ECE 417: Honors Option
(Instructor: Dr. Vaibhav Srivastava)

Student Name: Vedant K. Naik
(PID: 181197242)

May 7, 2026

Contents

1 Introduction

The objective of the project is to model the dynamics of a mobile blimp robot and develop a simulation environment to develop and test various control strategies. The project is divided mainly in following sections in 2 the full 6-Dof model of the blimp with physical system constraints is discussed for an assumed form of the robot Fig. 2, in section 3 a controller is developed to track an arbitary trajectory in a plane which is further improved by usage of EHGO to compensate for any unmodeled dynamics and reject any external disturbances.

2 System Model

2.1 Reference Frame Definitions and Assumptions

For simplicity of the model the blimp was assumed to be a rigid body so that dynamics could be derived in 2.2. There were two major reference frames chosen for the discussion in subseqent sections. An inertial frame Σ_0 is chosen to be primary reference frame, and a body fixed refernce frame Σ_b is chosen at the COM of blimp robot which helps in easily referencing all local forces experienced on the body.

2.2 Rigid Body Dynamics

Assume the blimp robot as a rigid body in space, thus it will follow dynamics of the form Eq. (1), (2) as shown in [1], where m is the net mass of the blimp robot $\mathbf{I}$ is the inertia tensor defined about Σ_b, v^b and Ω_b are linear and angular velocities expressed in body fixed frame of blimp robot.

$$m \cdot \dot{v}_b + m \cdot \Omega^b \times v^b = F^b(x, u) \tag{1}$$

$$\mathbf{I} \cdot \dot{\Omega}^b + \Omega^b \times (\mathbf{I} \cdot \Omega^b) = M^b(x, u) \tag{2}$$

2.3 External Forces and Moments

Majorly following forces and subsequent moments acting on the blimp are modeled in the simulator,for simplicity Drag force modeling is left out from further discussion since its explained in the [1].

1. Thruster
2. Gravity
3. Buoyancy
4. Drag/ Aerodynamic

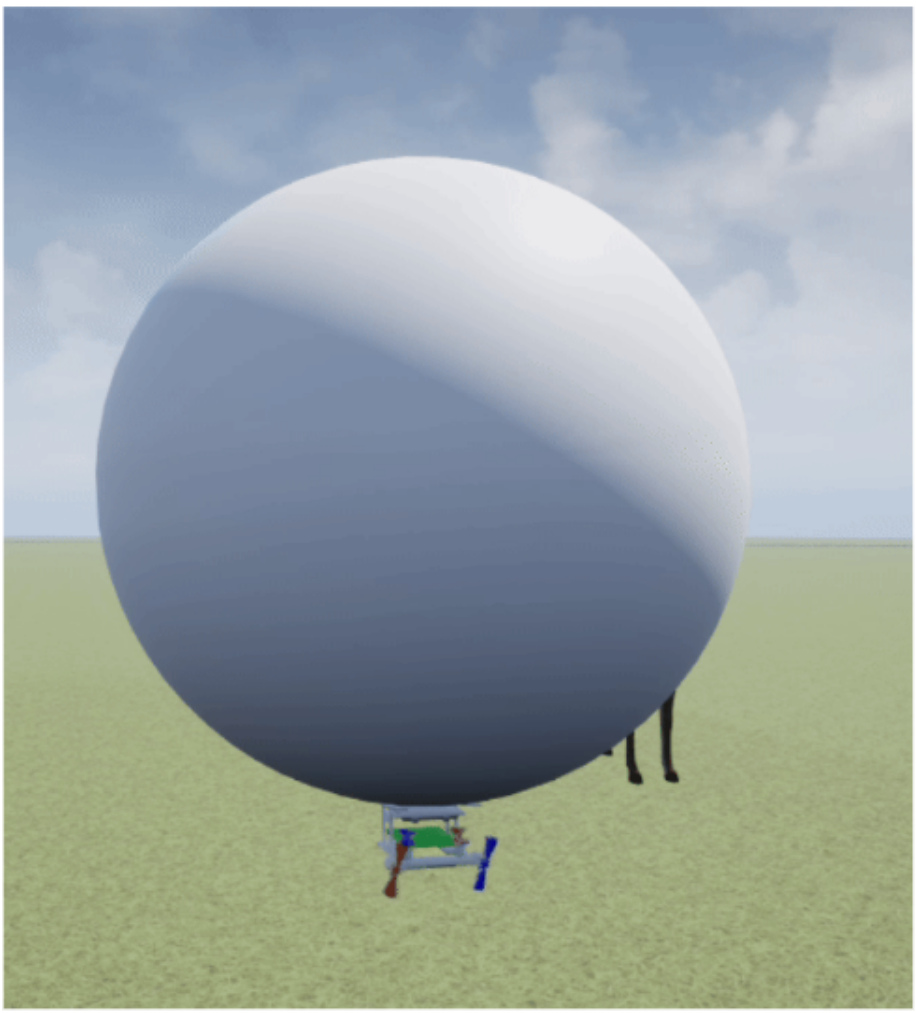

Figure 1: Assumed form of Blimp Robotic Platform

3 Control Strategy

The assumed form of the blimp provides us with 4-actuators, out of which 2 actuators helps in actuating roll and $\hat{z}^b$ control and other 2 actuators help in $\hat{x}^b$ and yaw. Given that due to inherent nature of blimp roll and pitch are naturally stable due to system construction, thus $\hat{z}^b$ and $\hat{z}^0$ are very closely aligned, thus we can easily assume 2 thrusters can help the robot achieve arbitrary z^0. Consequently allowing to decouple the 6-DOF problem into a planar 3-DOF problem.

Decomposing the problem into planar problem also removes the gravity and bouyancy relations from the dynamics. To further simplify the complicated dynamics introduced by the aerodynamic coefficients we can lump them all in a disturbance term δ. As shown in [2] an extended high gain observer (EHGO) is shown to be effective to estimate unmodeled dynamics and disturbances effectively and compensate it.

3.1 Simplified Planar Problem

To address the planar problem we assume an unicycle-like dynamic model described in Eq. 3, where δ_n terms represent the lumped unmodelled dynamics and the disturbances.

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{y} \end{bmatrix} = \begin{bmatrix} f \cdot cos(\theta) + \delta_x \\ f \cdot sin(\theta) + \delta_y \\ M + \delta_\theta \end{bmatrix} \tag{3}$$

Here the robot could actuate its longitudinal (f) or angular acceleration (M) about its COM. As shown in Fig. 2

3.2 Reference Planar Trajectory

We want to track a desired planar trajectory, meaning we are provided $x_d(t)$ and $y_d(t)$. We want the robot's COM to track the desired coordinates.

For the sake of experimentation we assume we want the robot to follow a circular trajectory in plane defined by Eq. 4

$$\begin{bmatrix} x_d \\ y_d \end{bmatrix} = 1.5 \cdot \begin{bmatrix} cos(0.5t) \\ sin(0.5t) \end{bmatrix} \tag{4}$$

3.3 Controller Design

Assuming we have estimates of disturbance and full state estimate, we can design a controller as follows.

3.3.1 Position tracking

Given that we cannot independently command forces in x_b and y_b, we need to design the control action to track x_d and y_d simultaneously

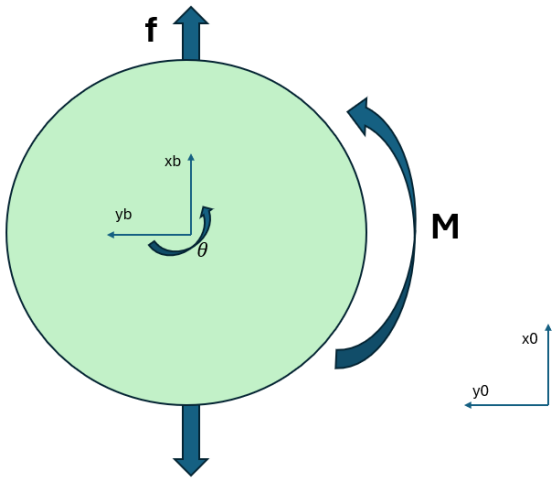

Figure 2: Unicycle-Like Model of Robot

$$\begin{bmatrix} f \cdot cos(\theta) \\ f \cdot sin(\theta) \end{bmatrix} = \begin{bmatrix} \ddot{x_d} + K_p(x_d - \hat{x}) + K_d(\dot{x_d} - \dot{\hat{x}}) - \hat{\delta}_x \\ \ddot{y_d} + K_p(y_d - \hat{y}) + K_d(\dot{y_d} - \dot{\hat{y}}) - \hat{\delta}_y \end{bmatrix} \tag{5}$$

For the robot to follow dynamics governed by the controller Eq. 5, we need to command f and also make sure that the robot is oriented correctly. Thus we infer the longitudnal force input and desired orientation like:

$$f = \sqrt{(f \cdot sin(\theta))^2 + (f \cdot cos(\theta))^2} \tag{6}$$

And subsequently θ_d control ref for yawing controller is:

$$\theta_d = atan2(f \cdot sin(\theta), f \cdot cos(\theta)) \tag{7}$$

3.3.2 Yaw controller

Using this calculated θ_d, we can design a yawing controller which is fast enough to correct the yaw orientation of the robot based on desired trajectory.

$$M = Kp \cdot sin(\theta_d - \hat{\theta}) - Kd \cdot \dot{\hat{\theta}} - \hat{\delta}_\theta \tag{8}$$

3.4 Observer Design

Since we have essentially 3 chains of integrators, we can directly use the example from [3] For the planar case we have 6 state variables as shown by Eq. 9

$$X = \begin{bmatrix} x \\ \dot{x} \\ y \\ \dot{y} \\ \theta \\ \dot{\theta} \end{bmatrix} \tag{9}$$

And to develop observer assume we only can infer the position value of each chain, thus outputs becomes Eq. 10

$$Y = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \tag{10}$$

Thus the 3-integrator chains become, Eq. 11, 12, 13

$$\begin{bmatrix} \dot{\hat{X}}_1 \\ \dot{\hat{X}}_2 \\ \dot{\hat{\delta}}_x \end{bmatrix} = \begin{bmatrix} \hat{X}_2 + (\alpha_1/\epsilon) \cdot (Y_1 - X_1) \\ \hat{\delta}_x + f \cdot cos(\theta) + (\alpha_2/\epsilon^2) \cdot (Y_1 - X_1) \\ (\alpha_3/\epsilon^3) \cdot (Y_1 - X_1) \end{bmatrix} \tag{11}$$

$$\begin{bmatrix} \dot{\hat{X}}_3 \\ \dot{\hat{X}}_4 \\ \dot{\hat{\delta}}_y \end{bmatrix} = \begin{bmatrix} \hat{X}_4 + (\alpha_1/\epsilon) \cdot (Y_2 - X_3) \\ \hat{\delta}_y + f \cdot sin(\theta) + (\alpha_2/\epsilon^2) \cdot (Y_2 - X_3) \\ (\alpha_3/\epsilon^3) \cdot (Y_2 - X_3) \end{bmatrix} \tag{12}$$

$$\begin{bmatrix} \dot{\hat{X}}_5 \\ \dot{\hat{X}}_6 \\ \dot{\hat{\delta}}_\theta \end{bmatrix} = \begin{bmatrix} \hat{X}_6 + (\alpha_1/\epsilon) \cdot (Y_3 - X_5) \\ \hat{\delta}_\theta + M + (\alpha_2/\epsilon^2) \cdot (Y_3 - X_5) \\ (\alpha_3/\epsilon^3) \cdot (Y_3 - X_5) \end{bmatrix} \tag{13}$$

4 Results

4.1 Simulator

A simulator using MATLAB Simulink was developed with Unreal Engine Plugin to effectively visualize the dynamics of the 3D robot. A screenshot of the front end of the visualizer is shown in Fig. 3 In the subseqent sections for simplicity we assume a simplified planar case as discussed in 3.2

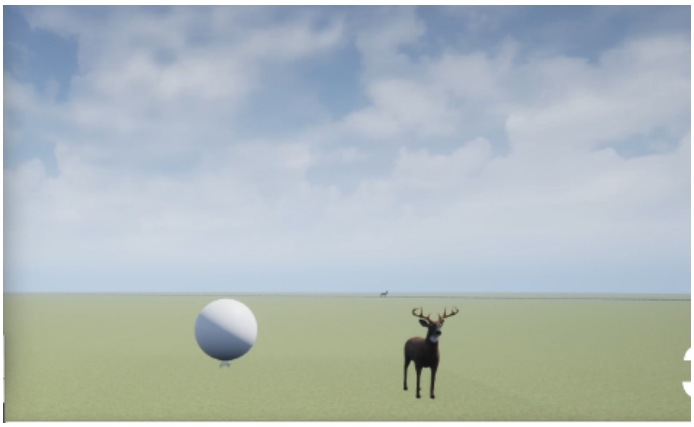

Figure 3: Simulation Visualizer using Unreal Engine(UE 5.3)

4.2 Planar Controller

Assume we apply a constant disturbance unknown to observer of the form Eq.14

$$\begin{bmatrix} \delta_x \\ \delta_y \\ \delta_\theta \end{bmatrix} = \begin{bmatrix} 0.4 \\ -0.3 \\ 0.25 \end{bmatrix} \tag{14}$$

4.3 Trajectory Tracking

To get a baseline of the improvement provided by EHGO, the same controller was applied to simmilar controller without the disturbance estimation and rejection. Fig. 4 shows the controller without disturbance rejection and Fig. 5 shows trajectory tracking performance with disturbance rejection.

4.4 Disturbance Estimates

The disturbance estimated by the observer over samples can be inferred from Fig. 6

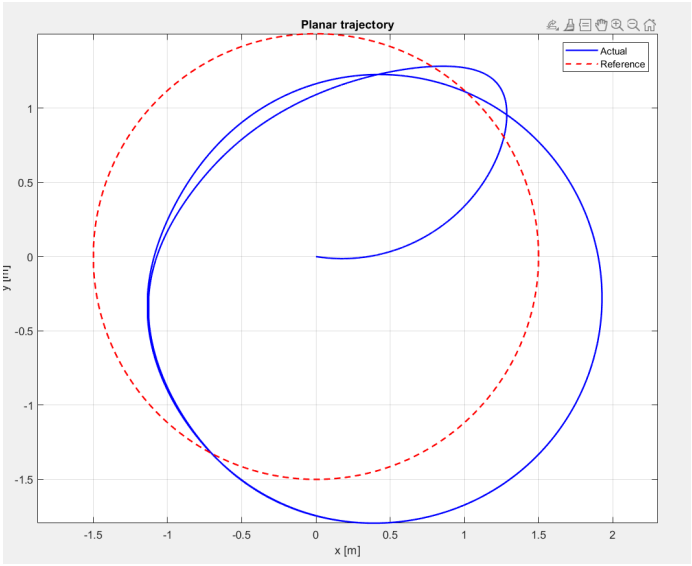

Figure 4: Trajectory tracking With No Disturbance Rejection

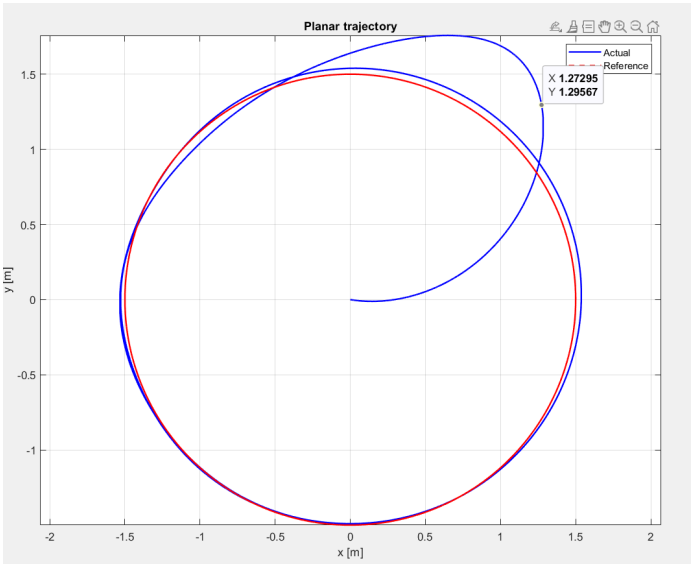

Figure 5: Trajectory Tracking Rejecting Disturbance

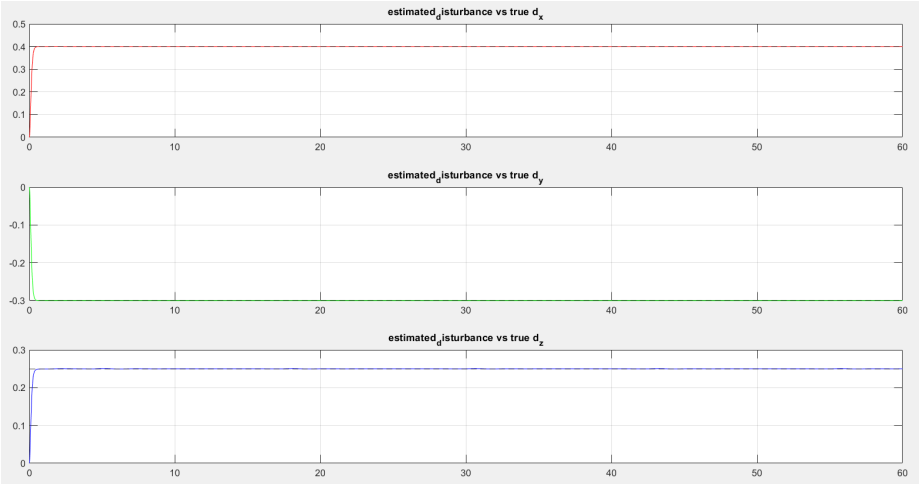

Figure 6: Estimated Disturbances

A MATLAB Code

Code as listed in Listing 1 used for the simulation of the planar EHGO case.

```
clear; clc; close all;
%% parameters
acc_lim  = 0.25;       % SI translation acceleration limit
alph_lim = 5;       % SI yaw angular acceleration limit
mass =  0.148;
I = 0.004;

L             = [3 3 1]';    % observer gains (copied from
    magazine)
eps           = 0.1;         % high gain eps
kp            = 1; kd = 1.5;  % PD for translation
kp_t          = 12; kd_t = 4;% PD for yaw

%simulation step
dt = 0.001;
T_circle = 60;
T  = T_circle*5;
N = T/dt;
t = (0:N-1)*dt;

%% STATE VECTORS
% plant       x = [p_x v_x  p_y v_y  theta omega]'
x   = zeros(6,1);

% true (unknown) disturbances  d = [d_x d_y d_theta]'
d   = [ 0.4;  -0.3;  0.25];
```

```
26
27  % EHGO augmented states  (three copies/ chain of integrators
        )
28  zx = zeros(3,1);
29  zy = zeros(3,1);
30  zt = zeros(3,1);
31
32  % logging
33  xlog = zeros(6,N);
34  zlog = zeros(9,N);
35  rlog = zeros(3,N);
36
37  %% SIMULATION LOOP
38  for k = 1:N
39      tk = t(k);
40      omega = 2*pi / T_circle;    % angular frequency [rad/s]
41
42      % reference trajectory
43      pr  = 1.5 * [cos(omega*tk);  sin(omega*tk)];
44      vr  = 1.5 * omega * [-sin(omega*tk);  cos(omega*tk)];
45      ar  = -1.5 * omega^2 * [cos(omega*tk); sin(omega*tk)];
46
47      % position and  velocity errors
48      e   = pr - [zx(1); zy(1)];
49      de  = vr - [zx(2); zy(2)];
50      dist_rej = 1;
51
52      % longitudnal controller
53      ax_cmd = ar(1) + kp*e(1) + kd*de(1) - dist_rej.*zx(3);
54      ay_cmd = ar(2) + kp*e(2) + kd*de(2) - dist_rej.*zy(3);
55      ax_cmd = max(min(ax_cmd, acc_lim),  -acc_lim);
56      ay_cmd = max(min(ay_cmd, acc_lim),  -acc_lim);
57      thetad = atan2(ay_cmd,ax_cmd);
58
59      % yaw_controller
60      et  = atan2( sin(thetad - zt(1)), cos(thetad - zt(1)));
61      thetadd_cmd   = (kp_t*sin(et) - kd_t*zt(2) - dist_rej.*
        zt(3));
62      thetadd_cmd = max(min(thetadd_cmd, alph_lim), -alph_lim)
        ;
63
64      % convert to force / torque AFTER the saturation
        -------------
65      F = mass * sqrt(ax_cmd^2 + ay_cmd^2);    % [N]
66      M = I    * thetadd_cmd;                  % [Nm]
67
68      x_dot = [ x(2);
69                (F/mass)*cos(x(5)) + d(1)*sin(0.05*tk);
70                x(4);
71                (F/mass)*sin(x(5)) + d(2)*sin(0.05*tk);
```

```
                x(6);
                (M/I)       + d(3) ];

    x = x + dt*x_dot;

    % EHGO UPDATES  (include the commanded input!)
    zx = ehgo_step(zx, x(1), F*cos(zt(1))/mass, L, eps, dt);
    zy = ehgo_step(zy, x(3), F*sin(zt(1))/mass, L, eps, dt);
    zt = ehgo_step(zt, x(5), (M/I), L, eps, dt);

    % LOG
    xlog(:,k) = x;
    zlog(:,k) = [zx; zy; zt];
    rlog(:,k) = [pr; thetad];
end

%% plots
figure;
plot(xlog(1,:),xlog(3,:),'b','LineWidth',1.4);
hold on
plot(rlog(1,:),rlog(2,:),'r--','LineWidth',1.4);
axis equal;
grid on;
xlabel('x [m]');
ylabel('y [m]');
legend('Actual','Reference');
title('Planar trajectory');

time = t;

figure;
subplot(3,1,1);
plot(time, zlog(3,:) ,'r');
hold on;
yline(d(1),'k--');
title('estimated_disturbance vs true d_x');
grid on;

subplot(3,1,2);
plot(time, zlog(6,:) ,'g');
hold on;
yline(d(2),'k--');
title('estimated_disturbance vs true d_y');
grid on

subplot(3,1,3);
plot(time, zlog(9,:) ,'b');
hold on;
yline(d(3),'k--');
title('estimated_disturbance vs true d_z');
```

```
grid on

figure;
plot(time,zlog(2,:))
hold on;
plot(time,xlog(2,:))
hold off;
grid on;
legend("Estimate","Truth")
title("x-speed")

figure;
plot(time,zlog(5,:))
hold on;
plot(time,xlog(4,:))
hold off;
grid on;
legend("Estimate","Truth")
title("y-speed")

figure;
plot(time,zlog(7,:))
hold on;
plot(time,xlog(5,:))
hold off;
grid on;
legend("Estimate","Truth")
title("theta comp")

figure;
plot(time,atan2(sin(rlog(3,:)),cos(rlog(3,:))))
hold on;
plot(time,atan2(sin(xlog(5,:)),cos(xlog(5,:))))
hold off;
grid on;
legend("Ref","Truth")
title("Yaw Tracking")

figure;
plot(time,rlog(1,:))
hold on;
plot(time,xlog(1,:))
hold off;
grid on;
legend("Ref","Truth")
title("X Tracking")

figure;
plot(time,rlog(2,:))
hold on;
```

```
plot(time,xlog(3,:))
hold off;
grid on;
legend("Ref","Truth")
title("Y Tracking")

%% EHGO HELPING FUNCTION
function z = ehgo_step(z, y, u, L, eps, dt)
% One Euler step of the standard 3-state EHGO for
    e   = y - z(1);
    z1d =                  z(2) + (L(1)/eps)*e;
    z2d =           u   + z(3) + (L(2)/eps^2)*e;
    z3d =                       (L(3)/eps^3)*e;
    z   = z + dt*[z1d; z2d; z3d];
end
```

Listing 1: MATLAB Planar Simulation Code

References

[1] H. Cheng, Z. Sha, Y. Zhu, and F. Zhang, "RGBlimp: Robotic gliding blimp - design, modeling, development, and aerodynamics analysis," *IEEE Robot. Autom. Lett.*, pp. 1–8, 2023.

[2] C. J. Boss and V. Srivastava, "A high-gain observer approach to robust trajectory estimation and tracking for a multirotor unmanned aerial vehicle," en, *J. Dyn. Syst. Meas. Control*, vol. 147, no. 1, pp. 1–16, Jan. 2025.

[3] H. K. Khalil, "High-gain observers in feedback control: Application to permanent magnet synchronous motors," *IEEE Control Systems Magazine*, vol. 37, no. 3, pp. 25–41, 2017. DOI: 10.1109/MCS.2017.2674438.