V is positive definite about the identity element $\left(e^{id},\ \tilde{W}_f^{id},\ \tilde{W}_g^{id}\right) \equiv (0,\ 0,\ 0)$. The time derivative of V is given by

$$\dot{V} = e\dot{e} + \frac{1}{\gamma_f}\tilde{W}_f \dot{\tilde{W}}_f + \frac{1}{\gamma_g}\tilde{W}_g \dot{\tilde{W}}_g \tag{17}$$

Substituting (13) in (17), the expression for $\dot{V}$ is written as

$$\dot{V} = -K_x e^2 + \tilde{W}_f \phi_f e + \tilde{W}_g \phi_g u e + \epsilon e + \frac{1}{\gamma_f}\tilde{W}_f(-\dot{\hat{W}}_f) + \frac{1}{\gamma_g}\tilde{W}_g(-\dot{\hat{W}}_g) \tag{18}$$

Using (15),

$$\dot{V} = -K_x e^2 + \epsilon e \tag{19}$$

Finally, (19) is less than zero if

$$|e| > \frac{|\epsilon|}{K_x} \tag{20}$$

Additionally, if $\epsilon = 0$, $\dot{V}$ becomes negative semi-definite over the complete error space. Thus, we have $\mathcal{M} = \left\{\left(e^{id},\ \tilde{W}_f^{id},\ \tilde{W}_g^{id}\right)\right\}$ as the largest invariant set in the complete error space. Asymptotic stabilization of the equilibrium $\left(e^{id},\ \tilde{W}_f^{id},\ \tilde{W}_g^{id}\right)$ in the error space using the weight update law in (15) follows from *LaSalle's Invariance Principle*.

The control law can be derived using the approximated dynamics model(i.e. Eq(11)) through dynamic inversion or other control techniques. Using dynamic inversion the control law is derived as follows,

$$u = [g(x) + \hat{W}_g \phi_g(x)]^{-1}(\dot{x}_d + K_x e - f(x) - \hat{W}_f\, \phi_f(x)) \tag{21}$$

The control design for quadrotors involes first and second order error dynamics, therefore the control design procedure for the second order system is given in the next subsection.

1. Stability proof of adaptive second order error dynamics

Using the same approach of the previous section, for control affine second order systems dynamics can be expressed as

$$\ddot{x} = f(x,\dot{x}) + g(x,\dot{x})u + W_f\, \phi_f(x,\dot{x}) + W_g \phi_g(x,\dot{x})u + \epsilon \tag{22}$$

Using the same reasoning used as done for first order systems, consider approximated dynamics as

$$\ddot{x}_a = f(x,\dot{x}) + g(x,\dot{x})u + \hat{W}_f\, \phi_f(x,\dot{x}) + \hat{W}_g \phi_g(x,\dot{x})u \tag{23}$$

where x_a is dummy variable and is defined as,

$$\ddot{x}_a \triangleq \ddot{x}_d + \omega^2(x_d - x) + 2\zeta\omega(\dot{x_d} - \dot{x}) \tag{24}$$

where ζ and ω are design parameters. Defining error as $e \triangleq x - x_d$, error dynamics for second order system is,(Subtracting Eq. (23) from (22))

$$\ddot{e} + 2\zeta\omega\dot{e} + \omega^2 e = \tilde{W}_f\, \phi_f(x,\dot{x}) + \tilde{W}_g \phi_g(x,\dot{x})u + \epsilon \tag{25}$$

ζ and ω are design parameters. For proving the stability of the system we choose the following Lyapunov function,

$$V = \frac{1}{2}\begin{bmatrix} e & \dot{e} \end{bmatrix}\begin{bmatrix} a & b \\ b & c \end{bmatrix}\begin{bmatrix} e \\ \dot{e} \end{bmatrix} + \frac{1}{2\gamma_f}\tilde{W}_f^2 + \frac{1}{2\gamma_g}\tilde{W}_g^2 \tag{26}$$

V is positive definite for $a > 0$ and $ac - b^2 > 0$ (Sylvester's criterion for positive definite matrices).
Differentiating Eq.(26)

$$
\begin{aligned}
\dot{V} &= \begin{bmatrix} e & \dot{e} \end{bmatrix} \begin{bmatrix} a & b \\ b & c \end{bmatrix} \begin{bmatrix} \dot{e} \\ \ddot{e} \end{bmatrix} + \frac{1}{\gamma_f} \tilde{W_f} \dot{\tilde{W}}_f + \frac{1}{\gamma_g} \tilde{W}_g \dot{\tilde{W}}_g \\
&= \begin{bmatrix} ae + b\dot{e} & be + c\dot{e} \end{bmatrix} \begin{bmatrix} \dot{e} \\ \ddot{e} \end{bmatrix} + \frac{1}{\gamma_f} \tilde{W_f} \dot{\tilde{W}}_f + \frac{1}{\gamma_g} \tilde{W}_g \dot{\tilde{W}}_g \\
&= ae\dot{e} + b\dot{e}^2 + (be + c\dot{e})\ddot{e} + \frac{1}{\gamma_f} \tilde{W_f} \dot{\tilde{W}}_f + \frac{1}{\gamma_g} \tilde{W}_g \dot{\tilde{W}}_g \\
&= ae\dot{e} + b\dot{e}^2 + (be + c\dot{e})(-2\zeta\omega\dot{e} - \omega^2 e + \tilde{W}_f \phi_f(x) + \tilde{W}_g \phi_g(x) u + \epsilon) + \frac{1}{\gamma_f} \tilde{W_f} \dot{\tilde{W}}_f + \frac{1}{\gamma_g} \tilde{W}_g \dot{\tilde{W}}_g \\
&= ae\dot{e} + b\dot{e}^2 - 2\zeta\omega be\dot{e} - \omega^2 be^2 - 2\zeta\omega c\dot{e}^2 - \omega^2 ce\dot{e} \\
&\quad + \tilde{W}_f \big(- \frac{\dot{\hat{W}}_f}{\gamma_f} + (be + c\dot{e})\phi_f \big) + \tilde{W}_g \big(- \frac{\dot{\hat{W}}_g}{\gamma_g} + (be + c\dot{e})\phi_g u \big) + (be + c\dot{e})\epsilon \\
&= -\omega^2 be^2 + (b - 2\zeta\omega c)\dot{e}^2 + (a - 2\zeta\omega b - \omega^2 c) e\dot{e} \\
&\quad + \tilde{W}_f \big(- \frac{\dot{\hat{W}}_f}{\gamma_f} + (be + c\dot{e})\phi_f \big) + \tilde{W}_g \big(- \frac{\dot{\hat{W}}_g}{\gamma_g} + (be + c\dot{e})\phi_g u \big) + (be + c\dot{e})\epsilon
\end{aligned}
$$

For stability $\dot{V} < 0$. Hence the necessary conditions for stability are,

$$b > 0 \tag{27}$$

$$c \geq \frac{b}{2\zeta\omega} > 0 \tag{28}$$

$$a = 2\zeta\omega b + \omega^2 c > 0 \tag{29}$$

$$\dot{\hat{W}}_f = \gamma_f (be + c\dot{e})\phi_f \tag{30}$$

$$\dot{\hat{W}}_g = \gamma_g (be + c\dot{e})\phi_g u \tag{31}$$

It is to be noted that the conditions for V to be positive definite, $a > 0$ and $ac - b^2 = 2\zeta\omega b(c - \frac{b}{2\zeta\omega}) + \omega^2 c^2 > 0$ are automatically satisfied. Hence Eq. (29)-(31) are also the sufficient conditions for stability.
With these conditions the equation for $\dot{V}$ reduces to

$$\dot{V} = -\alpha e^2 - \beta \dot{e}^2 + b\epsilon e + c\epsilon\dot{e} \tag{32}$$

where $\alpha \triangleq \omega^2 b$, $\beta \triangleq 2\zeta\omega c - b$.
Hence $\dot{V} < 0$, when $|e| > \frac{b|\epsilon|}{\alpha} and \ \ |\dot{e}| > \frac{c|\epsilon|}{\beta}$ This shows that both e and $\dot{e}$ always remain bounded. Using dynamic inversion the control law can be derived as follows

$$u = [g(x, \dot{x}) + \hat{W}_g \phi_g(x, \dot{x})]^{-1} (\ddot{x}_a - f(x, \dot{x}) - \hat{W}_f \phi_f(x, \dot{x})) \tag{33}$$

2. Function Approximators

The two types of uncertainties tackled in this report are structured and unstructured. Structured uncertainties are measurement errors in plant parameters such as mass, moment of inertia or aerodynamic coefficients. The basis functions for this type of uncertainty are known *a priori* and are obtained from the structure of functions $f(x)$ and $g(x)$ in plant dynamics (Eq. 22). Unstructured uncertainties are uncertain forces and moments which may arise due to modeling error or due to sudden changes in external conditions or actuator failure while in flight. Since the basis functions for these uncertainties are not known *a priori*, radial basis functions which (from universal approximation theorem [1]) are universal approximators are used. The single layer neural network formed in this way is called RBF-network. It is to be noted that the adaptive theory developed here is only valid for single hidden layer neural network. Hence other types of neural network model which require more than one hidden layer for universal approximation such as MLP(Multi-layer perceptron) cannot be used.

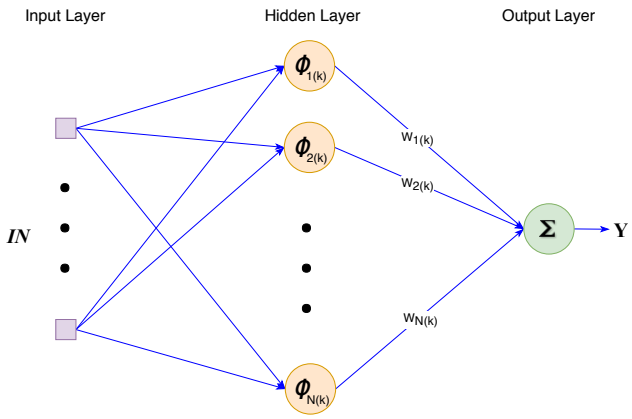


Fig. 13 RBF network

B. Adaptive control of quadrotors

In this section, the above developed theory is applied to quadrotor dynamics. Utilizing the timescale separation, the controller is still designed as two loops, with the inner loop still being a PID controller. The outer loop's dynamic inversion controller is however augmented using the adaptive control design method described in the previous section.

1. Outer loop control design

Outer loop provides tracking ability by driving the error in position variables ($e_x = x - x_d$, $e_y = y - y_d$, $e_z = z - z_d$) to zero. Following the theory developed in the previous section first the actual translational dynamics of a quadrotor with uncertain terms is presented. Next, the dynamics is approximated with the introduction of dummy variables, which are used for control derivation. Thereafter, stability of the approach is demonstrated using Lyapunov theory. Finally the control outputs of outer loop is derived from the approximate dynamics equation using Dynamic Inversion.

Translational dynamics of quadrotor in inertial coordinate system is written as,

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \frac{R_b^i}{M} \begin{bmatrix} 0 \\ 0 \\ -T \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} \tag{34}$$

Modifying the equation to account for variation in mass(structured uncertainty),

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \frac{R_b^i(1+W_M)}{M} \begin{bmatrix} 0 \\ 0 \\ -T \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} \tag{35}$$

where W_M is the 'weight' added to account for changes in mass. It is added in the numerator because the adaptive theory developed in the previous section requires the weights to be in linear form.
Modification to account for unmodelled forces (unstructured uncertainty)

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \frac{R_b^i(1+W_M)}{M} \begin{bmatrix} 0 \\ 0 \\ -T \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} + \frac{R_b^i}{M} \begin{bmatrix} \Sigma W_{i1}\phi_{i1} \\ \Sigma W_{i2}\phi_{i2} \\ \Sigma W_{i3}\phi_{i3} \end{bmatrix} + \epsilon \tag{36}$$

where ϵ=[ϵ_x, ϵ_y, ϵ_z] is the ideal approximation error and ϕ_{ij}'s ($j = 1, 2, 3$) are radial basis functions (RBFs), $\phi_{ij} = \exp^{\frac{-||x-c||^2}{r_j^2}}$. The design of this single neural network for unstructured uncertainty is discussed in Subsection C. Since the actual dynamics contain weights which are not known *a priori*, the following approximate dynamics equation

is used for control derivation,

$$\begin{bmatrix} \ddot{x}_a \\ \ddot{y}_a \\ \ddot{z}_a \end{bmatrix} = \frac{R_b^i(1+\hat{W}_M)}{M}\begin{bmatrix} 0 \\ 0 \\ -T \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} + \frac{R_b^i}{M}\begin{bmatrix} \Sigma\hat{W}_{i1}\phi_{i1} \\ \Sigma\hat{W}_{i2}\phi_{i2} \\ \Sigma\hat{W}_{i3}\phi_{i3} \end{bmatrix} \tag{37}$$

where hat ($\hat{}$) represents the approximate value of that term and,

$$\begin{bmatrix} \ddot{x}_a \\ \ddot{y}_a \\ \ddot{z}_a \end{bmatrix} \triangleq \begin{bmatrix} \ddot{x}_d + 2\zeta_x\omega_x(\dot{x}_d - \dot{x}) + \omega_x^2(x_d - x) \\ \ddot{y}_d + 2\zeta_y\omega_y(\dot{y}_d - \dot{y}) + \omega_y^2(y_d - y) \\ \ddot{z}_d + 2\zeta_z\omega_z(\dot{z}_d - \dot{z}) + \omega_z^2(z_d - z) \end{bmatrix} \tag{38}$$

where ζ_x, ζ_y, ζ_z, ω_x, ω_y and ω_z are design parameters. Subtracting Eq. (37) from Eq. (36) we get the following error dynamics,

$$\begin{bmatrix} \ddot{e}_x + 2\zeta_x\omega_x\dot{e}_x + \omega_x^2 e_x \\ \ddot{e}_y + 2\zeta_y\omega_y\dot{e}_y + \omega_y^2 e_y \\ \ddot{e}_z + 2\zeta_z\omega_z\dot{e}_z + \omega_z^2 e_z \end{bmatrix} = \tilde{W}_M\frac{R_b^i}{M}\begin{bmatrix} 0 \\ 0 \\ -T \end{bmatrix} + \frac{R_b^i}{M}\begin{bmatrix} \Sigma\tilde{W}_{i1}\phi_{i1} \\ \Sigma\tilde{W}_{i2}\phi_{i2} \\ \Sigma\tilde{W}_{i3}\phi_{i3} \end{bmatrix} + \begin{bmatrix} \epsilon_x \\ \epsilon_y \\ \epsilon_z \end{bmatrix} \tag{39}$$

where $\tilde{W}_{ij} = W_{ij} - \hat{W}_{ij}$, $\tilde{W}_M = W_M - \hat{W}_M$ and

$$\begin{bmatrix} e_x \\ e_y \\ e_z \end{bmatrix} \triangleq \begin{bmatrix} x - x_d \\ y - y_d \\ z - z_d \end{bmatrix} \tag{40}$$

Next, we show the stability of the above error dynamics

2. Weight update law and lyapunov stability proof

We choose the following Lyapunov function,

$$\begin{aligned} V = \frac{1}{2}\begin{bmatrix} e_x & \dot{e}_x \end{bmatrix} P_x \begin{bmatrix} e_x \\ \dot{e}_x \end{bmatrix} + \frac{1}{2}\begin{bmatrix} e_y & \dot{e}_y \end{bmatrix} P_y \begin{bmatrix} e_y \\ \dot{e}_y \end{bmatrix} + \frac{1}{2}\begin{bmatrix} e_z & \dot{e}_z \end{bmatrix} P_z \begin{bmatrix} e_z \\ \dot{e}_z \end{bmatrix} \\ + \frac{1}{2\gamma_M}\tilde{W}_M^2 + \Sigma\Sigma\frac{1}{2\gamma_{ij}}\tilde{W}_{ij}^2 \end{aligned} \tag{41}$$

where γ_M, γ_{ij} are learning rates and P_x, P_y, P_z are positive definite matrices whose elements satisfy the conditions (29)-(31). Differentiating, expanding Eq. (41) and using Eq. (40),

$$\begin{aligned} \dot{V} =& -\alpha_x e_x^2 - \beta_x\dot{e}_x^2 - \alpha_y e_y^2 - \beta_y\dot{e}_y^2 - \alpha_z e_z^2 - \beta_z\dot{e}_z^2 + \frac{1}{\gamma_M}\tilde{W}_M(-\dot{\hat{W}}_M) + \Sigma\Sigma\frac{1}{2\gamma_{ij}}\tilde{W}_{ij}(-\dot{\hat{W}}_{ij}) \\ &+ (\tilde{W}_M\frac{R_b^i(1,3)}{M}(-T) + \frac{R_b^i(1,1)}{M}\Sigma\tilde{W}_{i1}\phi_{i1} + \frac{R_b^i(1,2)}{M}\Sigma\tilde{W}_{i2}\phi_{i2} + \frac{R_b^i(1,3)}{M}\Sigma\tilde{W}_{i3}\phi_{i3} + \epsilon_x)(b_x e_x + c_x\dot{e}_x) \\ &+ (\tilde{W}_M\frac{R_b^i(2,3)}{M}(-T) + \frac{R_b^i(2,1)}{M}\Sigma\tilde{W}_{i1}\phi_{i1} + \frac{R_b^i(2,2)}{M}\Sigma\tilde{W}_{i2}\phi_{i2} + \frac{R_b^i(2,3)}{M}\Sigma\tilde{W}_{i3}\phi_{i3} + \epsilon_y)(b_y e_y + c_y\dot{e}_y) \\ &+ (\tilde{W}_M\frac{R_b^i(3,3)}{M}(-T) + \frac{R_b^i(3,1)}{M}\Sigma\tilde{W}_{i1}\phi_{i1} + \frac{R_b^i(3,2)}{M}\Sigma\tilde{W}_{i2}\phi_{i2} + \frac{R_b^i(3,3)}{M}\Sigma\tilde{W}_{i3}\phi_{i3} + \epsilon_z)(b_z e_z + c_z\dot{e}_z) \end{aligned}$$

where, $\alpha_x, \beta_x, \alpha_y, \beta_y, \alpha_z, \beta_z$ are appropriate positive constants similar to α and β in Eq. (32). (e.g.: $\alpha_x = \omega_x^2 b_x$, $\beta_x = 2\zeta_x\omega_x c_x - b_x$)

Therefore the weight update laws for $\dot{V} < 0$ are,

$$\begin{aligned}
\dot{\hat{W}}_M &= \gamma_M \left[\frac{R_b^i(1,3)}{M}(b_x e_x + c_x \dot{e}_x) + \frac{R_b^i(2,3)}{M}(b_y e_y + c_y \dot{e}_y) + \frac{R_b^i(3,3)}{M}(b_z e_z + c_z \dot{e}_z) \right](-T) \\
\dot{\hat{W}}_{i1} &= \gamma_{i1} \left[\frac{R_b^i(1,1)}{M}(b_x e_x + c_x \dot{e}_x) + \frac{R_b^i(2,1)}{M}(b_y e_y + c_y \dot{e}_y) + \frac{R_b^i(3,1)}{M}(b_z e_z + c_z \dot{e}_z) \right] \phi_{i1} \\
\dot{\hat{W}}_{i2} &= \gamma_{i2} \left[\frac{R_b^i(1,2)}{M}(b_x e_x + c_x \dot{e}_x) + \frac{R_b^i(2,2)}{M}(b_y e_y + c_y \dot{e}_y) + \frac{R_b^i(3,2)}{M}(b_z e_z + c_z \dot{e}_z) \right] \phi_{i2} \\
\dot{\hat{W}}_{i3} &= \gamma_{i3} \left[\frac{R_b^i(1,3)}{M}(b_x e_x + c_x \dot{e}_x) + \frac{R_b^i(2,3)}{M}(b_y e_y + c_y \dot{e}_y) + \frac{R_b^i(3,3)}{M}(b_z e_z + c_z \dot{e}_z) \right] \phi_{i3}
\end{aligned} \tag{42}$$

Control is found by solving Eq. (37) for T, θ_d and ϕ_d.

$$\begin{aligned}
T &= M\sqrt{a^2 + b^2 + c^2} \\
\theta_d &= tan^{-1} \frac{a \cos\psi_d + b \sin\psi_d}{c} \\
\phi_d &= tan^{-1} \frac{a \sin\psi_d - b \cos\psi_d \cos\theta_d}{c}
\end{aligned} \tag{43}$$

where,

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} \triangleq \frac{1}{(1 + \hat{W}_M)} \left\{ \begin{bmatrix} \ddot{x}_d + 2\zeta_x \omega_x (\dot{x}_d - \dot{x}) + \omega_x^2 (x_d - x) \\ \ddot{y}_d + 2\zeta_y \omega_y (\dot{y}_d - \dot{y}) + \omega_y^2 (y_d - y) \\ \ddot{z}_d + 2\zeta_z \omega_z (\dot{z}_d - \dot{z}) + \omega_z^2 (z_d - z) \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} - \frac{R_b^i}{M} \begin{bmatrix} \Sigma \hat{W}_{i1} \phi_{i1} \\ \Sigma \hat{W}_{i2} \phi_{i2} \\ \Sigma \hat{W}_{i3} \phi_{i3} \end{bmatrix} \right\} \tag{44}$$

3. Simulation Results

In this section, the performance of the adaptive controller is compared with that of a nominal controller (Nonlinear Dynamic Inversion) through some simulations. First, the effect of uncertainty in mass on the tracking performance of both the controllers is shown. Next, the effect of unstructured uncertainty is inspected, and the performance of both the controllers are studied at two different velocities. Finally, the effect of actuator failure on the tracking ability of both the controllers is presented. The controller gains have been adjusted so that its tracking performance is satisfactory when there is no uncertainty in the dynamic model. After that the uncertainties in the dynamic model are brought in and the adaptive portion of the controller is turned on. Neural network uses online training. Hence its learning rate is started from a low value (around 10^{-10}) and is increased until either the performance of the adaptive controller is satisfactory with uncertainties in the dynamic model or the numerical method used for solving the dynamics destabilizes.

In the following simulation, the ability of both the controllers to track a circular trajectory in the presence of variation in mass only is studied. Fig. 14 shows that the effect of uncertainty in mass for a DI-controller is a steady state error in height.

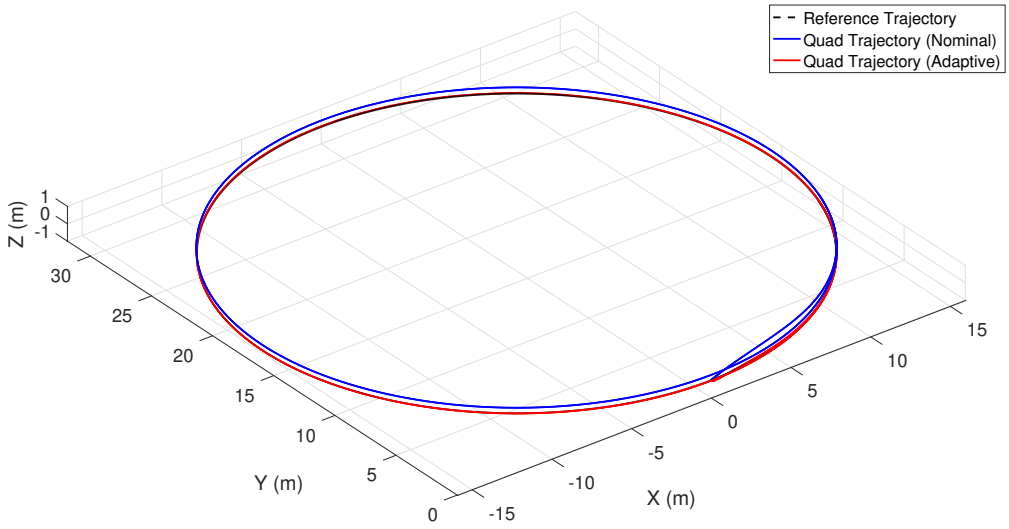


Fig. 14 3-D tracking with uncertain mass

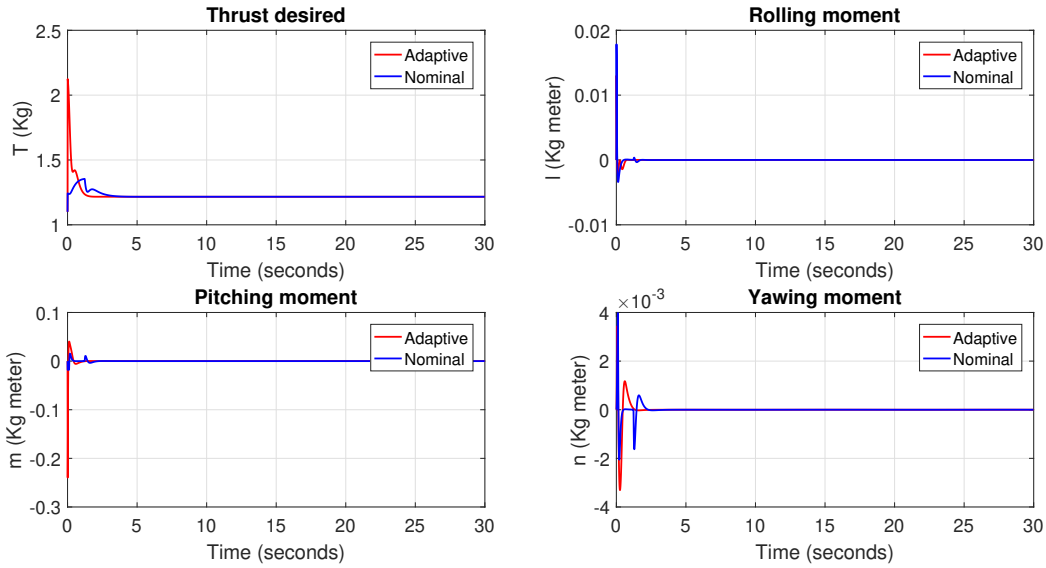


Fig. 15 Control Efforts with uncertain mass

As observed in Fig. 14 the uncertainty in mass is appropriately learnt by the adaptive controller and the error hieght is nullified . Fig. 15 shows the control history for both the controllers.

In the next simulation, the dynamics model uses the advanced rotor kinetics developed from BEMT and published in [2]. Hence for the controller, the simulation model now contains unstructured uncertain terms. The performance of both the controllers is studied at both low and high speeds.

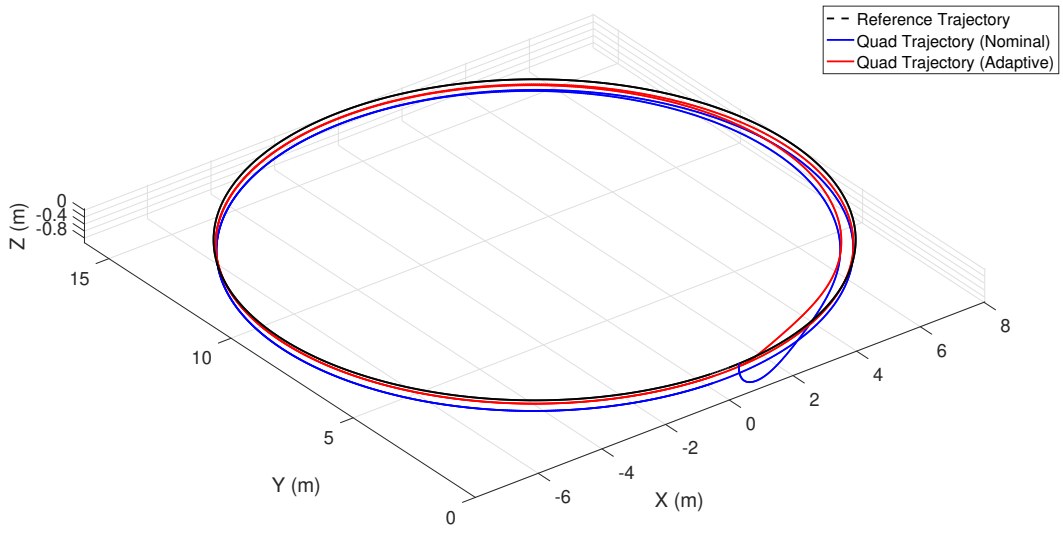


Fig. 16 3-D tracking at low speed (5 m/s)

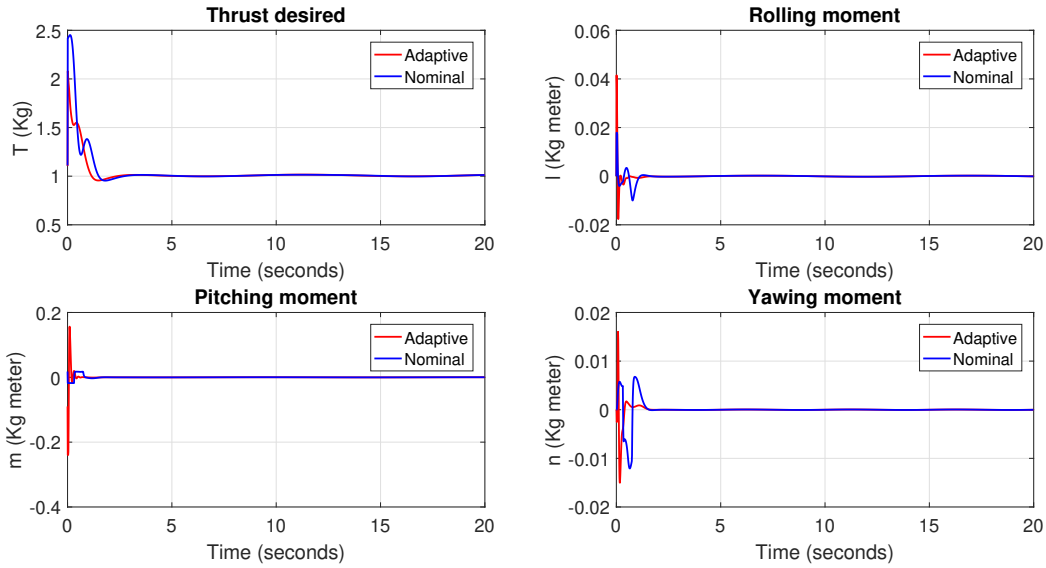


Fig. 17 Control Efforts at low speed (5 m/s)

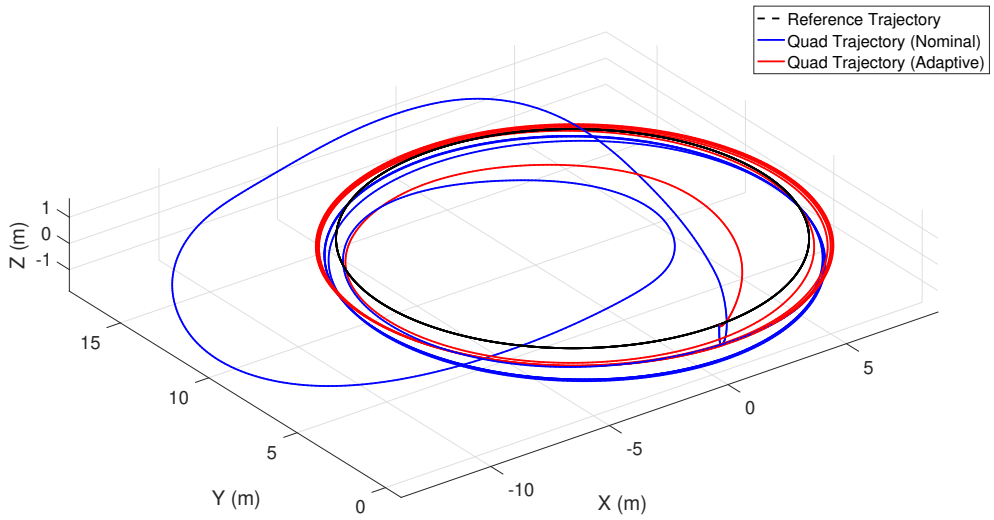


Fig. 18 3-D tracking at high speed (15 m/s)

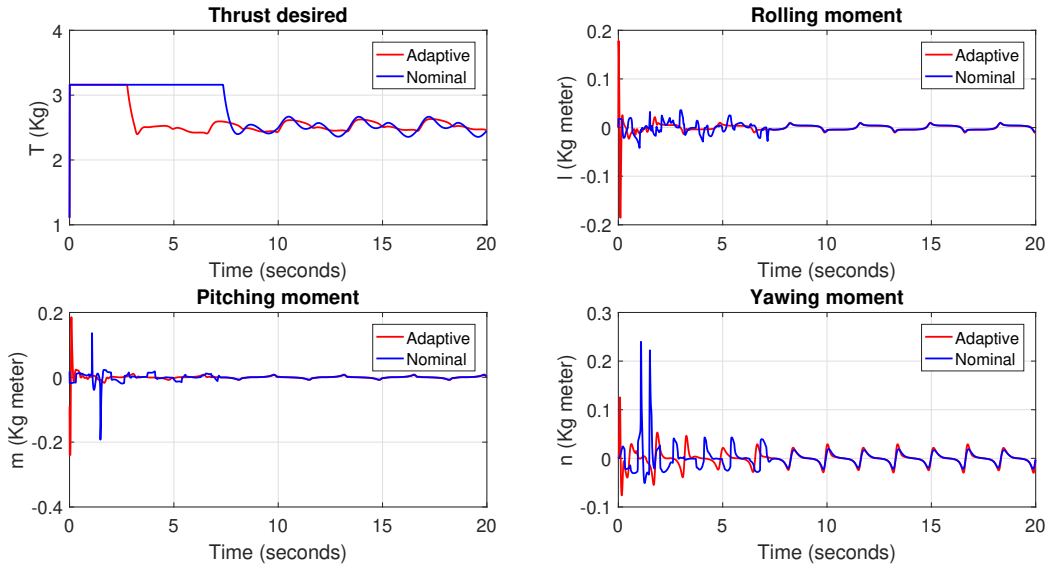


Fig. 19 Control Efforts at high speed (15 m/s)

The aerodynamics of the rotor is the source of uncertainty here. Fig. 16 shows that at low speed both the controllers are capable of tracking the given trajectory because at low speeds the aerodynamics effects which are accounted for in the nominal controller are small. The control forces and moments required to be generated by the controllers are shown in Fig. 17. At high speeds the aerodynamics effects become significant and as it is observed from Fig. 18, the performance of the nominal controller degrades considerably. As also observed from the same figure, the adaptive controller still provides good tracking ability. The effects of rotor aerodynamics on control history for both the controllers is shown in Fig. 19.

Finally, the fault tolerant ability of the controllers is tested. The fault is introduced into one of the motors of the quadrotor in the simulation model after 10 seconds. The faulty motor operates at 60% of the desired rpm.

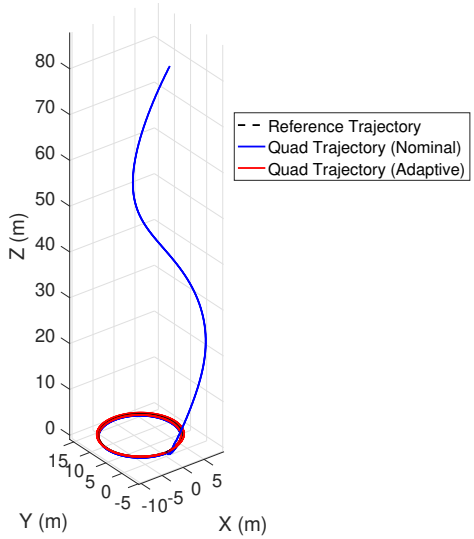


Fig. 20 Fault Tolerant tracking

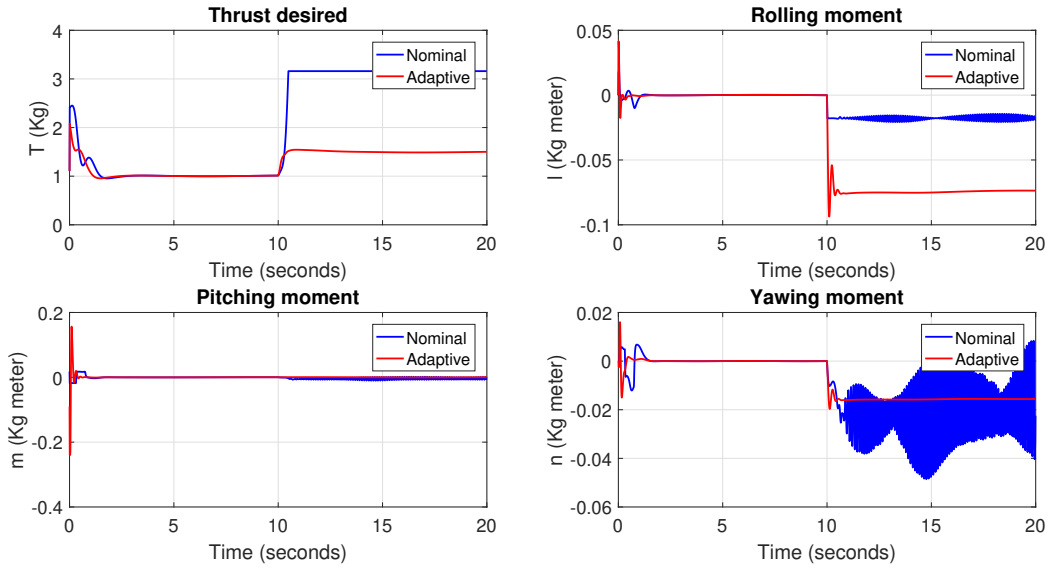


Fig. 21 Fault Tolerant Control Efforts

It is observed from Fig. 20 and Fig. 21 that the nominal controller fails to track the given trajectory and destabilizes while the adaptive controller can control the vehicle even in this situation. The extra rolling moment needed by the adaptive controller after 10 seconds, to balance the thrust from the faulty motor is evident in control history, as shown in Fig 21. These results demonstrate the fault-tolerant capability of the adaptive controller. It should, however, be pointed out that further simulations showed that the fault-tolerant capability of the current controller is limited because of the specific design of its unstructured neural network. However, the controller can be optimized for more robust operations by increasing the size of the RBF-network, but the computing power required for such networks will be enormous.

C. Model-free Reinforcement Learning Control

The basis of all Reinforcement Learning (RL) algorithms employing neural networks for control is Markov Decision Process (MDP) model, which is represented by the following figure:

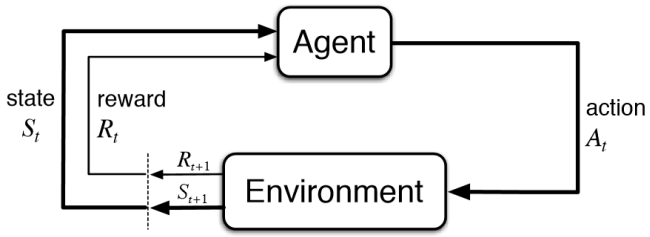


Fig. 22 Markov Decision Process

The environment in MDP is the external system to be controlled. The agent is the program executing actions on environment and receiving rewards and the current state of the environment in return. Action by an agent is same as control input to a system in classical control terminology. A policy in MDP is any function which outputs an action for a given state of the environment. It is to be noted that an agent can either follow a policy or act randomly, which is common during training phase of RL algorithms. All RL algorithms are focussed on finding optimum policy functions for any given environment.

RL controllers have primarily two benefits over traditional analytical controllers namely model-free algorithm and emergent behaviour.

Model-free algorithm: RL controller design does not require the knowledge of system's dynamics apriori. It learns the dynamics during training phase. Hence by nature, it can adapt to changes in plant's dynamics if the training is continued online. This adaptive nature of RL controllers, also make it more robust than PID controllers. It can by design tackle problems like gust, turbulence, ground effect (on one or multiple rotors), partial actuator failure, etc.

Emergent behaviour: For RL controllers, system behaviour emerge during training so as to perform the task as decided by the reward function optimally. For PID these are by design limited to the linear regime and are fixed throughout the system operation. The behaviour of an RL system can be non-linear and change as the environment changes. This comes with the downside that since the behaviour is not known apriori, it could be undesirable in environments the algorithm has not been trained before.

1. Controller structure

Controller structure for a DQN controller is shown in Fig 23. The Q-network is a neural-net that given a state, calculates the Q-values of all possible control inputs(actions). The Q-value of a control input represents the average future rewards for executing that control. Under the optimum policy the control with the maximum Q-value is executed. However, for the optimum policy to be valid the network has to be completely trained to represent to the exact Q-value of the task at hand. Hence we need an algorithm that can train a generic neural-net to optimally represent the Q-values of all possible (state, control) pairs. This training algorithm is discussed next.

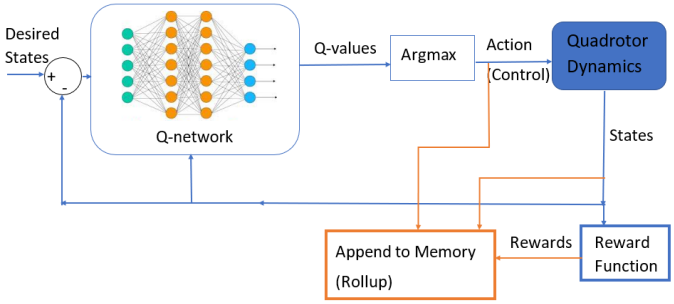


Fig. 23 Deep Q-Network structure. The neural network takes the vehicle state as input and calculates the Q-value of all actions. Under greedy policy, the action with max Q-value is executed.

2. DQN parameter update

The feedback on whether a performed action(control) is appropriate or not is obtained through rewards obtained after performing that action on the environment. Value of a state under a policy is defined as the average rewards that can be obtained by following the policy from that state and is given by the following equation:

$$V_\pi(s_i) = \sum_{j=i}^{\infty} \gamma^{j-i} r(s_j, a_k = \pi(s_j))$$

where $r(s_j, a_k)$ is the reward obtained for taking action a_k in state s_j following a policy π. $\gamma < 1$ is a discount factor (a constant) to indicate that rewards in near future have preference over long-term rewards. This is also necessary for a stable training. Using the value function, now the Q-value function of a (state, action) pair is defined as follows:

$$Q(s_i, a_k) = r(s_i, a_k) + \gamma V_\pi(s_{i+1})$$

It is evident by definition that under the optimal policy(π^*), which is defined as the one that maximizes the value function, the following relation holds:

$$V_{\pi^*}(s_i) = \max_{a_k} Q^*(s_i, a_k) = \max_{a_k}(r(s_i, a_k) + \gamma V_{\pi^*}(s_{i+1}))$$

Henceforth, we get the following:

$$Q^*(s_i, a_k) = r(s_i, a_k) + \gamma \max_{a_k} Q^*(s_{i+1}, a_k)$$

This is the Bellman equation of optimality [8] written in terms of Q-value function. The update law for the Q-value function is derived from the bellman equation using fixed-point iteration method as given by the following equation:

$$Q_{new}(s_i, a_k) = (1 - \alpha) Q_{old}(s_i, a_k) + \alpha(r(s_i, a_k) + \gamma \max_{a_k} Q_{old}(s_{i+1}, a_k)) \tag{45}$$

where α is the relaxation factor also known as 'learning rate' in reinforcement learning literatures. The above along with gradient descent algorithm is used to train the Q-network. In the above algorithm, both learning rate and discount factors are hyperparameters or constants that need to be 'tuned' for the algorithm. Low learning rate and high discount factors lead to stable but slow learning algorithms, while high learning rate and low discount factor leads to faster but possibly numerically 'unstable' learning algorithms.

Using this framework, the process of finding the optimal policy is simply reduced to the process of finding a good reward function for a given environment. They can range from very simple generic functions to ones that are well crafted for the problem at hand, which have higher tendency to produce more robust control solutions(or policies) [9].

3. Simulation Results

The simulation experiment is designed to test the ability of a Q-learning algorithm to control a hovering quadrotor. Neural networks used are simple feed forward neural networks. The state of the environment is defined as the 6 Degree-of-Freedom (DoF) pose representing the current state, concatenated with the 6 DoF pose error. Since DQN requires discrete action space, the continuous action space of quadrotor is discretized into 'bins'. The bins in this study is to increase or decrease or no change of motor rpm of quadrotor by a fixed value. This set of bins is chosen because it requires a coarser discretization of the control action space, yet it allows for a high granularity of control actions to be achieved. Simulation validation of the DQN learning algorithm is performed using a simple set of reduced degree of freedom non-linear quadrotor rotational dynamics model. The 1-DOF model tests the ability of DQN to control only the pitch of the quadrotor. The 2-DOF model incorporates the control of roll in addition to pitch.

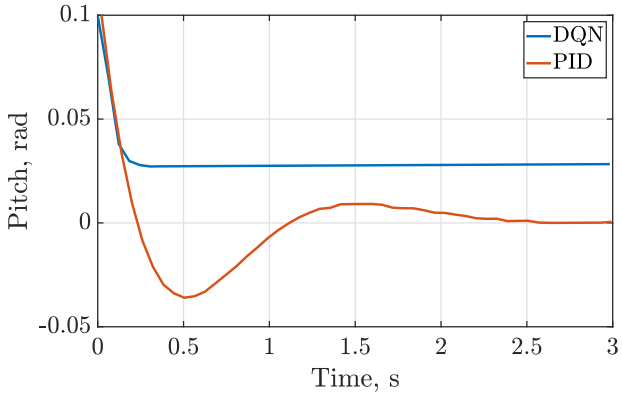


Fig. 24 Validation of Deep Q-Learning on a quadrotor to perform pitch control. The steady state error in DQN is believed to be the result of coarse discretization of continuous controls for quadrotor

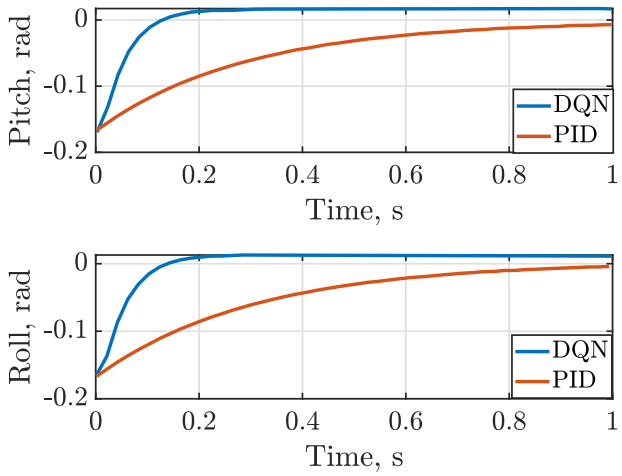


Fig. 25 Validation of Deep Q-learning on a quadrotor for 2d-attitude control.

As observed from the figures, DQN has atleast similar performance to that of a PID controller. This validates the generic idea of using neural nets for quadrotor stabilization. The DQN controller though has a steady state error, the source of which is believed to be the discretization of the continuous action set for the controller. It is important here to point out that even though in the above results RL controllers seems to have similar performance to traditional PID, the benefits of RL controllers as model-free algorithm, being adaptive to changing environments and free from extensive online manual tuning, makes it worthwhile to explore it as a flight controller for the quadrotor especially for the shipboard landing task. Also, it is important to recognize that unlike traditional controllers model-free reinforcement learning controllers have no stability proofs, but so is the case for current human pilots flying the aircrafts today. Hence, the only way to satisfactorily validate such controllers is to have extensive rigorous testing with exhaustive set of test cases. Hence it is only in the long-run that such controllers can completely replace human pilots.

Current status

Though DQN algorithm demonstrates good performance in the above simulation environment, it cannot be used in real-world environments primarily because it requires the action space to be discrete and finite. To overcome this limitation, continuous action reinforcement learning algorithms are required. This is currently being developed at the Alfred Gessow Rotorcraft Center (AGRC). Model-based adaptive control has been tested in simulation but remains to be proven using hardware. This will be carried out next year after the basic landing algorithm discussed in Sec. 3 for a dynamic platform has been satisfactorily validated.

V. Conclusions and Future Work

In this work, controllers for landing an UAV on a shipboard are developed. The controller has been successfully implemented in hardware and tested for landing on a static platform. For landing on a dynamic platform, the controller has been tested in simulations and hardware validation is ongoing(delayed due to closure of campus facilities as a result of COVID-19) and should be completed by the end of the year. Model-based adaptive control has also been demonstrated in simulations and will be validated in hardware next year. This will be followed by testing of reinforcement learning controllers whose continuous variant for quadrotor control is currently under development.

References

[1] Park, Jooyoung, and Sandberg, I. W., "Universal approximation using radial-basis-function networks,"Neural computation 3.2,1991, pp. 246–257

[2] Shastry, A. K., Kothari, M., and Abhishek, A., "Generalized Flight Dynamic Model of Quadrotor Using Hybrid Blade Element Momentum Theory", Journal of Aircraft, 2018, doi:10.2514/1.C034899

[3] Solomon, E., Hrishikeshavan, V., Chopra, I., (2018). "Autonomous Quadrotor Control and Navigation with Snapdragon Flight". 74th Annual American Helicopter Society Forum: Phoenix, AZ.

[4] Solomon, E., Vorwald, C., Hrishikeshavan, V., Chopra, I.. (2017). "Visual Odometry Onboard a Micro Air Vehicle Using Snapdragon Flight. 7th AHS Technical Meeting on VTOL Unmanned Aircraft Systems and Autonomy": Mesa, AZ.

[5] Solomon, E., Shastry, A., Hrishikeshabvan, V., Chopra, I. (2019), "Reinforcement Learning Control for Quadrotors using Snapdragon Flight": Vertical Flight Society Autonomous VTOL Technical Meeting and Electric VTOl Symposium: Mesa, AZ.

[6] Zhang, T., Kahn, G., Levine, S. and Abbeel, P., "Learning Deep Control Policies for Autonomous Vehicles with MPC-Guided Policy Search", 2016 IEEE International Conference on Robotics and Automation (ICRA).

[7] Polvara, R., Patacchiola, M., Sharma, S.K., Wan, J., Manning, A., Sutton, R. and Cangelosi, A. "Autonomous Quadrotor Landing using Deep Reinforcement Lerarning", CoRR, 2017.

[8] Bellman, R.E., "Dynamic Programming", Princeton University Press, Princeton, NJ. Republished 2003: Dover, ISBN 0-486-42809-5

[9] Heess, N., et al, "Emergence of Locomotion Behaviours in Rich Environments", arXiv:2707.02286v2, 2017.

[10] Berrios, M., Berger, T., Tischler, M., Juhasz, O., Sanders, F., (2017). "Hover Flight Control Design for UAS Using Performance-based Disturbance Rejection Requirements". 73rd Annual American Helicopter Society Forum: Fort Worth, TX.

[11] Weiss, S., Achtelik, M., Lynen, S., Achtelik, M., Kneip, L., Chli, M., Siegwart, R. (2013). "Monocular Vision for Long-term Micro Aerial Vehicle State Estimation": A Compendium. Journal of Field Robotics. Vol 30 Iss 5.

[12] Wang, J., Olson, E. (2016). "AprilTag 2: Efficient and robust fiducial detection". Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).