In-Situ Replanning for Autonomous Thermal Soaring

Nick Rypkema and Dehann Fourie

***Abstract*— Aerospace control paradigms classically separate inner loop control, catering for immediate stability dynamics, from guidance, which is concerned with way-point and flight progress management. Making an unpowered glider climb, so as to reach a specific way-point at a desired altitude, is an underactuated problem. We propose an in-situ replanning based control and trajectory optimization strategy to address the control aspect of centering a glider at an estimated thermal updraft location. This work emphasizes high fidelity control and trajectory planning of the glider rather than estimating thermal updraft positions, but feel this control approach is a natural compliment to integrate with atmospheric estimation process.**

I. Introduction

Humans pilots are able to fly efficient unpowered gliders, such as the one shown in fig. 1, over long distances tapping energy from atmospheric updrafts. These updrafts, known as thermals, are essentially columns of hot rising air and can be exploited by glider pilots in order to gain altitude. Other energy sources are from winds blowing over obstructions, such as ridges or mountains, or wave-gliding. This work seeks to contribute towards long term autonomous glider soaring through trajectory optimization.

This paper is organized as follows: We present the approach to trajectory replanning and the assumptions made. Trajectory optimization, focused on circling the glider about an estimated thermal updraft position, is used to predict the glider flight path with high fidelity. Trajectory optimization is done with nonlinear convex optimization. Two control strategies are presented through which the glider flight path is maintained on the desired trajectory. The first is a classic cascaded type control, while the second is a work in progress L_2 gain optimized controller design. Implementation details and results, followed by analysis and conclusion follow.

A. Motivations

We focus on thermal updrafts, but the work may be extended to other forms of updrafts as well, and limit the scope of work by assuming that the thermal location is known through some atmospheric estimation process. While this is a strong assumption, autonomous aircraft have sufficient sensor information to measure air currents, and a lot of work has been done on estimating updraft locations. We hope this in-situ replanning approach facilitates the control aspects of presumed autonomous soaring.

Furthermore, the dynamics of the aircraft are considered sufficiently predictable for the flight speeds in question. That said, we know from experience that huge energy savings can be had when flying near minimum sink rate. Trivial PID laws are insufficient to robustly stabilize vehicle dynamics in this condition. That said, our approach is motivated by easily understandable linear control strategies, however, we do use a few non-linear tricks to simplify and improve the control architecture robustness.

The authors are with the Massachusetts Institute of Technology, Cambridge, MA 02139, USA {rypkema,dehann}@mit.edu

Fig. 1. Screenshot of the ASK-13 glider model from the open source flight simulator, *Flightgear*, the vehicle used for this project.

This work is inspired by the MIT class 6.832 (Underactuated Robotics) and is motivated by advances made with software frameworks such as Drake [1].

B. Problem statement

An unpowered glider starts with some initial altitude at distance from a known thermal updraft. The glider must be able to repeatably center in the thermal updraft and climb to a designated altitude.

II. Approach

Current approaches to autonomous thermal soaring have focused primarily on the use of way-points during flight to the thermal, and the implementation of feedback rate-of-turn controllers using changes in rate of climb for thermal centering, as in [2], [3]. A disadvantage of such an approach is that the centering performance is very sensitive to tuning, and poorly tuned controllers can cause the glider to exit and lose the thermal prematurely.

In this work, we propose the use of trajectory optimization, using a simplified dynamic model of our chosen glider, the ASK-13 (approximated via experimental analysis). The use of a carefully formulated trajectory optimization allows us to autonomously guide the glider into a thermal, and have it automatically center and circle within the thermal, without the use of mode switching. In addition, by using a conservative dynamic model, trajectory optimization produces

inherently stable control inputs for the glider. Our platform for this work is flight simulator "Flightgear", which uses YASim to simulate the dynamics of the ASK-13 glider [5]. Our controller and trajectory optimization program interface with Flightgear. An example trajectory from our approach is shown in fig. 2.

1) Line Following: Although our approach produces a trajectory and the corresponding control inputs for the aircraft, note that since the exact dynamics of the glider are unknown and have been approximated by our simplified model in the trajectory optimization, these control inputs will not necessarily result in the simulated glider following the desired trajectory; in fact, simply using the produced control inputs causes the glider to approximately mimic the trajectory, often with an offset. In order to remove this offset and correctly follow the desired trajectory, we have implemented a feed-forward controller which balances the trajectory control inputs with inputs based on the distance from, and angle of attack toward the trajectory, to create a line-following controller.

2) Replanning: An issue to consider in our approach is the computational cost of trajectory optimization; as the complexity of the non-linear optimization grows, the time it takes to compute a solution increases. In our case, as we increase the length of the trajectory (the number of knot points), the longer it takes to produce a trajectory; however, the shorter the trajectory, the less it is able to mimic our desired behaviours of centering and circling. In addition, in order to successfully control the glider, it is critical that a new trajectory is computed before the vehicle completes its current trajectory, i.e. we require dynamic, in-situ trajectory replanning. During the course of this work, we found that 35 knot-points, spaced 0.6 seconds apart, would result in a quick enough computation time, and a long enough trajectory to balance this trade-off. As the glider reaches the 30th knot point of the current trajectory, and crosses an imaginary 'wall' perpendicular to the trajectory (shown in purple in fig. 2 and fig. 5), it is given a new trajectory (computed during traversal of the current trajectory), whose initial conditions are given by the state of the current trajectory at that knot point.

Even though it was found in the vast majority of cases that the next trajectory is computed 'in time', in rare cases the computational time can be too long. When this occurs, our system checks for divergence from the trajectory using a distance/time metric, and recomputes a new trajectory using the current state of the glider as initial conditions.

A. Assumptions

In order to simplify the problem tackled in this work, a number of assumptions were made. It is assumed that the location of the thermal is known (i.e. we do not address the issue of searching for thermals), and the thermal location is static. In addition, to simplify the trajectory optimization formulation, we work in a thermal-centric frame of reference (i.e. the thermal is located at (0,0)). However, future work should enable more capable adjustments for centering onto an unknown thermal.

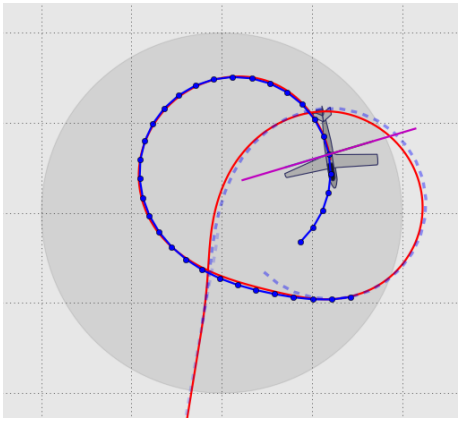

Fig. 2. Visualized output of the approach presented in this paper.

III. Trajectory Optimization

Trajectory optimization requires an understanding of feasible glider dynamics. Using constrained nonlinear optimization, we construct an objective to guide the glider towards a desired goal location, enter into a circular flight pattern and constantly adjust the pattern to properly center the loop about the desired goal point. This approach allows us to dynamically adjust the desired goal point around which the glider will circle. While not yet implemented, the formulation allows smooth trajectory transitions as the estimate of where the thermal core is improved through incremental adjustments.

We construct an optimization objective based on predicted vehicle locations, which may be influenced by certain control inputs. The vehicle dynamics are enforced through first order discrete state constraints. We model closed loop center of mass dynamics, avoiding many of the possible complexities in high fidelity modeling of the vehicle. The trajectory states are x, y position, heading ψ, bank angle ϕ and scaled heading rate $\dot{\psi}'$

$$\mathbf{x} = \begin{bmatrix} x & y & \psi & \phi & \dot{\psi}' \end{bmatrix}^T, \quad \mathbf{u} = \begin{bmatrix} {}^b v_x & \text{aileron} \end{bmatrix}^T \tag{1}$$

Furthermore, the trajectory is influenced by two inputs, body forward velocity ${}^b v_x$ and aileron input commands. We need only consider aileron set points to change aircraft heading. Vehicle velocity is maintained at a defined set point by the controller, and is used to force time evolution during trajectory optimization – an aircraft must keep moving.

Next we must model how vehicle state evolves over time, and assume a vector mapping $\mathbf{f}$ exists:

$$\frac{\partial}{\partial t}\mathbf{x} = \mathbf{f}\left(\mathbf{x}, \mathbf{u}\right) \tag{2}$$

We linearize the dynamics about the nominal operating

point and denote the perturbed state and input as $\bar{\mathbf{x}}$ and $\bar{\mathbf{u}}$:

$$\frac{\partial}{\partial t}\bar{\mathbf{x}} \approx \mathbf{A}\bar{\mathbf{x}} + \mathbf{B}\bar{\mathbf{u}}$$

$$\mathbf{A} = \bigtriangledown_{\mathbf{x}} \mathbf{f} \approx \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & m_{\phi,\dot{\psi}} \\ 0 & 0 & 0 & \tau_\phi & 0 \\ 0 & 0 & 0 & \tau_\psi & -\tau_\psi \end{bmatrix}$$

$$\mathbf{B} = \bigtriangledown_{\mathbf{u}} \mathbf{f} \approx \begin{bmatrix} \cos\psi & 0 \\ \sin\psi & 0 \\ 0 & 0 \\ 0 & -\tau_\phi \\ 0 & 0 \end{bmatrix} \tag{3}$$

We will assume this first order approximation is sufficient for trajectory optimization, even though capable non-linear convex optimization methods exist. The main motivation for linearizing the center of mass dynamics stems from parameter identification, matching the model to the test vehicle.

We introduce the non-zero terms in $\mathbf{A}$ and $\mathbf{B}$ matrices as follows. Top down two dimensional x, y position of the glider is propelled via constant velocity input in $u_1 = {}^b v_x$. The heading ψ results in a single rotational degree of freedom, which maps forward velocity onto $\dot{x}, \dot{y}$ according to cos and sin terms – top left in $\mathbf{B}$.

Next we assume a first order transient delay between aileron input and change in roll rate (Laplace transform):

$$\frac{\Phi}{\text{Aileron}} \approx m_{\phi,\dot{\psi}} \left(\frac{\tau_\phi}{s + \tau_\phi} \right) \tag{4}$$

which, when converted to state space and included into the transition model, results in the two τ_ϕ terms. Similarly, we introduce a first order transient delay between roll angle ϕ and rate of change in heading $\dot{\psi}$, using characteristic time τ_ψ and scaled by $m_{\phi,\dot{\psi}}$.

The last term, $m_{\phi,\dot{\psi}}$, is just the steady state gradient between roll angle ϕ and rate of heading change $\dot{\psi}$. The steady state gradient $m_{\phi,\dot{\psi}}$ was computed by linear least squares fit to experimental data, as shown in fig. 3.

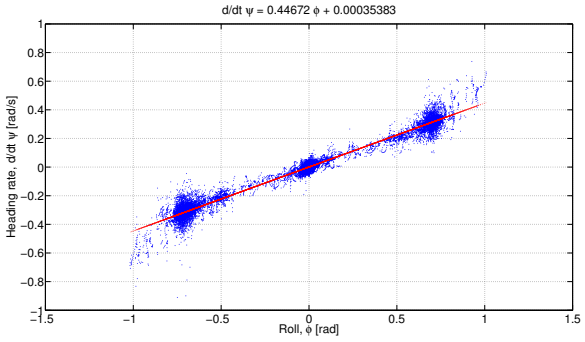

Fig. 3. Red line shows regression of roll to heading rate steady state gradient, given a long data sequence with many continuous turns.

A. Center of mass dynamics gradients

We find these gradients using linear least squares fit, and capturing data with basic controllers feeding input and the existing *ASK-13, Flightgear/YAsim* output:

$$\frac{\partial f_4}{\partial \phi} \approx -\frac{\partial f_4}{\partial u_2} \approx \tau_\phi = 0.5 \; rad/s \quad \frac{\partial f_3}{\partial \dot{\psi}'} = m_{\phi,\dot{\psi}} = 0.447$$
$$\frac{\partial f_5}{\partial \phi} \approx -\frac{\partial f_5}{\partial \dot{\psi}'} \approx \tau_\psi = 1.5 \; rad/s \tag{5}$$

Visual interpretation of results, for example fig. 2 and fig. 5, suggest the center of mass dynamics model does a reasonable job in predicting glider motion. Remember this approximate equality between seemingly separate gradients in the Jacobian stems from the first order delay argument made above. It is also important to note that for this implementation we are interested in the feasible closed loop dynamics of the vehicle and not full out model predictive type control strategies.

We would argue this approach of setting up center of mass dynamics does not follow the conventional kinematics derivations, but is sufficient for the task at hand.

B. Setting up the convex optimization problem

We predict the glider trajectory as a sequence of discrete knot points and must therefore convert the continuous dynamics to a pre-integrated discrete dynamics equivalent, also known as the direct transcription method. Using symbolic expressions for time varying nonlinear quantities in the $\mathbf{B}$ matrix, we solve for arbitrary discrete time solutions using matrix exponential as approximated closed form solution of the first order state space model:

$$\begin{matrix} \mathbf{A}_d = \mathbf{M}_{d,11} \\ \mathbf{B}_d = \mathbf{M}_{d,12} \end{matrix} \quad \mathbf{M} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}, \quad \mathbf{M}_d = e^{\mathbf{M}\,\Delta t} \tag{6}$$

We find the matrices with parameters from section III-A:

$$\mathbf{A}_d = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0.083 & 0.178 \\ 0 & 0 & 0 & 0.741 & 0 \\ 0 & 0 & 0 & 0.501 & 0.407 \end{bmatrix}$$

$$\mathbf{B}_d = \begin{bmatrix} 0.822\cos\psi & 0 \\ 0.822\sin\psi & 0 \\ 0 & 0.011 \\ 0 & 0.360 \\ 0 & 0.116 \end{bmatrix} \tag{7}$$

and selected knot time $\Delta t = 0.6 \; s$ as a function of how fast we were able to compute the next trajectory segment. Trajectory replanning is discussed in section II-.2, but it is worth mentioning that our testing was done by computing only the next 21 s of trajectory each time.

The precise construction of the cost function is as follows: Distance from the goal is quadratically penalized at each knot point; as well as a penalty on jiggling the aileron input, squared, too much with weight $r_1 = 100$. Only the first $4/5^{\text{th's}}$ of aileron input is penalized, a feature that stems from the replanning approach. Penalty on last couple of aileron

commands tends to ignore the long term progression of glider and results in unsatisfactory effects in tail portion of the trajectory plan.

The third term is more interesting, smoothness between every fourth roll angle is rewarded via weight $p_1 = -50$. This objective function therefore promotes indefinite circling in either, but a consistent direction, near the goal; while simultaneously guiding the glider towards the goal.

$$\begin{aligned}\mathbf{x}^d, \mathbf{y}^d, \phi^d = \underset{\mathbf{x},\mathbf{u}}{\operatorname{argmin}} \quad & \sum_{l=1}^{N} \left({x_l}^2 + {y_l}^2\right) \\ & + \; r_1 \sum_{l=1}^{0.8N} {u_{l,2}}^2 \\ & + \; p_1 \sum_{l=4}^{N} \phi_{l-3}\, \phi_l \end{aligned}$$

$$\begin{aligned}\text{s.t.} \quad & \mathbf{A}_d \bar{\mathbf{x}}_l + \mathbf{B}_d \bar{\mathbf{u}}_l - \bar{\mathbf{x}}_{l+1} = 0 \\ & \mathbf{A}_d \bar{\mathbf{x}}_0 + \mathbf{B}_d \bar{\mathbf{u}}_0 - \bar{\mathbf{x}}_1 = 0 \\ & |\phi_l| \leq 40^o, \quad \forall l \in [1, N] \end{aligned} \tag{8}$$

Lastly, the trajectory objective function is equality constrained by the feasible closed loop glider dynamics. The initial condition for each trajectory segment is equally constrained. To prevent the glider from nose diving in a fervent spin at the goal, the roll angle at each knot point is constrained to less or equal to 40^o. This is a convex optimization problem and details on solving it is discussed further in section VI-A.3.

Solving the optimization problem produces a trajectory tape into the immediate future. The next task is actually being able to follow the trajectory prescribed by the knot points given by $\mathbf{x}^d, \mathbf{y}^d$. Note, we also extract the desired roll angle throughout the trajectory tape, ϕ^d, which is used as a feed-forward control term. How the trajectory tape and controller are integrated is discussed in section IV.

IV. Control

A control system maintains the glider on the desired trajectory. We implemented a cascaded controller (fig. 4) to use the predicted vehicle roll angle, computed during trajectory optimization, together with current vehicle state to achieve the desired flight path. Cascaded control is intuitively simple but has sub-optimal performance. Limitations of cascaded control are the prime motivator for a single L_2 gain optimized controller, discussed in section V.

A. Feed-forward control

The cascaded controller consists of two individual controllers: The first producing a roll angle set-point, using trajectory optimization predicted roll, distance from intended trajectory track and heading angle variation from the current incremental portion of predicted trajectory. The second controller takes the output of the first, along with current glider state, and controls the glider velocity and roll angle.

The cascaded control actuates the glider ailerons and elevator, but is able to follow a precomputed trajectory in 3D space. A visual representation of the cascaded control is presented in fig. 4.

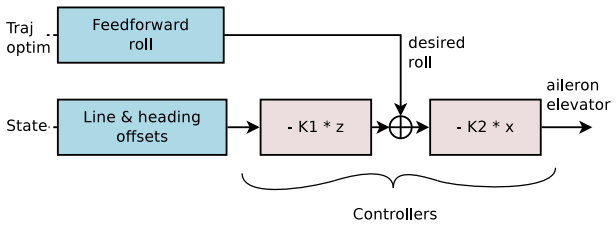

Fig. 4. Diagram of the cascaded controller used in this work, with trajectory optimization feed-forward term.

Special care is needed with the heading and line offsets. The perpendicular distance ϵ_L from intended trajectory can potentially be a large number, and cannot be directly used to influence the roll angle set point – if a linear relation is used, a large line offset could topple the glider, or result in over-correcting the current heading away from the desired trajectory. Second, the delta heading offset $\Delta\psi$ is non-linear.

We use the tangent and arctangent functions to map both ϵ_L and $\Delta\psi$ into similar near linear spaces:

$$\mathbf{z} = \begin{bmatrix} \phi^{ff} \\ \frac{1}{2}\arctan(\epsilon_L) \\ \tan(\Delta\psi) \end{bmatrix}$$

$$-K1 = \begin{bmatrix} 1 & 1 & -0.75\arctan(200\,\frac{\pi}{180}) \end{bmatrix}$$

where $200\,\frac{\pi}{180}$ indicates a 45^o heading approach angle to the intended trajectory at $\epsilon_L = 200\ m$, and 0.75 a step response softening factor found through experimentation with the closed loop system. The $\tan\Delta\psi$ is excessive, but empirical performance favored the use of the $\tan$ function.

The trajectory is maintained via the second and third terms. The second component of eq. (9) uses a scaled $\arctan$ function to force the glider to roll in toward the trajectory at a steeper angle the further it is away (note that this offset is negative when the glider is to the right of the trajectory segment, and positive when it is to the left).

The third component of eq. (9) serves a similar purpose, but uses the difference in angle between the trajectory segment and the heading of the glider; this function is similar to an inverted 'bowl' whose vertical scaling is controlled by the $\arctan$ term, so the greater the difference in angle, the harder the roll. The combined effect of these last two components is to push the glider toward the trajectory segment, reducing the steepness of approach as the glider gets closer to the line defined by this segment. The feed-forward component of eq. (9) allows the glider to anticipate changes in roll of the trajectory as it reaches each knot point.

The desired roll, which is the set point input to the second phase controller, is computed by:

$$\phi_{desired} = -K1 \cdot \mathbf{z} \tag{9}$$

The second phase is a usual gain matrix and state controller as follows:

$$x = \begin{bmatrix} \Delta_\phi \\ \Delta_\theta \end{bmatrix} \quad K2 = \begin{bmatrix} 5 & 11 \end{bmatrix}$$

where $\Delta\phi$ is the difference between the current roll of the vehicle and the desired roll, and $\Delta\theta$ is the difference between the current pitch of the vehicle and the desired pitch (not shown in fig. 4). The overall effect is that this cascaded controller works to control the roll of the glider in order to follow the desired trajectory, while pitch is used to maintain a constant flight speed, as assumed in our trajectory optimization.

Note this second phase controller could be designed through LQR type methods, but again, we favor a more encompassing design to replace the cascaded control architecture and is discussed in section V.

B. Line following

In the previous section we explained the inner workings of our controller and how it works to maintain the desired trajectory. The cascaded controller actually does this with respect to the line defined by two consecutive knot points of the trajectory (a trajectory segment), controlling the vehicle roll to follow this line. When the vehicle passes a virtual 'wall' (the current 'wall' can be seen in green/cyan in fig. 5) at the end of the segment defined by the two knot points, the controller then works with the line defined by the next two consecutive knot points. We use simple trigonometric formulas to calculate the perpendicular distance between the glider and the line defined by the segment, on which side the glider is with respect to this line, and whether or not the vehicle has passed the 'wall' at the end of the segment. An example of our trajectory following cascaded controller can be seen in fig. 5, where the glider is purposefully started at an offset to the desired trajectory, and is able to converge onto, and follow it.

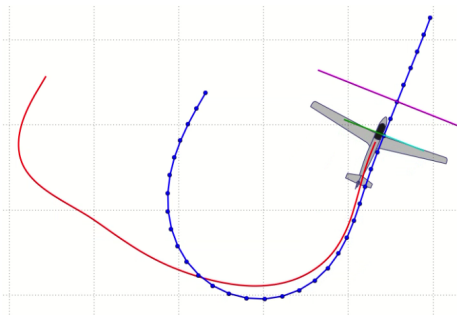

Fig. 5. Visualization of a desired trajectory with knot points, and a trace of the vehicle path using our cascaded controller.

V. H_∞ / L_2-GAIN OPTIMIZED CONTROLLER

To improve performance, we consider designing a H_∞ optimized controller which would take measurements from the system and produce aileron and elevator control outputs. A thorough controller design is also motivated as a mechanism to establish some guarantee of optimal disturbance rejection. We suspect a glider would certainly encounter strong disturbances when entering a thermal or other atmospheric updraft affects.

We would like to have a single controller which produces control signals to maintain the glider on a trajectory flight path, where trajectory tape segments are generated as described in section III. To keep the glider on track, we do H_∞ controller design on an approximated all linear dynamical model.

Another item on the controller wish list, is to keep flight control states in a local relative coordinate frame. We represent this as a clear distinction, since human glider pilots follow features on the ground to find and hone in on thermals. We imagine a system whereby navigation state is estimated based on ground relative features, which may be augmented by GPS aids if they are available.

As a last note, the controller design presented here is computed using an *OpenSource* work-in-progress H_∞ / L_2-gain optimized controller design software[1], which uses *Octave, SeDuMi* and *Yalmip* to solve the H_∞ design through Semidefinite Programming.

A. Plant dynamics

We follow the standard linear multi-input multi-output plant model for multivariable control design:

$$\begin{aligned} \dot{\bar{\mathbf{x}}} &= \mathbf{A}\bar{\mathbf{x}} + \mathbf{B}_1\mathbf{w} + \mathbf{B}_2\bar{\mathbf{u}} \\ \mathbf{e} &= \mathbf{C}_1\bar{\mathbf{x}} + \mathbf{D}_{11}\mathbf{w} + \mathbf{D}_{12}\bar{\mathbf{u}} \\ \mathbf{y} &= \mathbf{C}_2\bar{\mathbf{x}} + \mathbf{D}_{21}\mathbf{w} \end{aligned} \tag{10}$$

were we enforce the low-pass plant restriction of $\mathbf{D}_{22} = \mathbf{0}$. Noise and unknown inputs are vectorized in $\mathbf{w}$, with system errors collected in $\mathbf{e}$. The optimization is focused on minimizing L_2 gain γ:

$$\gamma = \frac{||\mathbf{e}||_2}{||\mathbf{w}||_2} \tag{11}$$

We assume measurements $\mathbf{y}$ are available from the plant and will influence the plant state $\mathbf{x}$ using input $\mathbf{u}$. Where,

$$\begin{aligned} \mathbf{x} &= \begin{bmatrix} \phi & \theta & {}^r\psi & {}^bv_x & \epsilon_L \end{bmatrix}^T \\ \mathbf{u} &= \begin{bmatrix} aile & elev \end{bmatrix}^T \\ \mathbf{w} &= \begin{bmatrix} \Delta v^d & \Delta\psi^d & \Delta\phi^d & \Delta\theta^d & \Delta{\epsilon_L}^d \end{bmatrix}^T \\ \mathbf{e} &= \begin{bmatrix} \tilde{\psi} & \tilde{\phi} & {}^b\tilde{v_x} & \tilde{\epsilon_L} & \tilde{aile} & \tilde{elev} \end{bmatrix}^T \\ \mathbf{y} &= \begin{bmatrix} \phi & \theta & \Delta\psi & {}^bv_x & \epsilon_L \end{bmatrix}^T \end{aligned} \tag{12}$$

Our linearization of nonlinear dynamics is about the operating point, $\mathbf{x}_0$, of the *ASK-13* glider

$$\mathbf{x}_0 = \begin{bmatrix} 0 & 1.5^o & 0 & 25.5\ m/s & 0 \end{bmatrix}^T \tag{13}$$

1) Incorporating trajectory tape following in a linear model: Next we need to define the gradients associated with transformation matrices in eq. (10), and is discussed further in the next section on parameter estimation.

[1] github.com/dehann/HinfControl

B. Parameter Estimation

Rather than indirectly estimate system parameters, such as fuselage drag coefficient, we find the system gradients directly through experiment. System gradients can only be measured if the system is perturbed about the linearization point $\mathbf{x}_0$. We choose to use sinusoidal stimulus and a lock-in amplifier detection scheme to explicitly measure each of the gradients. In the glider case, we use a basic sinusoidal generation and logging mechanism in the control scripts to stimulate the system.

We present our method for gradient measurement by example of one of the parameters and then present the remaining parameters of interest directly. The major advantage of this method is that we avoid nonlinear modeling, followed by inevitable system identification – here we skip straight through and measure the linearized dynamics of the system directly. Furthermore, with this approach we also implicitly measure the scaling between SI units in the state space and actuator output forces. The disadvantage of this approach is that if the plant model changes, many of the parameters must be remeasured, however, this is not a sticking point for this project.

1) Lock-in amplifier method: Stimulus to the system is generated by cosine function at some desirable frequency. This cosine setpoint is then fed into one of the system inputs and the response of the system is measured. All measurable parameters listed in eq. (12), along with cosine and an equal sine setpoint, are logged for post-processing. Since the system states are always changing, we are able to perform valid numerical differentiations and compute the relevant gradients.

For example, if we wish to measure the gradient of pitch rate to elevator input, we could take

$$h_{\dot{\theta}\,e}(t) = \frac{\dot{\theta}}{elev} \tag{14}$$

but is difficult to measure accurately in the presence of other disturbances.

If we drive $\dot{\theta}$ at the stimulus frequency $\omega = 2\pi f$ and average over a window, we can find a more accurate estimate of the gradient. That is:

$$\begin{aligned} elev &= \alpha \cos(\omega t) \\ \frac{\partial \theta_k}{\partial t} &\approx \frac{\theta_k - \theta_{k-1}}{\Delta t} \\ SPC_k &= \frac{\partial \theta_k}{\partial t} \cos(\omega t) \\ SPS_k &= \frac{\partial \theta_k}{\partial t} \sin(\omega t) \end{aligned} \tag{15}$$

from here we compute the low-pass filtered magnitude response, or average amplitude over N samples:

$$\begin{aligned} \bar{S}_c = \frac{1}{N}\sum_{k=2}^{N} SPC_k, \qquad \bar{S}_s = \frac{1}{N}\sum_{k=2}^{N} SPS_k \\ \bar{h}_{\dot{\theta}\,e} \approx \frac{1}{\alpha}\sqrt{\bar{S}_c{}^2 + \bar{S}_s{}^2} \end{aligned} \tag{16}$$

If the stimulus scale α is small enough and frequency f low enough transient affects become negligible, however, transient affects can be measured through the phase response at increased frequencies:

$$\text{phase} = \text{atan2}\left(\frac{\bar{S}_s}{\bar{S}_c}\right) \tag{17}$$

Using this approach, we measure each of the gradients of interest. As a last step in finding a linear control system, we define all gradients for the multi-input multi-output system as follows:

$$\begin{aligned} \mathbf{A} &= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ -0.447 & 0 & 0 & 0 & 0 \\ 0 & -41.7 & 0 & 0 & 0 \\ 0 & 0 & -0.9\,{}^{b}v_x & 0 & 0 \end{bmatrix} \\ \mathbf{B}_1 &= \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & -20 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \mathbf{B}_2 = \begin{bmatrix} 0.27 & 0 \\ 0 & -0.38 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \end{aligned} \tag{18}$$

The error output is generated by:

$$\begin{aligned} \mathbf{C}_1 &= \begin{bmatrix} 0 & 0 & -1 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \\ \mathbf{D}_{11} &= \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \mathbf{D}_{12} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 10 & 0 \\ 0 & 10 \end{bmatrix} \end{aligned} \tag{19}$$

Lastly, sensor measurement output of the plant is modeled as:

$$\begin{aligned} \mathbf{C}_2 = -\mathbf{I}_{5\times 5} \qquad \mathbf{D}_{22} = \mathbf{0}_{5\times 2} \\ \mathbf{D}_{21} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{20}$$

Running this through the H_∞ design synthesis code produces the following controller, but it remains untested at this point in the project. The augmented state space controller terms are $\begin{bmatrix} \mathbf{A}_f & \mathbf{B}_f \\ \mathbf{C}_f & \mathbf{D}_f \end{bmatrix}$:

$$\left[\begin{array}{rrrrr|rrrrr} -43700 & -0.0222 & 1.8e+5 & -0.00184 & -6276 & -0.21 & -0 & 0.0045 & 0 & 0.0174 \\ -0.0452 & -4.649 & 0.1907 & 0.4323 & -0.0065 & 0 & 0.223 & 0 & 0.0112 & -0 \\ -0.404 & 0 & -20.1 & -0 & 0.0175 & 0.0045 & 0 & -4.47 & -50 & 0.00316 \\ 0 & -41.65 & -5e-05 & -2.04 & 1e-05 & -0 & 0.0112 & -0 & -0.456 & 0 \\ -1.405 & -1e-05 & -19.32 & 0 & -0.5641 & -0.146 & -0 & 0.00316 & 0 & -0.102 \\ \hline -7.3e+05 & -0.3709 & 3.076e+06 & -0.03073 & -1.05e+05 & 0 & 0 & 0 & 0 & -1 \\ 0.529 & 66.08 & -2.232 & -4.475 & 0.0760 & 0 & 0 & 0 & 0 & 0 \end{array}\right]$$

VI. Implementation

Given the theoretical description of our system, we now provide a brief explanation on its practical implementation.

A. Architecture

Our system has a number of interacting sub-systems, as detailed in the block diagram of fig. 6, with communication achieved via TCP/UDP and JSON encoded messages.

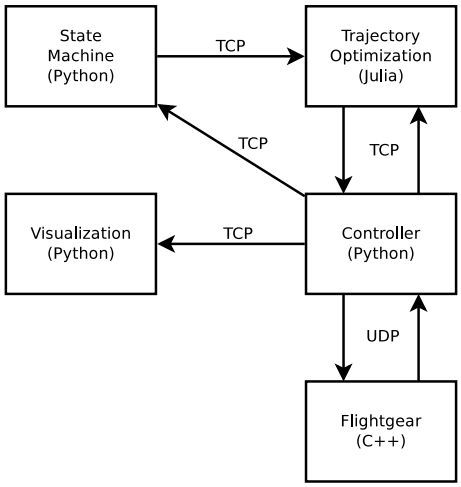

Fig. 6. Diagram of our system architecture.

1) Flightgear: Flightgear is an open-source flight simulator supporting a variety of platforms [5]. Flightgear uses an integrated flight dynamics model called YASim, which uses the geometry of an aircraft model to generate its base flight characteristics and dynamics. Its weather engine is able to simulate thermals and conditions for soaring, among other realistic effects. This work utilizes Flightgear to simulate our chosen glider, the ASK-13, as well as the thermal updraft used for soaring. FlightGear allows access to a very large number of internal state variables (organized in a hierarchal "property" tree) via a number of access mechanisms, which allow both the monitoring of vehicle state variables, and the external control of vehicles. Glider state is obtained via its 'native protocol' which exposes the property tree over UDP, and glider control is achieved via its 'generic protocol' which accepts elevator and aileron commands over UDP. These protocols interface the Flightgear ASK-13 model directly with our controller, implemented in Python.

2) Controller: Our controller, implemented in Python, continuously monitors the state of the glider and exposes these values to our trajectory optimization, state machine, and visualization programs; as well as this, it receives desired trajectories from our trajectory optimization program, and implements our cascaded controller detailed in section IV, sending aileron and elevator control commands over UDP to Flightgear. Communication to all other subsystems besides Flightgear is achieved using TCP, with the controller acting as a TCP server. Messages between these subsystems (such as glider state and trajectory information) are stored in dictionaries, and encoded and transmitted over TCP as JSON strings.

3) Trajectory Optimization: Our trajectory optimization program, written in Julia, implements and continuously solves the optimization formulation detailed in III as quickly as possible, allowing us to achieve on-line trajectory re-planning. Solving of the non-linear optimization utilizes the JuMP (Julia for Mathematical Programming) package, part of the JuliaOpt project, and NLopt package, finding solutions via the SLSQP sequential quadratic programming algorithm. In addition, the trajectory optimization program receives way-point commands from our state machine, allowing us to command the glider to other desired points outside the thermal. It also uses a simple time metric to check if the glider has completed the desired trajectory in a reasonable amount of time, and if it fails to do so, recomputes a new trajectory to recover from undesired states.

4) State Machine: We implemented a simple finite state machine in Python to perform the following task: Enter the thermal until the glider reaches a target altitude ($1500\ m$); randomly select a way-point in a $10000 \times 10000\ m$ box centered at the thermal and travel to it; perform this action until the glider reaches an altitude where it is determined that traveling back to the thermal will cause it to fall below a target altitude ($1000\ m$); return to the thermal and repeat. The glider should stay above a virtual altitude limit cone centered at the thermal.

In effect, this state machine causes the glider to perpetually randomly explore a $10000 \times 10000\ m$ box around the thermal.

Way-points are sent to the trajectory optimization process via TCP. It is anticipated that this state machine can be used as a basis for more complicated behaviors in future work, namely searching for thermals, optimal surveying, or long-distance flight.

5) Visualization: Our final program is a visualizer written in Python, which uses PyLab (incorporating NumPy, SciPy, and Matplotlib) to visualize the state of the glider, the calculated desired trajectory, and our controller performance in real-time, receiving this information from the controller over TCP. This tool proved invaluable during development, enabling the debugging of issues related to the controller, trajectory optimization, and state machine. Examples of the visualizer output can be seen in fig. 2 and fig. 5.

As a final note, we mention that all elements of this project are open-source and freely available. In addition, the code for all sub-systems is available at [2].

VII. Results

We would like to show a few interesting results from the in-situ trajectory replanning system. As a general overview, fig. 7 shows a portion of glider trajectory showing the glider just before it enters a known thermal location at low altitude. The glider is then led up in near continuous circles, riding the thermal, until a targer trigger altitude of $1500\ m$ is reached.

Once at altitude, the state machine sends a new waypoint command to the trajectory replanner. From here the glider is led to two different waypoints before the gliding limit cone is reached. The glider is then led back to the last known thermal location. The trajectory replanner would then lead the glider back into the thermal, recenter and climb to the designated altitude.

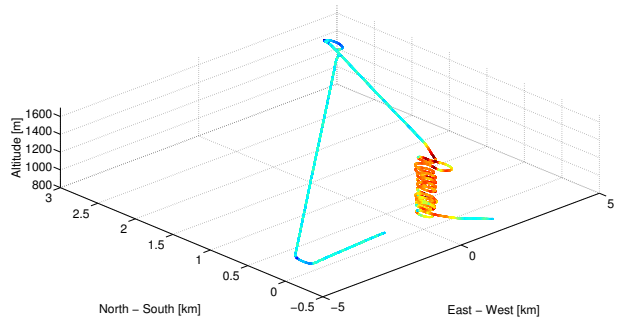

Fig. 7. Three dimensional plot of a climb, explore and return to last thermal location. Blue trace segments show decreasing altitude, while orange portions indicates the glider gaining altitude inside the thermal.

Regarding thermal centering, fig. 2 shows how the trajectory optimization centers the glider flightpath around the estimated thermal core location. Fig. 7 shows a full climbing event of circular trajectory segments near the core of the thermal.

[2]"Dynamic replanning of sailplane trajectories for autonomous thermal soaring". https://github.com/dehann/ThermalSoaring

We have focused on high fidelity trajectory control in this work. The high fidelity control makes it possible to introduce small roll angle adjustments near the estimated thermal location. Human glider pilots are trained to either reduce or increase the roll angle of the glider based on certain 'seat-of-the-pants' cues from glider. If the glider climb rate is increasing, roll less, if the climb rate passes the peak value, turn harder for a short segment. These adjustments are key to successfully controlling a glider into core lift.

Fig. 8 shows predicted and actual roll angles near the end of a thermal climbing cycle. This segment of glider trajectory is shown as it shows several interesting aspects of the entire closed loop system – note several comments included in the figure.

The glider is rolled to the right at time $0\ s$ in fig. 8 and thereby circling clock-wise inside a thermal air-current. The trajectory replanner then throws the glider into an opposite roll limit of -40^o, that is a counter clockwise circle. This flip in direction is a parasitic case in the trajectory replanning objective function, and is caused due to limited time horizon trajectory planning (local minima in the objective). The reward term, in eq. (8), is used to promote circling in the same direction, while penalty on aileron input is used to reduce the variations in roll. However, this reward and penalty cannot be increased too much, since the solution then tends to converge into a well centered loop very slowly. The occasional 'S-bends' are therefore tolerated.

Next we note the lag between the feed-forward (predicted) and actual roll angles, even though the turn is anticipated by the line following controller. These results are generated with the cascaded control setup. The feed-forward term is used as a set-point to the second controller – resulting in the step response type delay from the trajectory plan. This delay is annotated by the cascaded control delay tag, following a large 80^o roll change event.

These control delay dynamics are not modeled during trajectory optimization and results in variations between the predicted and actual angles. There are two important points to mention here: Atmospheric disturbances were enabled during this data. The variation between the predicted (blue) and actual (red) traces in fig. 8 are a result of random (wind gust) noise, unmodeled dynamics and cascaded controller lag. The second important aspect is the need for non-cascaded, disturbance rejection optimized control and is the prime motivator for section V – please refer to that section for more details.

The next important aspect shown in fig. 8 is annotated as centering adjustments. These show how the trajectory optimization is able to predict the motion of the glider with high fidelity. The roll set-point and actual roll is deliberately reduced for a few seconds, near the $35\ s$ mark, as a centering adjustment to better align the cicle flightpath around the goal point. The glider then returns to the maximum constrained roll angle of $\pm 40^o$.

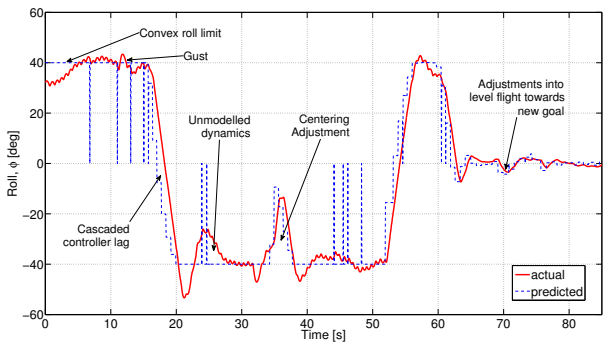

Fig. 8. Predicted and actual roll angle during final portion of a climb, before the glider is led away to a new goal location. *Flightgear* simulator's random atmospheric turbulence/disturbances was enabled producing several moderate gust disturbances.

VIII. Conclusion

The proposed mechanism of control is intended to reduce the glider trajectory and thermal centering task to that of nudging an estimated goal position around in a local relative coordinate frame. We have shown, at least in simulation with *Flightgear*, that the approach is feasible and able to center and climb a known thermal. It remains to be seen if this control approach will work on a real-world vehicle, but hope our reliance on an established, and external, simulator will win some confidence in the proposed method.

Growth paths for this work clearly include cooperation with thermal position estimation processes. We argue it is important to relax the 'known' thermal location, to an 'approximate' thermal location, whereby the trajectory optimization goal point is nudged around as more sensor data is collected in the region of a possible thermal. The next step would then probably be the use of Doppler radar, infra-red or color cameras to predict possible thermal search regions.

Lastly, we have used a basic state machine to transition between different flight modes – we hope this approach can be a good interface to other potential users of such a system.

References

[1] Russ Tedrake, "Drake: A planning, control, and analysis toolbox for nonlinear dynamical systems", 2014, "http://drake.mit.edu".

[2] Andersson, Klas, et al. "Thermal Centering Control for Autonomous Soaring; Stability Analysis and Flight Test Results." Journal of Guidance, Control, and Dynamics 35.3 (2012): 963-975.

[3] Allen, Michael J., and Victor Lin. "Guidance and control of an autonomous soaring vehicle with flight test results." AIAA Aerospace Sciences Meeting and Exhibit, AIAA Paper. Vol. 867. 2007.

[4] Twigg, Shannon, Anthony Calise, and Eric Johnson. "On-line trajectory optimization for autonomous air vehicles." AIAA guidance, navigation, and control conference. 2003.

[5] "FlightGear Flight Simulator." https://www.flightgear.org, N.p., n.d. Web. 12 Dec. 2014.