Autonomous Blimp with Non-Linear Controls.

***Abstract*— The idea for an autonomous blimp began this summer as a potential independent study and research project for the Morphable BioRobotics Lab at BU. This is meant as a foundation for future work on lighter than air vehicles utilizing the jet propulsion system developed by Pranav Sultania with help from Liam ThirtyAcre, myself, and others at the same lab. The current iteration of the blimp is designed to use propellers mounted on the outside of the blimp with the idea that they can be later replaced by the jet.**

I. Dynamics

A. Introduction

Before deriving the equations of motion for the blimp I will specify how we will define the state space and general nomenclature. Calculating the dynamics efficiently requires switching reference frames and creating a solid notation system is a key first step. The character X will be used to refer to the systems entire state space. This includes: position (x), orientation(ϕ), velocity(v) and angular velocity($\dot{\phi}$) in that order. Position is measured from the blimps center of mass and the center of the motion capture room is set as zero. For the purposes of effectively calculating the controls all X will always refer to the global reference frame. We will also have to use the rotation matrix extensively which switches from the global to the local reference frame. To switch back we can use the transpose. All of these are ordered in x,y,z such that x is forward, y is left and is up. The blimp is controlled by four bi-directional motors. The controls are written as $\vec{u}$ (4x1) with each value between -1 and 1 representing the portion of the maximum thrust. It is also important to note that the drag equations use of the .* tool in MATLAB. There isn't a great way to represent this in Latex so I've just written it as .*.

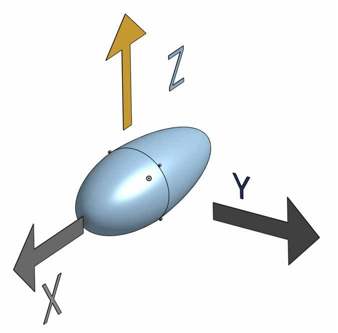

Fig. 1: Orientation

As of now it seems that manufacturing imprecision makes accurate modeling of the motion of the blimp beyond the scope of a one semester project. However, work done in anticipation is still a solid basis for understanding the dynamics and controls of a future blimp. The drag coefficients for all of the below equations were calculated using COMSOL simulations. Regardless of difficulties in manufacture these should be seen as estimates and would have to be updated experimentally.

B. Net Force

The net force measured on the blimp can be calculated as the sum of the drag and the thrust from the jets.

$$F_{net} = F_{thrust} + F_{drag} \tag{1}$$

1) Drag: The equation for drag in a single direction is relatively simple $F_d = -.5 * v^2 * A * C_d * \rho * sign(v)$ where ρ is the density of the fluid and A is the cross sectional area. Technically the drag coefficient changes based on the Reynolds number, but within our expected range of velocities the changes are minor. Our blimp will have velocity components in three different directions, with three different drag coefficients and three different cross sectional areas. I also need to account for the orientation of the blimp. While annoying to do by hand this can run extremely fast in code. By first calculating the rotation matrix we can decompose the global velocity into the local components. We can calculate drag as:

$$\begin{aligned} v_{local} &= Rot * v \\ drag_{local} &= -.5 * v_{local}.^2 * .C_d. * A * \rho. * sign(localv) \\ drag_{global} &= Rot' * drag_{local} \end{aligned} \tag{2}$$

By defining C_d and A as vectors and using the $.^2$ function (which is default in python) we can significantly decrease the compute requirements which is essential for running the controls.

2) Thrust: This is the most straightforward of our dynamics equations.

$$F_{prop} = Rot' * [sum(maxthrust * u); 0; 0] \tag{3}$$

C. Torque

Torque is significantly more difficult. Not only are there more factors that lead to torque, but each is also relatively complex. Net torque is:

$$\tau_{net} = \tau_{prop} + \tau_c + \tau_{drag} + \tau_b \tag{4}$$

1) Torque From Propellers: As the equation for torque is r x F we can start in the local reference frame. r is the location of each jet with regard to the blimps center of mass.

$$\tau_{prop} = Rot' * [\sum_{n=1}^{4}(\vec{r_i} \times (u_i * maxthrust, 0, 0] \tag{5}$$

2) Angular Velocity Damping: The angular velocity of the blimp will also induce torque on the blimp. Calculating this precisely in simulation is difficult so the relevant constants will need to be updated through experimentation. In COMSOL this as calculated using a surface integral. Measuring torque in along the x axis due to $\dot{\phi}_1$ (angular velocity along the x axis):

$$\tau_c = \oint pressure_e 1 * ydS. \tag{6}$$

Where y = 0 is based on the COM. I was unsure exactly how the torque would correlate to angular velocity, particularly whether it would be a multiple of $\dot{\phi}$ or $\dot{\phi}^2$, but based on the simulations it seems that it based on $\dot{\phi}$. Our total torque due to angular velocity is therefore:

$$\begin{aligned} \dot{\phi}_{local} &= Rot * \dot{\phi}_{global} \\ \tau c_{local} &= -\dot{\phi}_{local} . * \vec{C_d} \\ \tau c_{global} &= Rot' * \tau_{local} \end{aligned} \tag{7}$$

3) Drag Induced Torque: The torque induced by drag is requires similar simulation techniques and will also have to be updated experimentally. Initially I was able to dismiss it by assuming that the COM would be the same as the center of volume (COV), but this design is impractical (unstable with regard to roll and adds weight). One key difference here is that velocity along one axis can result in torque along the other two axis. For this scenario each flow direction has it's own vector of resultant torque that is linearly related to the velocity squared. Right now I am calculating the torque as:

$$\begin{aligned} v_{local} &= Rot * v_{global} \\ \tau_{drag} &= \sum_{i=1}^{3} v_i^2 . * \vec{Td_j} \end{aligned} \tag{8}$$

Where Td is a 3x3 matrix and Td_{ij} is the torque induced along the the i axis in by velocity in the j direction. This is the portion I am the least confident in and will likely require significant changes based on experimentation. Based on simulations this torque will be three orders of magnitude smaller than other torques. As it requires the most compute as well I've decided to ignore it for now. If the dynamics are inaccurate we may add it back, but currently it is not deemed significant.

4) Restorative Buoyancy: While intimidating this is relatively simple to calculate.

$$\begin{aligned} F_b &= [0, 0, Vol * (\rho_{air} - \rho_{liftingagent}) * g] \\ \tau_b &= Rot' * [Rot * COV \times F_b] \end{aligned} \tag{9}$$

In a more intuitive sense, the blimp is stable when the center of buoyancy is above the center of mass. This can be thought of as a boat swaying in the water. If the boat starts tilted it will oscillate until the COV is at or above the COM.

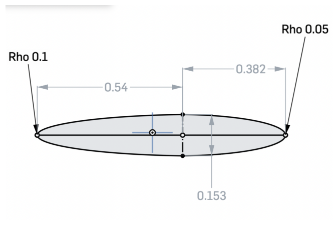

Fig. 2: Relevant Dimensions

II. Localization

The weight of the sensors make most kinds of localization prohibitive. IMUs (Inertial Measurement Units) have low weight, but are prone to drift. However, the envelop of the blimp is also prone to deformation which would combined with the drift in the IMU would lead to inaccurate results over time. The other alternative for onboard sensing is visual odometry. The downsides here are cost and compute. As the controls already have a high compute we decided to use the motion capture room in RASTIC.

Using the motion capture also has a number of challenges and downsides. RASTIC has lightweight fiducial stickers that we can attach to the blimp. As the envelope is reflective we need to spray paint the blimp which does add a non-negligible amount of weight. If we attach three fiducial to the blimp we can identify the orientation of the balloon at any time. We will attach them to the front F_f, back F_b and top F_t of the envelope.

Given the length of the envelope L and the height H we can identify which of the two fiducial are at the ends by calculating which two fiducial are L distance apart. ϕ_y and ϕ_z can be calculated from the unit vector.

III. Controls

A. Background

We came into this project completely green in the world of controls. After viewing the MIT courses online I bounced around between exploring a few different algorithms (PID, LQR, and HJB) but ultimately decided on MPC as a happy balance of non-linear optimal control without the compute requirements of HJB. The actual implementation is relatively simple with the main difficulty with the main difficulties in debugging the dynamics equations and properly setting weights to account for the non-linear dynamics. Using the equations of motion in I the algorithm finds the optimal path subject to those dynamics using interior point optimization. The true challenge turned out to be defining cost equation.

$$C = \int_{t_0}^{t_f} (l(x, u) + \Delta u' R * \Delta u) + m(x_f) \tag{10}$$

R is a diagonal matrix that acts to penalizes changing the controls. This leads to less drastic changes in the controls more robust paths less subject to "jitter." The most straightforward solution is to write the Langrage and Meyer terms as:

$$l(X,u) = \vec{weights}_l * (X_{target} - X)^2$$
$$m(X_f) = \vec{weights}_m * (X_{target} - X_f)^2 \quad (11)$$

This does run into problems where the ϕ values need to be corrected. For example if our target ϕ_1 = 0 and our calculated $\phi_1 = 2 * \pi$ This would have a heavy penalty. Each loop the ϕ values are fixed with a modulus so that the angle is never above 2π and I just made a local function "smallest angle" that returns the correct error given the current and target angle. $Error = min(abs((\phi_{target} - \phi_{current}), (\phi_{target} - \phi_{current} - 2 * \pi)))$ This only needs to be applied to ϕ values.

After some experimentation I decided to remove the values forϕ, $\dot{\phi}$, and $\dot{x}$ from the Lagrange term. Since the blimp can only accelerate forward complex paths can require that the blimp turn of accelerate off of the targets. For example if the blimp was given the target of x = [1,1,0] and ϕ = [0,0,0] the ϕ term would be penalized if the blimp turned in order to translate along the Y axis.

B. *Results*

There are a number of factors that go into the performance of the controls are for it to run in real time:

- Complexity of task
- Hardware
- Complexity of Dynamics
- Complexity of Cost Equation
- Number of steps and step-size

The biggest surprise running this was how much the compute cost changed based on the complexity of the task. For example a single command such as moving forwards 1 m and stopping took half as long to compute (23.15 sec) compared to a more complicate maneuver that required drifting and rotation along multiple axis (42.24 sec.) In implementation this will run on the SCC available to students at RASTIC which runs orders of magnitudes faster than the jupyter notebook on my old laptop.

C. *Scenarios*

After some basic testing I was curious to see how the controls algorithm would perform with more complex tasks. To clarify all of the below control tests ran in simulation with an initial condition of X = $\vec{0}$

1) Sideways movement: For this run the blimp was given the target of x = [0,4,0] and zeros for all other DOF. Initially the blimp didn't move, but it succeeded when I changed the target ϕ to .001. I don't know enough about the internal controls in *do_mpc*, but this is probably due to some kind of local minima or equivalent. There are two paths with a completely equal cost, but by adding a small deviation we can avoid the symmetry.

2) Peel out: Another scenario I was curious about was setting all target DOF to zero except for dx[0] = 2. More intuitively this is the blimp trying to peel out. This is a difficult path for a number of reasons. While the blimp can reach a this speed, its acceleration is low. A human would intuitively back up or try to move in a circle to build up speed, but these

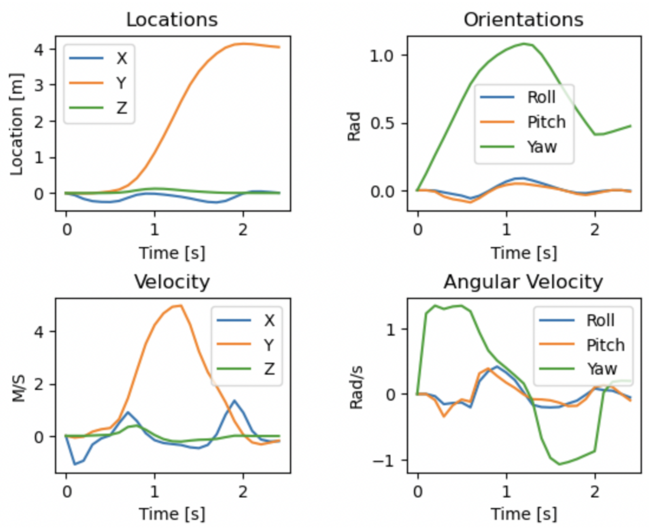

Fig. 3: Translate Left

are both penalized heavily in the cost function. I was not able to successfully update the controls for it to perform this task. There are a few methods that would probably work including generally increasing the weights on the Meyer function, but this requires that the n_steps value be increased or the cost will grow exponentially for the early calculations. I decided to ignore the compute costs and checked what happened if the n_steps increased. This compute cost was extremely high (took 167.22 sec), but didn't seem to improve the performance. I was finally able to get it to back up in order to reach the target eventually, by setting the Lagrange term to zero and with the increased n_steps it was able to reverse and try to build speed, but it seems that the system ran into other problems.

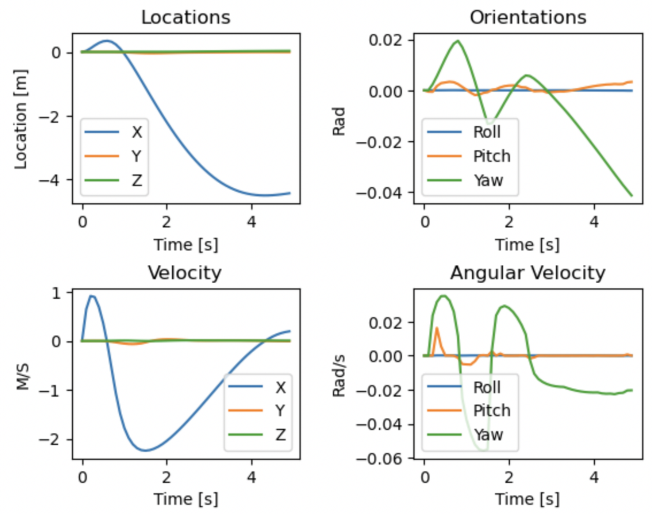

Fig. 4: Peel Out

My current theory is that the each iteration is in conflict with the previous iterations. Each time the MPC is called to calculate the optimal path it returns a path with the final step as close to the target state as possible. The problem is that the final step is also pushed into the future by one time increment. As the target state is inherently unstable this means that each call of MPC is predicting a different path. More succinctly even if the state space was equal to the target space or close the cost function would return based on the final state and only the finial state. This could be improved in the future by setting a target time as well and updating the n_steps value to accommodate this.

3) Banking Turn: A key goal personally when designing this algorithm was to achieve a banking turn. Because of the different profiles of the blimp it is able to generate velocity through drag by changing it's orientation. This is counter-intuitive mathematically, but by looking at the equations for drag and specifically the role of the rotation matrices it becomes clear. If the blimp has a forward velocity dx = [1,0,0] and the target is to translate in the positive y direction the blimp can change it's yaw and the resultant drag in the local y will result in a positive global y velocity. Based on

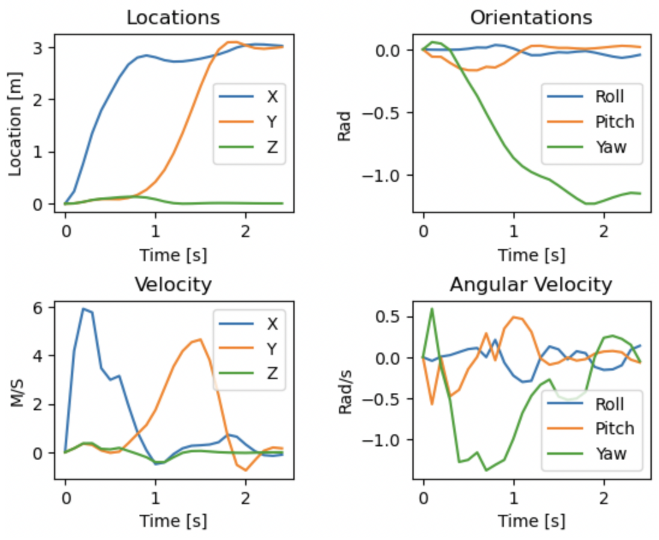

Fig. 5: Banking Turn

III-C.2 I decided to remove the derivatives form the target function. Unstable targets have consistently caused problems with the MPC. Based on these results I consider the goal of a banking turn achieved!

4) Unstable Equilibrium: The question of unstable states in III-C.2 I was curious to see how the blimp could turn itself upside down. As the double pendulum is a classic problem in controls it seems like a fun test to see how the controls perform. In reality the instability of the system in three dimensions and the imprecision inherent in blimp construction would likely make this impossible, but I was curious to see how it would perform in simulation. I set the target as x = [0,0,1] and $\phi = [0, \pi, 0]$

This is largely what I expected and shows why quaternions can be better than Euler angles. The weight for ϕ_x is 0 because the blimp has no control over it's roll. However, since the orientation is not uniquely defined by Euler angels

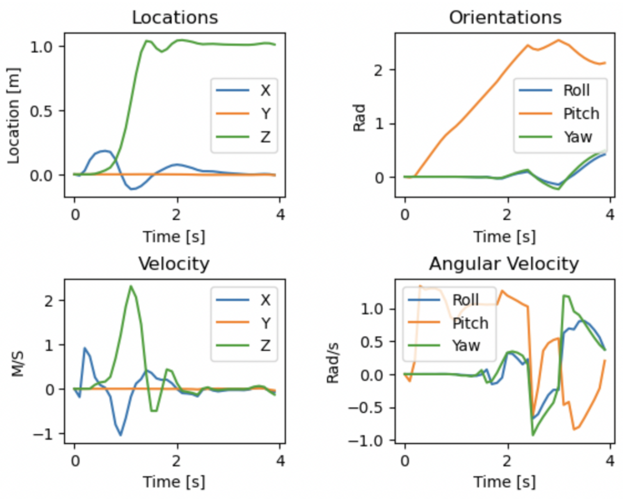

Fig. 6: Unstable Equilibrium

and it can reach singularities. For example if ϕ = $[\pi, \pi, 0]$ is a stable equilibrium and the cost function will return as 0.

IV. Adapting the Jet for Air

A. Disclaimer

As the paper this is based on is not yet published and not my work I will not divulge details and will assume a working knowledge of the technology.

B. General Construction

The weight of the jet is the most straightforward problem in a lighter than air vehicle. Helium and hydrogen both have around a 1 gram per liter ($1kg/m^3$). Depending on the source of lifting gas there is also the concern of purity. The source we found claimed to be above 80% pure by volume, but there are likely better sources as ours was from target and intended for party balloons. Increasing the size of the blimp is always an option, but this will increase the drag force and more importantly increase the cost of each inflation. Helium is not cheap and hydrogen comes with it's own challenges IV-B.1. There are a few components that we can easily reduce the weight for. The solenoid is a prime candidate. Using resin embedded carbon fiber would actually increase the efficiency of the jet in water as well. Not only is it thinner which would improve the performance of the magnets, but is has a much lower friction coefficient than the current resin printed solenoid. Using a heat set resin would also allow us to run the jets in air without worrying about melting the resin or loctite which was a serious problem during testing. The next obvious steps are just changing components (magnets, fewer copper wire layers) which would reduce the efficiency, but may be required for lighter than air flight.

1) Hydrogen: Electrolysis is an unlimited source of hydrogen, but a setup that could safety collect the volume required is it's own difficult project. I also called and emailed industrial hydrogen suppliers, but they were unwilling to sell in the small quantities that we were looking for. As another difficulty can't be shipped normally since it's highly flammable so it is mostly transported in dedicated vehicles. Even if we managed to create a source the blimp would be a serious danger if it used hydrogen. The Hindenburg disaster was as much the result of the flammable doping on the envelope, aluminum powder specifically, but Mylar also has aluminum powder on it. Hydrogen flames are also invisible and not particularly hot so it's possible we would not know the blimp is on fire until serious consequences are unavoidable. In short weight is going to be the main limiting factor for any lighter than air vehicle. The buoyancy of the blimp is straightforward mathematically $F_b = (\rho_{air} - \rho_{liftingGas}) * g * V$ so while hydrogen is half the weight of helium the change in lifting force is only .09 grams/liter. In short hydrogen is a difficult option with serious safety concerns. The only benefit (which is substantial) is the reduced cost and saving a non-renewable resource. The global helium shortage is a non-trivial problem with no solution on the horizon. Talking with Kenn Sebesta we managed to set up a hydrogen blimp protocol he was confident we could use in the RASTIC motion capture room. A lot of it would need to be re-worked based on the jet technology though. The high ignition temperature of hydrogen (535° C) meant that our propellers would be relatively safe, but experimenting with the jets in air we burned loctite within a few seconds which is already at least 250° C.

C. Volume

Assuming the fluid flow will be similar in air and water the force from the jet in air will be $\sim$ 1/1000th of the force in water. Increasing the frequency from 3.3 hz to 50-75hz will help somewhat, but that is still an upper limit of $\sim 7.5\%$ of the force in water. Placing the jets on the rear would allow us to increase the cross section without changing the C_d too much, but this has downsides. Unless we add weight to the front we'll only be able to fly vertically up. Thrust vectoring may be possible, but the restorative buoyancy is strong and would likely overpower the thrust and cause oscillation. It's still worth exploring, but it may not be viable. Placing the jets on the side would allow for better steering, but increasing the cross section would have serious effects on the C_d We could add a Mylar shroud to the front, and I am also curious about a non-circular jet as well as trying to adapt the one way valve to work with the shroud.

Retaining the current bullet shape also may run into problems with bouncy and imprecision. More specifically if the jet is mounted to the bottom of the blimp pulsing the jet may cause the jet to rotate and the resultant force vector may result in more rotation than translation. The best solution here would probably require a rigid skeleton attached to the blimp. We initially planned on a similar method for mounting the propellers, but realized that the weight would quickly become prohibitive. Some clever design with regard to the shape of the blimp could likely solve this, but

References